



Universidad
Zaragoza

Trabajo Fin de Grado

Armygeddon: desarrollo de un videojuego de estrategia en tiempo real basado en StarCraft

Autor:

Antonio Gallén Recio

Director:

Eduardo Mena Nieto

Titulación:

Grado en Ingeniería Informática

Escuela de Ingeniería y Arquitectura, Zaragoza

Agosto 2023

RESUMEN

El proyecto ha consistido en la creación de Armygeddon, un videojuego de estrategia en tiempo real¹ basado en StarCraft [9] [2] y ampliado con nuevas características conforme avanzaba su desarrollo.

Tomando como base lo aprendido en las prácticas de laboratorio [1] de la asignatura Videojuegos, que consistía en la creación de un clon simplificado de StarCraft utilizando *Pygame*², se distinguen las siguientes etapas en el proyecto:

1. Construcción de un videojuego base. Para conseguir un mayor rendimiento se ha utilizado C++ y la biblioteca SDL2 (*Simple DirectMedia Layer 2.0*) [7].
2. Creación de un editor de escenarios. Siendo un escenario todos los datos necesarios para poder jugar una partida (configuración de los elementos del mapa, recursos iniciales, ...) se desarrolla un editor para crear, visualizar y modificar dichos escenarios.
3. Aumento de la profundidad del juego. En vez de centrar las partidas en la lucha entre dos ejércitos, ahora se permiten hasta ocho ejércitos y, estos pueden estar a su vez agrupados en distintas facciones de aliados. Además, ya no se trata solo de exterminar a los rivales sino que se añaden otros tres tipos de objetivos distintos.
4. Creación de un modo multijugador en línea. Basado en el paso de las órdenes de cada jugador con una arquitectura cliente-servidor.
5. Desarrollo de la inteligencia artificial. Además de búsqueda de caminos y movimiento automático de cámaras para seguir el transcurso de la partida, también se han creado jugadores artificiales con distintos comportamientos para controlar algunos de los ejércitos.

Puesto que se trata de una construcción evolutiva, la arquitectura software ha tenido que ser abierta y, ha requerido de una depuración continua para facilitar lo máximo posible la incorporación de las nuevas características conforme surgieran.

¹Un videojuego de estrategia en tiempo real o RTS (*Real Time Strategy*) es un subgénero de los videojuegos de estrategia en el que todos los jugadores progresan tomando decisiones estratégicas en tiempo real y de manera simultánea, como contrapunto a los juegos de estrategia por turnos. https://es.wikipedia.org/wiki/Videojuego_de_estrategia_en_tiempo_real (último acceso: 05/08/2023).

²Pygame es una biblioteca diseñada para la creación de videojuegos en *Python*. <https://www.pygame.org> (último acceso: 5/8/2023).

AGRADECIMIENTOS

Agradecer a mi director Eduardo, por la idea del proyecto, por animarme a hacer siempre más, por enseñarme tanto, y por discutir sinceramente conmigo todo lo que hiciera falta hasta encontrar la mejor opción.

Gracias también a mi padre, por desvivirse intentando conseguirme todo el tiempo posible, y porque si me hubiera dejado convencer, él hubiera seguido así hasta que hubiera introducido hasta la última de las posibles mejoras del proyecto.

Finalmente, agradecer también a mi compañero de la carrera y amigo Alberto Lardiés Getán, por permitirme tomar prestado su ordenador para realizar pruebas, y por jugar durante horas en cualquier momento del día partidas y partidas para probar Armygeddon.

En definitiva, y sobre todo, gracias a los que habéis estado en lo bonito y en lo no tan bonito del proyecto.

Para obtener más información, o para discutir algún aspecto relacionado con este trabajo de fin de grado, ponerse en contacto conmigo enviando un correo electrónico a antonio.gallen.98@gmail.com.

Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
2. Diseño del videojuego	3
2.1. Análisis de requisitos	3
2.2. Arquitectura software	4
2.2.1. Bucle de juego	4
2.2.2. Pantallas	6
2.3. Núcleo de juego	7
2.4. Diseño del jugador artificial	9
2.5. Arquitectura del modo multijugador en línea	11
2.5.1. Controladores cliente-servidor	12
2.5.2. Primer diseño: Actualizaciones gráficas completas	14
2.5.3. Segundo diseño: Actualizaciones gráficas parciales	15
2.5.4. Tercer diseño: Simulación perfecta	15
3. Desarrollo del videojuego	17
3.1. Interacción con el usuario	17
3.2. Jugar una partida	19
3.2.1. Dibujo del tablero	19
3.2.2. Velocidad de la partida	21
3.2.3. Entidades y las casillas	22
3.3. Editor de escenarios	25
3.4. Jugador artificial	26
3.4.1. Agresividad	26
3.4.2. Velocidad de respuesta	27
3.4.3. Comportamiento	27
3.4.4. Protocolos	28
3.4.5. Variantes según el tipo de objetivo	29

3.4.6. Bolsa de protocolos	30
3.5. Modo multijugador en línea	30
3.6. Pruebas	32
3.7. Conclusiones	34
3.7.1. Resultado del proyecto	35
3.7.2. Principales dificultades	35
3.7.3. Medición de esfuerzos	36
3.7.4. Posibles expansiones	38
3.7.5. Opinión personal	40
Bibliografía	41
Anexos	42
A. Lista de requisitos de Armygeddon	43
A.1. Videojuego base	43
A.2. Partida en Armygeddon	45
A.3. Editor de escenarios	46
A.4. Multijugador en línea	48
A.5. Jugadores artificiales	48
A.6. Otros	49
B. Lógica de las entidades	50
B.1. Unidades	50
B.2. Estructuras	51
B.3. Refineras	53
B.4. Géiseres	54
B.5. Minerales	55
B.6. Obstáculos	55
C. Inteligencia artificial	57
C.1. Jugador artificial	57
C.1.1. Análisis sobre Armygeddon	58
C.1.2. El usuario de Armygeddon	58
C.1.3. Protocolos	60
C.1.4. Invadir	62
C.1.5. Defender	64
C.1.6. Construir	65
C.1.7. Trabajar	67
C.1.8. Generar soldados	67
C.1.9. Generar trabajadores	69
C.1.10. Parámetros	69

C.1.11. Velocidad de respuesta	70
C.1.12. Comportamiento	71
C.1.13. Implementación de los protocolos	72
C.1.14. Variantes según el objetivo	78
C.1.15. Bolsa de protocolos	80
C.2. Ejército neutral	81
C.3. Búsqueda de caminos	84
C.4. Inteligencia del antiguo invasor	85
D. Escenarios de ejemplo	88
D.1. StarCraft	88
D.2. HelmAbyss	90
D.3. Thermopylae	92
D.4. OpenStarCraft	95
E. Pruebas	100
E.1. Sistema de versiones para depuración	100
E.2. Pruebas de balance	101
E.2.1. StarCraft	101
E.2.2. Thermopylae	102
E.2.3. HelmAbyss	102
E.2.4. OpenStarCraft	103
E.3. Pruebas del jugador artificial	104
E.3.1. Agresividad al máximo	104
E.3.2. Agresividad al mínimo	105
E.4. Pruebas de requisitos	105
E.5. Pruebas de conectividad multijugador	106
F. Organización de archivos, directorios y mensajes	108
F.1. Árbol de directorios	108
F.2. Archivo de configuración	111
F.3. Archivo de escenario	114
F.4. Archivo de guardado y mensajes en línea	116
G. Ingeniería inversa	118
G.1. Tamaño de las casillas	118
G.2. Nombre ficheros sonido e imagen	119
G.3. Animaciones entidades	119
H. Principales dificultades del desarrollo	122
H.1. Sistemas de distancia	122
H.1.1. Distancia real	122

H.1.2. Distancia optimizada	124
H.2. Gestión y acceso de la memoria	125
I. Interfaces	127
I.1. Pantallas	127
I.2. Elementos reactivos	132
I.2.1. Implementación	132
I.2.2. Respuesta audiovisual	135
I.2.3. Organización	136
I.3. Estándares cromáticos	136
I.4. Contenido audiovisual	139
I.4.1. Gráficos	139
Gráficos originales	139
Gráficos modificados	139
Gráficos nuevos	141
I.4.2. Música	141
I.4.3. Efectos	143
J. Posibles expansiones	144
J.1. Fusión de terrenos	144
J.2. Mejora del objetivo investigación	145
J.3. Reproductor de música	146
J.4. Efectos en partida del terreno	147
J.5. Otras mecánicas de las razas	147
J.6. Sistema manual de creación de elementos	148
J.7. Carga multijugador	149
J.8. Navegación por teclado completa	149
J.9. Rehumanización del jugador artificial	150
J.10. Customización	150
J.11. Nuevo sistema de casillas y posiciones	151
K. Resumen del trabajo práctico StarCraft	152
L. Registro de cambios	155
M. Manual de usuario	203
M.1. Resumen	204
M.2. Requisitos	205
M.3. Navegación	205
M.4. Controles	207
M.5. Ajustes	210
M.5.1. Teclado	210

M.5.2. Sonido	211
M.5.3. Vídeo	212
M.5.4. Extra	213
M.6. Reglas del juego	214
M.6.1. Partida de Armygeddon	215
M.6.2. Facciones	215
M.6.3. Jugador	215
M.6.4. Ejércitos	216
M.6.5. Raza	217
M.6.6. Cristal	217
M.6.7. Gas	217
M.6.8. Población	218
M.6.9. Unidad	218
M.6.10.Estructura	220
M.6.11.Refinera	220
M.6.12.Obstáculo	220
M.6.13.Órdenes del jugador	220
M.6.14.Objetivo de facción: exterminar	221
M.6.15.Objetivo de facción: sobrevivir	222
M.6.16.Objetivo de facción: conquistar	223
M.6.17.Objetivo de facción: investigar	224
M.7. Protoss	225
M.7.1. Trabajador: Sonda	225
M.7.2. Soldado básico: Fanático	226
M.7.3. Soldado avanzado: Dragón	227
M.7.4. Edificio base: Nexo	228
M.7.5. Edificio poblacional: Pílon	229
M.7.6. Edificio reclutamiento: Portal	230
M.7.7. Edificio avanzado: Centro cibernético	231
M.7.8. Instalación refinera: Asimilador	231
M.8. Zerg	232
M.8.1. Unidad especial: Larva	233
M.8.2. Unidad especial: Huevo	234
M.8.3. Unidad poblacional: Soberano	235
M.8.4. Trabajador: Zumbido	236
M.8.5. Soldado básico: Zergling	237
M.8.6. Soldado avanzado: Hidralisco	238
M.8.7. Edificio base: Criadero	239
M.8.8. Edificio especial: Estanque de desove	239
M.8.9. Edificio avanzado: Guarida de hidraliscos	240

M.8.10.Instalación refinera: Extractor	241
M.9. Terran	242
M.9.1. Trabajador: VCE	243
M.9.2. Soldado básico: Marine	244
M.9.3. Soldado avanzado: Murflama	244
M.9.4. Edificio base: Centro de operaciones	245
M.9.5. Edificio poblacional: Almacén de suministros	246
M.9.6. Edificio reclutamiento: Cuarteles	247
M.9.7. Edificio avanzado: Academia	248
M.9.8. Instalación refinera: Refinería	249
M.10 Editor de escenarios	250
M.10.1.Modos de edición	251
M.10.2.Cuadrícula	252
M.10.3.Herramientas de selección	253
M.10.4.Ajustes de visibilidad	254
M.10.5.Edición de ejércitos	256
M.10.6.Edición de facciones	256
M.10.7.Descripción del escenario	256
M.11 Problemas frecuentes	256

Índice de figuras

2.1. Esquema del bucle de juego de Armygeddon.	5
2.2. Última evolución del modelo de pantallas de Armygeddon.	6
2.3. Tamaño real y en casillas de una larva y de un criadero.	8
2.4. Esquema de los tipos de entidad de Armygeddon.	8
2.5. Relación de los protocolos del jugador artificial y las acciones del jugador humano que engloban junto con los tipos de órdenes usadas.	11
2.6. Creación de sala en línea y emparejamiento.	12
3.1. Menú principal del editor de escenarios de Armygeddon.	18
3.2. Posibles estados de un elemento reactivo simple: botón ‘Guardar’ de ajustes.	18
3.3. Parte del tablero de una partida sin y con cuadrícula activa.	20
3.4. Diagrama de flujo del dibujo del escenario.	20
3.5. Distintos bucles de juego en función de la velocidad.	22
3.6. Posiciones dentro de cada casilla.	23
3.7. Diagrama de flujo de la actualización de estado de una unidad recolectando.	24
3.8. Esquema de las transiciones entre estados de las unidades.	24
3.9. Ejemplo de pegar una selección de 2x3 en un espacio de 7x8.	26
3.10. Elección y reequilibrio de los protocolos del jugador artificial en una actualización.	30
3.11. Rutas de mensajes y estructura de los controladores de conexiones. . .	31
3.12. Captura del foro para la depuración de Armygeddon: reportes (izquierda) e hilos (derecha).	33
3.13. Casillas afectadas en 3,5 casillas en distancia real (izquierda) y en distancia optimizada (derecha).	36
3.14. Medición de esfuerzos en el proyecto según el mes y el tipo de tarea. . .	37
3.15. Cómputo global en horas de esfuerzos en el proyecto según el tipo de tarea.	38
B.1. Esquema de las transiciones entre estados de las unidades.	51

B.2. Esquema de las transiciones entre estados de las estructuras.	52
B.3. Distancia de deposito de recursos alrededor de las estructuras.	53
B.4. Esquema de las transiciones entre estados de las refineras.	53
B.5. Distancia respecto a un géiser para instalar una refinería en el.	54
B.6. Esquema de las transiciones entre estados de los minerales.	55
B.7. Alcance de un trabajador relacionado con minerales cercanos.	56
C.1. Relación de los protocolos del jugador artificial y las ideas del jugador humano que siguen junto con los tipos de órdenes usadas.	61
C.2. Esquema de algoritmo de invasión.	72
C.3. Esquema de algoritmo de actualizar invasiones.	74
C.4. Esquema de algoritmo de Defender.	75
C.5. Esquema de algoritmo de Construir.	76
C.6. Esquema de algoritmo de Trabajar.	77
C.7. Esquema de algoritmo de Generar Soldados.	78
C.8. Elección y reequilibrio de los protocolos del jugador artificial en una actualización.	80
C.9. Diagrama de flujo de la actualización del ejército neutral.	82
C.10. Diagrama de flujo de la búsqueda de caminos.	84
C.11. Diagrama de flujo de la generación de un movimiento de invasor.	86
D.1. Escenario de ejemplo ‘StarCraft’.	89
D.2. Estado inicial del escenario ‘StarCraft’ con las celdas y accesos indicados.	90
D.3. Escenario de ejemplo ‘HelmAbyss’.	91
D.4. Estado inicial del escenario ‘HelmAbyss’.	92
D.5. Fortaleza de ‘HelmAbyss’ señalizada.	93
D.6. Escenario de ejemplo ‘Thermopylae’ (izquierda) y estado inicial (derecha).	94
D.7. Estado inicial del escenario ‘OpenStarCraft’.	95
D.8. Creación de ‘OpenStarCraft’ paso 1.	96
D.9. Creación de ‘OpenStarCraft’ paso 2.	97
D.10. Creación de ‘OpenStarCraft’ paso 3.	97
D.11. Creación de ‘OpenStarCraft’ paso 4.	98
D.12. Escenario de ejemplo ‘OpenStarCraft’.	99
E.1. Captura del foro en Discord para la depuración de Armygeddon.	101
E.2. Captura de pantalla de uno de los nodos durante la prueba multijugador.	106
F.1. Árbol de directorios: Parte I.	108
F.2. Árbol de directorios: Parte II.	109
F.3. Árbol de directorios: Parte III.	110
F.4. Árbol de directorios: Parte IV.	111

G.1. Murflama ejemplo en todas las direcciones de una unidad.	120
G.2. Intento de obtención de la animación de un murflama.	121
G.3. Intento de obtención de la animación de un zergling.	121
H.1. Ejemplo de distancia real.	123
H.2. Ejemplo de distancia optimizada.	124
I.1. Primera evolución del modelo de pantallas de Armygeddon.	127
I.2. Segunda evolución del modelo de pantallas de Armygeddon.	128
I.3. Tercera evolución del modelo de pantallas de Armygeddon.	129
I.4. Cuarta evolución del modelo de pantallas de Armygeddon.	129
I.5. Quinta evolución del modelo de pantallas de Armygeddon.	131
I.6. Posibles estados de un elemento reactivo simple: botón ‘Guardar’ de ajustes.	133
I.7. Ejemplo de marcas: ajustes de trucos al crear una partida.	133
I.8. Ejemplo de botones de radio: tipo de jugador al crear una partida. . . .	133
I.9. Ejemplo de botones de transición menor: Ajustes de avatar en línea. . .	134
I.10. Ejemplo de botón de transición mayor: Botón principal del editor. . . .	134
I.11. Ejemplo de botón de escritura: Nombre del escenario en la creación de escenarios del editor.	134
I.12. Ejemplo de botón deslizante: Barra deslizante en ajustes de teclado. . .	135
I.13. Ejemplo de respuesta visual sobre elemento reactivo.	136
I.14. Ejemplo de organización de los elementos reactivos: Creación de partidas.	137
I.15. Icono de Armygeddon.	140
I.16. Ejemplo de cambio de color de una guarida de hidraliscos.	140
I.17. Mapa de colores de una entidad de StarCraft.	140
I.18. Interfaz de las tres interfaces de razas combinadas del editor de escenarios.	140
I.19. Algunos gráficos que se han requerido crear desde cero y que eran cos- tosos de generar por el programa.	141
J.1. Unión de terrenos en Armygeddon.	144
J.2. Fusión de terrenos.	145
K.1. Captura del trabajo práctico StarCraft.	152
M.1. Título de Armygeddon.	203
M.2. Una simulación en Armygeddon.	204
M.3. Mapa de navegación.	206
M.4. Panel derecho de acciones protoss señalado.	209
M.5. Lista de controles en ajustes de teclado.	211
M.6. Ajustes de sonido.	212
M.7. Ajustes de vídeo.	213

M.8. Ajustes extras.	214
M.9. Variación gráfica de las facciones en función de su tipo de objetivo. . .	215
M.10.Variación gráfica de los ejércitos en función de su color.	216
M.11.Ejército neutral en el escenario oficial StarCraft.	217
M.12.Progreso de objetivos de exterminio dentro de partida.	222
M.13.Tabla de facciones dentro de partida en el escenario StarCraft.	222
M.14.Defensa del ejército violeta.	223
M.15.Zona de conquista.	223
M.16.Panel de información de la maravilla dentro de partida.	224
M.17.Distintas apariencias de la maravilla del ejército rojo.	224
M.18.Un protoss.	225
M.19.Cartel: Sonda.	226
M.20.Cartel: Fanático.	227
M.21.Cartel: Dragón.	228
M.22.Cartel: Nexo.	228
M.23.Cartel: Pílon.	229
M.24.Cartel: Portal.	230
M.25.Cartel: Centro cibernético.	231
M.26.Cartel: Asimilador.	232
M.27.Un hidralisco zerg.	233
M.28.Cartel: Larva.	233
M.29.Cartel: Huevo.	234
M.30.Cartel: Soberano.	235
M.31.Cartel: Zumbido.	236
M.32.Cartel: Zergling.	237
M.33.Cartel: Hidralisco.	238
M.34.Cartel: Criadero.	239
M.35.Cartel: Estanque de desove.	240
M.36.Cartel: Guarida de hidraliscos.	241
M.37.Cartel: Extractor.	241
M.38.Un marine terran.	242
M.39.Cartel: Vehículo de construcción espacial.	243
M.40.Cartel: Marine.	244
M.41.Cartel: Murflama.	245
M.42.Cartel: Centro de operaciones.	246
M.43.Cartel: Almacén de suministros.	247
M.44.Cartel: Cuartel.	247
M.45.Cartel: Academia.	248
M.46.Cartel: Refinería.	249
M.47.Captura general del editor de escenarios con señalizaciones.	250

M.48.Ventana emergente del editor de cambios sin guardar.	251
M.49.Escenario con la cuadrícula activa y el control de cuadrícula activo. . .	252
M.50.Herramientas de selección.	253
M.51.Selección en el editor.	253
M.52.Selección en el editor tras pulsar en eliminar.	254
M.53.Ejemplo de pegar una selección de 2x3 en un espacio de 7x8.	255
M.54.Escenario siendo afectado por los ajustes de visibilidad en el editor. . .	255
M.55.Descripción de un escenario vista desde la selección de escenario individual.	257
M.56.Ventana emergente de edición de la descripción de un escenario vista desde el editor de escenarios.	257
M.57.Error al intentar ejecutar Armygeddon sin todos los archivos DLL. . . .	258
M.58.Error al intentar ejecutar Armygeddon sin todos los recursos.	258
M.59.Error al intentar cargar partida sin existir la carpeta './saves'.	258

Índice de tablas

2.1. Algunas diferencias básicas entre los tipos de entidades	9
2.2. Posibles órdenes enviadas en función del estímulo según el primer diseño en línea.	14
3.1. Posibles lapsos temporales entre actualizaciones del jugador artificial según su velocidad de respuesta.	27
3.2. Resultado de las pruebas de balance sobre StarCraft.	33
3.3. Resultado de las pruebas	34
C.1. Relación entre los problemas de diseño del primer modelo de Invadir con las decisiones del segundo.	63
C.2. Posibles lapsos temporales entre actualizaciones del jugador artificial según su velocidad de respuesta.	70
C.3. Variaciones sobre el parámetro de las funciones de retroalimentación del jugador artificial en función de su comportamiento.	71
E.1. Resultado de las pruebas de balance sobre StarCraft.	101
E.2. Resultado de las pruebas de balance sobre Thermopylae.	102
E.3. Resultado de las pruebas de balance sobre HelmAbyss.	102
E.4. Resultado de las pruebas de balance sobre OpenStarCraft variante Ex- terminio.	103
E.5. Resultado de las pruebas de balance sobre OpenStarCraft variante In- vestigación.	103
E.6. Resultado de las pruebas del jugador artificial con agresividad máxima.	104
E.7. Resultado de las pruebas del jugador artificial con agresividad mínima.	105
I.1. Posibles pistas de música para las partidas	142
I.2. Posibles pistas de música para fin de partidas	142
K.1. Comparativa de las características del trabajo práctico de videojuegos que están en Armygeddon.	153
M.1. Estadísticas: Sonda.	226

M.2. Estadísticas: Fanático.	227
M.3. Estadísticas: Dragón.	228
M.4. Estadísticas: Nexo.	229
M.5. Estadísticas: Pilón.	230
M.6. Estadísticas: Portal.	230
M.7. Estadísticas: Centro cibernético.	231
M.8. Estadísticas: Asimilador.	232
M.9. Estadísticas: Larva.	234
M.10. Estadísticas: Huevo.	235
M.11. Estadísticas: Soberano.	236
M.12. Estadísticas: Zumbido.	237
M.13. Estadísticas: Zergling.	238
M.14. Estadísticas: Hidralisco.	238
M.15. Estadísticas: Criadero.	239
M.16. Estadísticas: Estanque de desove.	240
M.17. Estadísticas: Guarida de hidraliscos.	241
M.18. Estadísticas: Extractor.	242
M.19. Estadísticas: VCE.	243
M.20. Estadísticas: Marine.	244
M.21. Estadísticas: Murflama.	245
M.22. Estadísticas: Centro de operaciones.	246
M.23. Estadísticas: Almacén de suministros.	247
M.24. Estadísticas: Cuarteles.	248
M.25. Estadísticas: Academia.	249
M.26. Estadísticas: Refinería.	249

Capítulo 1

Introducción

En este capítulo se resumen la motivación y los objetivos para este proyecto.

1.1. Motivación

La motivación de este proyecto (*Armygeddon*) surge de los resultados del trabajo práctico realizado [1] en la asignatura de Videojuegos¹ dentro del plan de estudios² del Grado en Ingeniería Informática de la Universidad de Zaragoza. Dicho trabajo práctico consistía en realizar en grupos de cuatro personas un clon simplificado de un videojuego y añadirle características que usasen inteligencia artificial. Puesto que desde que se comenzó a impartir la asignatura nunca se había realizado un videojuego RTS, se escogió *StarCraft* por ser el más relevante en este subgénero³.

El resultado del trabajo no fue satisfactorio del todo, a pesar de haber sido galardonado como tercer mejor videojuego del curso dentro de la asignatura y haber obtenido una calificación de notable. Se trataba de un clon muy simplificado y cuya jugabilidad era mínima, como se detalla en el anexo K.

Habiendo visto así en primera persona la complejidad no explorada de *StarCraft* [9], y al existir interés en la materia (desarrollo de videojuegos), aparece la posibilidad de

¹<http://webdiis.unizar.es/asignaturas/videojuegos/> (último acceso: 10/5/2023).

²https://estudios.unizar.es/estudio/asignatura?anyo_academico=2022&asignatura_id=30262&estudio_id=20220148¢ro_id=110&plan_id_nk=439 (último acceso: 10/5/2023).

³Fue entre otros hitos, el videojuego de ordenador más vendido de 1998 [9]. Tanto es así, que existe una copia física del videojuego de 1998 en el Museo de Informática Histórica de la Universidad de Zaragoza <http://mih.unizar.es/> (último acceso: 10/5/2023). El 2 de mayo de 2022, durante la guerra de Ucrania, el asesor de la oficina presidencial ucraniana, Oleksii Arestovych, hizo una referencia al videojuego refiriéndose al ejército ruso como “una avalancha zerg”: <https://odessa-journal.com/arrestovich-zerg-rush-is-being-prepared-for-us-about-the-further-situation-at-the-front/> (último acceso: 22/5/2023)

crear un proyecto base similar y continuar evolucionándolo con características de otros videojuegos para satisfacer ambas ideas.

Pero para postular la propuesta de Trabajo de Fin de Grado es requisito sentar unas metas generales sobre qué se va a trabajar, así que se fijaron unos objetivos mínimos.

1.2. Objetivos

Una vez se construya un videojuego base con un diseño abierto y escalable, que permita al menos superar las limitaciones del videojuego original⁴, el proyecto se orienta por cuatro líneas que son a su vez sus características mínimas y sobre las que se diseña y añaden todas las demás:

1. Editor de escenarios. Se consideraba esencial para Armygeddon, pues al ampliar la profundidad del videojuego, también aumentan las posibles configuraciones y con el editor deja de limitarse al usuario a solo los escenarios incluidos.
2. Rediseño de las reglas de juego. Intentando conservar la esencia de StarCraft, pero que sea algo más que solo StarCraft.
3. Opcional: Modo multijugador en línea. Para permitir varios jugadores conectados desde distintos equipos a través de internet.
4. Distintos perfiles de jugadores artificiales con inteligencia artificial. La inteligencia artificial en este caso además de suplir la falta de jugadores humanos sirve para estudiar al propio videojuego y generar datos útiles para su desarrollo.

De esta manera se consigue:

- Abarcar el máximo número de áreas distintas y enfrentarse al mayor número de problemas diferentes dentro del desarrollo de videojuegos.
- Enfrentar un proyecto que va evolucionando sin un enunciado definido.
- Poder utilizar ingeniería inversa sobre un proyecto real y sofisticado⁵.
- Terminar el estudio sobre *StarCraft* [9], como una de las piezas clave en la historia del desarrollo de videojuegos.

⁴Por ejemplo, los escenarios estaban limitados a un tamaño máximo de 128x128 casillas, y el número máximo de unidades tenía un “límite” sorteable de 200: https://starcraft.fandom.com/wiki/Gameplay_of_StarCraft#Resources (último acceso: 18/7/2023).

⁵Desde el anuncio del equipo de desarrollo de StarCraft hasta su primer lanzamiento, *Blizzard* (la entidad desarrolladora de StarCraft) tardó tres años [9].

Capítulo 2

Diseño del videojuego

En base a los objetivos del proyecto, se destacan diferencias sustanciales en el enfoque del diseño para este nuevo videojuego respecto al trabajo anterior [1]:

- De intentar clonar solo el resultado, a intentar clonar *también cómo lo hace*. Durante el trabajo práctico solo se analizaba el videojuego como usuario, ahora se añade el uso de ingeniería inversa a la documentación externa y a los archivos del juego para intentar obtener un diseño similar al original y no solo en apariencia.
- De la completitud a *la diversidad de características*. Para intentar explotar al máximo el tiempo disponible se intenta no explorar del todo ninguna característica para enfrentarse al mayor número de problemas distintos posible. Por ejemplo, introducir solo parte de las unidades en vez de todas y, pasar a la siguiente característica.
- Del desarrollo en cascada al *desarrollo en espiral*. Ahora no existe un enunciado claro del proyecto con unas metas a cumplir, sino que este se va construyendo conforme avanza. Mientras que en el trabajo práctico había un enunciado, un reparto de tareas, y finalmente una integración entre todo el grupo para obtener un prototipo, ahora se requiere un diseño abierto y evolutivo para un prototipado constante.

2.1. Análisis de requisitos

El primer enunciado del proyecto era obtener un núcleo lógico similar a StarCraft, desde el cual continuar añadiéndole funcionalidades nuevas para conseguir crear algo distinto. La lista de requisitos básicos fue ampliándose hasta que se estimó que se podía jugar una partida completa en una simplificación de StarCraft.

A partir de ahí se siguieron las líneas objetivo de Armygeddon, con reuniones por lo general semanales en las que se iban trazando nuevas rondas de requisitos divididas en cada una de las principales funcionalidades del proyecto. Se trataban con rondas de requisitos pues es un diseño evolutivo y se buscaba responder a las necesidades que el videojuego fuese teniendo en las sucesivas versiones intentando siempre equilibrar jugabilidad, diversidad en las mecánicas, accesibilidad y experiencia del usuario utilizando el menor tiempo posible.

La lista completa de requisitos es muy extensa y se encuentra en el anexo A.

2.2. Arquitectura software

Al tratarse de un videojuego, hay que tratar la parte lógica con atención a la parte visual (la que lo convierte de juego a videojuego). Por eso se recurre a obedecer el diseño básico por norma general de un videojuego: el bucle de juego¹.

2.2.1. Bucle de juego

Un bucle de juego consiste en las acciones que de manera periódica tiene que hacer el programa para mantener una cierta tasa de imágenes mostradas al usuario, con el procesamiento lógico que conlleve, sin sufrir retrasos en ninguna de las dos partes. En este proyecto la parte lógica está vinculada a la gráfica ya que el flujo del tiempo afecta a las partidas y no puede avanzar un ciclo de partida sin haber completado su correspondiente procesamiento lógico. Por ejemplo, si una unidad tarda 10 ciclos en desplazarse de una casilla a otra, esos 10 ciclos tienen que durar lo mismo, además de ser un tiempo adecuado, para que el usuario lo perciba correctamente.

Al determinar una cantidad fija de imágenes por segundo que tiene que mostrar el programa, los fotogramas por segundo o *FPS* (*Frames Per Second*), aparecen dos problemas según la carga del procesador:

- El tiempo de procesamiento lógico y gráfico es *menor* al esperado para un ciclo. El usuario percibirá que el juego está acelerado, se soluciona añadiendo una espera variable que retrase el ciclo hasta el tiempo esperado.
- El tiempo de procesamiento lógico y gráfico es *mayor* al esperado por un ciclo. El usuario percibirá que el juego está ralentizado, se genera latencia (conocida como *FPS lag*).

Partiendo de esto, la estructura básica del bucle de juego consistiría en que mientras

¹https://en.wikipedia.org/wiki/Video_game_programming#Game_structure (último acceso: 26/5/2023).

se esté jugando se realicen las siguientes acciones (ver figura 2.1):

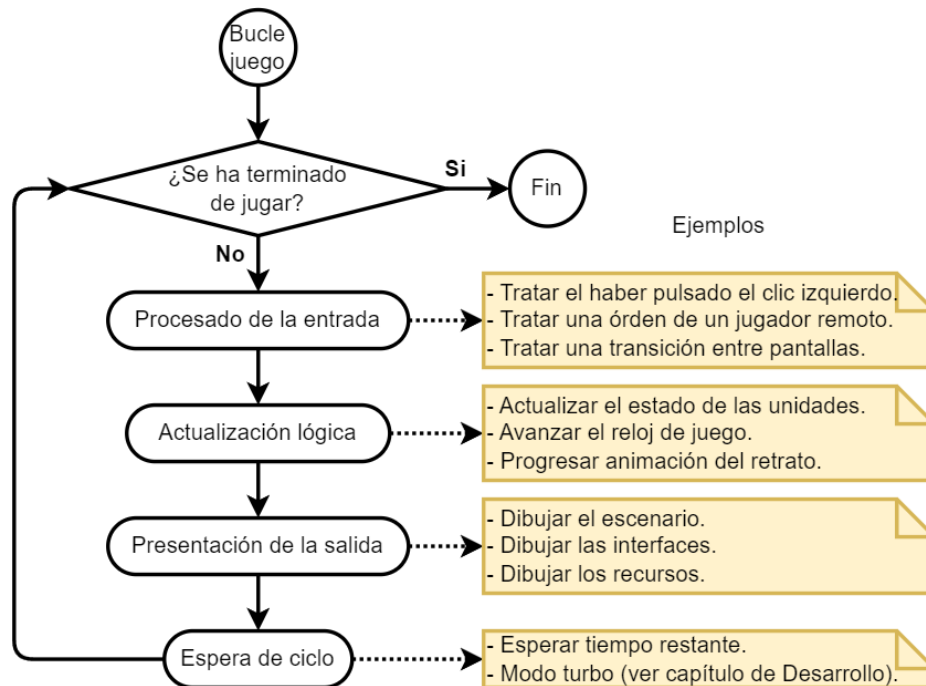


Figura 2.1: Esquema del bucle de juego de Armygeddon.

- **Procesado de la entrada.** Tratamiento de todas las órdenes del sistema (como órdenes de cierre o de redimensionado del programa), del usuario (a través del teclado y del ratón), o del propio programa si hay más de un jugador (órdenes de jugadores artificiales o de otros usuarios si se juega en línea).
- **Actualización lógica.** La aplicación de un ciclo lógico a todos los elementos de una partida. Esto es, avanzar en una mínima unidad temporal toda la lógica de Armygeddon (por ejemplo, avanzar mínimamente la posición de una unidad que se está moviendo).
- **Presentación de la salida.** Sustituir la imagen mostrada al usuario por pantalla por el resultado calculado en el último ciclo lógico.
- **Espera de ciclo.** Dicha espera tiene que, por ende, variar en cada ciclo en función de cuánto tiempo sobre hasta alcanzar el tiempo de ciclo estimado. Puede llegar a no haber espera si se da el caso de que el coste temporal de las otras acciones juntas (teniendo en cuenta el tiempo que pueda tardar el sistema por sí solo) supere al tiempo de ciclo, es decir, si hay latencia.

Como las funcionalidades del videojuego original StarCraft están asociadas y divididas en sus distintas pantallas, se conserva el concepto de *pantallas* como elemento clave para estructurar el diseño de Armygeddon.

2.2.2. Pantallas

La figura 2.2 es el esquema de navegación final por las distintas pantallas del interfaz de usuario.

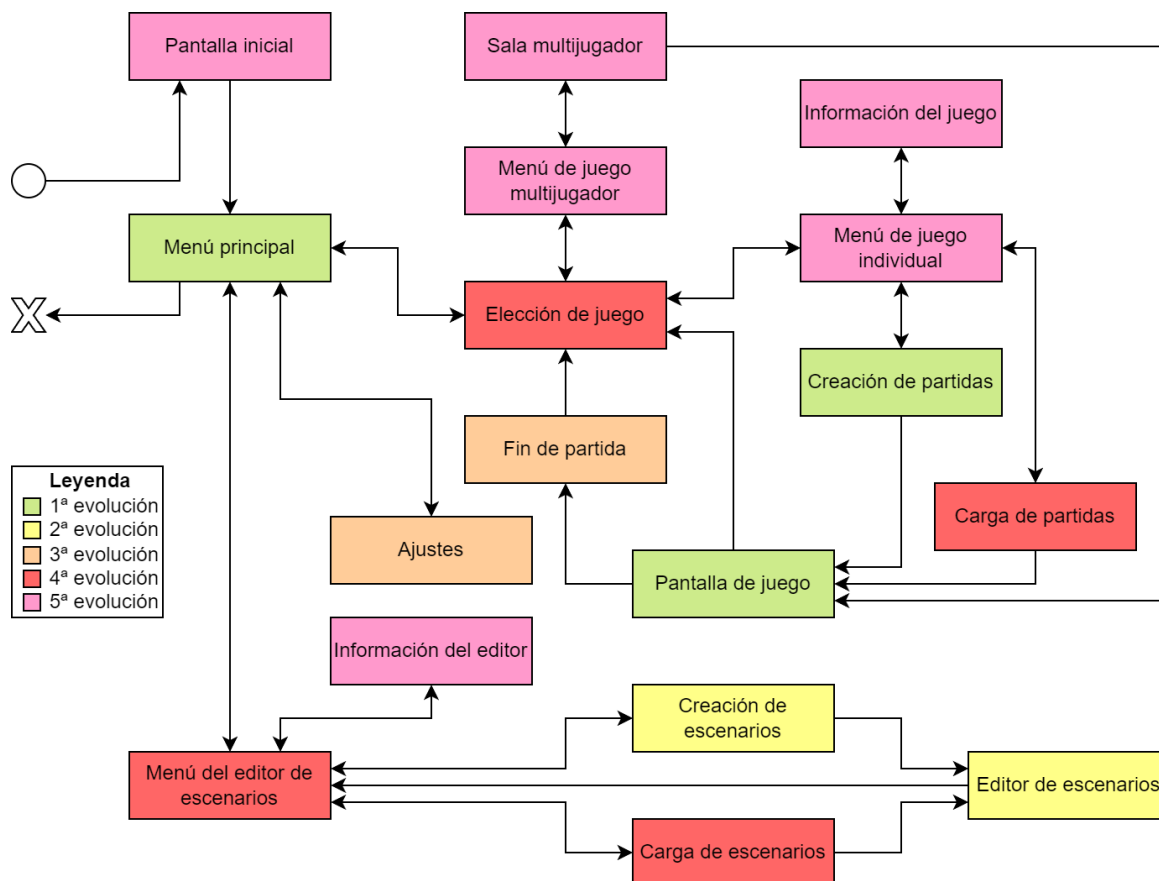


Figura 2.2: Última evolución del modelo de pantallas de Armygeddon.

Al ser un diseño evolutivo el modelo de pantallas ha tenido que ir evolucionando sobre la marcha, distinguiendo cinco etapas según sus adiciones:

1. Clon básico de StarCraft. Con lo mínimo para crear y jugar una partida.
2. El editor de escenarios. Permitiendo crear escenarios que luego jugar.
3. Personalización y fin de partida. Añade ajustes de usuario y los resultados de la partida.
4. Carga de partidas y de escenarios. Permitiendo continuar partidas guardadas y editar escenarios ya creados.
5. Red, información sobre el juego, multijugador en línea, pantalla inicial. Son aspectos opcionales importantes comunes en otros videojuegos.

Los cambios de una pantalla a otra se tratan en el procesado de la entrada, ya que de-

penden de la entrada del propio usuario en la mayoría de casos². La evolución completa de este modelo de pantallas está detallada en el anexo I.1.

2.3. Núcleo de juego

Tanto la pantalla de juego como la del editor de escenarios trabajan con el mismo núcleo de juego, el cual se ha diseñado de forma abierta para facilitar los distintos posibles cambios que fueran surgiendo. En esta subsección se trata la parte lógica del diseño, la parte gráfica se trata en los anexos I.4.1 y G.3.

Entendiendo el escenario como unidad básica y suficiente de información para jugar una partida, su información se puede clasificar en dos tipos:

- Información tangible: El conjunto de casillas que conforman el mapa y sus ocupantes, es decir, *el tablero*.
- Información intangible: El tiempo de partida, información sobre las facciones, recursos de los ejércitos, etc.

La dificultad del diseño reside entonces en la información tangible, a las casillas y a sus ocupantes; ya que la información intangible son solo datos que afectan a la tangible en ciertos momentos en los que se evalúa. Por ejemplo, poder permitirse una construcción, poder atacar a un ejército, terminar la partida si ha pasado suficiente tiempo, etc.

Lo primero es explicar por qué se opta por un diseño mixto del tablero basado en casillas con un sistema de posicionamiento por debajo:

- Las entidades de StarCraft son de muy variado tamaño, la mayoría con formas irregulares ovaladas lo cual es muy costoso de tratar (ver figura 2.3). Es más simple englobar a las entidades en casillas para solo tener en cuenta su altura y anchura en casillas.
- El movimiento se espera uniforme, no a saltos entre casilla y casilla, por lo que el desplazamiento tiene que trabajar con *posiciones* más pequeñas a las propias casillas.
- La *búsqueda de caminos* de las entidades móviles (ver anexo C.3) es más rápida si trabaja con menos elementos. Al englobar a las entidades en casillas solamente hay que evaluar si las casillas vecinas están ocupadas para encontrar caminos y

²Solo en algunos casos al acabar una partida deben ocurrir transiciones a la pantalla de fin de partida en la parte de la actualización lógica, la cual “tardaría” en mostrarse al usuario hasta el siguiente ciclo.

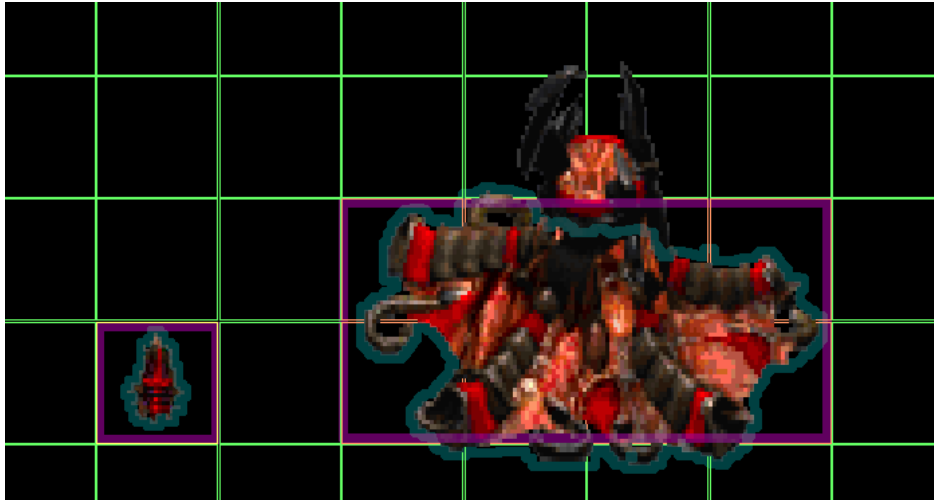


Figura 2.3: Tamaño real y en casillas de una larva y de un criadero.

no todas las posiciones.

- Hay entidades que requieren saber la distancia hasta otras entidades para el funcionamiento de las mecánicas en partida, y es mucho más sencillo de calcular trabajando con casillas que con posiciones. Resulta más fácil analizar secciones del mapa si solo evalúas ciertos puntos (casillas) y no todos (posiciones). Esto se detalla más en el anexo H.1.

Los elementos que ocupan las casillas, llamadas *entidades* u *ocupantes*, pueden ser de distintos tipos y se intentan distinguir por su funcionalidad en la partida antes que por sus diferencias (ver figura 2.4).

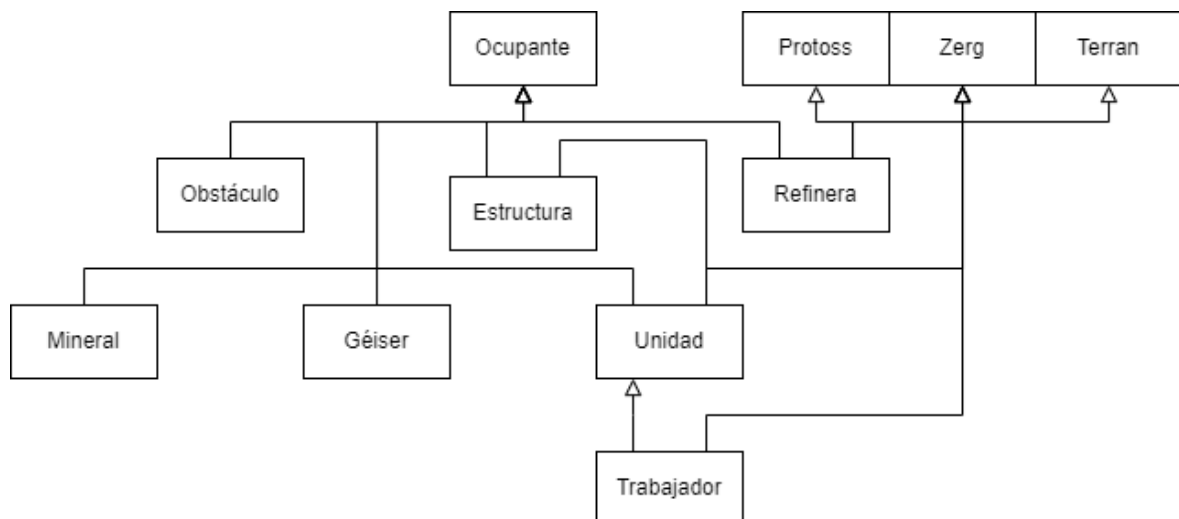


Figura 2.4: Esquema de los tipos de entidad de Armygeddon.

Se distinguen seis tipos de entidades de las cuales tres dependen de la raza a la que pertenezcan ya que tienen mecánicas únicas o ligeramente diferentes (ver tabla 2.1). El

único caso especial de los tipos de entidad es el de las unidades, puesto que un grupo de estas, los trabajadores, son notablemente diferentes al resto ya que tienen muchas más mecánicas.

Rasgos	Unidad	Estructura	Refinera	Obstáculo	Géiser	Mineral
Destruible Varía por raza Controlable Se puede crear	Si			No		
Móvil Puede atacar Puede trabajar	Sí	No		No		
Recibe recursos	No	Si	No	No		
Sustituye géiser	No	No	Si	No		
Inalterable	No	No	No	Si	No	
No explotable así Se sustituye	No	No	No	No	Si	No
Se agota	No	No	No	No	No	Si

Tabla 2.1: Algunas diferencias básicas entre los tipos de entidades

Existe un análisis más detallado de cada tipo de entidad en el anexo B. También se explican los tipos de entidades, así como de las reglas del juego en el anexo M.

2.4. Diseño del jugador artificial

El mayor uso de inteligencia artificial se aprecia en el jugador artificial³ y por ello se le dedica esta sección distinguida. Para las otras aplicaciones de inteligencia artificial tanto como para un mayor detalle del jugador artificial consultar el anexo C.

El diseño del jugador artificial pasa por un análisis preliminar del propio videojuego y de sus normas, observando que aun sin ser demasiadas acciones distintas las que se ofrecen al usuario, menos aún afectan al escenario y son relevantes para el jugador artificial. Por ejemplo, el jugador artificial no tiene por qué querer cambiar el volumen de la música o desplazar la cámara. No se considera interesante ya que no se reflejaría en los demás jugadores.

Sin embargo, las acciones que sí repercuten en el escenario son demasiado abiertas como para poder calcular las órdenes perfectas. Es por eso que el siguiente paso fue

³Como se indica en el manual (ver anexo M.6.3) un jugador artificial es un agente artificial diseñado para controlar un ejército de manera autónoma y automática.

estudiar al usuario y entender cómo se enfrenta a un RTS, siendo este un subgénero tan dinámico y abierto. Se concluye que el número de estrategias es incalculable, ya que también lo son las combinaciones de rivales, escenarios, etc. Aunque sí se pueden observar ciertos aspectos intrínsecos a las mecánicas del juego que tienen las estrategias:

- Agresividad. Cómo atacar, cuando, con cuántos recursos... Todo lo relacionado con las actitudes ofensivas.
- Exploración. El algoritmo de búsqueda utilizado, el momento de la búsqueda, etc.
- Economía. Toda la gestión de los recursos bien en otras materias o bien en inversión de la propia economía.
- Expansión. Cubrir mayor terreno, o menos y defenderlo mejor. Mejorar la infraestructura horizontal (primero más estructuras básicas) o verticalmente (primero estructuras avanzadas).
- Investigación. La velocidad a la que se intenta alcanzar algunos objetivos, o bien se mejoran las tropas, además de la prioridad entre tropas avanzadas o básicas.

Aun con todo, no solo la estrategia define las tácticas que sigue un usuario. También afectan el escenario, los rivales, la habilidad del usuario, etc. En definitiva, se entiende que en una partida pueden ocurrir tácticas muy contrarias entre sí, pues aunque se guíen por una estrategia, siguen estando condicionadas por la partida.

Es por esto que un usuario, por elaborada que sea su estrategia, juega atendiendo estímulos concretos, teniendo en cuenta sus recursos en cada momento (crystal, ejército, etc.) y, aunque vaya priorizando alguno antes que otro, tiene que atenderlos todos o podría perder la partida. Es con esta idea con la que se crea el jugador artificial y es que se entiende que un jugador, si fuese ordenado, podría jugar una partida repasando todas sus necesidades periódicamente y realizando acciones según convenga en cada aspecto del juego.

Se diseñan seis protocolos, grupos de las mencionadas acciones, que intentan cubrir las principales ideas que tiene un usuario mientras juega una partida (ver figura 2.5):

- Invadir. Utilizar las tropas para lograr los objetivos de la facción de manera ofensiva. Se crean secciones lógicas del tablero sobre las cuales, según su amenaza e interés, se gestionan escuadrones de tropas mediante invasiones.
- Defender. Consiste en preservar el estado del ejército para que todo siga funcionando. Esto es, movilizar tropas para protegerse mediante patrullas, llegando a

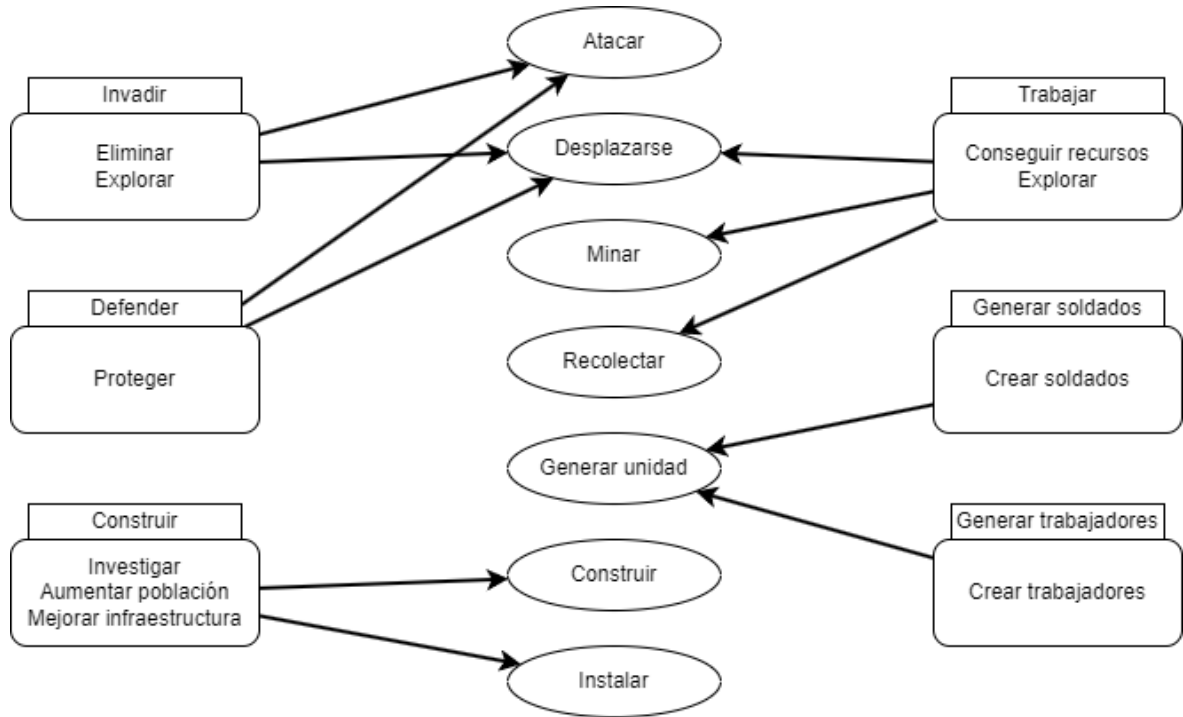


Figura 2.5: Relación de los protocolos del jugador artificial y las acciones del jugador humano que engloban junto con los tipos de órdenes usadas.

atacar a las posibles amenazas si es necesario.

- Construir. Comandar a los trabajadores para crear estructuras o instalar refineras⁴, según sean las necesidades.
- Trabajar. Gestionar las rutas de los trabajadores para obtener y explotar los recursos naturales, llegando a mandar unidades a explorar si es necesario.
- Generar soldados. Aumentar el poder militar, en función de las necesidades y de la situación de la partida.
- Generar trabajadores. Crear trabajadores para tener suficiente mano de obra y capacidad logística con los recursos.

2.5. Arquitectura del modo multijugador en línea

El modo multijugador en línea ha sido la característica más compleja, pues ha llegado a tener tres diseños distintos, y por eso tiene una sección distinguida.

En las tres versiones cada usuario asume el rol de servidor o de cliente en función

⁴Las refineras (de *refiners*) como se explica en el manual M.6.11, son las ‘estaciones refineras’ que los ejércitos instalan en los géiseres para obtener gas. La refinería (de *refinery*, ver manual M.9.8) es la versión terran de una refinería.

de si ha creado él la sala o no. Además, se usa por ello un controlador distinto para el usuario según si este actúa de servidor o de cliente.

2.5.1. Controladores cliente-servidor

El porqué se usa una arquitectura cliente-servidor responde a mantener la seguridad de que los eventos van a suceder correctamente y con el mismo orden en todo el sistema en su conjunto. Es decir, que las órdenes que ejercen todos los jugadores sobre el escenario se apliquen con el mismo orden para todos y sin condiciones de carrera, por lo que se ordenan todas las órdenes y se aprueban en su caso⁵ en un punto común que es el servidor.

Al tratarse Armygeddon de un videojuego en estrategia en tiempo real, el flujo de información va a ser constante, tanto del servidor hacia los clientes para actualizarles el estado del escenario⁶, como de los clientes al servidor para enviar las órdenes que procese el jugador remoto. Por esta razón se utilizan conexiones TCP ya que hay que garantizar la llegada de la información.

Como se aprecia en la figura 2.6 se destacan unos parámetros pero hay más. Se mencionan estos parámetros porque son los que hay que introducir o modificar en el momento y que condicionan la entrada del cliente en la sala; el nombre y el avatar del jugador en línea se deciden antes, en ajustes. El proceso en detalle sería:

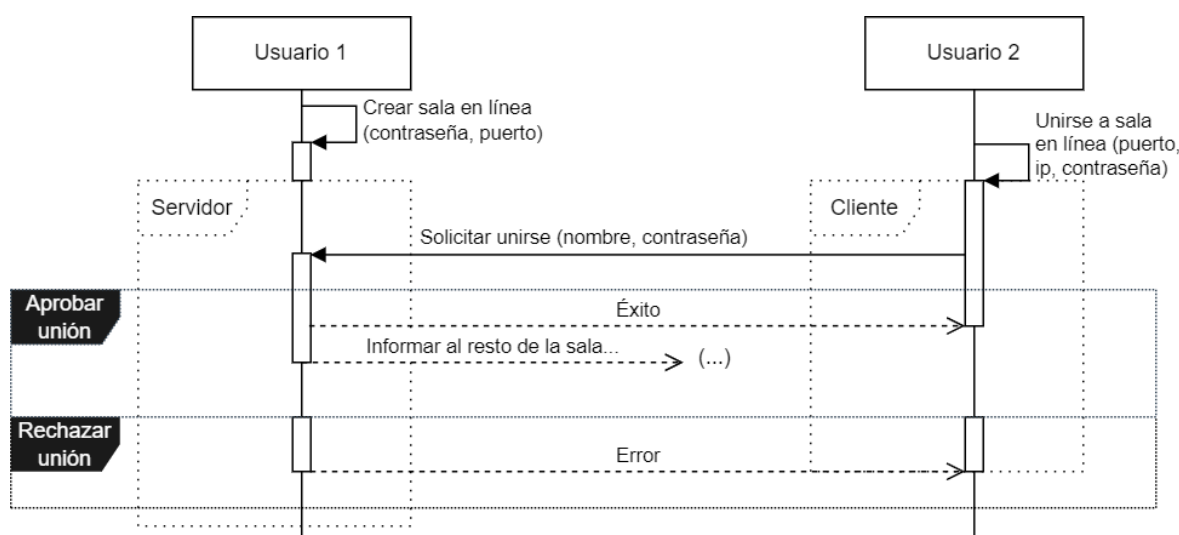


Figura 2.6: Creación de sala en línea y emparejamiento.

⁵Las órdenes de los jugadores pueden cancelar las de otros, p.e., si en el mismo instante dos jugadores intentan ocupar una casilla, solo se tiene que permitir a uno realizar la acción. Es por eso que es necesario un único punto de evaluación de las órdenes, o podrían crearse distinciones entre los escenarios de cada jugador.

⁶A razón de 30 veces por segundo ya que es la tasa de actualización gráfica (y lógica) de Armygeddon.

1. Un usuario crea una sala en línea, con un puerto para escuchar la conexión y una contraseña opcional para restringir el acceso a la sala.
2. Cuando otro usuario intenta unirse a la sala creada introduce puerto, dirección IP y contraseña (en el caso de haberla), creando una conexión TCP con el servidor.
3. El servidor responde a la solicitud de unión del cliente:
 - a) Si el servidor acepta, responde con éxito al cliente e informa a los demás en caso de haberlos de la nueva unión a la sala.
 - b) En caso de ser rechazada la solicitud se corta la conexión informando del error que haya sucedido (nombre en uso, contraseña incorrecta, etc.⁷).

A partir de la creación de la sala o de una solicitud de unión se crean los respectivos controladores de la conexión, con sus dos variantes, la del servidor y la del cliente. Estos controladores se encargan de enviar y recibir mensajes a la par que mantienen la conexión a lo largo de la actividad de los usuarios en las distintas pantallas que requieran interacciones en línea.

Una vez creada la sala, seleccionado el escenario a jugar, configurado la partida y puesto todo en consenso con todos los usuarios, se empieza la partida. Es destacable lo que ocurre con el jugador artificial en las partidas en línea:

1. Si el jugador servidor de la partida está designado bien como jugador artificial (espectador) o bien como jugador híbrido, el manejo de las órdenes del jugador artificial es tratado como en las partidas individuales: En la actualización se calculan las órdenes en la máquina del servidor y se juntan con las de los demás jugadores antes de ser procesadas todas de vez y por orden de llegada.
2. Si hay ejércitos no asignados a ningún usuario, y están configurados por ende como jugadores artificiales, estos son manejados de la misma manera que en el caso anterior: Se calculan las órdenes en la máquina del servidor y se procesan de la misma manera.
3. Si hay usuarios clientes, es decir, jugadores remotos, que actúan como jugador artificial/espectador o como jugador híbrido, las órdenes de la inteligencia artificial se calculan en esas máquinas remotas durante el procesamiento de los mensajes en línea de esos clientes. Esto es así para descargar a la máquina servidor de procesamiento y para testar la funcionalidad en línea. Este procesamiento del jugador artificial ocurre en el tratamiento de los mensajes del cliente ya que el cliente tiene que recibir del servidor cuándo ocurre un cambio de ciclo en la

⁷Para consultar la prevención de fallos realizada en Armygeddon al completo ver sección 3.5

partida y eso es lo que sirve de señal para las actuaciones del jugador artificial.

Es en partida cuando comienza la verdadera problemática del modo en línea que es la sincronización en conjunto de los jugadores y, lo que ha suscitado 3 diseños distintos del paso de mensajes.

2.5.2. Primer diseño: Actualizaciones gráficas completas

En vista del relativamente pequeño tamaño de los escenarios⁸ se comenzó con un diseño que tratase de compartir el escenario al completo, o al menos una versión simplificada, entre los jugadores de una partida.

Como se procesaba todo desde el servidor, por evitar las ya mencionadas condiciones de carrera, no era necesario que los clientes tuvieran todos los datos lógicos sobre el escenario, por lo que solo había que pasar los gráficos. Es decir, cosas como qué posición tienen las entidades, de qué tipo son, hacia dónde miran o qué hacen, y no cosas como qué camino están siguiendo, cuánto les queda para realizar el próximo ataque, o qué orden tienen que realizar.

Desde los clientes solo tenían que enviarse las órdenes que realizaran los jugadores. Pero al no disponer de ningún mecanismo de identificación de las entidades se añadió la orden invisible de selección, creando en el servidor la problemática de tener que almacenar e ir actualizando qué está seleccionando cada jugador para en momentos posteriores realizar las órdenes oportunas. Es decir, no se clasificaban las órdenes como órdenes del juego, sino como órdenes del usuario (ver tabla 2.2).

Estímulo	Orden enviada
Clic izquierdo	Se ha seleccionado lo que hay en X casilla. Se ha pulsado el botón de acción X.
Movimiento de ratón haciendo clic izquierdo	Se ha seleccionado lo que hay de la casilla X a la casilla Y.
Clic derecho	Se ha ordenado accionar la casilla X.
Tecla de botón de acción	Se ha pulsado el botón de acción X.
Botón de salir	Se ha deseleccionado.

Tabla 2.2: Posibles órdenes enviadas en función del estímulo según el primer diseño en línea.

De esta forma, se calculaba si generaba una orden sobre el escenario en la máquina del servidor, y solo se indicaba desde el cliente lo que se hacía, ya que al disponer solo

⁸Thermopylae ocupa ~80 KB, HelmAbyss ~150 KB, y los OpenStarCraft y Starcraft ~370 KB. El ancho de banda de las conexiones ADSL, que empezó a ser predominante en el año 2000, ya dispone de hasta 20 MBps de bajada y 1 MBps de subida. <https://en.wikipedia.org/wiki/ADSL> (último acceso: 2/7/2023).

de una perspectiva gráfica desde el cliente no era posible hacer más.

El diseño funcionaba correctamente en escenarios pequeños de 20x20 con 400 casillas, pero no para el tamaño máximo de 128x128, es decir, el objetivo de 16384 casillas. A pesar de haber reducido el número de caracteres enviados al tener en cuenta sólo la parte gráfica, el serializar los datos del escenario y crear toda la información de partida aumenta el volumen de los datos. Ocupaba un mensaje de un ciclo más que el propio fichero del escenario⁹.

Esto motivó la creación de un nuevo diseño para el modo multijugador.

2.5.3. Segundo diseño: Actualizaciones gráficas parciales

Al ser el problema enviar los datos del servidor a los clientes y no al revés, se intentó sacrificar el tráfico de datos hacia el servidor para añadir una nueva orden invisible (como lo había sido el seleccionar en el primer diseño): Actualizar la posición de la cámara. De esta manera, al saber el servidor qué está visualizando cada cliente, no era necesario enviar la información gráfica de todo el escenario, sino solo la necesaria que cada jugador estaba observando. La reducción fue notoria, a razón de hasta 55 veces menor el volumen de datos en los escenarios más grandes¹⁰.

Pero en ese entonces aún se conservaba el primer diseño de los jugadores artificiales y, al aparecer como requisito la elaboración de un nuevo diseño de jugador artificial, también apareció como precaución el requisito de permitir participar como un jugador artificial desde un cliente. Se decidió así para descargar de tanto uso de recursos computacionales al servidor, ya que todos los jugadores artificiales se procesaban en dicha máquina.

Al funcionar el jugador artificial con la lógica del tablero, y no con la parte gráfica como los usuarios¹¹, era necesario entonces que los clientes tuvieran acceso también a la parte lógica, surgiendo así la necesidad de un tercer y último diseño.

2.5.4. Tercer diseño: Simulación perfecta

A pesar de los esfuerzos por integrar de alguna manera todos los datos lógicos del escenario en los envíos entre ciclos, y que cada cliente borrara la información que tuviera del escenario en el ciclo anterior, y se recreara uno nuevo cada ciclo aunque solo fuera

⁹StarCraft, un escenario con tamaño máximo de 128x128 se serializaba con ~660.000 caracteres (~660 KB), siendo que el fichero en el formato de escenarios solo pesa ~373 KB.

¹⁰Por ejemplo, StarCraft pasó de pesar el envío del escenario de ~660 KB a ~12 KB.

¹¹Esto no quiere decir que el jugador artificial tenga acceso a más información que un usuario. Es solo que al usuario se le mostraba la información seleccionada en cada ciclo, mientras que el jugador artificial al actuar en un solo ciclo cada varios, requiere toda la información en el mismo momento.

en los ciclos en los que actuase el jugador artificial, el coste de recursos computacionales era inasumible.

Desde un principio se había establecido que para evitar discrepancias entre los escenarios, las órdenes de los jugadores tenían que pasar por un único punto (el servidor) para ordenarse y evitar condiciones de carrera. Pero eso no era motivo para centralizar toda la lógica de las partidas a solo el servidor: todos los clientes disponen del mismo programa, así que puede obtenerse la misma partida en todos los clientes si todos procesan las mismas órdenes en el mismo ciclo y bajo el mismo orden. Así solo sería necesario enviar el escenario al principio, y todas las órdenes que ejecuta cada jugador durante la partida, para que todas las máquinas dispusieran del mismo escenario en todo momento.

Solo existía un problema: la generación de números aleatorios. Para garantizar cosas como que en todos los clientes aparezca una unidad en la misma casilla aleatoria alrededor de una estructura no servía este modelo, ya que la semilla para el algoritmo generador de números aleatorios era la hora interna de la máquina. Entonces si ofrecer la semilla basada en la hora ocasiona inconsistencias porque no existe un reloj común y las máquinas no son igualmente veloces, se procede a utilizar una semilla común para todas las máquinas de la partida.

Es por eso que este diseño se denomina *simulación perfecta*, porque se basa en que todas las máquinas de los jugadores van a actuar como autómatas perfectos donde las entradas (la semilla para la generación de números aleatorios y las distintas órdenes que se van dando cada ciclo), al ser idénticas, y el programa (transiciones del autómata), el mismo en todas las máquinas, se va a poder simular la misma partida en todas las máquinas sin discrepancias.

Capítulo 3

Desarrollo del videojuego

Al igual que en el capítulo anterior, al ser Armygeddon un proyecto evolutivo han existido distintas versiones que han ido perfeccionándose conforme progresaba el trabajo y, por brevedad, se va a comentar solo la implementación final. Puede consultarse la evolución completa de la implementación en el registro de cambios (ver anexo L).

3.1. Interacción con el usuario

Armygeddon es de ámbito espacial y futurista, lo cual permite flexibilidad a la hora de representarlo, haciendo que se puedan usar una estrella o un planeta como botones (ver figura 3.1). No solo eso, al no estar muy estandarizadas las interfaces en la fecha que salió el videojuego en el que se basa (StarCraft), no al menos en los RTS ya que es de los primeros¹, hay *elementos con reacción a la entrada al usuario* que tienen que estar visual y auditivamente señalizados como tal. Por esas razones conceptuales y por otras como fomentar la inmersión o la accesibilidad, se implementan los gráficos con especial distinción a los “elementos reactivos”.

Los elementos reactivos son de distintos tipos (de escritura, botones de radio, marcas, deslizantes, etc.) pero tienen comportamientos comunes:

- Son activables. Ocasionan un evento respuesta a su activación. El refuerzo auditivo y visual de la activación depende de la naturaleza del elemento y de su evento respuesta.
- Pueden estar bloqueados. Impidiendo su activación según las circunstancias. Intentar activar un elemento reactivo bloqueado también produce una respuesta

¹El término de videojuego de estrategia en tiempo real es de la década de los 90 https://en.wikipedia.org/wiki/Real-time_strategy (último acceso: 6/7/2023).

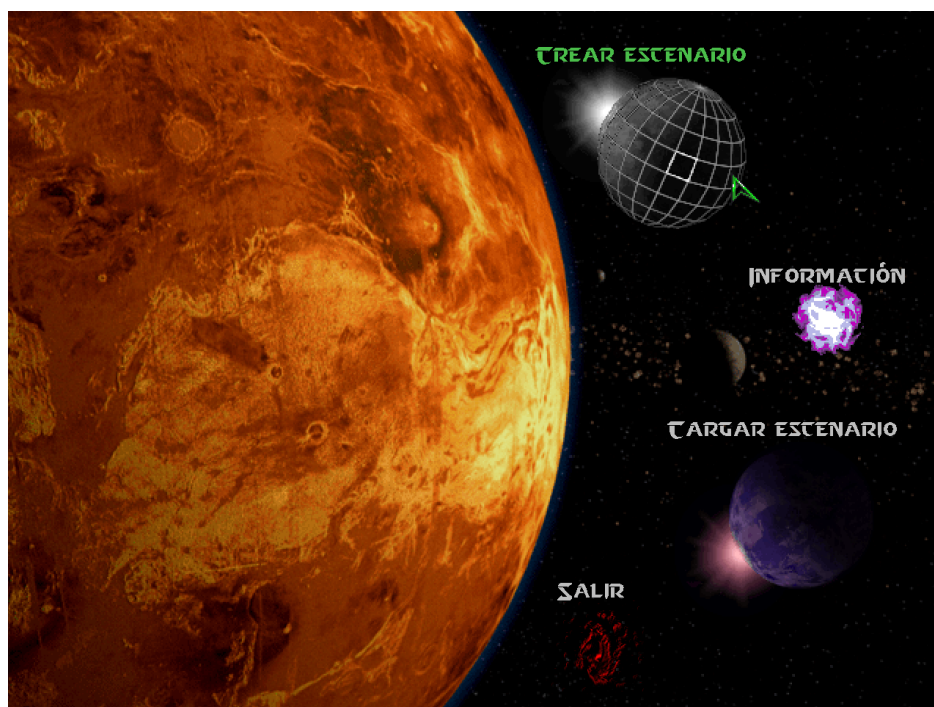


Figura 3.1: Menú principal del editor de escenarios de Armygeddon.

audiovisual según el caso.

- Son seleccionables. Por señalización al usuario de que se trata de un elemento reactivo, indicando la naturaleza del evento que va a ocasionar con el refuerzo auditivo y visual además de la propia forma canónica del elemento. Por ejemplo, los botones de radio aparecen como círculos alineados o agrupados, las marcas como rectángulos con fondo uniforme, etc.



Figura 3.2: Posibles estados de un elemento reactivo simple: botón 'Guardar' de ajustes.

En el anexo I.2 se muestra más información sobre los elementos reactivos, sus respuestas o sobre la posición de dichos elementos. En el anexo I.3 se hablan de los estándares seguidos para el uso de los colores en Armygeddon y, en el anexo I.4 se habla finalmente de la obtención de los gráficos, la modificación y hasta la creación de algunos para el

proyecto, además de los efectos y música utilizados.

3.2. Jugar una partida

Se define una *partida* como el transcurso del tiempo sobre las entidades de un escenario, las cuales varían según las órdenes que los jugadores ejercen sobre el ejército que cada uno controla. Bien hasta que una facción de ejércitos logra sus objetivos, o bien hasta que todas las facciones desaparezcan. Dichas órdenes son la única forma de interacción de los jugadores con el *tablero del escenario* el cual está conformado por casillas, que a su vez son agrupaciones de posiciones.

Una vez explicadas las normas en el manual (ver anexo M), y el diseño en el capítulo anterior (ver sección 2.3), ahora se trata cómo funciona internamente el tablero de una partida desde el punto de vista de los ocupantes de sus casillas.

Al inicio de la partida se instancian todas las entidades en función de la configuración del escenario y el modo de partida escogido, ya sea creando partida desde fichero de escenario o cargándola desde guardado de partida. A partir de ahí, en cada ciclo de actualización de la partida se calcula la parte lógica y la gráfica (la imagen que va a mostrar cada entidad).

Para referirse a las entidades a actualizar, cada entidad tiene un identificador único. Esto sirve para facilitar las operativas multijugador ya que no se pueden señalar regiones de memoria de una máquina externa y, con los identificadores se entiende en red a qué entidades se refiere cada jugador.

3.2.1. Dibujo del tablero

Del tablero del escenario solo se dibuja lo que está viendo el usuario siempre que tenga visibilidad de dicha zona (ver figura 3.3). Para saber qué casillas se consideran para dibujar se toma la posición actual de la cámara y se calcula cuáles visualiza el usuario con el zum actual. Esta zona de dibujo, algo mayor que la zona que se muestra al usuario por si algún efecto o sombra sobresale a sus casillas ocupadas, es la que se tiene en cuenta también a la hora de reproducir sonidos. Si la entidad que genera el sonido está dentro de la zona, entonces al usuario que está enfocando dicha zona se le reproduce dicho sonido.

Una vez delimitada la zona a tratar, se sigue un orden que permita dibujar los ocupantes en el orden correcto (de izquierda a derecha y, de arriba a abajo) distinguiendo varias capas en el dibujo (ver figura 3.4):

1. Se dibuja el suelo. Las baldosas declaradas, las que no están declaradas (ver



Figura 3.3: Parte del tablero de una partida sin y con cuadrícula activa.

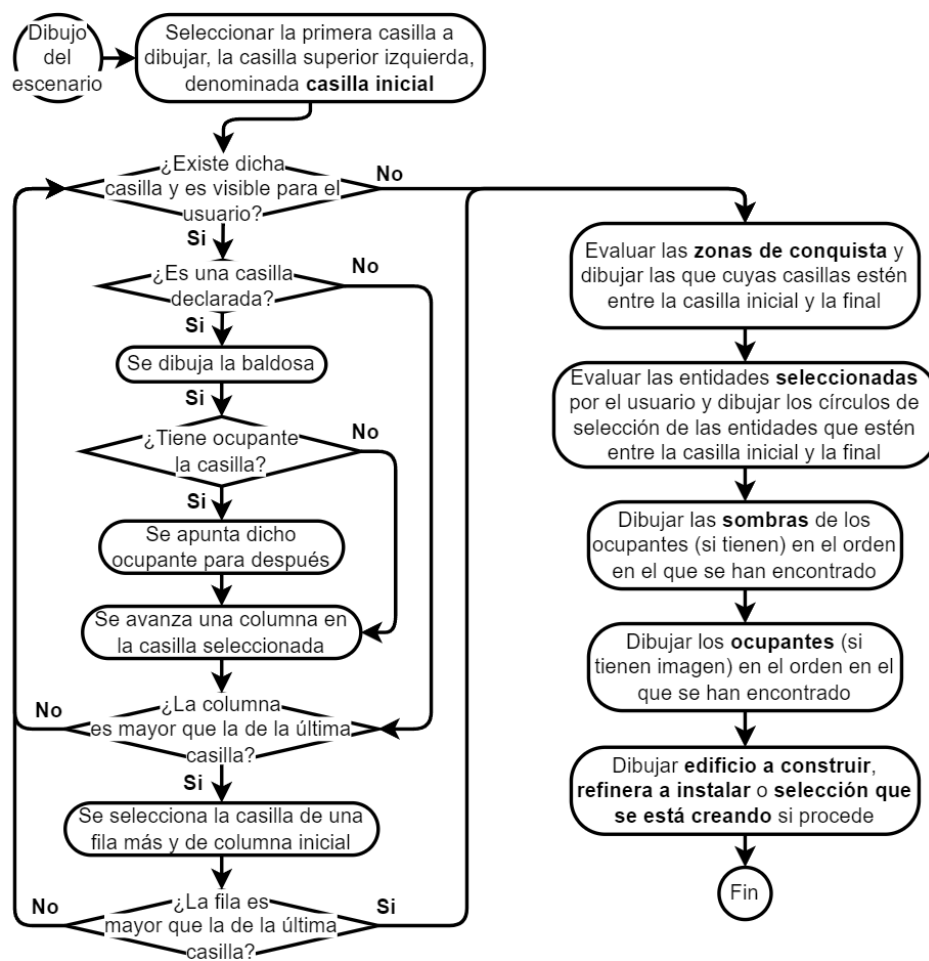


Figura 3.4: Diagrama de flujo del dibujo del escenario.

manual M.10.2) se pueden dejar en fondo negro.

2. Se dibujan las zonas de conquista. Al ser zonas lógicas, indicadores, se dibujan pegadas al suelo como si fueran una alfombra o una iluminación.
3. Se dibujan los círculos de selección. Van a rodear al ocupante y se quiere que el ocupante tape parte del círculo.
4. Se dibujan las sombras. En este caso se prefiere que si bien el círculo de selección es algo lógico que solo ve el usuario que selecciona, aun así se trate los círculos como algo tangible sobre los que se refleja la sombra.
5. Se dibujan los ocupantes. Dibujar las entidades en sí, también las acciones que estén realizando como por ejemplo, la explosión que ocasione un ataque enemigo.
6. Se dibuja la edición y selección del usuario. Si bien está construyendo un edificio o instalando una refinería, que se muestre dónde va a construirla por encima de todo, ya que tiene que verse claramente si la ubicación que está apuntando es apta o no. También se dibuja en su caso el rectángulo lógico de la selección del usuario mientras la crea.

3.2.2. Velocidad de la partida

Armygeddon ejecuta un ciclo lógico por cada ciclo gráfico, y los ciclos gráficos tienen una frecuencia de 30, es decir, 30 imágenes por segundo y por ende, 30 avances de la lógica del escenario. Como se ha de poder pausar y acelerar una partida de distintas maneras, lo que se ha procedido es lo siguiente sobre el bucle del juego (ver figura 3.5):

1. Modo pausa. Se omite la parte de cálculo lógico, manteniendo el funcionamiento del resto del bucle de juego.
2. Modos de aceleración. Se distinguen dos tipos, aceleraciones fijas y aceleración adaptable:
 - a) Aceleración fija. Son los modos x2 y x4, que intentan duplicar o cuadruplicar la velocidad de la partida. La implementación se ha basado en realizar dos o cuatro ciclos lógicos por cada ciclo gráfico. Esto irremediamente puede generar latencia de la misma manera que se genera en un ciclo normal (a modo de velocidad normal: x1) si no se llega a completar a tiempo el cálculo lógico y el dibujado de la salida.
 - b) Aceleración adaptable. Es para el modo de velocidad máxima (mostrado en

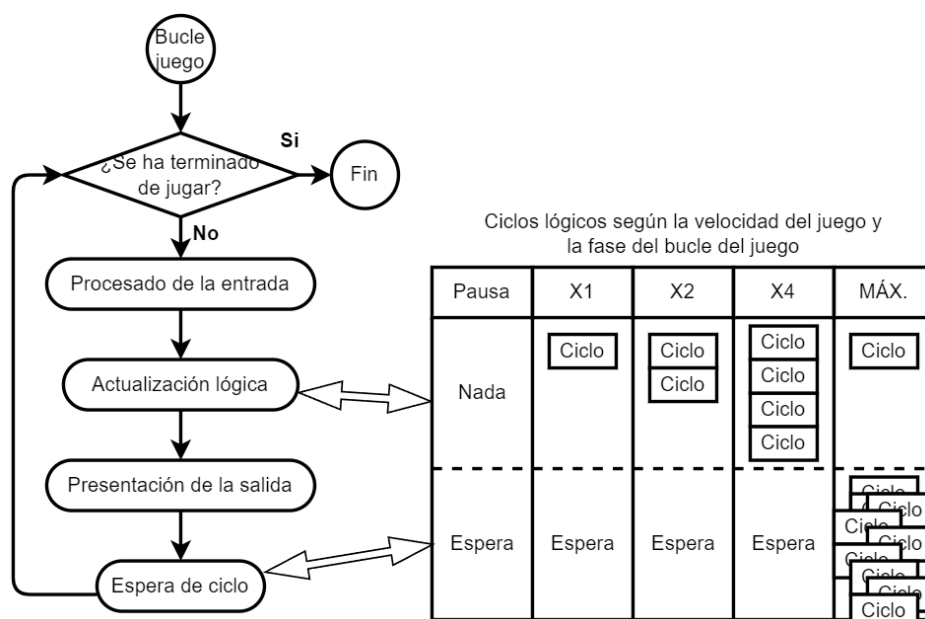


Figura 3.5: Distintos bucles de juego en función de la velocidad.

la interfaz como “MÁX”), el modo turbo. Como se ha dicho, los modos de aceleración fija pueden a ralentizar el programa si no se llega a completar a tiempo los ciclos lógicos porque van antes de la parte de presentación de la salida. Mientras tanto, la aceleración adaptable, se implementa como la ejecución de todos los ciclos extra posibles en el tiempo de espera del bucle hasta que se exceda, sin límite. Esta aceleración, puede acabar la partida entera en un solo ciclo realizando todos los ciclos lógicos necesarios, como solamente llegar a calcular uno (el de la parte *actualización lógica*) si hay latencia de base y no queda tiempo para la *espera de ciclo*.

3.2.3. Entidades y las casillas

El tipo de entidad que más trabaja con las posiciones es la unidad, ya que aunque una unidad solo ocupe una casilla por vez, se tiene que reflejar el desplazamiento entre estas gradualmente. Es por esto que las posiciones están asociadas al número de píxeles, 32 por 32 cada baldosa (casilla), ya que son la expresión mínima que se va a reflejar al usuario².

Una unidad busca siempre colocarse en la posición más centrada dentro de una casilla. Cuando va a ocurrir el paso de una unidad de una casilla a otra colindante³ el proceso

²O al menos lo era hasta que se introdujo el aumento de zum por encima de la escala natural 1:1 en la resolución original de 640x480. Desde entonces se permite ver píxeles de imágenes ocupando más de un píxel de la pantalla.

³Se puede pasar de una casilla a otra en las ocho direcciones cardinales e intercardinales. Aun si las direcciones de una unidad son 16 y pueden orientarse en todas ellas para atacar u otras acciones, en el sistema de movimiento solo se usan esas ocho.

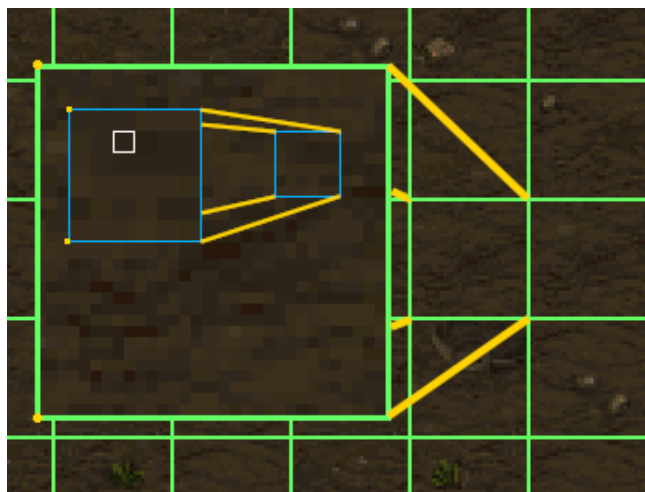


Figura 3.6: Posiciones dentro de cada casilla.

era el siguiente:

1. Se reserva la casilla objetivo.
2. Se libera la casilla actualmente ocupada por la unidad.
3. Cada ciclo se desplaza una cantidad de píxeles/posiciones independientemente de la dirección⁴ equivalente a la velocidad de la unidad hasta llegar al centro de la casilla objetivo.
4. Al llegar al centro de otra casilla puede que no haya recorrido todos los píxeles que podía en ese ciclo, en ese caso, y de no haber terminado el camino, se desplazará hacia la próxima casilla los píxeles restantes tras repetir los 2 primeros pasos.

Este proceso condiciona las actualizaciones de las unidades, de forma que entre casillas lo único que se hace es actualizar el movimiento, y cuando en un ciclo lógico se alcanza el centro de una casilla es cuando se valora todo lo demás en función de su estado.

Las unidades son más complicadas que el resto de las entidades, teniendo distintos comportamientos que tener en cuenta dentro de cada estado (ver figuras 3.7 y 3.8). Por ejemplo, actualizar una unidad que está recolectando tiene tres tratamientos principales diferentes en función de si está yendo hacia la refinería, recolectando gas ya en la refinería, o yendo a una estructura a devolver el gas. Las principales diferencias serían que mientras está dejando el gas evalúa constantemente si entra en contacto con una estructura, mientras está en la refinería si la refinería explota, y mientras está yendo hacia la refinería que si está en contacto con esta y si está disponible.

⁴De forma que un desplazamiento en diagonal dura lo mismo que en una dirección cardinal.

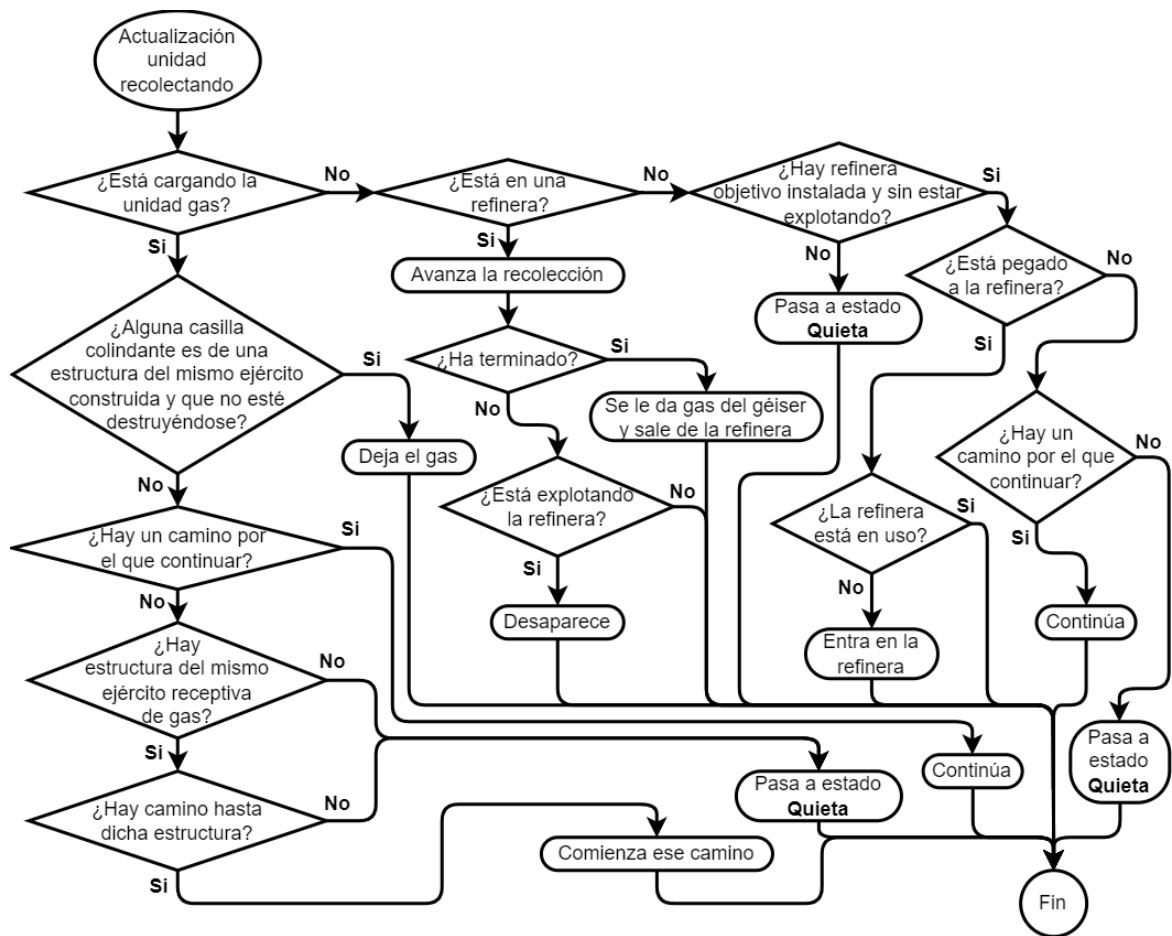


Figura 3.7: Diagrama de flujo de la actualización de estado de una unidad recolectando.

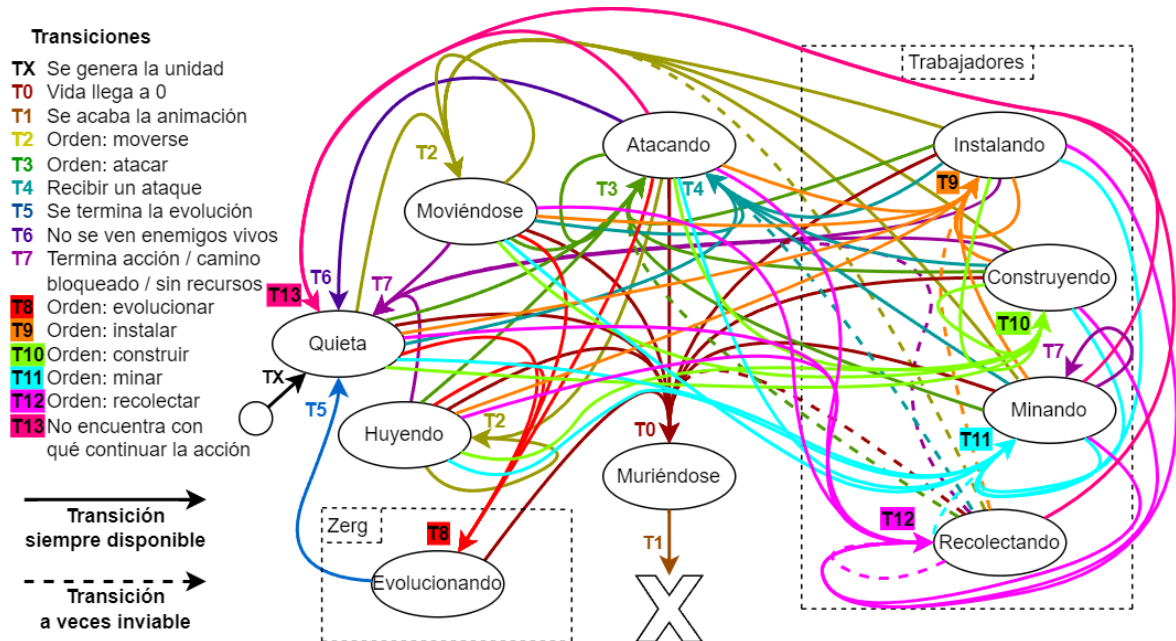


Figura 3.8: Esquema de las transiciones entre estados de las unidades.

Es al final de cada actualización de una unidad cuando se tratan los datos internos como la regeneración de salud o escudo, o la imagen que se va a mostrar en cada ciclo.

Para ver más detalles sobre las entidades consultar el anexo B.

3.3. Editor de escenarios

En el editor de escenarios se presenta la cuadrícula completa de casillas que tiene el escenario que se está editando con ajustes para que actúe como un visor personalizable. De esta forma, se permite editar directamente las entidades que aparecen en cada casilla nada más comenzar una partida y visualizarlas de distintas maneras. El formato de los escenarios sobre cómo se almacenan los escenarios está detallado en el anexo F.3.

El *dibujado* de la matriz de casillas o tablero funciona de la siguiente manera:

1. Se dibuja el suelo si está visible en los ajustes de visibilidad.
2. Se dibuja la cuadrícula si está activa.
3. Se evalúa los ajustes de visibilidad para discernir qué tipo de entidades se dibujan y cuales no.
4. De las entidades que se dibujan solo se dibuja una imagen distintiva, en el caso de las unidades una en la que salgan mirando al frente.
5. Finalmente se dibuja la entidad que se está editando marcando qué casillas ocupa y la imagen correspondiente con cierta transparencia y con los colores modificados para que se vea mejor.

La *edición* funciona con la casilla que está apuntando el cursor. Con entidades con tamaño mayor a una casilla se toma la casilla de la esquina superior izquierda como la que se va a colocar donde se está apuntando con el cursor.

Las *herramientas de selección* también trabajan con las casillas seleccionadas, pero teniendo en cuenta los ajustes de visibilidad. Cuando se pega el contenido del portapapeles en un espacio mayor al que ocupa el propio contenido lo que se hace es extender la operación como se aprecia en la figura 3.9.

Las funcionalidades del editor se detallan en el manual (anexo M.10).

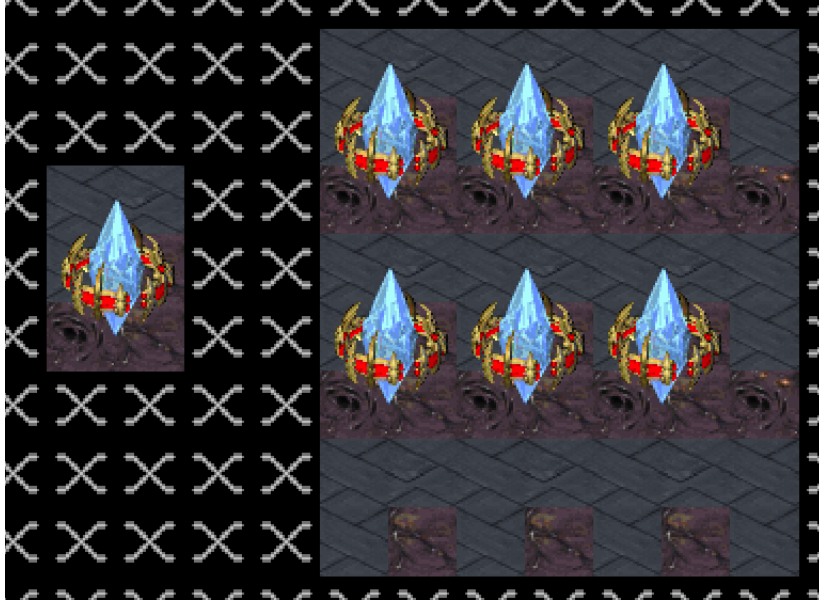


Figura 3.9: Ejemplo de pegar una selección de 2×3 en un espacio de 7×8 .

3.4. Jugador artificial

Tras definir el diseño del jugador artificial y para obtener distintos perfiles de jugadores artificiales como se indicaba en los requisitos, se implementan una serie de parámetros que definen la estrategia a seguir.

3.4.1. Agresividad

En un análisis de cómo un usuario interacciona con Armygeddon, se distinguen varios aspectos comunes respecto a las estrategias en partida, siendo el más importante y general la agresividad. Por eso se define como guía para la toma de decisiones del jugador artificial un parámetro que representa precisamente su *agresividad*.

La agresividad del jugador artificial entonces definirá las tácticas que realiza en cada uno de sus protocolos. A su vez, la agresividad puede aumentar o disminuir durante cada protocolo según el cómo le esté yendo la partida al jugador artificial en respuesta a unos *estímulos* que se ejemplifican más adelante en la sección 3.4.4.

Existen otros parámetros que también condicionan al jugador artificial y son los que se les da a escoger al usuario. Principalmente son la *velocidad de respuesta* y el *comportamiento*, pero hay más (ver anexo C.1.10).

3.4.2. Velocidad de respuesta

El jugador artificial en un ciclo de partida⁵ escoge un protocolo, realiza una serie de tácticas asociadas a ese protocolo a través de órdenes que se envían al escenario como un usuario, y luego se queda a la espera una determinada cantidad de ciclos de partida hasta que le vuelva a “tocar” realizar acciones en otro ciclo.

La *velocidad de respuesta* es un parámetro configurable por el usuario que define dichas cantidades de ciclos de partida que esperan los jugadores artificiales entre cada actualización. Dichas cantidades no son fijas, varían levemente para que los jugadores artificiales parezcan más diferenciados y fomentar así la inmersión del usuario (ver tabla 3.1).

Velocidad de respuesta	Factor	Horquilla en ciclos	Horquilla en segundos
Lenta	x4	[60, 180]	[2, 6]
Algo rápida	x3	[45, 135]	[1.5, 4.5]
Rápida	x2	[30, 90]	[1, 3]
Muy rápida	x1	[15, 45]	[0.5, 1.5]

Tabla 3.1: Posibles lapsos temporales entre actualizaciones del jugador artificial según su velocidad de respuesta.

Ver más en detalle la velocidad de respuesta en el anexo C.1.11.

3.4.3. Comportamiento

El *comportamiento* del jugador artificial es otro parámetro que afecta a su agresividad de manera indirecta.

Como se ha mencionado, cuando el jugador artificial recibe un estímulo de agresividad, este puede ser positivo (más agresividad) o negativo (menos agresividad). Lo que hace el comportamiento es afectar a la intensidad con la que interpreta estos estímulos. Un jugador artificial con un comportamiento muy agresivo tendrá más en cuenta los estímulos positivos de agresividad y menos en cuenta los estímulos negativos, y viceversa; con un comportamiento defensivo interpretará los estímulos positivos con menor intensidad y tendrá mucho más en cuenta los estímulos negativos.

Es decir, el comportamiento del jugador artificial define su estrategia, pero no impide que haya tácticas contrarias a la estrategia. Más detalle en el anexo C.1.12.

⁵Los *ciclos de partida* son actualizaciones que ocurren mientras transcurre la partida, no confundir con los *ciclos del programa* que siempre están en marcha aun con la partida en pausa.

3.4.4. Protocolos

Los protocolos, aunque varían en algunos aspectos en función de la raza del ejército, se implementan de la siguiente manera y, con los siguientes estímulos⁶ sobre la agresividad:

- **Invadir.** Se analiza la peligrosidad y el interés por sectores lógicos⁷ del escenario para gestionar invasiones. Las invasiones son grupos de soldados que tienen un sector objetivo y que disponen de dos tipos de retiradas: una retirada individual si la unidad tiene poca vida y otra a nivel de invasión si la agresividad es demasiado baja.
 - Aumenta la agresividad: Conquistar sectores con las invasiones. Crear invasiones.
 - Disminuye la agresividad: Sufrir bajas en las invasiones. Tener invasiones activas.
- **Defender.** Las tropas que no estén invadiendo se denominan *patrulladores*. En este protocolo, se comprueban poco a poco los distintos tipos de entidades propias del jugador artificial. Si estas entidades tienen enemigos cercanos o, están en combate, se van gestionando los patrulladores para ir a defender. Al final los patrulladores que no estén en combate siguen patrullando en busca de amenazas.
 - Aumenta la agresividad: No encontrar amenazas cercanas.
 - Disminuye la agresividad: Quedarse sin patrulladores. Encontrar amenazas en entidades importantes.
- **Construir.** Se continúan las obras de estructuras sin terminar si es el caso. Además se consulta el estado de la infraestructura para ver qué puede ser necesario construir. Ejemplo: Se alcanza el 90 % de la población máxima. → Es necesaria una ampliación.
 - Aumenta la agresividad: No necesitar ningún tipo de construcción.
 - Disminuye la agresividad: No poder permitirse una construcción necesaria. Necesitar una estructura base.
- **Trabajar.** Se separan los trabajadores según si están quietos, minando o recolectando. Hasta un umbral se mandan a recolectar siempre que haya suficientes minando, todos los demás se envían a minar y en algunos casos a explorar. Si no hay suficientes recursos para explotar se envían más trabajadores a explorar.

⁶Solo se listan los ejemplos de estímulos más relevantes.

⁷Es decir, transparentes al usuario.

- Aumenta la agresividad: Enviar trabajadores a explorar.
- Disminuye la agresividad: No tener suficientes trabajadores para el mínimo de recolección.
- Generar soldados. En función de cuántos soldados básicos se pueden comprar en ese momento contando solo la cantidad de recursos de la que se dispone, se generan soldados gastando una parte de los recursos. El número depende de varios factores, y dicho número no será mayor al necesario de poner a trabajar a todos los generadores de soldados disponibles.
 - Aumenta la agresividad: Utilizar todos los generadores de soldados.
 - Disminuye la agresividad: No permitirse generar soldados.
- Generar trabajadores. Se genera un trabajador en función del número de trabajadores inactivos.
 - Aumenta la agresividad: Nada.
 - Disminuye la agresividad: No poder permitirse generar un trabajador.

Para un análisis completo, y los esquemas de los algoritmos de los protocolos, ver el anexo C.1.13.

3.4.5. Variantes según el tipo de objetivo

Además, hay que destacar que los jugadores artificiales alteran algunos de sus protocolos comentados si el objetivo es uno de los siguientes tres:

1. Supervivencia. Se reorganizan las prioridades de los patrulladores y se añade una rutina que los devuelve a una estructura aliada si se alejan demasiado. Se añaden muchos refuerzos negativos de la agresividad.
2. Conquista. La invasión estudia una nueva relación en el interés de los sectores teniendo en cuenta la distancia al sector a conquistar priorizando movimientos hacia la zona de conquista antes que a combates.
3. Investigación. La generación de soldados avanzados se controla para tener reservas de gas para las investigaciones. Se priorizan las *maravillas* (anexo M.6.17) y sus mejoras (las investigaciones) en las construcciones.

Estas variaciones se explican en detalle en el anexo C.1.14.

3.4.6. Bolsa de protocolos

Finalmente para ganar naturalidad haciendo que el uso de estos protocolos sea imprevisible, se utiliza una “bolsa de protocolos balanceada” de la siguiente forma:

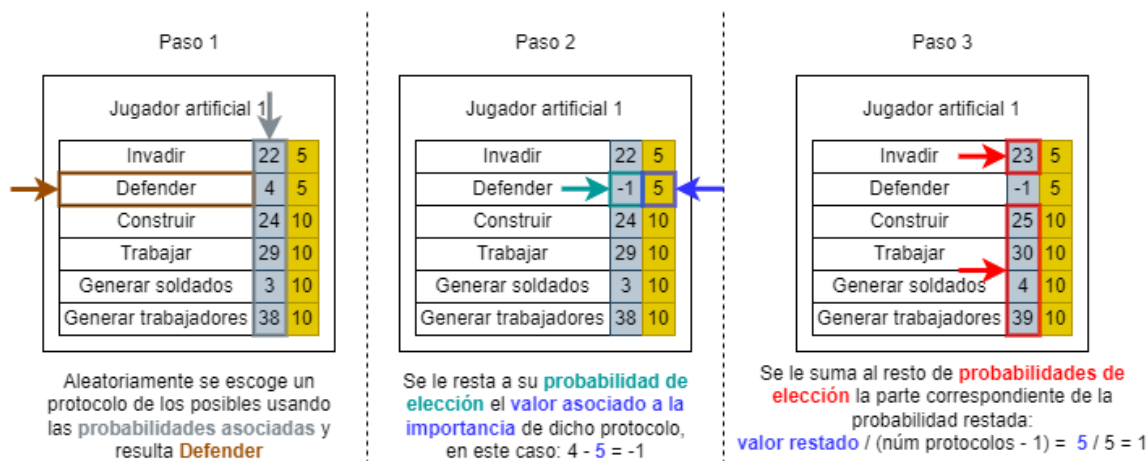


Figura 3.10: Elección y reequilibrio de los protocolos del jugador artificial en una actualización.

1. Cada vez que el jugador artificial vaya a realizar un protocolo, este se escoge al azar entre uno de los seis protocolos teniendo en cuenta una serie de probabilidades.
2. Se reduce la probabilidad de que en posteriores elecciones vuelva a elegirse dicho protocolo en función de cuán importante sea el protocolo restándole parte de dicha probabilidad.
3. El valor restado se reparte entre todas las probabilidades de los demás protocolos para que la probabilidad no desaparezca.
4. Se realiza el protocolo escogido y en su caso se procesan órdenes sobre el escenario.

De esta manera se garantiza que no habrá inanición y se dificulta saber qué protocolo va a escoger el jugador artificial en el futuro. El proceso se detalla más en el anexo C.1.15.

3.5. Modo multijugador en línea

En el menú multijugador se escoge un controlador, según se actúe como cliente o como servidor, como ya se ha mencionado en el diseño. La decisión es en función de si el usuario se conecta a una sala multijugador o si este la crea. Según el tipo de controlador, este funcionará de una manera o de otra (ver figura 3.11):

- En el caso del controlador del cliente se crean dos hilos, uno para atender los

mensajes del servidor (recibidor de servidor) y, otro para ir manejando la cola de mensajes a enviar (controlador de envíos).

- Si se trata de un controlador del servidor, entonces se crea un hilo para escuchar las conexiones de los clientes (*listener*, escuchador) y, otros dos hilos por cada conexión de cliente que aparezca para recibir (recibidor de cliente) y enviar mensajes a este (controlador de envíos).

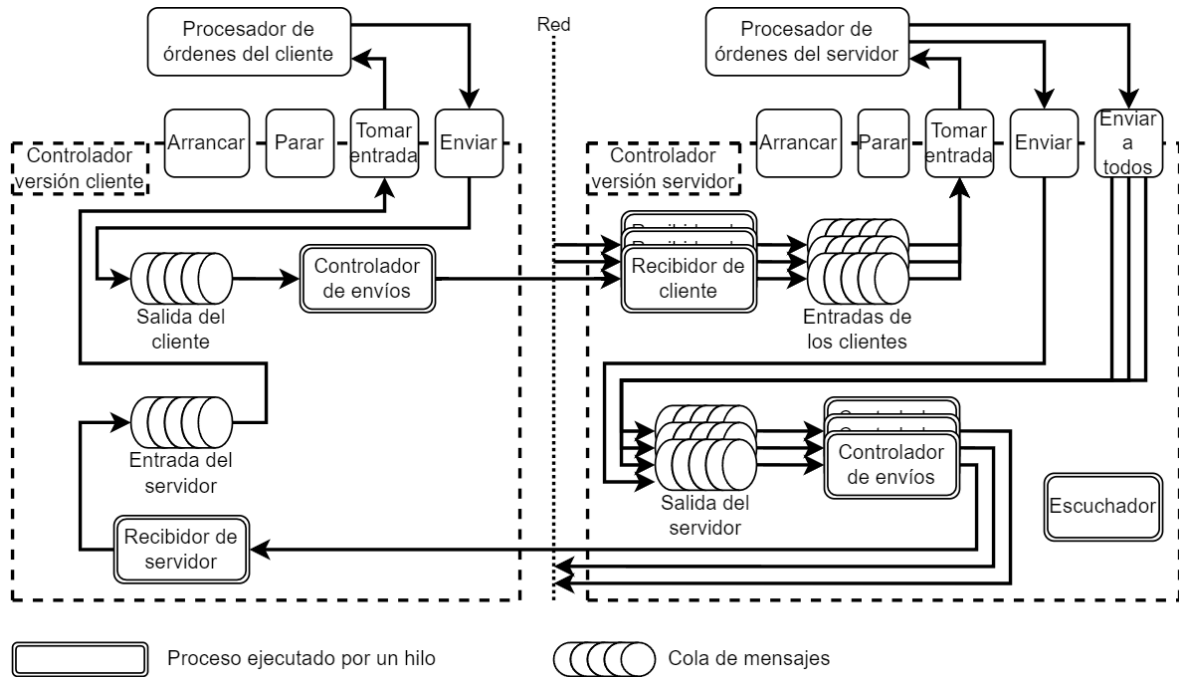


Figura 3.11: Rutas de mensajes y estructura de los controladores de conexiones.

Los mensajes se mandan en fragmentos de tamaño de 10 kB como máximo, que se unen antes de traducirlos. Cada mensaje tiene una estructura compuesta de tres partes:

1. Código de mensaje. Cuatro caracteres numéricos⁸ que representan el tipo de mensaje que se está mandando. Algunos ejemplos: Echar a un cliente, servidor caído, solicitar unirse, ciclo de partida, mensaje del servidor...
2. Cuerpo de datos. La información adicional si la hay para el tipo de mensaje en concreto que se envía. Por ejemplo, puede enviarse un mensaje desde un cliente a un servidor una solicitud de unión a la sala y en este campo añadir la contraseña, el avatar a usar, el nombre... El cuerpo de datos se envía serializado siguiendo el formato mencionado en el anexo F.4.
3. Carácter distintivo. Se utiliza el carácter nulo '\0'.

⁸Se usan cuatro caracteres numéricos ya que se traduce directamente desde un tipo enumerado por legibilidad, que se prioriza en este caso a ahorrar en consumo de red.

A pesar de no ser el objetivo del proyecto, se ha implementado cierta tolerancia a fallos más allá de que sea mínimamente jugable el modo en línea:

- Se da un aviso si la contraseña introducida para la sala es incorrecta, además de impedir la unión del jugador a la sala.
- Se da un aviso si el nombre en línea está ya en uso en esa sala, además de impedir la unión del jugador a la sala.
- No se consigue conectar, se muestra un aviso.
- Servidor se desconecta durante la sala multijugador, sacando a todos de la sala.
- Cliente se desconecta durante la sala multijugador, actualizando a todos los demás miembros de la sala que dicho cliente ha salido.

3.6. Pruebas

Las principales pruebas de un videojuego son las de depuración, no solo para saber que no hay errores, sino también para comprobar que lo que funciona lo hace de la manera esperada o, incluso si la manera esperada es la adecuada.

En Armygeddon se ha utilizado *Discord*⁹ para crear un foro común (ver figura 3.12) donde se compartía contenido sobre los errores para corregirlos si procede para la siguiente versión. Cada reporte de un fallo daba lugar a un hilo donde se obtenía la información necesaria para identificar y, en su caso, replicar el error y solventarlo.

Aunque cerca de la mitad del esfuerzo temporal dedicado al proyecto (ver sección 3.7.3) ha sido la depuración de las versiones, también se han realizado pruebas con otras finalidades haciendo uso de los escenarios de ejemplo (más detalle en el anexo D):

- Pruebas de balance. Intentan medir el equilibrio de las razas y de los escenarios oficiales. Este se contrasta en escenarios que aíslan lo suficiente a los jugadores como para mitigar las diferencias entre razas que generan resultados similares, al contrario que para escenarios menos limitados que se aumentan las diferencias (ver ejemplo, tabla 3.2).
- Pruebas del jugador artificial. Se estudian los máximos y los mínimos del comportamiento del jugador artificial y su repercusión sobre las tácticas tomadas y el resultado de la partida. Lo más relevante es ver como a pesar de los cambios

⁹Discord es una red social de mensajería instantánea y de llamadas utilizada en este caso por su capacidad de crear foros llamados servidores en los que es muy fácil aportar imágenes y vídeos, ordenar las publicaciones, o incluso compartir la pantalla. <https://discord.com/> (último acceso: 11/8/2023).

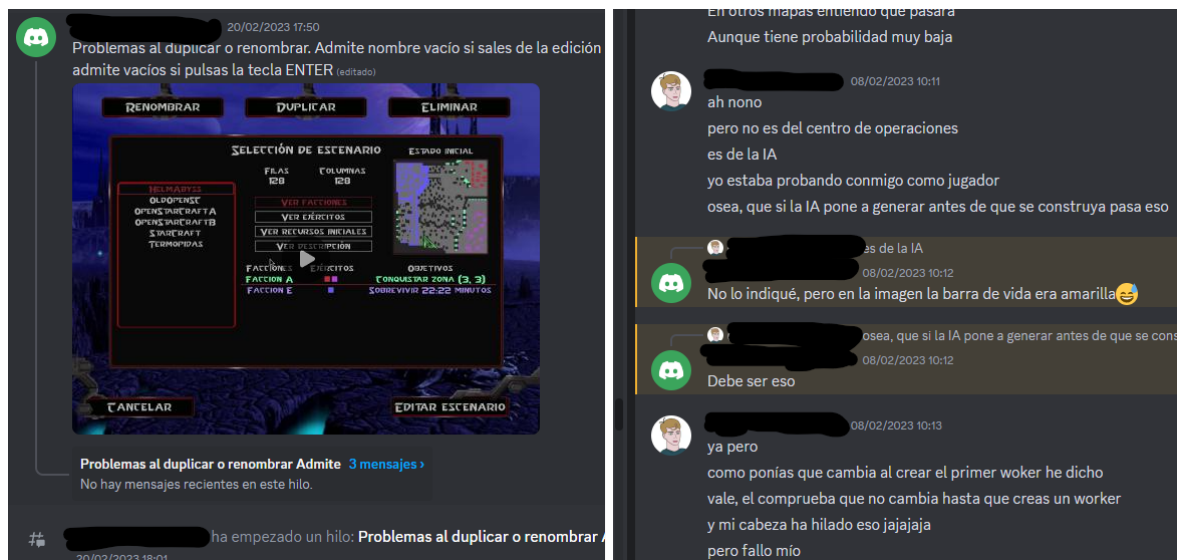


Figura 3.12: Captura del foro para la depuración de Armygeddon: reportes (izquierda) e hilos (derecha).

Facción	Raza	Victorias
Ejército rojo	Zerg	7
Ejército naranja	Terran	6
Ejército amarillo	Protoss	7
Ejército verde	Terran	7
Ejército azul	Terran	4
Ejército rosa	Zerg	10
Ejército violeta	Zerg	8
Ejército blanco	Protoss	5

Tabla 3.2: Resultado de las pruebas de balance sobre StarCraft.

sobre la estrategia se permite tomar tácticas que en principio pueden parecer contrarias y aun así obtener más victorias (ver tabla 3.3).

Facción	Victorias agresividad máxima	Victorias agresividad mínima
Defensores	10	2
Invasores	7	10

Tabla 3.3: Resultado de las pruebas

- Pruebas de requisitos. Se miden los recursos necesarios para poder jugar a Armygeddon ejecutando una simulación del escenario StarCraft (el considerado más complejo de los de ejemplo, ver anexo D.1) con el truco de recursos infinitos activado¹⁰. La partida de 45 minutos se ejecutó en menos de 10 segundos, casi *300 veces* más deprisa que en tiempo de partida y con la carga gráfica ocupando menos del 15 % de los recursos usados de procesador. Se concluye que en cualquier ordenador de los últimos años se puede ejecutar Armygeddon sin tener en consideración requisitos mínimos.
- Pruebas de conectividad multijugador. Aquí se comprueba que el protocolo de paso de mensajes utilizado es suficiente para ejecutar una partida con el máximo de jugadores en el escenario más complicado. Para ello se ejecuta una partida al completo con ocho jugadores híbridos en remoto al doble de velocidad para asegurar que a una velocidad normal seguiría sin sobrecargarse ningún cliente. El uso de ancho de banda por Armygeddon en este caso (en el servidor: 720 kBps subida, 14 kBps bajada) resultó muy inferior al ancho banda medio en España (135,73 mBps), al menos 100 veces menor.

Más información sobre las pruebas realizadas en el anexo E.

3.7. Conclusiones

En esta sección se concluyen los siguientes aspectos de Armygeddon:

1. Resultado del proyecto (ver sección 3.7.1).
2. Principales dificultades encontradas (ver sección 3.7.2).
3. Medición de esfuerzos realizados (ver sección 3.7.3).
4. Posibles expansiones no implementadas (ver sección 3.7.4).

¹⁰Al no limitar los recursos la generación de unidades aumenta de manera destacada, exigiendo más recursos computacionales.

Además, se hace una valoración personal del proyecto (ver sección 3.7.5).

3.7.1. Resultado del proyecto

El proyecto a lo largo de sus 65 versiones ha ido ampliando sus características, mejorando las ya existentes y, corrigiendo errores que iban surgiendo. Estas versiones están divididas en tres grupos:

1. Recreación de StarCraft para obtener un núcleo básico para el proyecto. Lo necesario para conseguir jugar una partida mínima en StarCraft. Un total de 12 versiones.
2. Evolución de ese núcleo básico a Armygeddon. Todos los objetivos marcados para el proyecto. Un total de 40 versiones.
3. Expansiones opcionales sobre Armygeddon. Contenido adicional como el modo multijugador en línea, o un mayor perfilado de los agentes artificiales. Un total de 13 versiones.

En cuanto a datos cuantificables sobre Armygeddon:

- El proyecto se compone de más de 25.000 líneas de código repartidas en más de 70 clases.
- En el foro de depuración se han superado los 300 hilos de reportes de errores.
- El registro de cambios ocupa unas 1500 líneas de características añadidas, mejoradas o arregladas.

Para más información sobre el trabajo realizado, consultar el registro de versiones en el anexo L.

3.7.2. Principales dificultades

De todas las problemáticas encontradas durante el proyecto, se destacan solo dos:

- Sistemas de distancias. En muchas ocasiones dentro de una partida se requiere qué entidades hay alrededor de una entidad dentro de una cierta distancia. Sin embargo, el proyecto se fundamenta en el tablero del escenario y, este está hecho de casillas. Entonces aparecen dos maneras de medir la distancia (ver figura 3.13):
 - Distancia real (circular). Menos eficiente pero más realista. Es la distancia que percibe el usuario como correcta pues es un círculo aproximado de

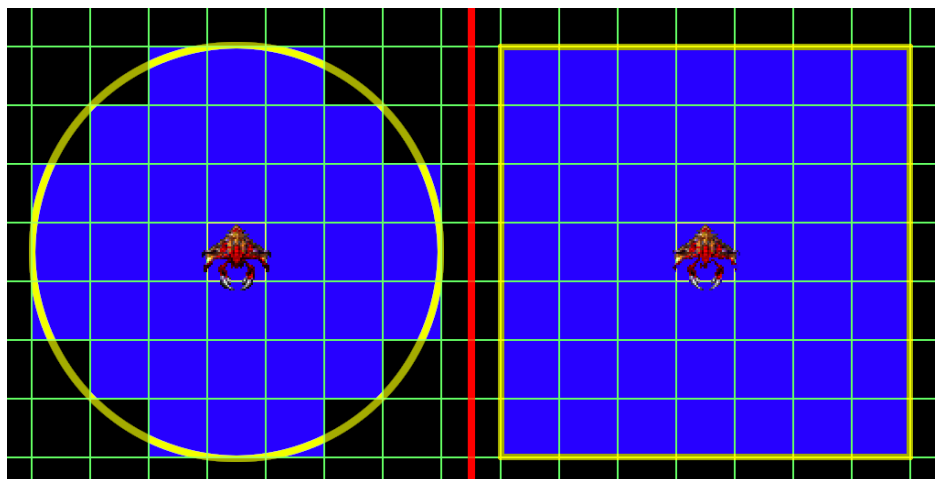


Figura 3.13: Casillas afectadas en 3,5 casillas en distancia real (izquierda) y en distancia optimizada (derecha).

casillas.

- Distancia optimizada (cuadrada). Más eficiente pero inexacta, pues simplemente se refiere a una sección cuadrada del tablero que tenga el doble de la distancia buscada en cada lado.
- Gestión y acceso de la memoria. Debido al formato en el que se obtuvieron los gráficos del juego, que es en columnas muy largas con todas las posibles variaciones de cada entidad, existían problemas durante la carga debido a su longitud en según qué ordenadores debido a una biblioteca utilizada (SDL [7]). La solución pasó por alterar ciertas estructuras de datos de la biblioteca con operativas obsoletas para poder dividir los gráficos durante la carga. En general, el lenguaje utilizado (C++) permite una gestión de la memoria tan precisa que ha frenado el progreso tanto dentro de la biblioteca utilizada como fuera.

Estas dificultades se detallan más en el anexo H.

3.7.3. Medición de esfuerzos

Se clasifica el esfuerzo temporal invertido en Armygeddon durante este año¹¹ en cuatro tareas fundamentales (ver la figura 3.14):

1. Reuniones. Son las horas dedicadas a reunirse con el director del proyecto para acordar la temática del trabajo, los requisitos, analizar el progreso, discutir soluciones y organizar tanto la propuesta como la memoria. Durante los meses de octubre a diciembre de 2022 consistía en reuniones semanales en las que se buscaba contenido que añadir y se decidía el rumbo del proyecto.

¹¹De mediados de agosto de 2022 hasta mediados de agosto de 2023.

Cronograma

Medición del esfuerzo por tareas y por meses

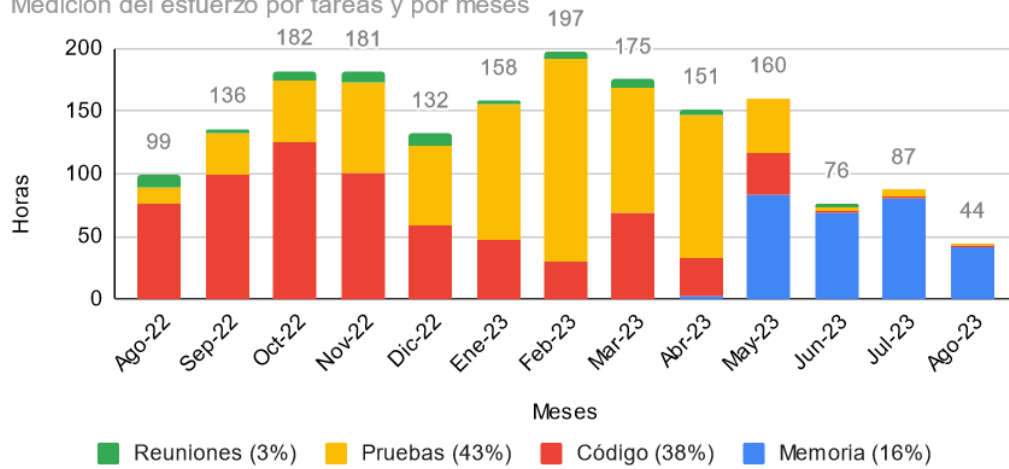


Figura 3.14: Medición de esfuerzos en el proyecto según el mes y el tipo de tarea.

2. Pruebas. Depurar el proyecto para encontrar errores nuevos o para simular errores ya encontrados por otras personas y también el corregirlos. Además se tiene en cuenta la investigación sobre el propio proyecto para diseñar su progreso. Supone la tarea más costosa al ser un videojuego y las posibilidades del usuario tan abiertas.
3. Código. Añadir características a Armygeddon. Desde el diseño en pseudocódigo, hasta la implementación final. También se incluye en esta tarea el arreglo de los errores reportados, de los que se conoce exactamente su ubicación y, la investigación sobre el videojuego original y sobre las tecnologías utilizadas [5] [3].
4. Memoria. Redacción de este documento, incluyendo la creación de las figuras y tablas.

Si se juntan las horas de las tareas de código y pruebas, se puede separar los esfuerzos totales dedicados a cada objetivo del proyecto (ver figura 3.15) donde:

- *Videojuego base* engloba a los sistemas básicos de las partidas y sus mecánicas.
- *Inteligencia Artificial* son todas las diversas características con agentes artificiales (ver anexo C) como por ejemplo, el jugador artificial.
- *Interfaces* se refiere a todas las pantallas que no son expresamente dentro de partida, dentro del editor de escenarios o una pantalla del modo multijugador en línea.
- *Editor de escenarios* trata todo lo relacionado con el editor de escenarios.

Clasificación del diseño e implementación

Código y pruebas dividido según los tipos de tareas

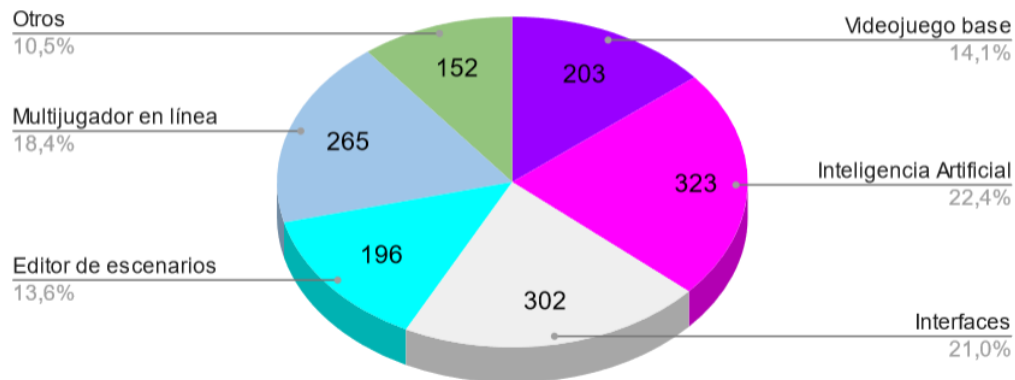


Figura 3.15: Cómputo global en horas de esfuerzos en el proyecto según el tipo de tarea.

- *Multijugador en línea* engloba los tres modelos de arquitecturas en línea realizados.
- *Otros* es el tratamiento de los ficheros y sus formatos, creación de gráficos (ver más en detalle en el anexo I.4.1), así como de mejoras o simplificaciones del sistema o cualquier otra tarea no mencionada anteriormente.

3.7.4. Posibles expansiones

Aquí se comentan brevemente algunas de las características con las que se iba a expandir el contenido de Armygeddon, o con las que se había teorizado un diseño en algún momento del proyecto:

- Fusión de terrenos (coste aproximado: Un día). Mejora visual para que el usuario no distinga el cambio de tipos de baldosas entre casillas tan bruscamente. La mejora consiste en hacer que en partes de algunas casillas haya dos baldosas superpuestas de distinto tipo. Esto trata de conseguir que el cambio entre terrenos sea curvo, en vez de completamente recto y dependiente de las casillas.
- Mejora del objetivo de investigación (coste aproximado: Un día). El objetivo de investigación no está del todo diseñado, pues pueden quedarse bloqueados jugadores al no haber suficientes recursos en el escenario para completar las maravillas (ver manual M.6.17). La solución pasaría por introducir mecánicas de generación pasiva de recursos, o por un cálculo de los recursos disponibles en cada momento y declarar la derrota del jugador según toque.
- Reproductor de música (coste aproximado: Un día). Permitiría al usuario pausar

la pista de música actual, volverla a reproducir o cambiar a otra distinta, tanto del juego como externa del usuario. Habría que distinguir una carpeta en el directorio del juego para este propósito, además de implementar la información de la pista sonando o de los controles del reproductor en la interfaz de la partida.

- Efectos en partida del terreno (coste aproximado: Una semana). Serviría para hacer de los terrenos una característica relevante más allá de lo estético, provocando por ejemplo que según el tipo de terreno, las unidades que lo pisen se desplacen más despacio o sufran daño. Sería necesario alterar muchas mecánicas y revisar el equilibrio de los escenarios y de las razas pues puede no afectarles por igual.
- Otras mecánicas de las razas (coste aproximado: Un mes). Que las unidades suban de nivel, que puedan tener mejoras según el tipo de unidad que sean, o más distinciones entre razas como por ejemplo, la matriz de energía protoss¹².
- Sistema manual de creación de elementos (coste aproximado: Hasta varios meses, dependiendo de la profundidad del sistema). Aumentar el metajuego que proporciona el editor de escenarios para permitir no solo modificar los tipos de entidades que ya hay en el juego sino también poder añadir otros nuevos aprovechando los sistemas ya existentes.
- Carga multijugador (coste aproximado: Una semana). Una mezcla de la carga de escenarios, la configuración de escenarios y la sala multijugador que se descartó por ser solo eso.
- Navegación por teclado completa (coste aproximado: Menos de una semana). Completar el sistema de navegación por teclado para que sirva también en las pantallas de juego y del editor. Requeriría hacer controles alternativos para el ratón y varios atajos para facilitar selecciones y órdenes.
- Rehumanización del jugador artificial (coste aproximado: Una semana). Añadir capacidades humanas al jugador artificial como despistes (probabilidades pequeñas de fallar), manías (tendencias a realizar algunas acciones o a realizarlas con ciertos parámetros) o cambios en la concentración (alteración de los periodos de espera entre cálculos de órdenes) para añadirle más naturalidad.
- Customización (coste aproximado: Una semana). Transliterar todas las constantes de Armygeddon en un fichero donde se le permita al usuario modificarlas y, así alterar todas las estadísticas de las entidades, parámetros del jugador artificial, límites, etc.

¹²Los protoss podrían teletransportarse o construir estructuras solo según una red de energía psi que conecta ciertas estructuras como si de una red inalámbrica se tratase.

- Nuevo sistema de casillas y posiciones (coste aproximado: Tres semanas). Separar del todo el sistema de posiciones y de casillas para las unidades, permitiendo que trabajen sobre las posiciones y con ello permitir que haya unidades de distintos tamaños o movimientos en las 16 direcciones. Requeriría una nueva búsqueda de caminos.

Estas características están más detalladas en el anexo J.

3.7.5. Opinión personal

Entiendo el impacto que debió tener StarCraft hace 25 años con solo estudiarlo desde fuera, sin haber tenido la oportunidad de jugar a este RTS en concreto. Creo que en aquel entonces ya, las desarrolladoras se toparon con que la esencia de un videojuego de estrategia en tiempo real, aun teniendo tan pocas mecánicas, es tan abierto que existe muchísimo recorrido humano que las máquinas aún no saben entender. Es un recorrido donde sigue pareciendo que se aprende más sobre cómo pensamos las personas estudiando al jugador artificial, que “enseñándole” a este cómo mejorar, a pesar de que se analicen los patrones del usuario y se estudien los distintos perfiles de estrategias. Es un recorrido que me ha llegado a abrumar llenándonos pizarras enteras y que dudo que deje de interesarme nunca (y de divertirme tampoco).

Me alegra haber podido estudiar este género de los videojuegos mientras que he dispuesto de tanta libertad para trabajar con otras características ajenas, siendo que no tenía apenas experiencia en ninguna de las dos cosas. Simplemente haber podido hacer el trabajo de fin de grado sobre esta materia ya me supone un importante paso de realización personal, ya que a mi parecer los videojuegos son una vía artística en la que además también participa el usuario.

Finalmente, quedarme con que cualquier idea que suene a despropósito tiene cabida en este ámbito siempre que al jugador le parezca divertida. Es lo que creo que guía los videojuegos y lo he podido apreciar de primera mano a pesar de que para mí fueran al principio todo locuras.

Bibliografía

- [1] Alexandre Zheng, Alonso del Rincón de la Villa, Antonio Gallén Recio, Juan Plo Andrés. Trabajo práctico de Videojuegos — StarCraft. <http://webdiis.unizar.es/asignaturas/videojuegos/#pdf-starcraft>, 2022. [En línea; acceso 18-Jul-2023].
- [2] Blizzard Entertainment. Manual de StarCraft. <http://ftp.blizzard.com/pub/misc/StarCraft.PDF>, 1998. [En línea; acceso 5-Ago-2023].
- [3] Foo'. Lazy Foo' Productions - Beginning Game Programming v2.0. <https://lazyfoo.net/tutorials/SDL/index.php>. [En línea; acceso 29-Ago-2023].
- [4] Gimp community. GNU Image Manipulation Program (Gimp) - Tutorials. <https://www.gimp.org/tutorials/>. [En línea; acceso 29-Ago-2023].
- [5] La comunidad de cplusplus. C++ Reference. <https://cplusplus.com/reference/>. [En línea; acceso 29-Ago-2023].
- [6] La comunidad de GameBanana. GameBanana - StarCraft Tutorials. <https://gamebanana.com/tuts/games/1424>. [En línea; acceso 29-Ago-2023].
- [7] La comunidad de SDL. Simple DirectMedia Layer 2.0 (SDL2). <https://wiki.libsdl.org/SDL2/FrontPage>. [En línea; acceso 18-Jul-2023].
- [8] Vinyals, O., Babuschkin, I., Czarnecki, W.M. et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. — *Nature* 575, 350–354. *Nature*, 2019. [En línea; acceso 6-Jun-2023].
- [9] Wikipedia contributors. Starcraft (video game) — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/w/index.php?title=StarCraft_\(video_game\)&oldid=1148552965](https://en.wikipedia.org/w/index.php?title=StarCraft_(video_game)&oldid=1148552965), 2023. [En línea; acceso 4-May-2023].

Anexos

Anexos A

Lista de requisitos de Armygeddon

Antes de empezar con los requisitos de Armygeddon, hace falta una base como ya se ha comentado. El requisito inicial en sí era tener un núcleo del juego que permita jugar una partida de StarCraft. Se escogió por cuenta propia una lista con lo básico en cuanto a la jugabilidad, accesibilidad, y el mejorar la experiencia del usuario. Todos los demás requisitos de Armygeddon se listan por rondas, ya que se ha seguido un diseño evolutivo, agrupadas según las líneas objetivo ya mencionadas en la sección 1.2.

A.1. Videojuego base

Los requisitos para crear un videojuego base que se considere que abstrae lo suficiente StarCraft han sido los siguientes:

1. Una partida tiene que tener un terreno, compuesto de casillas.
 - a) Las casillas tienen que tener varios aspectos.
 - b) Las casillas han de poder ser ocupadas por los elementos de la partida.
2. Tiene que haber hasta ocho equipos jugables en una partida, que son conjuntos de entidades pertenecientes a un jugador diferenciados por un color distintivo.
 - a) El usuario tiene que poder manejar su propio equipo.
 - b) Los equipos tienen que ser distinguibles a simple vista.
 - c) Tienen que estar disponibles los tres tipos (razas) de equipos del juego original: protoss, zerg y terran.
 - d) Tiene que haber un recurso denominado *Población* que limite la cantidad

de unidades que el equipo puede tener.

3. Tiene que haber minerales, que alberguen una cantidad determinada de cristal.
 - a) Al agotarse el cristal de un mineral, el mineral en sí tiene que desaparecer.
4. Tiene que haber géiseres, que alberguen una cantidad determinada de gas de alta calidad y una cantidad infinita de baja calidad.
 - a) Al recolectar gas de un géiser se obtendrá de alta calidad si lo hay, si no, se obtendrá de baja calidad otorgando menor cantidad de gas por recolección.
5. Tiene que haber refineras.
 - a) Las refineras tienen que pertenecer a un equipo y solo a uno.
 - b) Las refineras tienen que poder instalarse solo en un géiser para permitir la recolección de gas de este.
6. Tiene que haber unidades.
 - a) Las unidades tienen que pertenecer a un equipo y solo a uno.
 - b) Las unidades tienen que poder atacar a otras unidades y estructuras que no sean de su mismo equipo.
 - c) Tiene que haber varios tipos de unidades que hagan distintas cosas y que tengan distintas estadísticas en cada raza.
 - d) Tiene que haber unidades especiales denominadas trabajadoras, que sean más débiles que las otras unidades, para disociarlos del resto de unidades que no pueden realizar tareas distintas del combate.
 - e) Los trabajadores tienen que poder minar cristal.
 - f) Los trabajadores tienen que poder recolectar gas de las refineras.
 - g) Los trabajadores tienen que poder instalar refineras de gas en los géiseres disponibles.
7. Tiene que haber estructuras.
 - a) Las estructuras tienen que pertenecer a un equipo y solo a uno.
 - b) Tiene que haber una estructura base por cada equipo al iniciar una partida, que una vez destruida se acabe la partida.

- c) Los edificios tienen que tener funcionalidades como generar unidades o permitir construir otros edificios.
- 8. Tiene que haber obstáculos.
- 9. Tiene que poder ampliarse la población a través de un coste durante la partida.
- 10. Tiene que ser justo y estar medianamente balanceado el poder entre las razas.
- 11. Tiene que poder haber jugadores artificiales.
- 12. El cristal y el gas tienen que ser los recursos que actúen como moneda de cambio en el juego para obtener ventajas para el equipo dueño de dichos recursos.
- 13. El recurso población tiene que variar en función de la raza tanto lógica como gráficamente.
- 14. Tiene que haber colores en los equipos independientemente de la raza, para poder jugar la misma raza dos jugadores.
- 15. Tiene que poder seleccionarse elementos en la partida, como mínimo los propios de un equipo.
- 16. Si se trata de unidades tiene que poder seleccionarse más de una unidad a la vez.
- 17. Tiene que haber información de los elementos seleccionados por el usuario.

A.2. Partida en Armygeddon

En cuanto a la propia lógica de juego de Armygeddon, se han ido desarrollando los siguientes requisitos cronológicamente, como ya se ha dicho, en función de cómo iba avanzando el juego:

1. Primera ronda de requisitos:
 - a) Tiene que poder guardarse y cargarse partidas.
 - b) Tiene que haber sonidos y animaciones.
 - c) Tiene que haber agentes de inteligencia artificial que reciban información del escenario (ver requisitos jugadores artificiales en A.5).
 - d) Tiene que poder hacerse partidas de solo jugadores artificiales (simulaciones).

- e) Tiene que poder pausarse la partida.
2. Segunda ronda de requisitos:
- a) Tiene que haber agrupaciones de ejércitos denominadas facciones en las que los componentes no puedan atacarse entre sí.
 - b) Los ejércitos tienen que cumplir un objetivo, que es común en la facción, para ganar; perdiendo en caso de que ningún ejército de la facción los pueda cumplir o que alguna otra facción logre sus objetivos antes.
 - c) El juego tiene que poder acelerarse todo lo que se quiera (y se permita por rendimiento).
3. Tercera ronda de requisitos:
- a) Tiene que haber varios tipos de objetivos, ligados a las facciones, y estos tienen que poder mezclarse en las partidas.
 - b) Los posibles objetivos de las facciones tienen que ser:
 - 1) *Exterminar* a los ejércitos rivales.
 - 2) *Sobrevivir* una determinada cantidad de tiempo.
 - 3) *Conquistar* una zona determinada.
 - 4) *Investigar* para desarrollarse armamentísticamente lo suficiente.
 - c) Tiene que poder verse el progreso del objetivo en partida, al igual que los ejércitos que hay y quiénes son aliados o enemigos.

A.3. Editor de escenarios

Aquí se listan los requisitos para el editor de escenarios:

1. Primera ronda de requisitos:
- a) Tiene que permitir ver un escenario moviendo la cámara, añadir y eliminar elementos.
 - b) Tiene que haber modos de edición para clasificar todos los elementos.
 - c) Tiene que permitirse editar entidades de un ejército en concreto (unidades, estructuras y refinerías que sean del color seleccionado).

2. Segunda ronda de requisitos:

- a) Tiene que poder eliminarse de vez todos los elementos que pertenezcan a los ejércitos, para poder reutilizar la parte decorativa de los escenarios con otra configuración.

3. Tercera ronda de requisitos:

- a) Tiene que poder configurarse la cantidad de recursos iniciales de cada ejército para aumentar las posibilidades de juego.
- b) Tiene que poder visualizar y editarse las facciones, ya que son fundamentales en un escenario.
- c) Tiene que poder renombrarse tanto ejércitos como facciones.
- d) Tiene que poder moverse la cámara con el ratón para aumentar accesibilidad.
- e) Tiene que poder configurarse el tipo de jugador que controla por defecto cada ejército.

4. Cuarta ronda de requisitos:

- a) Tiene que existir zum ajustable, para poder verse y poder analizar así el escenario entero en la pantalla.
- b) Tiene que haber ajustes de visibilidad, para poder enfocar distintos aspectos del escenario por separado y actuar solo sobre lo visible.
- c) Tiene que poder configurarse el tipo de los objetivos de cada facción y sus parámetros.

5. Quinta ronda de requisitos:

- a) Tiene que poder seleccionarse y desplazar los elementos del editor directamente, para evitar tener que estar borrando y añadiendo elementos en su defecto.
- b) Tiene que poder editarse una descripción textual del escenario para saber luego en qué consiste.
- c) Tiene que poder editarse la dimensión del escenario, ya que no tiene porqué saberse de antemano la dimensión exacta a la hora de crearlo.

A.4. Multijugador en línea

Se han considerado requisitos para el modo multijugador en línea los siguientes:

1. Primera ronda de requisitos:
 - a) Podría establecerse una conexión entre usuarios para crear una sala multijugador.
 - b) Podría haber chat para poder comunicarse los usuarios entre sí.
2. Segunda ronda de requisitos:
 - a) Podría haber consenso entre los jugadores.
 - b) Podría haber un mínimo de tolerancia a fallos.

A.5. Jugadores artificiales

El jugador artificial ha estado ligado a los siguientes requisitos mínimos:

1. Tiene que tener en cuenta los recursos, unidades y estructuras de las que se dispone.
2. Tiene que tener inteligencia conjunta entre todas las entidades y no solo separada por unidades, simulando a un usuario que cumple el papel de comandante militar.
3. Tiene que tener estado: información entre ciclos respecto a tácticas que aún está realizando el jugador artificial.
4. Tiene que tener memoria: capacidad de recordar información entre ciclos de ejecución del jugador artificial.
5. Tiene que tener mecánica de retirada.
6. Tiene que tenerse en cuenta al menos los siguientes aspectos para tomar sus decisiones:
 - La geografía del escenario.
 - La disposición de los enemigos y de sus puntos débiles.
 - Las bajas sufridas por el propio ejército.

A.6. Otros

Aquí se listan otros requisitos que se han estimado oportunos para Armygeddon:

1. Tiene que mostrarse toda la información del escenario antes de jugarlo.
2. Tiene que poderse ver Armygeddon en modo pantalla completa y en modo ventana, conservando el ratio 4:3 de anchura por altura en la resolución, cubriendo con barras negras la zona no utilizada.
3. Tiene que permitirse ajustar los controles del usuario.
4. Tiene que haber un sistema de navegación alternativo al ratón en las pantallas que sea sencillo implementarlo.

Anexos B

Lógica de las entidades

En este anexo se detalla brevemente la lógica de las entidades en función de su tipo y de su estado.

B.1. Unidades

Para simplificar el comportamiento de las unidades de cara al usuario se sigue un diseño de autómata cuyos estados refieren a la acción que está realizando la unidad en perspectiva y no actualmente. Por ejemplo, al estar *luchando* una unidad contra un objetivo, se entiende que la unidad está *luchando* incluso cuando se está desplazando hasta dicho objetivo. Lo mismo que por ejemplo para el minado, la unidad estará minando mientras va a por cristal, extrae el mineral o lo trae de vuelta a un edificio. Por tanto, solo cuando se esté moviendo a un punto en concreto sin ninguna acción subsecuente la unidad estará en estado *Moviéndose*.

Hay que destacar que en el caso de las unidades se intenta permanecer en el estado en el que se está, ya que al representar los estados las acciones que realizan, y dada la dinámica de las partidas basada en la gestión continua de una alta cantidad de unidades, es más conveniente para el usuario de esta manera. En concreto, los estados *Atacando*, *Instalando*, *Construyendo*, *Minando* y *Recolectando* han sido objetivo de mecánicas adicionales para intentar conservar a la unidad realizando dicha actividad tras tratar el estímulo (ya sea orden del jugador o evento externo) que obliga a la unidad a adoptar uno de esos estados. Dichas mecánicas de conservación se detallan en el manual junto con sus estados en la sección del manual M.6.9.

El estado *Muriéndose* es especial y solo refleja la muerte de la unidad. Aunque en este estado ya no se esté ocupando ninguna casilla ni mostrando sus barras de vida y escudo, sí que se muestra la animación de muerte de la unidad. Cuando termina

la animación es cuando se considera la unidad muerta del todo y que en breve va a ser liberada de memoria. Este estado es apenas perceptible por el usuario pues si una unidad seleccionada muere enseguida se quita de la selección automáticamente. El único caso de derrota de la unidad que no se refleja en la figura B.1 y, que no pasa por este estado, es cuando un trabajador está recolectando dentro de una refinera mientras esta es destruida. Esto es así ya que se pretende simular la muerte de la unidad bajo la explosión de la refinera y no requiere de animación alguna.

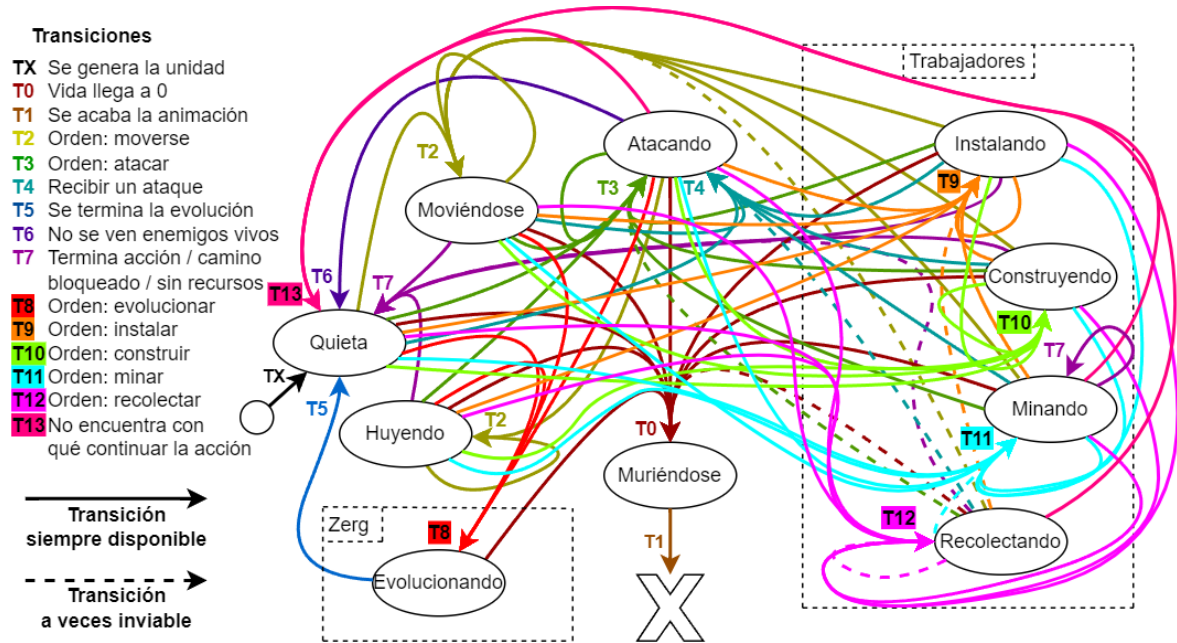


Figura B.1: Esquema de las transiciones entre estados de las unidades.

Por último destacar que el estado *Quieta* representa la ausencia de acción y es el estado inicial y el de por defecto.

B.2. Estructuras

Las estructuras en Armygeddon tienen una limitada funcionalidad más allá de habilitar otras construcciones, aportar población o ser puntos de entrega de gas o cristal, que se consideran características pasivas (ver detalles en anexo M.6.10). Estas características pasivas solo se relacionan con el estado de la estructura en que están disponibles cuando esta ya está construida y no está colapsando.

El resto de características de las estructuras, es decir, las características activas, están ligadas a su estado como se aprecia en la figura B.2 y que se detallan a continuación desde el punto de vista del estado de la estructura:

- *Construyéndose*. La estructura se está construyendo o está a la espera de que la terminen de construir, si es el caso de una estructura terran. Es el estado inicial

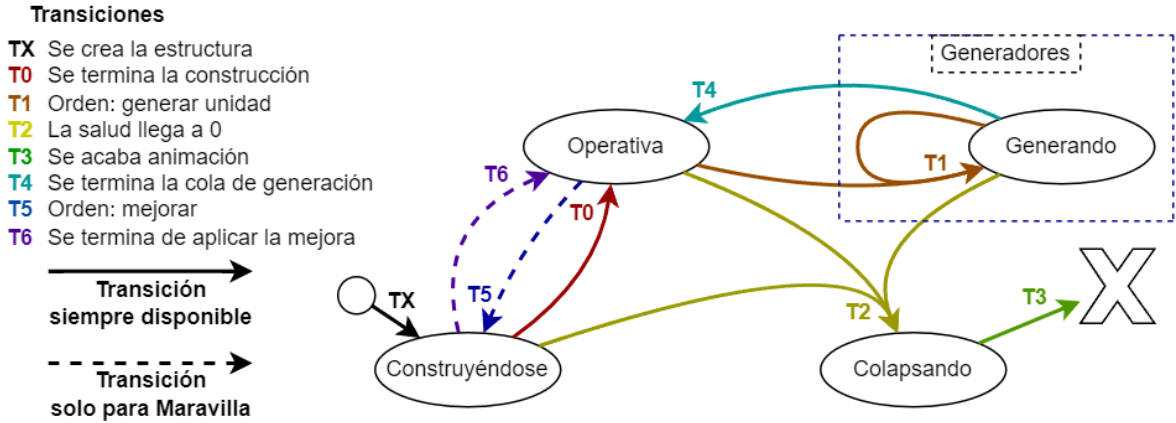


Figura B.2: Esquema de las transiciones entre estados de las estructuras.

salvo si se trata de una maravilla (ver anexo M.6.17) que también puede ser un estado transitorio ya que al hacer una mejora se pasa por este estado mientras esta se realiza.

- *Operativa*. La estructura está inactiva a la espera de recibir órdenes, ya sean de generar unidades o de mejora si es que el tipo de estructura lo permite.
- *Generando*. Este estado solo está disponible si la estructura puede generar unidades, ya que representa que está trabajando en la generación de una unidad. En este estado se pueden ordenar generar más unidades y por tanto puede haber más de una unidad a la espera de ser generada.
- *Colapsando*. La vida de la estructura ha llegado a cero y está siendo destruida. La estructura ya no ocupa casillas y se está mostrando su animación de colapso.

Como se ha comentado una estructura que no está ni en *Construyéndose* ni en *Colapsando* permite la recepción de recursos. Esto ocurre cuando un trabajador minando o recolectando, con una carga de gas o de cristal, se acerca lo suficiente a una estructura, como se aprecia en la figura B.3. En rojo se ve las casillas ocupadas por la estructura y en cian la distancia optimizada¹ desde la que se puede otorgar recursos. Dicha distancia es de una casilla pues es lo máximo que se puede acercar una unidad para simular que esta deposita el recurso en la estructura.

Existe un caso especial de estructura, el criadero (ver anexo M.8.7), que tras construirse está permanentemente en el estado *Generando* para justificar su mecánica de generación acotada de larvas.

¹Es importante aclarar que en el proyecto se utilizan dos sistemas de distancia distintos y que según el que se utilice en cada situación, en el documento se refiere a uno o a otro (ver anexo H.1).

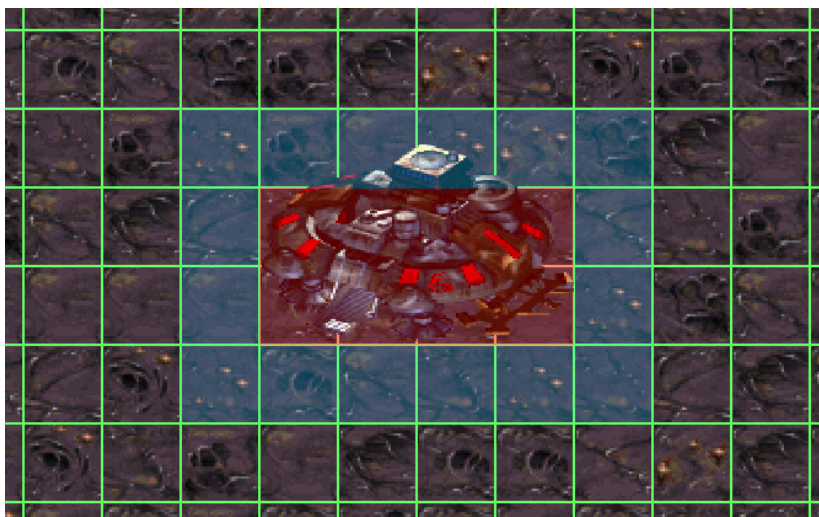


Figura B.3: Distancia de deposito de recursos alrededor de las estructuras.

B.3. Refineras

Una refineria es una entidad especial de los ejércitos que por funcionalidad no encaja ni en unidades ni en estructuras, y por eso está tan distinguida. No puede recibir recursos como las estructuras, sino que los otorga. Tampoco se puede crear en casillas libres, sino que requiere de un géiser para instalarse y que, en caso de ser destruida la refineria, dicho géiser reaparece. Además de no tener ninguna acción y permitir interactuar de una manera única a los trabajadores del mismo ejército. Ver figura B.4.

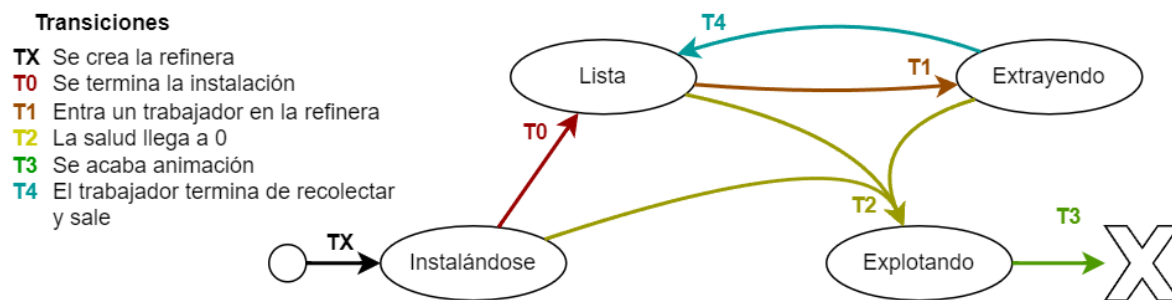


Figura B.4: Esquema de las transiciones entre estados de las refineras.

Los estados de una refineria son muy limitados, ya que se trata de una entidad débil y asequible, cuya principal intención es permitir la mecánica de extracción de gas en un proceso volátil y peligroso:

- *Instalándose.* La refineria está asentándose sobre el géiser. En este estado solo puede ser o bien destruida si salud llega a cero o terminada por el método de instalación de cada raza.
- *Lista.* La refineria está lista para alojar a un trabajador para que recolecte gas.

- *Extrayendo.* Hay un trabajador dentro de la refinera recolectando gas. Si la refinera se destruye en este estado el trabajador que está dentro muere instantáneamente. El gas que se recolecte al acabar este proceso depende del gas que le quede al géiser sobre el que está instalada dicha refinera. Es decir, la cantidad de gas de alta calidad del géiser va disminuyendo conforme se realiza una extracción en una refinera que esté instalada sobre él, afectando a dicho géiser incluso si la refinera es destruida.
- *Explotando.* La refinera está explotando, aludiendo al peligroso manejo del combustible que es el gas. Al acabar la animación desaparece y reaparece en la misma posición un géiser con el mismo gas restante que le quedase a la refinera.

B.4. Géiseres

Un géiser es distinto a los minerales ya que ni desaparece ni se puede obtener recursos de él en este estado. Cuando se instala una refinera en un géiser, este se ve sustituido por la refinera, pero conservando la cantidad de gas restante de alta calidad. Si la refinera es destruida, el géiser tiene que volver a aparecer, de forma que donde haya un géiser inicialmente esté siempre el géiser o en su defecto una refinera.

Para garantizar que se controla mínimamente la zona como para permitir a un jugador instalar una refinera sobre el géiser, el trabajador que vaya a instalar una refinera tiene que estar a una casilla de distancia optimizada. Como se ve en la figura B.5, en amarillo estaría el espacio ocupado por el géiser y en rosa alrededor estaría el rango donde un trabajador podría instalar una refinera en el géiser.

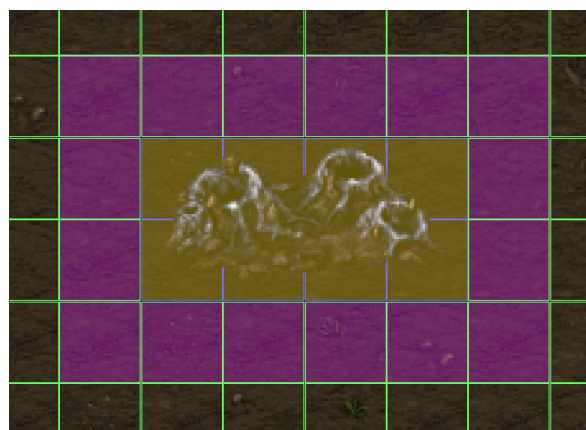


Figura B.5: Distancia respecto a un géiser para instalar una refinera en el.

B.5. Minerales

La funcionalidad de un mineral se reduce a ser minado por trabajadores para obtener cristal. Ver figura B.6.

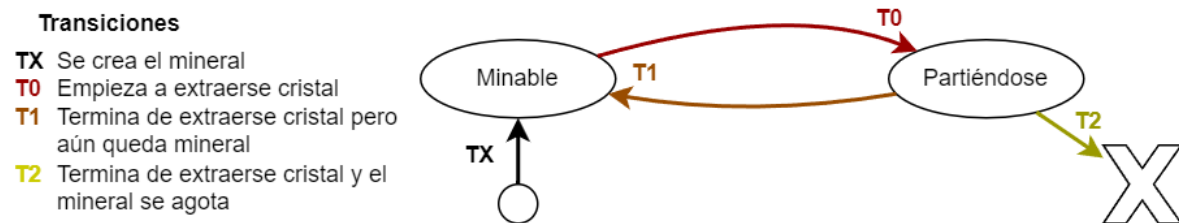


Figura B.6: Esquema de las transiciones entre estados de los minerales.

Cuando un trabajador comienza a extraer cristal de un mineral, este se “bloquea”, impidiendo a otro trabajador minar el mismo mineral mientras dure la extracción. Al acabar este proceso, el trabajador obtiene una carga de cristal (con un valor de ocho unidades de cristal) que poder llevar a una estructura y el cristal se libera, permitiendo de nuevo que un trabajador vuelva a utilizarlo. De esta manera varios trabajadores pueden estar trabajando sin molestarse entre sí en el uso de un mismo mineral mientras se devuelve el cristal obtenido a una estructura.

En sí no es necesario estar a una casilla de distancia del mineral para poder extraerlo, pues dicha acción consiste solo en romper el cristal, pero al ser el alcance de ataque de las unidades trabajadoras solamente de una casilla de distancia real (2,4 siendo exactos) el resultado es el mismo. Como se puede ver en la figura B.7 el círculo naranja representa el rango de alcance (distancia real) de un trabajador, en amarillo las casillas que están cubiertas por dicha distancia de alcance y en rojo los cristales que pueden ser minados desde esa posición. El objetivo de la figura es reflejar cómo el trabajador tiene que alcanzar a cubrir por completo al menos una casilla de las dos que tiene un mineral en su rango de alcance para poder comenzar el minado.

B.6. Obstáculos

Los obstáculos no tienen ninguna interacción, por lo que tampoco se permite seleccionarlos para por ejemplo saber el nombre del obstáculo, o simplemente para distinguirlos como obstáculos. Su manera por tanto, de ser distinguidos como elementos obstaculizadores y decorativos del escenario es precisamente esa, que no reaccionan de ninguna manera al usuario y que lo único que los distingue del tipo de suelo es que estos ocupan un espacio. Se trabajan distintos tipos de obstáculos con diferentes tamaños, principalmente pequeños para conseguir permitir al usuario la mayor flexibilidad a la hora de configurar la geografía del escenario.

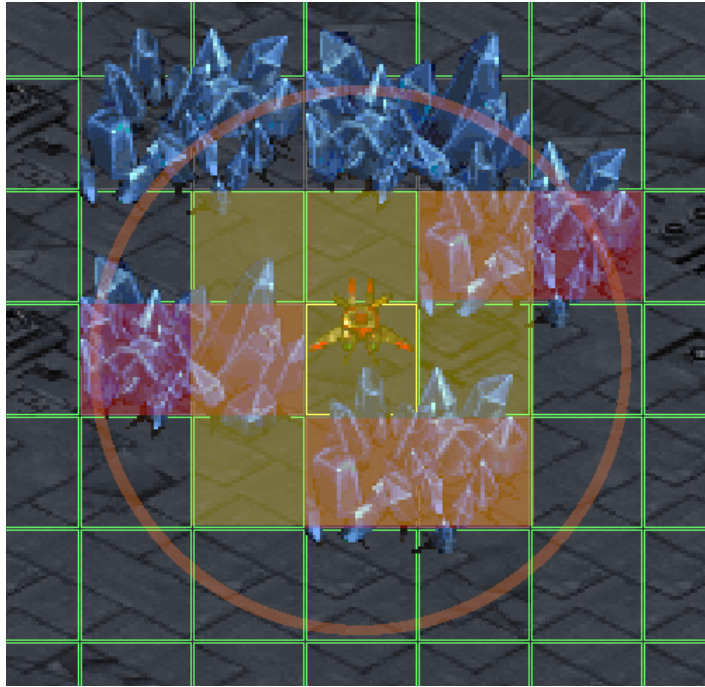


Figura B.7: Alcance de un trabajador relacionado con minerales cercanos.

Durante fases iniciales del proyecto, se trataron diversas características respecto a este tipo de entidades que no se llegaron a implementar y se detallan en el anexo J. Finalmente de todas estas características, se optó por implementar la defensa (ver anexo M.6.15), que es en sí una estructura con la funcionalidad de obstaculizar.

Esta *estructura-obstáculo* no se puede crear durante la partida y está hecha principalmente para llevar el modo de supervivencia a otro nivel, actuando de defensa temporal al recibir menos daño de ciertas unidades y, sin mostrar su indicador de vida ya que se intentan conceptualizar como *obstáculos destruibles*.

Anexos C

Inteligencia artificial

En este anexo se tratan las diversas aplicaciones de inteligencia artificial que tiene Armygeddon. En especial la del jugador artificial en el anexo C.1, pero también se detalla el ejército neutral en el anexo C.2, la búsqueda de caminos de las unidades en el anexo C.3 y finalmente, el antiguo sistema de invasores en el anexo C.4.

C.1. Jugador artificial

Requiere un apartado especial el diseño y desarrollo del *jugador artificial*¹, pues ha sido elemento clave del diseño tanto de sí misma como del diseño de Armygeddon en sí, ya que también suple la falta de testeo humano y, por tanto es además determinante en el rumbo y en el balance del propio proyecto.

Como en cualquier aplicación de inteligencia artificial en un videojuego, la principal característica del jugador artificial tiene que ser la imprevisibilidad². No solo porque poder predecir el comportamiento de otro jugador le resta dificultad, y por lo general esto empeora el disfrute del usuario, sino también porque la sensación de inmersión y la focalización en un videojuego se ve alimentada por la competitividad de tener rivales diseñados para parecer ser humanos. Aunque sepa el usuario perfectamente que está jugando con una máquina.

Primero, se realiza un análisis de cómo son las mecánicas de Armygeddon (anexo C.1.1), para poder después analizar cómo las afronta un usuario (anexo C.1.2) y así, poder finalmente crear un diseño de jugador artificial (anexos de C.1.3 a C.1.15) y

¹Como se indica en el manual (anexo M.6.3) un jugador artificial es un agente artificial diseñado para controlar un ejército de manera autónoma y automática.

²https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games#Video_game_combat_AI (último acceso: 8/8/2023).

posteriormente implementarlo (anexos de C.1.10 a C.1.14).

C.1.1. Análisis sobre Armygeddon

Lo que caracteriza principalmente a un videojuego de estrategia en tiempo real es el elevado número de acciones que se pueden realizar en todo momento³. En Armygeddon, haciendo una estimación optimista del número de acciones disponibles en un ciclo para un jugador, y solo teniendo en cuenta el posicionamiento de 10 unidades en un escenario de 40 filas y 40 columnas (un escenario pequeño) se obtiene:

$$C(n, r) = \frac{n!}{r!(n-r)!}$$
$$C(1600, 10) = \frac{1600!}{10!(1600-10)!} = 29 * 10^{24}$$

Donde $C(n, r)$ es la ecuación de las combinaciones de tamaño r en un set de n elementos.

Existen muchos tipos de acciones que puede realizar el usuario y no todas se van a querer replicar en el jugador artificial. Por ejemplo, el desplazar la cámara, o el tener que seleccionar una entidad para obtener su información, no son útiles para el jugador artificial, ya que son órdenes transparentes al resto de los jugadores al no afectar de ninguna manera al escenario. En el manual de usuario (anexo M.6.13) se detallan todas las órdenes posibles en Armygeddon.

A pesar de la complejidad intrínseca de un RTS, en una partida una persona solo sopesa un reducido número de ideas abstractas guiadas por un limitado conjunto de perfiles de estrategias.

C.1.2. El usuario de Armygeddon

Para definir la relación entre el usuario y Armygeddon, se analizan las estrategias que sigue un usuario para ganar una partida. Es incalculable el número de estrategias a seguir dentro de un RTS, ya que también lo es el número de ideas que se pueden seguir y, estas variar dentro de la partida en función del escenario, la raza, o incluso de los rivales.

Si se analiza las distintas mecánicas del videojuego, se pueden distinguir una serie de aspectos comunes que siempre se tienen en cuenta en mayor o menor medida en una estrategia:

³La revista *Nature* en un artículo [8] sobre inteligencia artificial en *StarCraft II* estimaba que las partidas, que duran unos 10 minutos cada una, tienen decenas de miles de elecciones en las que su agente inteligente, *AlphaStar*, posee un abanico de acciones de un tamaño de grado 10^{26} .

- *Agresividad.* Cuando atacar, si es que se va a atacar, en qué momentos y con qué intensidad. Puede por ejemplo, indicar que hay que atacar hasta con los trabajadores nada más empezar la partida, o que haya que ir mandando tropas cada pocos segundos para un acoso de los rivales constante.
- *Exploración.* Cuando explorar, cuanto, qué algoritmo de búsqueda seguir, cómo reaccionar a los estímulos que se van encontrando. Una persona puede por ejemplo, seguir una búsqueda en anchura para evitar sorpresas explorando primero las zonas cercanas, como por ejemplo ir a buscar los recursos más lejanos realizando una búsqueda en profundidad para poder explotarlos antes que sus contrincantes.
- *Economía.* Los recursos que un ejército genera están atados a la cantidad de recursos explotables y a la capacidad de explotación de este, ya que la extracción de los recursos está ligada al número de trabajadores. Entonces, si un ejército decide invertir sus recursos en mejorar su economía, tendrá por norma general menos capacidad militar al comienzo y, más conforme avance la partida, y viceversa: un ejército que priorice su poder militar antes que la economía tendrá mayor capacidad militar al principio y menor conforme avance la partida.
- *Expansión.* La ubicación es importante: Si las infraestructuras de un jugador están separadas es más difícil defenderlas, pues hay que desplazar más las tropas para ello. Sin embargo, es más fácil obtener recursos si las estructuras están más separadas, ya que la obtención de recursos está limitada por la distancia desde los propios recursos hasta las estructuras, al tener que traer de vuelta dichos recursos una vez extraídos para poder ser utilizados. También afecta a la capacidad militar, ya que permite generar tropas con mayor o menor versatilidad al generarse estas en las estructuras⁴.
- *Investigación.* En qué momento se invierte para poder obtener las tropas mejoradas influye. No solo hay que construir estructuras avanzadas, sino que hay que obtener gas para ello, requiriendo primero instalar una refinería para lograrlo. También se relaciona con la prioridad de un tipo de tropas sobre otro, ya que atacan de diferente forma, tienen distinto tiempo de generación, etc.

Con todo, una estrategia puede ser combinación de varias tácticas muy contrarias entre sí ya que como ya se ha dicho, tanto la configuración del escenario como el comportamiento de los contrincantes afecta. Por lo que en definitiva, además de ser el abanico de opciones muy amplio y por ello tener que seguir una estrategia muy abstracta, el jugador artificial a diseñar tiene que ceñirse por estímulos espontáneos. Siempre teniendo en cuenta la información obtenida a lo largo de la partida y responder

⁴Exceptuando en cierta medida a la raza zerg, ya que genera los soldados de las larvas y estas aunque salgan de los edificios, pueden desplazarse antes.

con tácticas que pueden ser muy volátiles.

C.1.3. Protocolos

Una persona a pesar de lo elaborada que sea su estrategia no deja de resumir sus tácticas en seguir estímulos como “necesito cristal” o “me faltan soldados”, y siempre realiza una táctica por vez⁵ afrontando dichos estímulos de una manera o de otra en función de su estrategia. Si bien es cierto que el usuario enfoca uno u otro aspecto (cristal, ejército, etc.) según se va acordando, la prioridad que el usuario le otorgue, o según los eventos de la partida, en realidad el usuario “se ve obligado” a tener en cuenta todo cada cierto tiempo, ya que descuidar un aspecto puede ocasionarle la derrota. En conclusión: Si un usuario fuese ordenado, y sin tener en cuenta sus cambios de contexto, este podría jugar una partida repasando todas sus necesidades periódicamente y realizando acciones según convenga en los distintos aspectos de la lógica de partida.

Es precisamente esta la idea con la que se diseñan los jugadores artificiales en Army-geddon, para intentar conseguir *naturalidad*. Se modela así el jugador artificial para simular a una persona con procesamiento monotarea⁶ que periódicamente va realizando solo una acción dentro de un conjunto posible de acciones abstracto.

Aparece un problema, y es que no siempre acompañan las órdenes que se pueden realizar en partida (detalladas en el anexo M.6.13) con las ideas que motivan las tácticas del jugador. Por ejemplo, un usuario puede querer obtener cristal pero para ello tener que:

1. Explorar para encontrar minerales (moverse).
2. Eliminar a los rivales cercanos de la zona (atacar).
3. Construir una estructura a la que devolver los recursos (construir).
4. Ordenar a los trabajadores ir a por el cristal (minar).

Y esto tiene que tenerse en cuenta para crear los protocolos que va a tener el jugador artificial, ya que se va a intentar obtener información de la partida y hacer que sus tácticas evolucionen conforme avanza la partida. Así que asociando los tipos de órdenes de la partida con las ideas que motivan las tácticas del jugador se definen seis protocolos:

⁵Es decir, el usuario no tiene varios ratones y hace varias cosas de vez, procesadas mentalmente en paralelo, aunque sería posible.

⁶El cerebro humano no es tan eficiente trabajando en distintas tareas a la vez de forma paralela como en serie: <https://es.wikipedia.org/wiki/Multitasking> (último acceso: 7/6/2023).

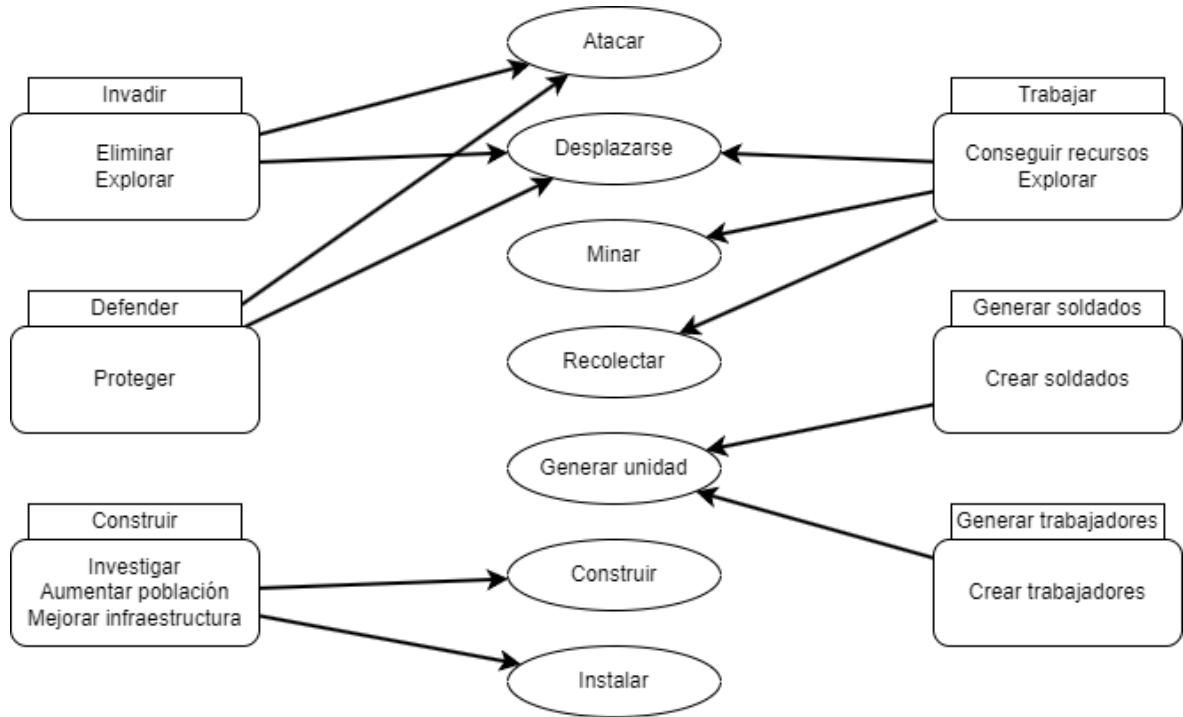


Figura C.1: Relación de los protocolos del jugador artificial y las ideas del jugador humano que siguen junto con los tipos de órdenes usadas.

- Invadir. Utilizar las tropas para lograr los objetivos de la facción de manera ofensiva.
- Defender. Consiste en preservar el estado del ejército para que todo siga funcionando. Esto es, movilizar tropas para protegerse, llegando a atacar a las posibles amenazas si es necesario.
- Construir. Comandar a los trabajadores para crear estructuras o instalar refineras, según sean las necesidades.
- Trabajar. Gestionar las rutas de los trabajadores para obtener recursos, llegando incluso a mandarlos a explorar si es necesario.
- Generar soldados. Aumentar el poder militar, en función de las necesidades.
- Generar trabajadores. Crear trabajadores para tener suficiente mano de obra y capacidad logística con los recursos.

En un momento determinado del desarrollo se cambió el modelo de inteligencia artificial añadiéndole *memoria* y *estado* más allá del propio ciclo en el que se ejecuta un protocolo, así que a continuación se explica cada protocolo desde el punto de vista de sus dos diseños.

C.1.4. Invadir

Este protocolo trata de gestionar los ejércitos de manera ofensiva para lograr los objetivos de la partida.

En una *primera aproximación* se diseñó como la actualización de todos los soldados como invasores independientes sin comunicación alguna entre sí. Se pretendía que las tropas exploraran el escenario aleatoriamente atacando a los rivales que se fueran encontrando, y para eso se creó la inteligencia del invasor ahora obsoleta (ver sección C.4). Este diseño de los invasores provocaba que las tropas estuvieran concentradas en su base, y que se fueran dispersando a medida que se alejaban de esta, pero cualquier emboscada o movimiento concentrado enemigo vencía fácilmente esta táctica. Además, con ese patrón de movimiento no se garantizaba que explorasen todo el escenario, aunque el sistema de generación de movimientos lo permitiera, ya que no se tenía en cuenta qué se había visitado ya y qué no.

Con el *segundo diseño* añadiendo memoria y estado al jugador artificial, se pasó a simular a una persona actuando con tácticas globales. Cada decisión tomada resuelve un problema visto con anterioridad (ver tabla C.1).

Con este diseño todas las tropas inicialmente se designan como patrulladores, y estos son gestionadas con el protocolo Defender (ver anexo C.1.5). El jugador artificial crea una invasión conforme estime que es el momento apropiado, y lo hace en base a unos estímulos de confianza. Por ejemplo, tiene muchos soldados, mantener pocas invasiones, etc.

Al crear una invasión se designa un grupo de patrulladores, pasando a ser estos invasores y se envían al sector más interesante para el jugador artificial. El poder militar de la invasión tiene que ser mínimamente superior a la amenaza que supone el sector objetivo para poder crearse.

Para calcular el interés de los sectores se tienen en cuenta 4 factores:

1. *La relevancia* del sector según el objetivo de la facción, llegando a no invadir nunca si es irrelevante.
2. *La distancia* al sector desde el sector que tenga la primera base del jugador artificial (sector base). Recorrer el escenario es peligroso, las invasiones pueden ser interceptadas y también resulta más complicado a mayor es la distancia.
3. *La amenaza* que supone ese sector para el jugador artificial. La amenaza en un sector se mide teniendo en cuenta las entidades enemigas que hay en él en ese momento: tropas, estructuras y refinerías.

Primer diseño	Segundo diseño
El invasor no tiene en cuenta el escenario, solo su entorno.	Se trabaja con secciones lógicas cuadradas del escenario llamadas sectores.
No se tiene en cuenta por dónde no está funcionando invadir.	Los sectores guardan recuento de muertes.
No se tiene en cuenta por dónde parece que es peligroso invadir.	Los sectores tienen asociado un indicador de amenaza.
No se quedan tropas defendiendo, todas se hacen invasores.	Se crea el patrullador, inicialmente todas las tropas son patrulladoras y su misión consiste en vigilar las cercanías a las estructuras.
Se envían todas las tropas siempre a invadir aunque haya pocas.	Se tiene en cuenta el número de patrulladores antes de convertirlos a invasores.
El invasor va solo inicialmente y es débil.	Se trabaja con invasiones, que son grupos de invasores.
Los invasores pueden ir por donde no hay nada que obtener.	Las invasiones solo se focalizan en sectores donde haya amenaza.
El invasor no se retira.	Se retiran los invasores que tienen poca vida, y también se retira la invasión si está yéndole mal al jugador artificial.

Tabla C.1: Relación entre los problemas de diseño del primer modelo de Invadir con las decisiones del segundo.

4. *Las pérdidas* en ese sector. Los tres factores anteriores, que son preventivos, pueden no ser suficientes si el jugador por ejemplo, desplaza las tropas para crear emboscadas o aprovecha muy bien el entorno. Es por esto que el jugador artificial tiene que aprender de sus errores sean cuales sean y evitar zonas donde las invasiones han tenido bajas o retiradas, provocando así alternativas en el ataque hasta dar con el punto débil de las defensas rivales.

Si una unidad invasora tiene poca vida, se le mandará retirarse y se le reasignará como patrullador. Se tiene en cuenta la retirada de una invasión al completo conforme a unos estímulos de confianza, como lo son las bajas, ataques en el sector base, pocos recursos, etc.

Finalmente, si una invasión logra su objetivo, sus integrantes no vuelven al cupo de patrulladores, sino que se cambia el sector objetivo a uno inmediatamente cercano para aprovechar la ubicación.

C.1.5. Defender

La tarea del protocolo consiste en proteger el territorio y las entidades del jugador de otros jugadores.

El primer diseño consistía en el análisis de todas las tropas, comprobar cuales no estaban luchando ya, y comandarlas siguiendo unas prioridades de defensa:

1. Defensa directa de los trabajadores. Si hay un trabajador luchando, manda a un defensor a atacar al objetivo del trabajador. Los trabajadores son el motor de la economía en una partida y, por eso son la máxima prioridad.
2. Defensa indirecta de las estructuras. Si hay un enemigo cerca de una estructura, manda a un defensor a atacarle.
3. Defensa indirecta de las refineras. Si hay un enemigo cerca de una refinería, manda a un defensor a atacarle. Ocupan una prioridad menor que las estructuras ya que solo pueden ser construidas en lugares específicos, y no siempre resulta rentable defenderlas.
4. Defensa directa de los soldados. Si hay un soldado luchando, manda a un defensor a atacar al objetivo del propio soldado. Los soldados son lo más fácilmente reemplazable y por eso ocupan la última prioridad. Este tipo de defensa conjunto al primer diseño de invasor (anexo C.1.4), provocaba que los invasores fueran obteniendo más y más refuerzos cuando se encontraban con enemigos, escalando el conflicto progresivamente.

Cada prioridad va restando defensores del cupo, deteniéndose el protocolo si no quedan más defensores. Si tras todas las evaluaciones siguen quedando defensores sin asignar, no se les ordena nada.

Con el segundo diseño se ganó en eficiencia y en precisión, ya que solo se tenían en cuenta a los patrulladores como defensores, y solo se ejercían ciertas defensas en función de cómo esté yendo la partida. Además, tampoco se envían refuerzos a los invasores, ya que eso se gestiona desde C.1.4. Por último, los defensores que no han sido asignados a ninguna defensa son reorganizados, con los antiguos movimientos de invasor (ver C.4) para patrullar la zona, y salvo si la partida está yendo bien siendo mandados de vuelta a un edificio cada poco.

C.1.6. Construir

Este protocolo busca satisfacer las necesidades de las distintas infraestructuras disponibles en Armygeddon, incluidas las refineras, ordenando sus respectivas construcciones.

En el *primer diseño* se establece un reconocimiento por las necesidades del jugador artificial que pueden cubrirse con una construcción (o instalación en el caso de las refineras). Al llegar a una necesidad que el jugador artificial se pueda costear, la ordena realizar. Solo se ordena una construcción por vez que se invoca al protocolo, y esto ocurre por dos razones:

1. Para simular al jugador humano: El usuario aunque abstraiga la idea de construir siempre construye un edificio por vez ya que las funcionalidades de estos son muy variadas. Si bien una persona puede ordenar construir varios edificios del mismo tipo por vez para satisfacer una necesidad grave, esto ocurre con muy poca frecuencia como para reflejarlo en el jugador artificial.
2. Para evitar construcciones inútiles: Una partida es muy volátil y el tiempo de construcción de los edificios muy elevado. Si se destinan muchos recursos de golpe a un aspecto de la partida es probable que estos se acaben desaprovechando.

Esto no impide sin embargo que se puedan mandar construir en distintas llamadas a este protocolo más infraestructura incluso si hay ya infraestructura en construcción.

La ubicación de las estructuras a construir se decide aleatoriamente siguiendo un algoritmo que busca espacios disponibles cercanos a los edificios ya existentes. Priorizando si se puede dejar cierto espacio para que puedan pasar las unidades. Al ser aleatorio se consigue que con mayor probabilidad se expanda la infraestructura de una manera agrupada a la par que imprevisible.

El orden de prioridades de construcción es el siguiente:

1. Estructura base. Es la única manera de crear trabajadores, que son el núcleo de cualquier ejército y que permite conseguir todo lo demás.
 - Se manda construir una base si no hay ninguna.
2. Aumento de población. Este recurso limita a los trabajadores, no solo a los soldados, y sin poder generar de ambas no se puede ni avanzar ni defenderse.
 - Si la población actual está cerca de la población máxima se ordena esta construcción o generación⁷.
3. Estructura de reclutamiento. Sin soldados el ejército es muy débil: Un soldado básico tiene mayor potencial militar que varios trabajadores juntos.
 - Se manda construir una estructura de reclutamiento si hay una estructura de reclutamiento que esté a la espera de órdenes.
4. Estructura avanzada. Es la siguiente prioridad no solo por las estadísticas mejoradas, sino por la combinación de distintos tipos de ataques que en conjunto aumenta las posibilidades de victoria.
 - Se manda construir una estructura avanzada si no hay ninguna.
5. Refinera. Puede haber escenarios en los que no haya siquiera géiser en los que instalar una refinera, y aún así el gas no es necesario para ganar una partida con la mayoría de objetivos de partida.
 - Se manda construir una refinera si no hay ninguna.

En la *segunda versión del diseño* se mejoran algunas condiciones de construcción para permitir mayor expansión del ejército y que esta sea más precisa:

- Estructura base. Solo si no hay ninguna → Si se es zerg, se ordena construir siempre que no haya larvas, ya que las larvas sirven para generar todo además de los trabajadores.
- Estructura avanzada. Solo si no hay ninguna → Si se es zerg, se ordena construir si se puede costear o si hay refineras, ya que es la única estructura avanzada que requiere gas y puede no haber géiseres en todo el escenario. Además, se tiene en cuenta el número de trabajadores y si hay pocos no se manda construir ya que se consume uno con cada construcción y la estructura avanzada se considera

⁷En el caso de los zerg, no tienen un edificio para aumentar la población sino una unidad especial.

opcional.

- Refinera. Solo si no hay ninguna → Si no hay ninguna y a partir de una se decide si se ordena su instalación con una probabilidad relacionada directamente proporcional con el número de trabajadores e inversamente proporcional con el número de refineras ya activas.

Además, en este segundo diseño, de no haber ninguna estructura el jugador artificial busca una posición cercana a un trabajador para construir una base. En el primer diseño si el jugador artificial desde el momento en el que se queda sin estructuras no manda construir ninguna.

C.1.7. Trabajar

En este protocolo se trata las labores de los trabajadores que no tienen que ver con la construcción. Esto es la obtención de cristal y gas, así como la exploración del escenario.

El primer diseño balancea los trabajadores priorizando los mineros. De haber una refinera mantiene hasta 3 trabajadores recolectando gas siempre que haya más trabajadores minando para respetar las prioridades de los recursos⁸. Todos los demás trabajadores son mandados a minar siempre que haya cristales suficientes para ello. Los cristales se encuentran realizando analizando las cercanías de los trabajadores. Finalmente se analizan las rutas de los trabajadores mineros para, de haber encontrado algún cristal disponible más cercano a las estructuras, redirigir al trabajador a ese cristal y mejorar así el tiempo de minado.

Este protocolo se sirve de que los minerales, al igual que en el videojuego original, suelen aparecer en formaciones de ocho, de forma que al acabar un trabajador un mineral este pasa a minar uno cercano y se detiene de no encontrarlo⁹.

C.1.8. Generar soldados

La tarea de este protocolo es ordenar generar tropas, decidiendo de qué tipo y cuántas.

En la primera versión del diseño se trataron diversas fórmulas que mezclaban los distintos datos que se creían relevantes en el momento de la llamada al protocolo. El objetivo era calcular una probabilidad con la que se decidía si generar soldados o no. Si se optaba por generar soldados, la cantidad generada dependía de los generadores de tropas inactivos y de los recursos que el jugador artificial dispusiera, creando todas

⁸El cristal tiene mayor prioridad que el gas ya que se utiliza en todo y el gas solo en estructuras y unidades más avanzadas.

⁹https://starcraft.fandom.com/wiki/Minerals#Game_Effect (último acceso: 20/6/2023).

las que se pudiera permitir, pero con el máximo de un soldado por generador inactivo. Los datos que se tenían en cuenta para tomar la decisión eran los siguientes:

- El número de tropas. Con ello se intentaba englobar todos los escenarios bajo un límite normativo por el videojuego original que se consideraba demasiadas tropas y que por ello no se permitía superar. De forma que a más se acercaba el número de tropas a ese límite, menor era la probabilidad de generar más.
- El número de trabajadores. Este dato permitía en algunas implementaciones asegurar que se iban a generar soldados hasta igualar o superar el número de trabajadores, bajo la premisa de que las unidades trabajadoras tienen que estar protegidas antes de invertir en nada más.
- El número de trabajadores inactivos. Conforme avanzaba el estudio sobre Armygeddon, se dedujo que la generación de recursos (economía) es clave para ganar las partidas y, para fomentarla, se invertían las reservas en generar trabajadores ya que estos la aumentan. Es cuando aparecen “trabajadores inactivos” que no encuentran manera de aumentar la economía, ya sea porque no quedan minerales o bien no los encuentran, que se considera buen momento para gastar en tropas.

Para ejemplificar, la ecuación más relevante de este primer diseño fue la siguiente:

$$Se\ generan\ soldados = rand(100 + 1) < 100 * \frac{NT_{Libres}}{3} + 100 * \frac{NT_{Totales}}{NS}$$

Donde $rand(V)$ devuelve un entero aleatorio en el rango de $[0, V - 1]$, NT es el acrónimo de *Número de Trabajadores* y NS el de *Número de Soldados*. Hay que aclarar que esta ecuación tenía un caso especial en el que no se aplicaba, y es que siempre decidía generar soldados cuando no se tenía ninguno.

En cuanto al tipo de tropas generadas, también se escogía aleatoriamente con la misma probabilidad para cada caso siempre que se pudieran costear, de no ser así se generaban solo soldados básicos.

Con el segundo diseño se intentó hacer más simple la decisión, ya que al aprovechar en todos los casos los trabajadores estos dejaban de estar inactivos. Además, la relación entre el número de soldados y trabajadores no siempre era la apropiada, puesto que por ejemplo cuando son pocas unidades cualquier baja afectaba mucho a la decisión.

Se tuvo en cuenta solamente la cantidad de soldados básicos que en ese momento se podían generar, y que haya o no trabajadores. La idea es asumir la generación de soldados como la inversión final de los recursos, teniendo siempre recursos reservados para todo lo demás y gastando siempre que se pueda una parte para esta tarea.

C.1.9. Generar trabajadores

Este protocolo consiste en, si es necesario, ordenar la generación de un trabajador.

Se trata solo de generar un trabajador ya que generar más por vez requeriría de más de una estructura base y, aunque esto se permite en Armygeddon, el coste temporal y de cristal es muy elevado¹⁰. Además de que esto serviría para aumentar la velocidad con la que aumenta la generación de recursos, es decir, una aceleración de la *economía de los recursos* que la escala actual de las partidas no necesita.

Exceptuando el caso de que no haya trabajadores, que está claro que hay que la decisión tiene que ser generar trabajadores, se estima que la decisión debe de estar condicionada por el número de *trabajadores inactivos*. Este número indica mejor la situación de la mano de obra mejor que la cantidad de cristales siendo minados, o los propios recursos del jugador artificial, ya que puede por ejemplo haber muchos cristales con el acceso bloqueado o sobrar recursos porque no se están generando soldados.

Se encuentran pocas respuestas al *para qué* podrían ser útiles más trabajadores siendo que ya hay una cierta cantidad de ellos inactivos. Una posible respuesta sería para acelerar las construcciones que surgieran si se trata de un ejército terran, pero a partir de cinco trabajadores inactivos cuesta más crear un trabajador extra que el tiempo que se ahorraría en una construcción. Además tendría que ser una construcción de un centro de operaciones (anexo M.9.4), que tampoco se iba a usar pues ya se tiene uno sobre el que se está decidiendo si generar otro trabajador. Otra respuesta sería para un eventual descubrimiento de más cristales o géiser disponibles, pero también se descarta pues si no se han encontrado hasta ahora quiere decir que se hayan lejos y, para esos casos, conviene empezar mandando pocos trabajadores y más tropas por el peligro de emboscada.

C.1.10. Parámetros

Para conseguir realizar todos los requisitos mencionados sobre los jugadores artificiales, plasmados en su correspondiente diseño se tiene que contrastar de alguna manera cómo le está yendo la partida para en los momentos de decisión saber si optar por una estrategia más o menos agresiva. Es por eso que hilando con el requisito de “tener distintos perfiles de agentes artificiales” se implementa un rasgo del jugador artificial que es la agresividad. Con la agresividad se condicionan todas sus acciones, y las acciones a su vez, afectan a la agresividad del jugador artificial.

Teniendo en cuenta la agresividad se postulan los siguientes parámetros a la hora de

¹⁰Una base cuesta de cristal lo que cuestan de seis a ocho trabajadores. De tiempo cuesta de dos a cinco minutos, mientras que un trabajador solo 20 segundos.

utilizar un jugador artificial:

1. Color del ejército. Otorga información sobre qué ejército es, qué aliados tiene...
2. Raza. Para saber de qué mecánicas de raza distintivas dispone.
3. Velocidad de respuesta. No es un parámetro propio del jugador artificial, pero sí de las partidas para saber cada cuánto se invoca al jugador artificial para que genere órdenes. A mayor velocidad, menor es el tiempo entre ciclos de actualización.
4. Comportamiento. Define cómo reacciona el agente artificial a los estímulos. Por debajo, lo único que hace es definir una variación que afecta a la manera en la que reacciona el jugador artificial, desconfiando más en los estímulos que le fomentan la agresividad si es un comportamiento defensivo, y todo lo contrario para los comportamientos agresivos.
5. Tipo de objetivo. En función del tipo de objetivo de la facción integrante del jugador artificial, esta tendrá una versión u otra de algunos de sus protocolos.

C.1.11. Velocidad de respuesta

Para mayor naturalidad se añaden además “buenas” y “malas” versiones de cada agente artificial, pues al crear una partida se incrementa en una proporción aleatoria hasta en un 100 % el tiempo de espera entre ciclo y ciclo de actualización (definido por la velocidad de respuesta). Esto intenta imitar la sensación de los videojuegos con varios usuarios en cada partida con sistema de emparejamiento de rivales, donde los usuarios perciben que cada rival es distinto de los demás, tanto entre sí dentro de una partida, como entre partidas que comparten un mismo escenario. Estas son las horquillas temporales posibles (ver tabla C.2):

Velocidad de respuesta	Factor	Horquilla en ciclos	Horquilla en segundos
Lenta	x4	[60, 180]	[2, 6]
Algo rápida	x3	[45, 135]	[1.5, 4.5]
Rápida	x2	[30, 90]	[1, 3]
Muy rápida	x1	[15, 45]	[0.5, 1.5]

Tabla C.2: Posibles lapsos temporales entre actualizaciones del jugador artificial según su velocidad de respuesta.

Las horquillas temporales que definen el tiempo entre ciclos de actualización del jugador artificial se calculan con la siguiente fórmula:

$$Horquilla_{ciclos} = \left(\frac{FPS}{2} + rand(FPS + 1) \right) * factor$$

Donde:

- FPS es el número de ciclos por segundo del *bucle de juego* (sección 2.2.1), que es 30.
- $rand(x)$ una función que devuelve un valor generado aleatoriamente entre 0 y $x - 1$.
- $factor$ es el factor dependiente de la velocidad seleccionada, mostrado en la tabla C.2.

C.1.12. Comportamiento

En cuanto al comportamiento del jugador artificial y su relación con la agresividad, se entiende mejor de la siguiente manera:

Suponiendo a un jugador artificial al cual le toca realizar un ciclo de *Construir*, y cuyo resultado es satisfactorio pues ya tiene contruidos todos los edificios que estima oportunos y no tiene que construir más. Se simula un aumento en la agresividad del jugador artificial al recibir un estímulo positivo invocando a la función de aumento de agresividad, debido a la buena situación en infraestructura. En base a estímulos como este se altera la agresividad, y a su vez, la agresividad condiciona el tipo de órdenes que ejecuta el jugador artificial en los distintos protocolos. Esto es especialmente notorio en el protocolo de invasión, ya que al tener alta agresividad es más fácil que el jugador artificial cree múltiples invasiones, mientras que al tener poca agresividad puede ordenar a sus tropas retirarse aunque se tuviera la batalla ganada.

El funcionamiento sería el siguiente: cada jugador artificial con un valor que representa la agresividad (entre 0 y 100) tiene estímulos que se generan con cómo vaya la partida que aumentan o disminuyen la agresividad utilizando las dos funciones mencionadas. Estas funciones tienen un parámetro que indica con qué intensidad se afecta a la agresividad, pero independientemente del valor introducido, este se ve modificado por las variaciones mostradas en la tabla C.3. Además dichas funciones están limitadas:

Comportamiento	Disminuir agresividad(x)	Aumentar agresividad(x)
Muy defensiva	+2	-2
Defensiva	+1	-1
Equilibrada	+0	+0
Agresiva	-1	+1
Muy agresiva	-2	+2

Tabla C.3: Variaciones sobre el parámetro de las funciones de retroalimentación del jugador artificial en función de su comportamiento.

- Si se invoca a reducir agresividad, siempre se va a reducir aunque sea en una unidad solo si el valor de agresividad no termina por debajo de 0.
- Si se invoca a aumentar agresividad, siempre se va a aumentar aunque sea en una unidad solo si el valor de agresividad no termina por encima de 100.

Solo queda ver cómo se ha implementado cada protocolo y qué eventos le indican que todo va bien y que sea más agresivo, o por el contrario que debe adoptar tácticas más defensivas.

C.1.13. Implementación de los protocolos

En esta subsección se trata la implementación de los protocolos, haciendo especial énfasis en la retroalimentación sobre la situación de la partida y su impacto en la agresividad del jugador artificial.

Invadir

El protocolo *Invadir* (ver figura C.2) se puede resumir en tres partes:

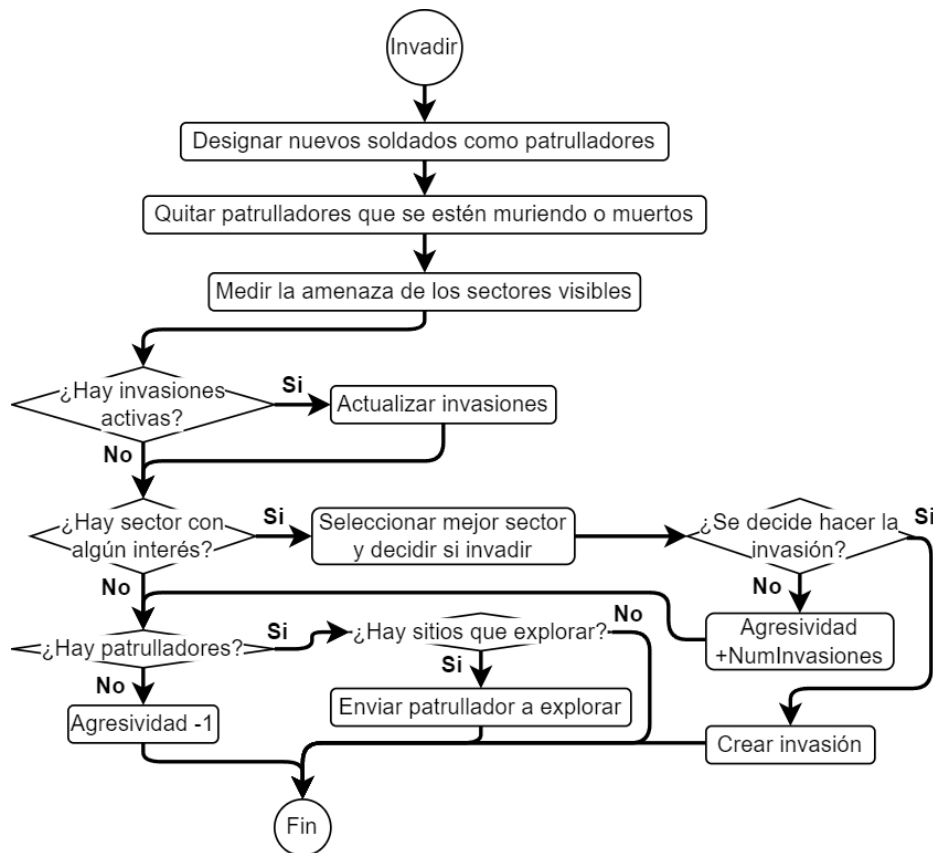


Figura C.2: Esquema de algoritmo de invasión.

1. Análisis general. Se comprueba el estado de las unidades y de la peligrosidad

del escenario según los sectores donde haya desbloqueado visibilidad alguna el jugador artificial.

2. Actualización externa. En esta parte se asignan nuevas órdenes a las invasiones actuales como cambiar su sector a atacar o, incluso retirar las invasiones. También se hace a nivel individual, se reasignan los soldados o se les ordena una retirada si se hayan en una situación crítica.
3. Actualización local. Aquí se decide si crear una nueva invasión, comprobando el estado de los patrulladores, y calculando una diferencia de poderes aceptable para movilizar el ejército necesario para invadir. En caso de no generar una invasión, se ordena una movilización de patrulladores para explorar, si es que queda algún sitio que explorar.

Hay que destacar dos conceptos que se tratan en la elección del mejor sector a invadir: el *interés* y la *amenaza*. La amenaza de un sector se mide con el número de entidades enemigas en dicho sector, ponderando cuatro veces cada soldado enemigo que lo que pondera un trabajador, estructura, o refinería enemiga. El sector con más interés¹¹ que no esté siendo invadido ya por el propio jugador artificial es el que se escoge para crear una invasión. Por ejemplo, para un sector R el interés se calcularía así:

$$Interes_R = distancia(sector_base, sector_R) * (amenaza_R + perdidas_R)$$

Donde:

- $distancia(sector_base, sector_R)$ es la distancia entre el sector donde se halle una estructura base del jugador artificial y el sector R en cuestión.
- $amenaza_R$ es el valor de la amenaza conocida por el jugador artificial en el sector R.
- $perdidas_R$ es el total de las pérdidas acumuladas en invasiones en dicho sector R.

En la actualización de las invasiones (ver figura C.3), existe la posibilidad de desplazar el sector objetivo de la invasión a otro sector vecino¹² cuando la amenaza en el sector objetivo desaparece. Esta mecánica existe para evitar retirar las tropas hasta la base tras terminar una invasión y luego volver a enviarlas en una eventual nueva invasión que puede ser en el mismo sector (si vuelven enemigos) o en uno vecino.

Hay que destacar que aquí se aprovecha mucho el jugador artificial de la mecánica

¹¹Tiene que tener amenaza para poder ser considerado un objetivo a invadir. Si se tratase de un sector interesante para el jugador artificial que no contiene amenaza alguna, no sería necesario tratarlo con una invasión.

¹²Un sector vecino es aquel que está a una distancia optimizada de un solo sector.

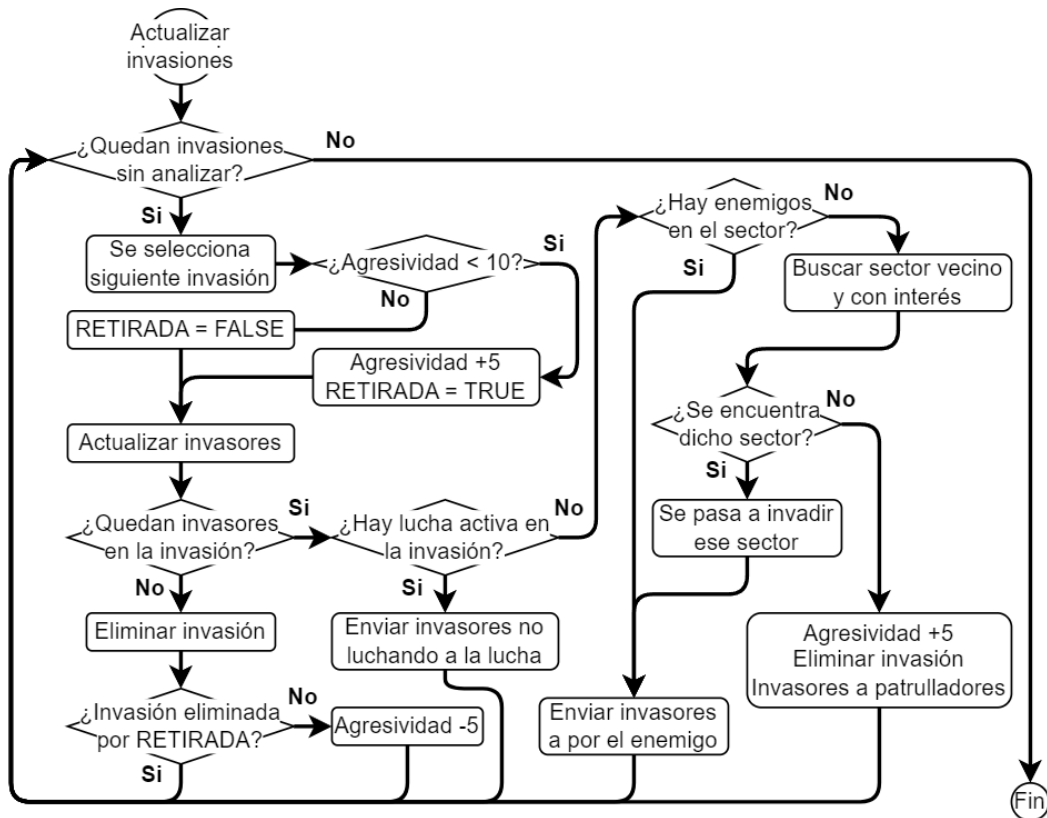


Figura C.3: Esquema de algoritmo de actualizar invasiones.

del *autoataque* (explicada en el anexo M.6.9), de forma que al eliminar un objetivo las tropas pueden desplazarse y atacar a enemigos en otros sectores cercanos, ya que no se comprueba si la lucha es en el sector, solo si los invasores están luchando. De esta forma, el combate inicial siempre es en el sector indicado, y si algún invasor está sin un objetivo, se le otorga uno dentro de dicho sector o uno que esté combatiendo un invasor distinto.

Defender

La defensa (ver figura C.4) tiene una rutina previa que trata si se es zerg, para gestionar todas las larvas y soberanos que están dificultando el paso y para reubicarlas. En resumen comprueba si están pegadas a una estructura y de ser así les ordena un *movimiento de invasor* (anexo C.4) de longitud uno.

Este protocolo tiene los siguientes pasos:

1. Recuento de patrulladores. Ver que no se ha muerto ningún patrullador desde la anterior actualización.
2. Refuerzo de agresividad. Si hay invasiones activas y hay patrulladores evalúa dos condiciones:

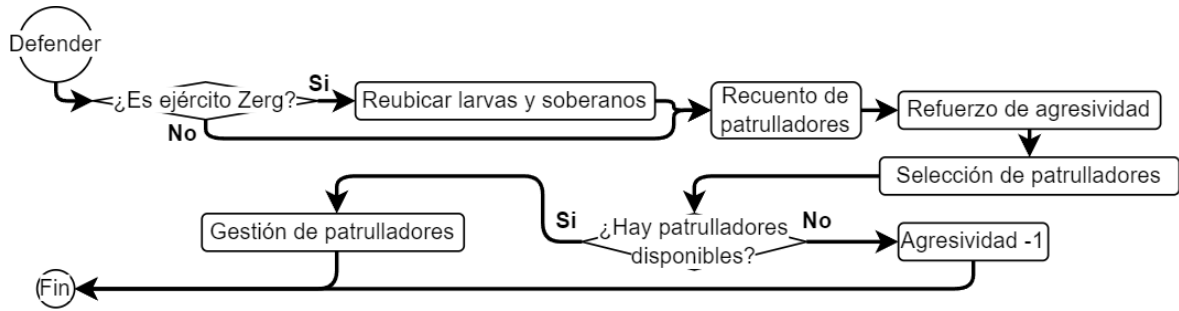


Figura C.4: Esquema de algoritmo de Defender.

- Si hay más patrulladores que invasores se aumenta la agresividad en cuantas veces más haya patrulladores que invasores.
 - Si hay más de cuatro veces el número de invasores que de patrulladores, se disminuye la agresividad en uno.
3. Selección de patrulladores. Todos los patrulladores que no estén luchando tienen una probabilidad¹³ de $\frac{1}{3}$ de analizar su entorno cercano y atacar al enemigo que vean. Los que no estén luchando tras esto, se seleccionan como patrulladores disponibles.
 4. Gestión de patrulladores disponibles. Si no hay patrulladores disponibles se disminuye la agresividad en uno. A partir de ahí se siguen unas prioridades:
 - a) Defensa indirecta de estructuras. Si se trata de una defensa existe una probabilidad de ignorarla.
 - b) Defensa de trabajadores. Un trabajador puede estar de exploración, es por eso que tiene que haber suficiente agresividad para realizar esta defensa.
 - c) Defensa indirecta de refineras. Esta defensa se realiza aquí, pues una refinería es más volátil y menos importante que los trabajadores o las estructuras.
 - d) Defensa de patrulladores. Requiere cierta agresividad por lo mismo que por los trabajadores, ya que los patrulladores pueden estar lejos y es arriesgado mandar tropas hasta allí.
 - e) Defensa de las cercanías. Se aumenta la agresividad si se ha llegado hasta aquí, pues se considera que hay buena situación de defensa al llegar a esta última sección. En esta sección hay una probabilidad de regresar a los edificios por si un patrullador patrulla demasiado lejos.

¹³Se usa una probabilidad para simular que los patrulladores pueden estar despistados.

Construir

Esta implementación (ver figura C.5) no tiene en cuenta problemas como falta de espacio a la hora de construir estructuras en el escenario, se asume que los escenarios están bien diseñados y en caso de no ser así no se trata la agresividad de ninguna forma pues no se sabe qué sería lo adecuado.

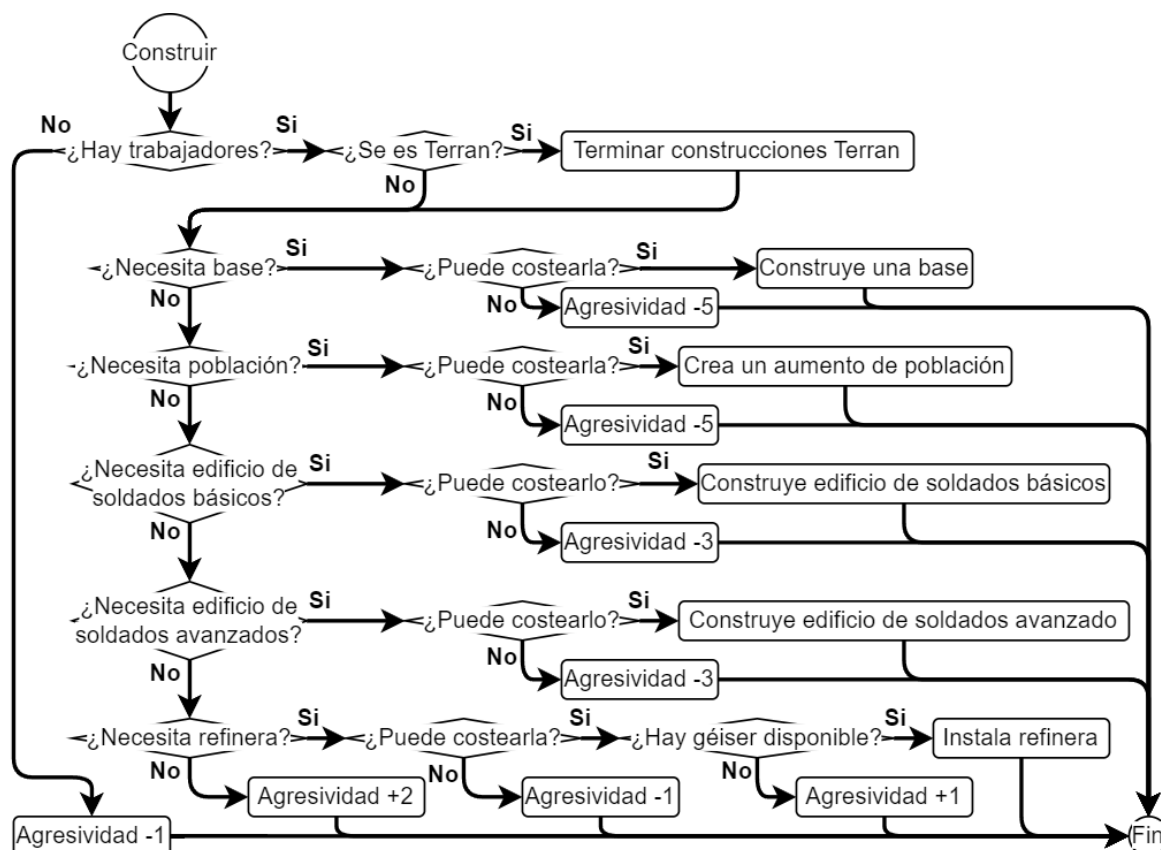


Figura C.5: Esquema de algoritmo de Construir.

Trabajar

En esta implementación (ver figura C.6) se asignan de haber una refinería y trabajadores suficientes hasta tres¹⁴ de ellos para recolectar si no los hay. Se escanean las cercanías de todos los trabajadores para encontrar minerales visibles que estén disponibles.

¹⁴Considerado tres el número perfecto de trabajadores asignados para recolectar gas de una refinería y que esté en funcionamiento el mayor tiempo posible, ya que es el cuello de botella en la recolección debido a su elevado tiempo. Se intenta asegurar que para cuando un trabajador salga de la refinería haya otro que haya devuelto el gas a la base y esté esperando para poder entrar. <https://starcraft.fandom.com/wiki/Refinery#StarCraft> (último acceso: 20/6/2023).

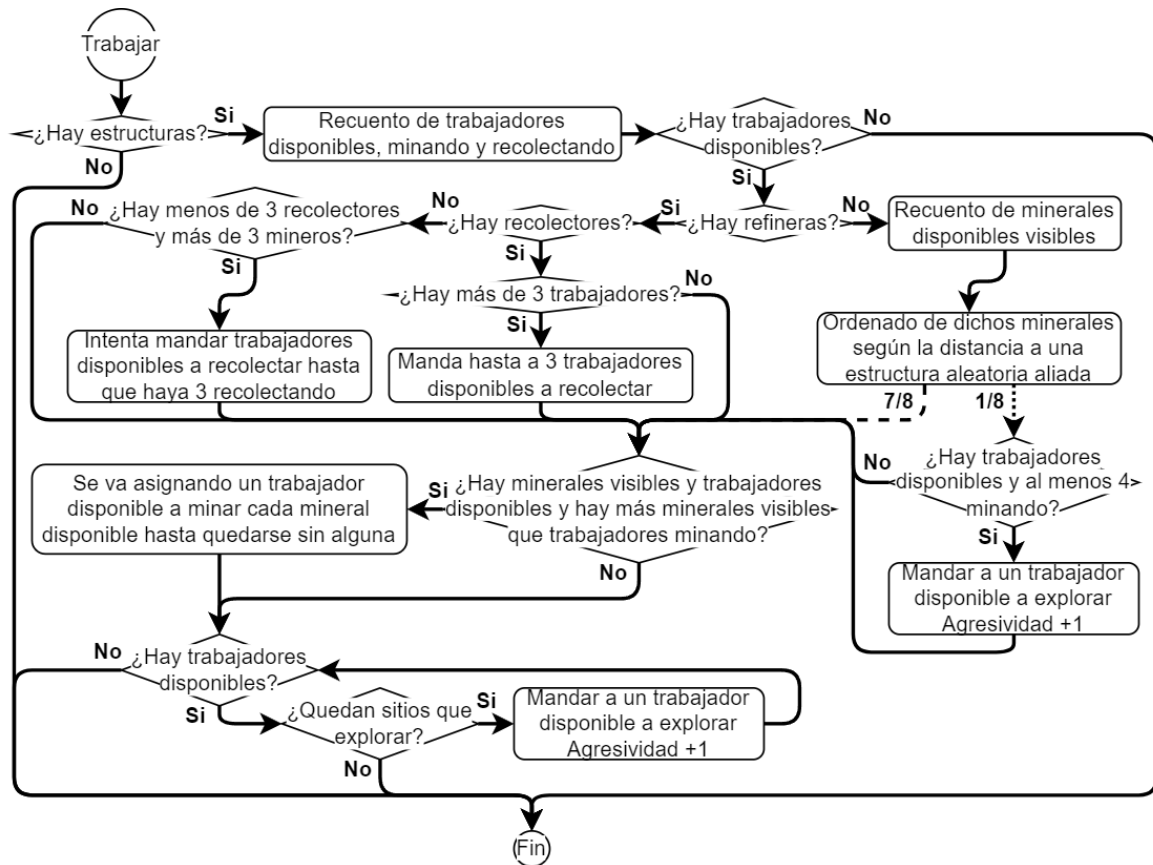


Figura C.6: Esquema de algoritmo de Trabajar.

Generar soldados

A la hora de generar soldados se tiene en cuenta solo los soldados costeables, es decir, los que el jugador en ese momento con los recursos actuales se podría permitir. En caso de decidir generar soldados y de disponer de la infraestructura y recursos necesarios para generar un soldado avanzado, existe un 50 % de probabilidades de que se genere de este tipo. Lo único que hay que destacar del algoritmo son los refuerzos de agresividad:

- Se incrementa la agresividad si aún quedan soldados por generar pero todos los generadores de soldados están ocupados.
- Se disminuye la agresividad si se decide que no hay que intentar generar ningún soldado porque no puede permitírselos.

Generar trabajadores

En este protocolo se siguen los siguientes pasos para decidir si se genera un trabajador o no:

1. Recuento de trabajadores disponibles. Se cuentan cuántos trabajadores están en estado Quieta.

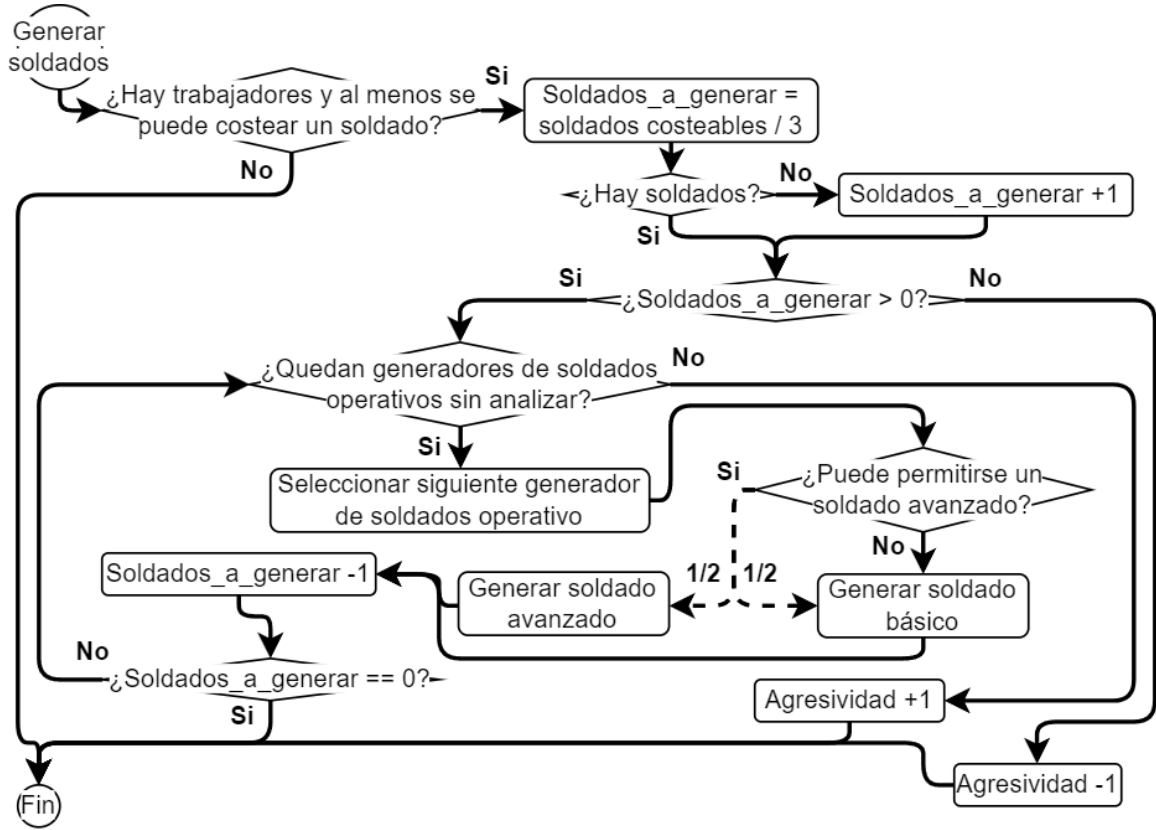


Figura C.7: Esquema de algoritmo de Generar Soldados.

2. Decisión de generar o no un trabajador. Existen dos casos:

- a) No hay trabajador alguno. Se decide generar un trabajador.
- b) Hay trabajadores ya. Se calcula la decisión de generar un nuevo trabajador con la expresión:

$$GenerarTrabajador = rand(100 + 1) < \frac{100}{(NT_{Disp} * NT_{Disp} + 1)}$$

donde:

- NT_{Disp} es el número de trabajadores disponibles calculado antes.

C.1.14. Variantes según el objetivo

Los jugadores artificiales tienen distintos perfiles no solo por la configuración de sus parámetros velocidad y comportamiento, sino también según el tipo de objetivo que sus respectivas facciones tengan en partida.

Exterminio

Para no definir protocolos completamente distintos para los cuatro objetivos posibles en partida, se parte del objetivo más genérico que es el de *Exterminio* y se realizan

modificaciones para generar las otras versiones. A continuación se detallan dichos protocolos genéricos usados en *Exterminio* y después se comentan las variaciones que hayan sufrido o no para los otros tipos de objetivos.

Supervivencia

De todos los protocolos base solo se sobrescribe el protocolo de defensa, al ser el más relevante para este objetivo, con los siguientes cambios:

- Se elimina el paso de refuerzo de agresividad. Siguen estando los cambios de agresividad en otras secciones como en la gestión de patrulladores.
- En la selección de patrulladores los patrulladores que estaban luchando se dejaban inalterados. Ahora se les ordena retirarse a un edificio aleatorio aliado si no tienen ninguno cercano.
- El refuerzo negativo de agresividad si no quedan patrulladores disponibles antes de la gestión pasa de ser de uno a cinco unidades.
- La defensa de los trabajadores pasa analizarse al final, justo antes de la defensa de las cercanías, y no en segundo puesto.

Conquista

Para esta variante se altera el protocolo Invadir, concretamente:

- A la hora de seleccionar el mejor sector a invadir, puede decidirse invadir un sector sin amenaza si se trata del sector que tiene la zona de conquista objetivo.
- Al calcular el interés por un sector R se utiliza la siguiente fórmula:

$$Interes_R = distancia(sector_base, sector_R) * (amenaza_R + perdidas_R) \\ * (distancia(sector_objetivo, sector_R) + 1)$$

donde:

- *sector_objetivo* es el sector en el que está la zona de conquista objetivo.
- Al actualizar el objetivo de una invasión hacia otro sector, no se cambia de sector en el caso de estar en el sector objetivo. Lo que se hace es enviar todos los invasores a la zona a conquistar.
- Cuando se crea una invasión en el sector objetivo y este sector no tiene amenaza, envía toda la invasión directa a la zona a conquistar.

Investigación

En esta variante se alteran los protocolos *Generar Soldados* y *Construir* de la siguiente manera:

- A la hora de generar un soldado avanzado se evalúa que se puedan permitir cuatro soldados y no solo uno.
- Se quitan todos los refuerzos de agresividad de si se podía o no haber generado la cantidad de soldados calculada.
- En la construcción se quita el refuerzo de agresividad negativo inicial si no hay trabajadores.
- Además, en la construcción se añade una rutina inicial que intenta mejorar todas las maravillas construidas que aun no estén maximizadas.
- Por último, en la construcción se añade al final una prioridad de construcción para construir también con el protocolo las maravillas que sean necesarias para lograr el objetivo.

C.1.15. Bolsa de protocolos

Finalmente, para respetar el principio básico de la *imprevisibilidad* los protocolos que vaya realizando el jugador artificial no tienen que seguir ningún orden claro que los jugadores humanos puedan descifrar. Tampoco se pueden escoger aleatoriamente, ya que eso es susceptible de inanición. Por lo que el diseño a seguir es de una “bolsa de protocolos con pesos” de forma que en cada actualización ocurra lo siguiente (ver figura C.8):

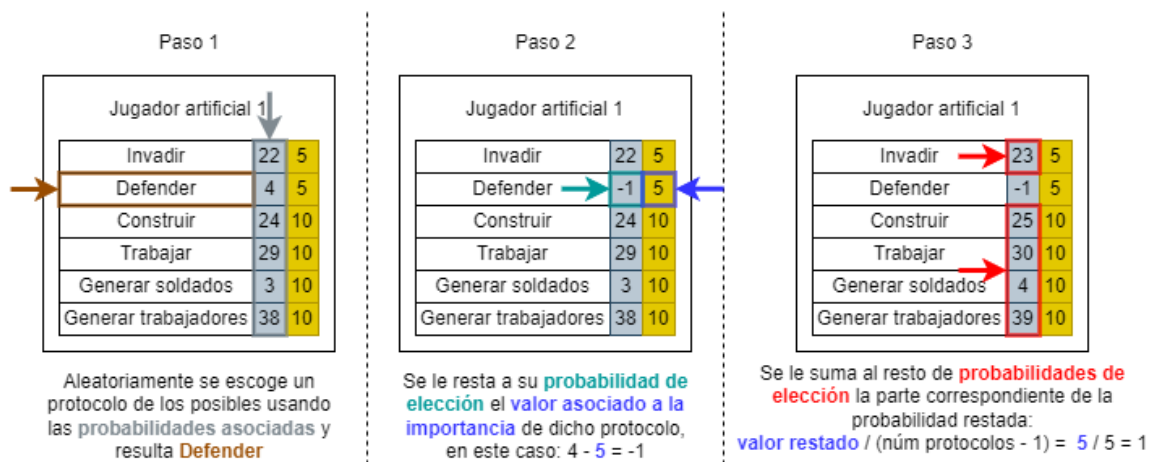


Figura C.8: Elección y reequilibrio de los protocolos del jugador artificial en una actualización.

1. Se escoge aleatoriamente¹⁵ un protocolo de los seis posibles teniendo en cuenta unas probabilidades asociadas a cada uno¹⁶.
2. Se resta un valor a la probabilidad del protocolo escogido que dependa de la frecuencia con la que se requiera el protocolo, siendo dicho valor menor a más importante sea el protocolo. En Armygeddon se estima que invadir y defender requieren el doble de atención que cualquier otro protocolo, y por eso cuando se escogen dichos protocolos se les reequilibran con menor intensidad.
3. El valor restado en el paso anterior se divide por el número de protocolos menos uno y, se suma el resultado a todas las demás probabilidades para que la probabilidad restada en el paso ‘2’ no desaparezca, sino que se traspase a los demás protocolos.
4. Se realiza el protocolo escogido y en su caso se procesan órdenes sobre el escenario.

Se garantiza así que no habrá inanición y que no se pueda adivinar qué protocolo va a seguir el jugador artificial después.

C.2. Ejército neutral

El ejército neutral (de color **gris oscuro**) es un ejército especial que no tiene objetivos, que no puede ser jugado, que no progresa y que no puede ser fijado como objetivo para ganar. Su única funcionalidad (además de decorativa) es perjudicar al jugador que se tope con el, ya que no está aliado bajo ninguna facción. Puede haber entidades del ejército neutral de la misma manera que de cualquier otro ejército (cuenta como un ejército más en el editor de escenarios), esto es, unidades, estructuras y refineras.

Las estructuras y las refineras neutrales no tienen ningún tipo de actividad. Actúan de obstáculos destruibles. Las unidades del ejército neutral conservan una posición dada desde el principio de la partida y actúan en función de unos estímulos.

Dado que este ejército tiene una actualización periódica cada ciclo, para evitar un efecto de “mente colmena” en el que todas las tropas actúan de vez como si compartieran los sentidos, estas se dividen equitativa y aleatoriamente en ocho particiones. De esta forma en cada actualización se actualiza una partición, rotando la que toca

¹⁵Cuando se habla en este documento de generar valores aleatoriamente se entiende con *pseudo-aleatoriedad*, ya que estos valores se generan utilizando una semilla y es solo aleatoriedad aparente https://es.wikipedia.org/wiki/Generador_de_n%C3%BAmeros_pseudoaleatorios (último acceso: 13/6/2023).

¹⁶En la figura ejemplo C.8 el conjunto de probabilidades suma 120, por lo que se generaría un número aleatoriamente entre 120 valores y por orden se evalúa a qué protocolo corresponde. En este caso serían de los valores de 0 al 21 Invadir, de 22 al 25 Defender, etc.

actualizar cada vez¹⁷.

En cada actualización, los miembros de cada partición, que pueden ser o bien soldados o bien trabajadores, se someten a unas comprobaciones en función de su estado, realizando una acción en su caso y pasando a analizar la siguiente unidad hasta que se analice la partición de dicho ciclo (ver figura C.9).

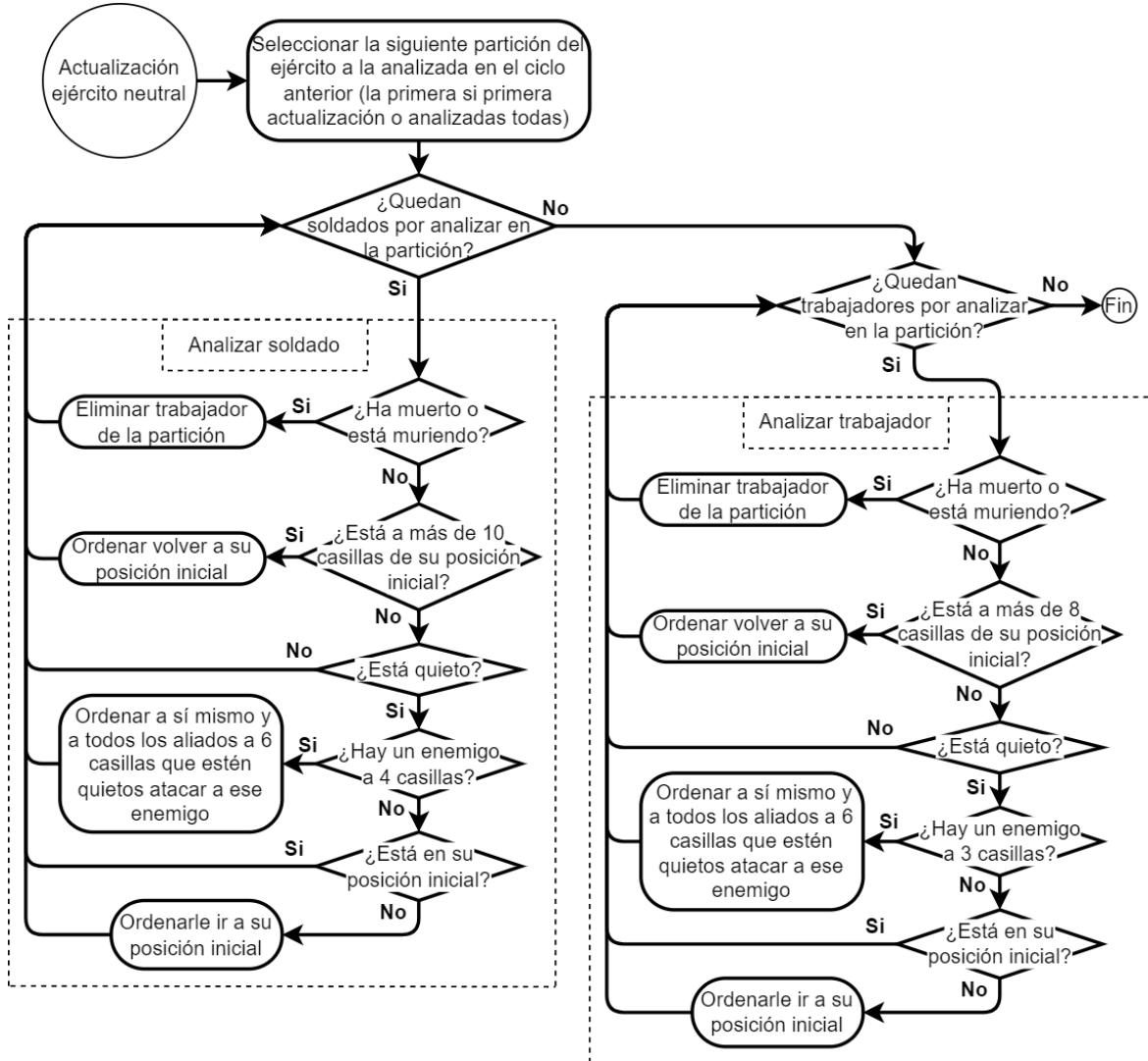


Figura C.9: Diagrama de flujo de la actualización del ejército neutral.

Hay que destacar el tipo de distancia¹⁸ utilizada en cada caso:

- Para saber si la unidad está demasiado lejos de la posición inicial se usa la distancia real, ya que esta es más precisa y costosa y se calcula como mucho una vez por unidad y por ciclo.

¹⁷Primero la primera partición, luego la segunda, y así sucesivamente, hasta la última y vuelta a empezar. Esto se repite hasta el final de la partida.

¹⁸En el anexo H.1 se explican los dos sistemas de distancia utilizados.

- Para *saber si hay un enemigo cercano* se utiliza la distancia optimizada, ya que hay que evaluar todas las casillas cercanas y eso sería muy costoso con distancia real.
- Para *avisar a los aliados cercanos* se utiliza la distancia real, si bien se utiliza sólo sobre las demás tropas neutrales, estas pueden ser muchas y ser costoso igualmente. La razón por la que se usa la distancia real en este caso es para evitar la frustración del usuario, ya que al llamar a los refuerzos se apreciaría que el juego no está siendo justo según la disposición de las tropas. Cosa que no ocurre en la detección de enemigos, al estar esta postergada y provocar la sensación de “guardia despistado”.

Este algoritmo se puede resumir en las siguientes características a las tropas neutrales:

1. Simulación de despiste. Como se acaba de comentar, al estar fracturada la evaluación de las tropas, los guardias no detectan instantáneamente a un foráneo en su zona de vigilancia, sino que puede tardar hasta siete ciclos. Esto en el usuario se traslada como que no hay un área delimitada fija en la que es detectado, sino que es “más humano” y que hay más factores involucrados.
2. Prioridad de guardia. Al igual que un personal de seguridad privado, que su prioridad suele ser proteger una zona por encima de atrapar a un transgresor, las tropas no se alejan demasiado de su posición asignada, volviendo a dicha posición incluso en mitad de un combate. Esto, entre otras cosas, impide que un jugador pueda separar tropas neutrales y llevarlas hacia una emboscada.
3. Aviso a aliados limitada. Se trata de un ejército neutral, que puede estar completamente dispuesto en un mismo asentamiento o no, y no se quiere que tenga inteligencia conjunta como ocurre en el resto de jugadores artificiales. Por lo que el aviso para pedir refuerzos al encontrar enemigos cercanos solo llegará a los aliados cercanos (como si de un grito de ayuda se tratase), y solo acudirán a este los aliados que estén disponibles (sin asignaciones por parte de una mente conjunta en función de la prioridad de la batalla).
4. Termina las peleas sin tener que cambiar de objetivo en una actualización posterior. Esto ocurre ya que las tropas neutrales también tienen la mecánica de *autoataque* (detallado en el anexo M.6.9), que consiste en atacar a un objetivo enemigo cercano al matar a su objetivo actual. De no estar esta mecánica, las tropas neutrales al matar a un invasor se quedarían quietas hasta que les tocara de nuevo una actualización.

C.3. Búsqueda de caminos

La búsqueda de caminos (ver figura C.10) está preparada (aunque no se hubiera terminado de implementar) para unidades de tamaños mayores, teniendo que comprobar varias casillas más en cada paso.

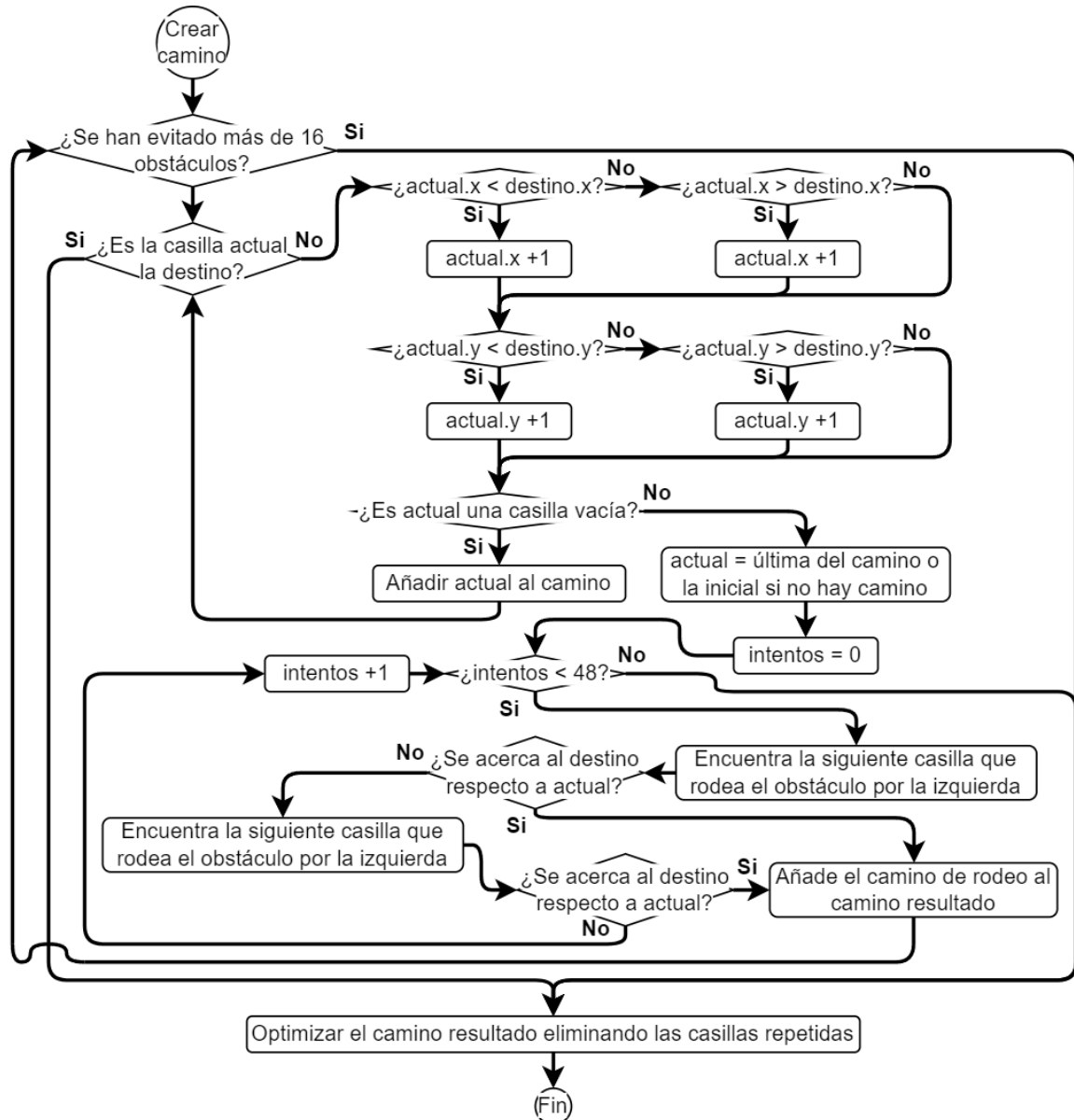


Figura C.10: Diagrama de flujo de la búsqueda de caminos.

El algoritmo utilizado trata de acercarse lo más directamente posible teniendo en cuenta que solo hay ocho direcciones de movimiento. En cuanto encuentra un obstáculo, se intenta rodearlo por ambos lados, y toma el camino que más rápidamente le aproxime algo al objetivo. Esto se repite una determinada cantidad de obstáculos a sortear y en cada obstáculo una determinada cantidad de intentos.

La peculiaridad del algoritmo es que no actúa omniscientemente y que tiene la me-

moria limitada. Es decir, cuando se utiliza es para mandar a una unidad desplazarse hacia una casilla objetivo, la cual no conoce el escenario y que no va a recordar todas las indicaciones que se le den. Este algoritmo simula que la unidad va acercándose en la dirección que le lleva hacia al objetivo hasta toparse con obstáculos y en ese momento ver por dónde continuar, pero está limitado, concretamente:

1. La unidad no recuerda más allá de 16 obstáculos, si hay más de 16 obstáculos en el camino, la unidad no tiene un camino completo y a medida este se agote o aparezca una entidad que le bloquee el camino, la unidad volverá a calcular un camino.
2. Si el obstáculo es demasiado grande, más allá de 48 casillas por ambos lados sin conseguir mejoría, la unidad tampoco va a saber qué camino tomar para rodearlo.

Por último destacar que se consideró optimizar aún más los caminos calculados teniendo en cuenta por dónde se esquivan los obstáculos, pero se descartó para simular que las unidades utilizan los obstáculos que se van topando para orientarse y que por ello se acercan a estos.

C.4. Inteligencia del antiguo invasor

En el primer diseño del jugador artificial se establece la primera versión del “invasor”, de la cual se ha preservado parte de la funcionalidad en las versiones posteriores. Un invasor es un soldado, de un jugador artificial, que actúa como si tuviera inteligencia propia.

Un invasor recorría el escenario por su cuenta, de manera aleatoria (ver figura C.11), y atacando a los enemigos que se fuese cruzando. Esto permitía al jugador artificial evitar tener que gestionar a sus tropas, ya que estas solas actuaban de *mercenarios errantes* enfrentándose a los enemigos de manera autónoma. Era un diseño que no tenía en cuenta la geografía o los enemigos del escenario más que la visión más inmediata de cada invasor, y lo que veía cada invasor solo se tenía en cuenta para sí mismo.

Lo destacable es el sistema de movimiento del invasor, que permitía de manera eficiente acabar explorando todo el escenario simulando un movimiento de *patrullador* con cierta inteligencia. Como se aprecia en la figura C.11 siempre existe una probabilidad de que el invasor altere su rumbo¹⁹, esto hace que no solo cuando se topa con un obstáculo decida explorar otros caminos. También hay aleatoriedad en la longitud del movimiento, permitiendo a la unidad llegar a cualquier casilla del escenario al ser esta

¹⁹Es decir, que altere la dirección en la que se está desplazando, ver las direcciones de movimiento de juego en la página del manual M.6.13.

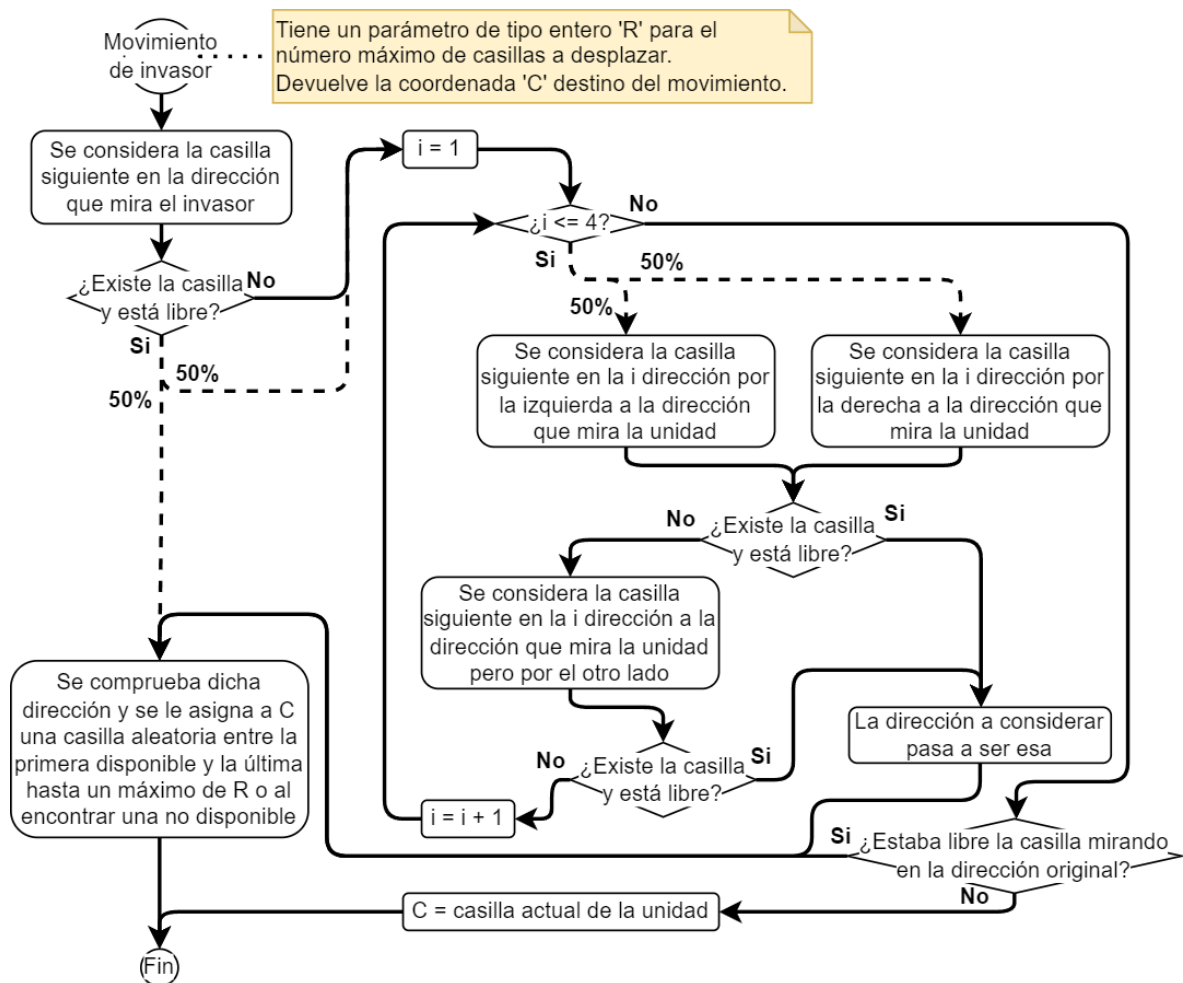


Figura C.11: Diagrama de flujo de la generación de un movimiento de invasor.

capaz de girar sin avanzar apenas aun sin presencia de obstáculos.

Este sistema de movimiento es útil para conseguir reubicar las unidades que están patrullando o que están mal posicionadas bloqueando el acceso a estructuras, ya que produce movimientos naturales e independientes de otras unidades.

Más adelante, el *invasor* pasó a obedecer a la inteligencia colectiva del jugador artificial y a estar gestionado en las invasiones de manera más compleja, como se ha mencionado en los anexos C.1.4 y C.1.13.

Anexos D

Escenarios de ejemplo

Todos los datos obtenidos en Armygeddon, así como para balancear el poder entre razas (ver manual M.6.5) o para desarrollar más las características de los jugadores artificiales, surgen de estudios sobre los escenarios de ejemplo.

Estos escenarios se desarrollaron para contrastar los distintos modos de juego, diferentes temáticas, y en su caso, representar batallas relevantes desde el punto de vista de la estrategia militar.

D.1. StarCraft

El escenario “StarCraft” (ver figura D.1) intenta abstraer la esencia de jugar una partida en el propio videojuego original StarCraft. Sin casillas sin declarar (explicadas en el manual M.10.2), con agrupaciones de minerales de ocho, géiseres por ejército a cierta distancia, y con el uso de obstáculos como elemento balanceador.

La principal tarea de este escenario es demostrar cierto equilibrio entre las razas:

- Todo el escenario está dividido conceptualmente en una matriz de nueve celdas cuadradas que están diseñadas para ser el espacio inicial de cada ejército. Distinguiendo a su vez tres tipos de celdas en función de la distribución de sus elementos y que responden al número de celdas vecinas colindantes (ver figura D.2):
 - Celdas de las esquinas. Al estar en las esquinas solo tienen tres celdas colindantes. Aunque se puede acceder a la celda desde las tres celdas colindantes, las entradas están muy próximas, por lo que para compensar esa defensa natural los recursos se disponen más dispersos que en las otras celdas.
 - Celdas con solo un lado bloqueado. Tienen cinco celdas colindantes. Tienen

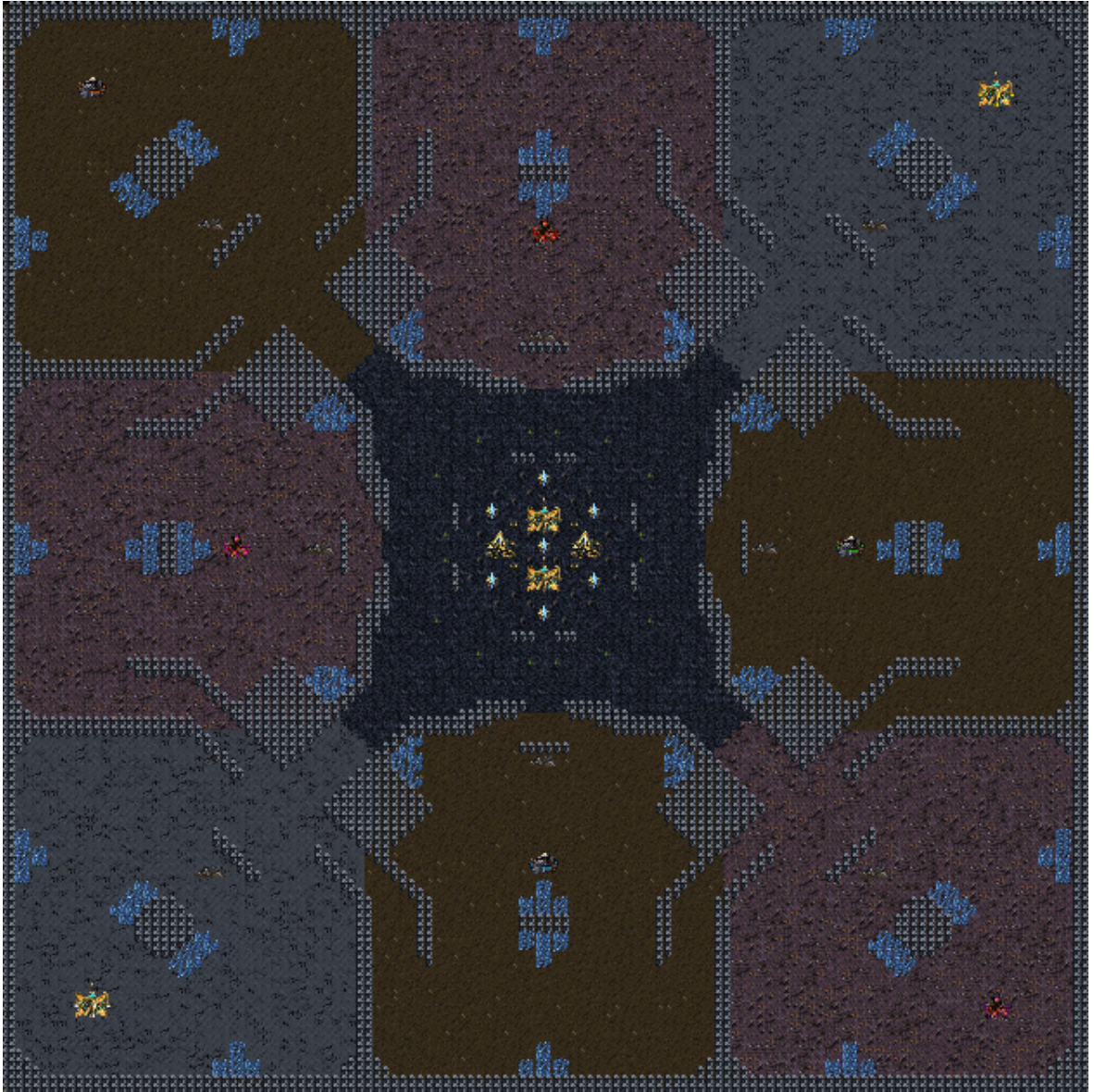


Figura D.1: Escenario de ejemplo ‘StarCraft’.

tres accesos separados, lo cual facilita que la celda sea flanqueada. A cambio, los recursos se encuentran más concentrados y cercanos a la posición inicial del ejército en estas celdas.

- Celda central. Tiene celdas colindantes en todas las direcciones. No posee recursos ya que está designada para el ejército neutral y este ya comienza la partida desarrollado. Tiene acceso a las otras ocho celdas.
- El escenario es prácticamente simétrico respecto del centro en todos los ejes. Con ligeras variaciones ya que no se pueden rotar las estructuras de Armygeddon¹ y muchos de estos son irregulares.
- Las razas de los ejércitos están repartidas equitativamente. De forma que en cada fila o columna de la matriz se encuentren las tres razas.

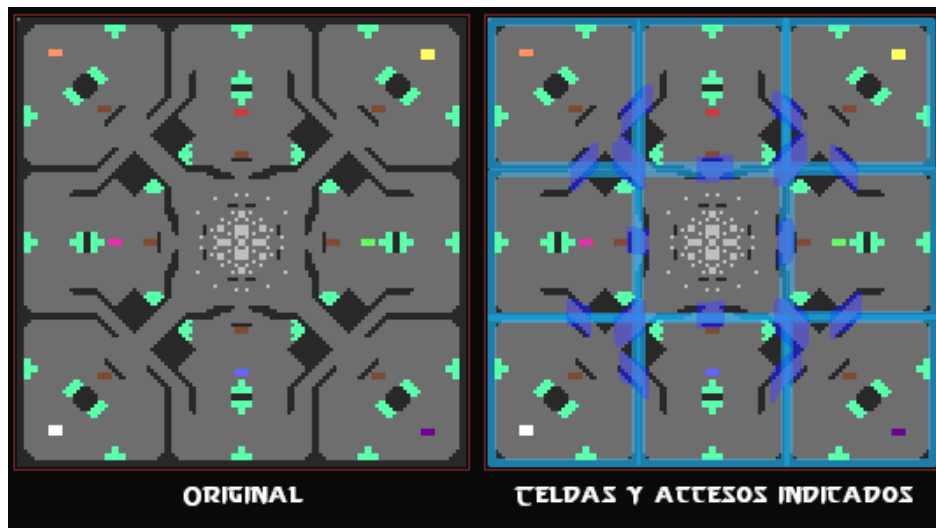


Figura D.2: Estado inicial del escenario ‘StarCraft’ con las celdas y accesos indicados.

D.2. HelmAbyss

Este escenario (ver figura D.3) está ambientado en “La batalla del abismo de Helm” o también conocida como “La batalla de los 5 ejércitos”² de *El Hobbit*³.

Sirve de escenario abierto de *supervivencia contra conquista* donde son varios los puntos en los que se puede librar la batalla. En una facción están los “Defensores”,

¹Todas las entidades de Armygeddon tienen gráficos en 2D pero las unidades se distinguen al tener gráficos hacia todas las direcciones para aparentar poder girar sobre sí mismas. https://en.wikipedia.org/wiki/Isometric_video_game_graphics (último acceso: 23/8/2023).

²https://es.wikipedia.org/wiki/Batalla_de_los_Cinco_Ejrcitos (último acceso: 12/8/2023).

³https://es.wikipedia.org/wiki/El_hobbit (último acceso: 12/8/2023).



Figura D.3: Escenario de ejemplo ‘HelmAbyss’.

compuestos de zerg, protoss y terran haciendo las veces de *Enanos*, *Elfos* y *Humanos*; mientras que por otro lado están los “Invasores”, compuestos de protoss y zerg haciendo de *Trasgos* y *Wargos* respectivamente. Ver figura D.4.

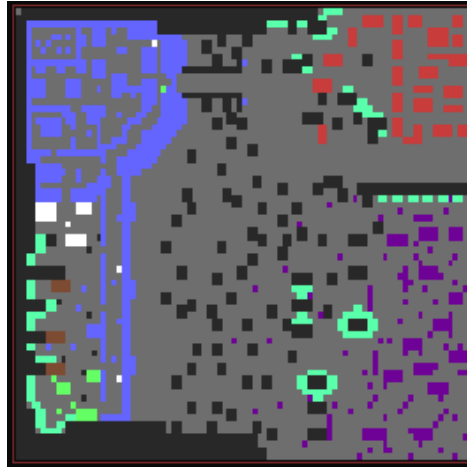


Figura D.4: Estado inicial del escenario ‘HelmAbyss’.

Todos los ejércitos de los Defensores tienen tropas iniciales alejadas para ser conscientes de la situación defensiva en la que están. Tanto el ejército blanco como el verde tienen tropas desperdigadas por el castillo para obtener visión de este, además del ejército azul que tiene visión de dos *guardias* en las afueras de la entrada principal del castillo para poder ver los enemigos venir.

El objetivo de los Defensores es sobrevivir 20 minutos, y el de los Invasores conquistar “la ciudadela”: la zona más custodiada por los Defensores (marcada con rojo en la esquina superior izquierda en la figura D.5). Como se aprecia en la figura, la parte más fortificada es la más cercana a la ciudadela, mientras que la zona de los recursos (muros inferiores) apenas tiene vigilancia y grosor en los muros. Es destacable que se ha intentado emular una sensación de altura con el grosor de los muros y el cambio de baldosas en el diseño de las murallas. La muralla más resistente es la que conecta con el puente, ya que es la externa y que separa el interior del exterior del castillo. El resto de las murallas son menos resistentes o tienen entradas para permitir el movimiento de los Defensores. En cuanto al puente, se intenta emular la altura de este colocando obstáculos en ambos lados.

D.3. Thermopylae

Este escenario (ver figura D.6) se crea en referencia a “La batalla de las Termópilas”⁴ de los ejércitos de las *polis* (ciudades-estado griegas), con los espartanos y los atenienses a la cabeza, defendiéndose del Imperio aqueménida (primer imperio persa). Su relevan-

⁴https://es.wikipedia.org/wiki/Batalla_de_las_Termopilas (último acceso: 13/8/2023).

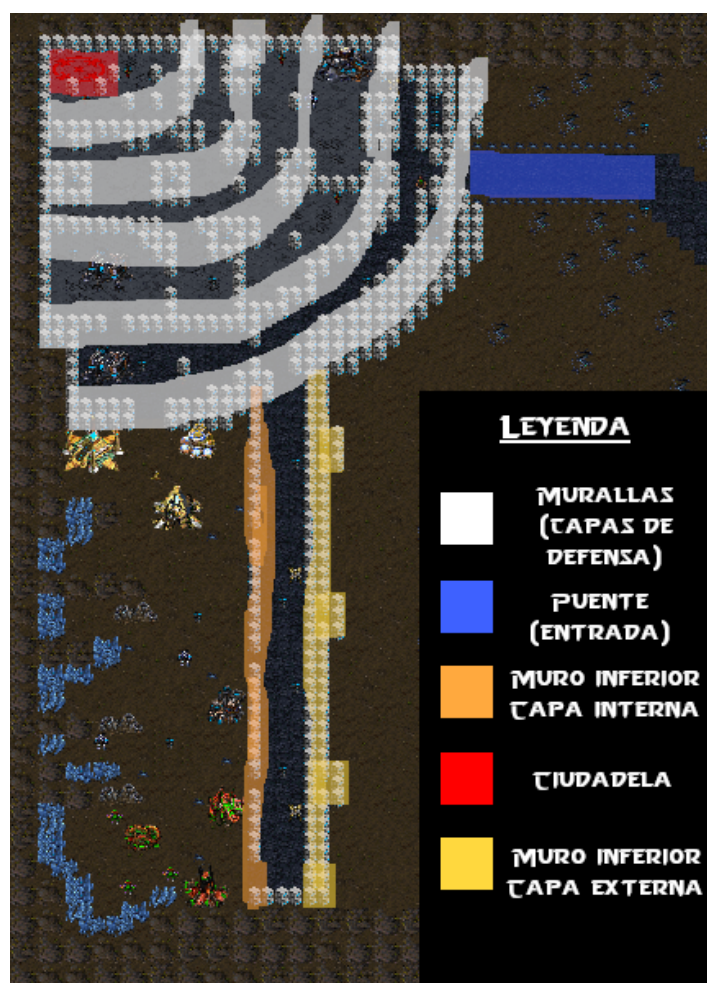


Figura D.5: Fortaleza de 'HelmAbyss' señalizada.

cia histórica es al reflejo de la alta rentabilidad que unos pocos soldados con buenas armas y un buen entrenamiento en un escenario estrecho acorde (se defendían desde un desfiladero infranqueable) puede ejercer contra un ejército mucho más numeroso⁵.

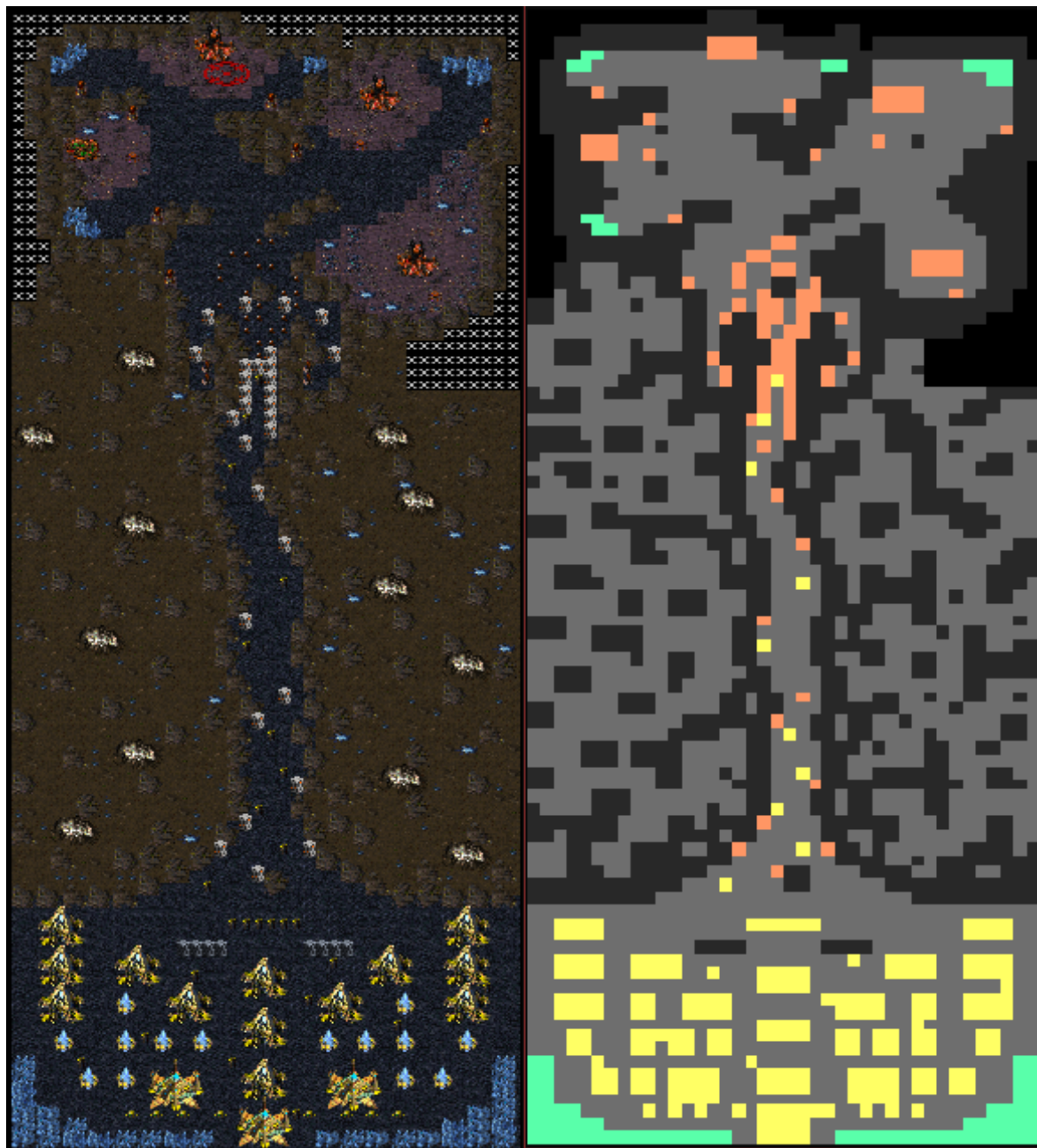


Figura D.6: Escenario de ejemplo ‘Thermopylae’ (izquierda) y estado inicial (derecha).

Se trata de otro escenario de dos facciones enfrentadas en el que los “Griegos” (situados en la parte norte del escenario) tienen como objetivo sobrevivir 15 minutos,

⁵“Guerrero persa: *Nuestras flechas taparán el sol.*

Rey Leónidas: *Mejor, así lucharemos a la sombra.*”

Diálogo de *300*, película ambientada en la misma batalla. <https://www.filmaffinity.com/es/film186830.html> (último acceso: 13/8/2023).

mientras que los “Persas” tienen que llegar a conquistar la ubicación situada en el centro de la base enemiga.

El ejército conquistador, tiene una amplia infraestructura y acceso a una gran cantidad de recursos, mientras que los defensores tienen acceso a una cantidad bastante más limitada de recursos y muchas menos estructuras que se ven compensadas con la ventaja estratégica del escenario al tener el control de un estrecho paso que es la única vía de acceso. Los defensores cuentan con varios pilares Defensa (ver anexo M.6.15) para hacer más estrecho todavía el paso y para proteger sus unidades a distancia.

Las simulaciones suelen resolverse de distintas maneras según la respuesta de ambos ejércitos. Si el ejército defensor cambia su ubicación estratégica o decide avanzar para atacar suele resultar derrotado, al igual que si el ejército invasor tarda demasiado en atacar o si no destruye parte de las defensas no conseguirá avanzar a tiempo y ganarán los defensores.

Para hacer más destacable la ventaja del escenario se asigna la raza zerg, que tiene las unidades con las estadísticas más débiles, a la facción defensora, y todo lo contrario para los invasores asignándoles la raza protoss. En la mayoría de los escenarios se hace más que notable que las tropas invasoras se bloquean a sí mismas para conseguir avanzar por el desfiladero.

D.4. OpenStarCraft

OpenStarCraft está ambientado para ser un escenario original de StarCraft como el primer escenario de ejemplo (anexo D.1) pero más abierto: sin apenas limitaciones del escenario para dividir los ejércitos o obstaculizarlos. Ver figura D.7.

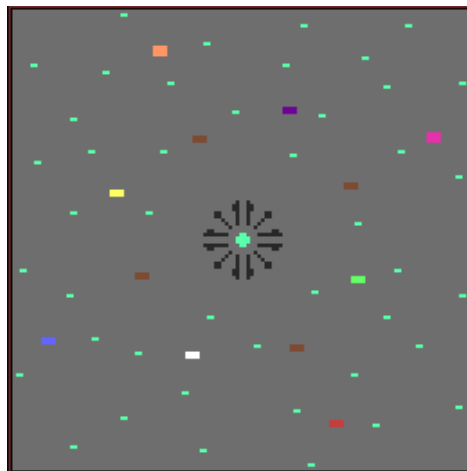


Figura D.7: Estado inicial del escenario ‘OpenStarCraft’.

Este escenario tiene dos variantes que tratan situaciones inexploradas por los otros

escenarios de ejemplo anteriores. Aunque ambas variantes comparten el mismo estado inicial (contenido y disposición de las casillas) y los recursos iniciales de los jugadores, la configuración de las facciones y sus objetivos varían. Dicha disposición del escenario se ha construido de la siguiente manera:

1. Se divide el escenario en ocho secciones similares cuyos límites pasan por el centro. De esta forma todas las secciones tienen otras dos secciones colindantes. Por ejemplo, la sección ‘3’ tiene a las secciones ‘2’ y ‘4’ de vecinas (ver figura D.8).

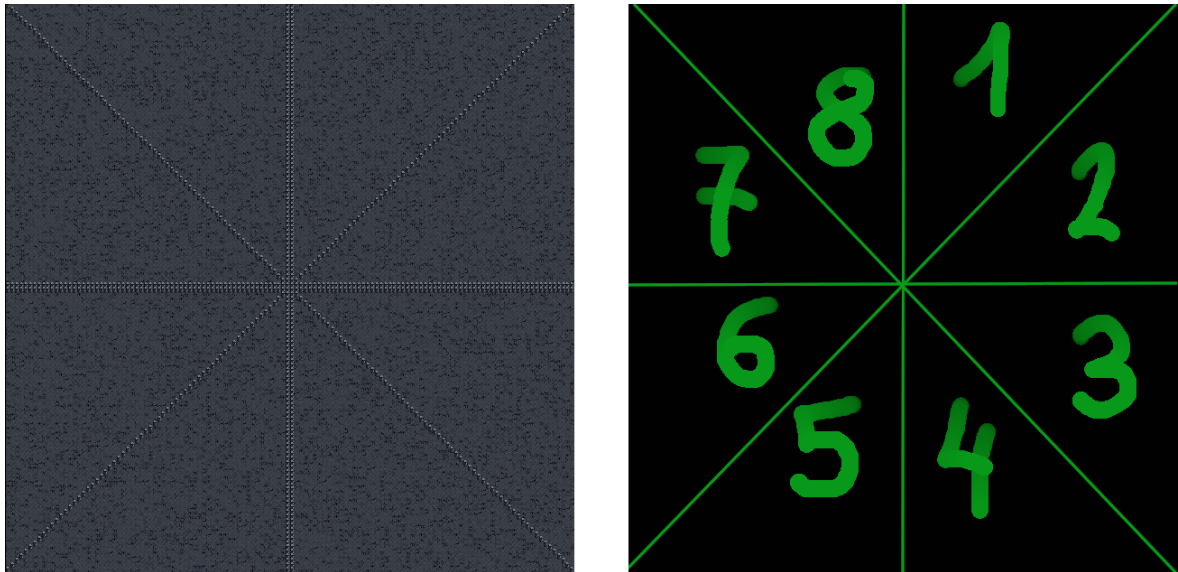


Figura D.8: Creación de ‘OpenStarCraft’ paso 1.

2. Se coloca una base de un ejército en cada sección sin repetir en ningún caso un triplete de razas. Cuatro bases se colocan más cercanas (secciones de número impar) al centro que las demás. Ver figura D.9.
3. Se distinguen las ocho bases con tres tipos de casillas asociándolo a las razas utilizadas y para el resto del suelo se utiliza el cuarto tipo de casillas. Se coloca también un géiser en las secciones con las bases alejadas, dando ventaja a los ejércitos céntricos ya que tienen dos géiseres cercanos mientras que las bases alejadas solo uno. Se le da ventaja a estos ejércitos pues son los más flanqueables. Ver figura D.10.
4. Se colocan los minerales en el escenario (ver figura D.11):
 - Cuatro cristales alrededor de cada una de las bases más o menos equidistantes a la base.
 - Un grupo de cristales en el centro del escenario.

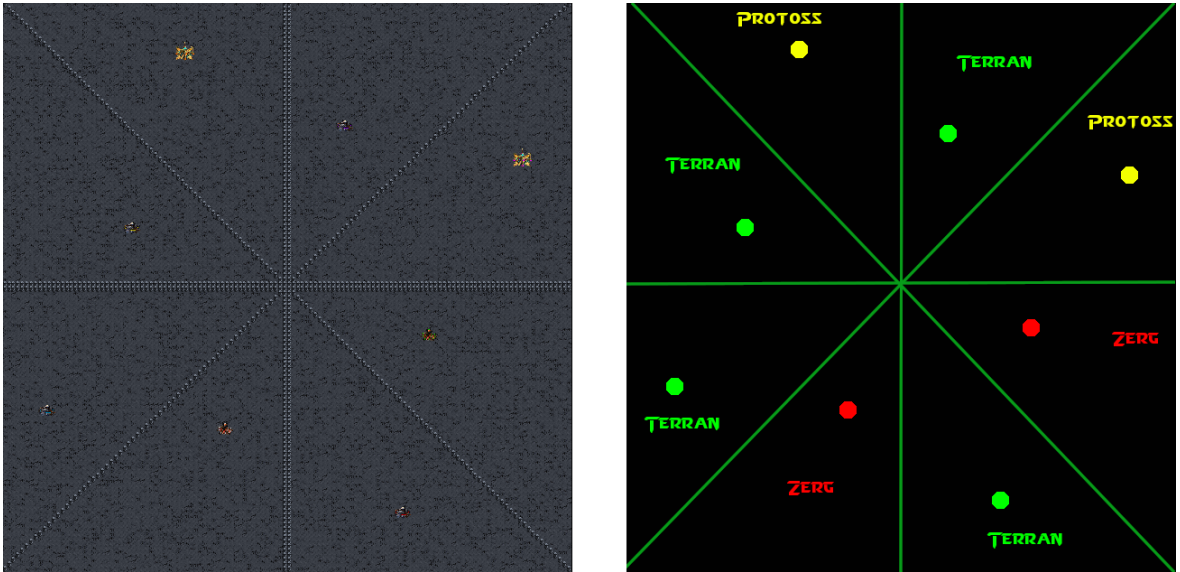


Figura D.9: Creación de 'OpenStarCraft' paso 2.

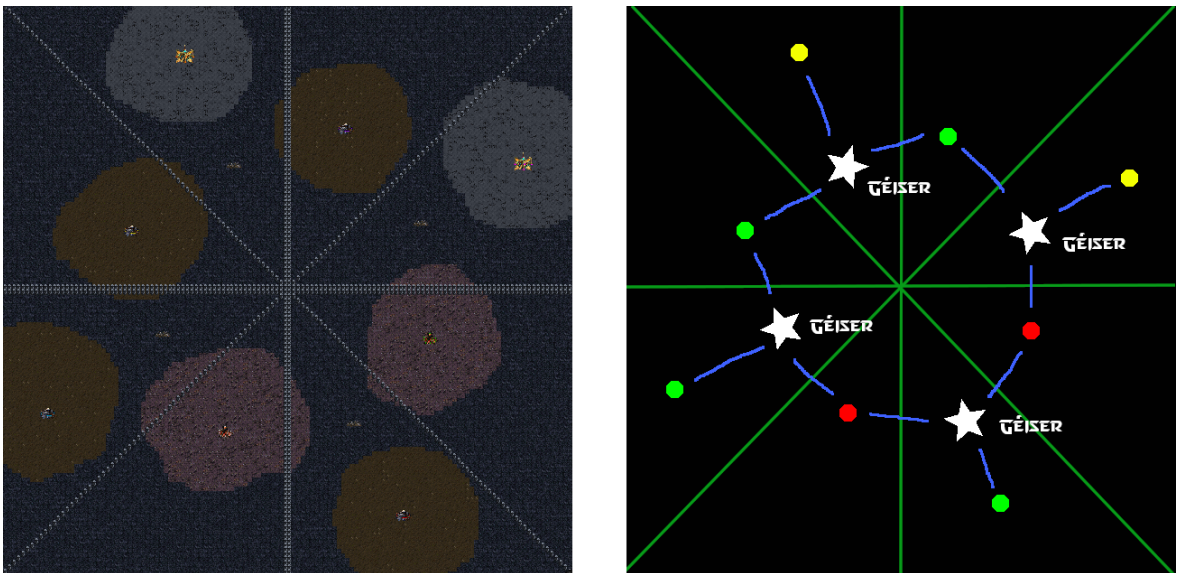


Figura D.10: Creación de 'OpenStarCraft' paso 3.

- Tres cristales en cada sector con una base céntrica para rellenar el hueco distante.

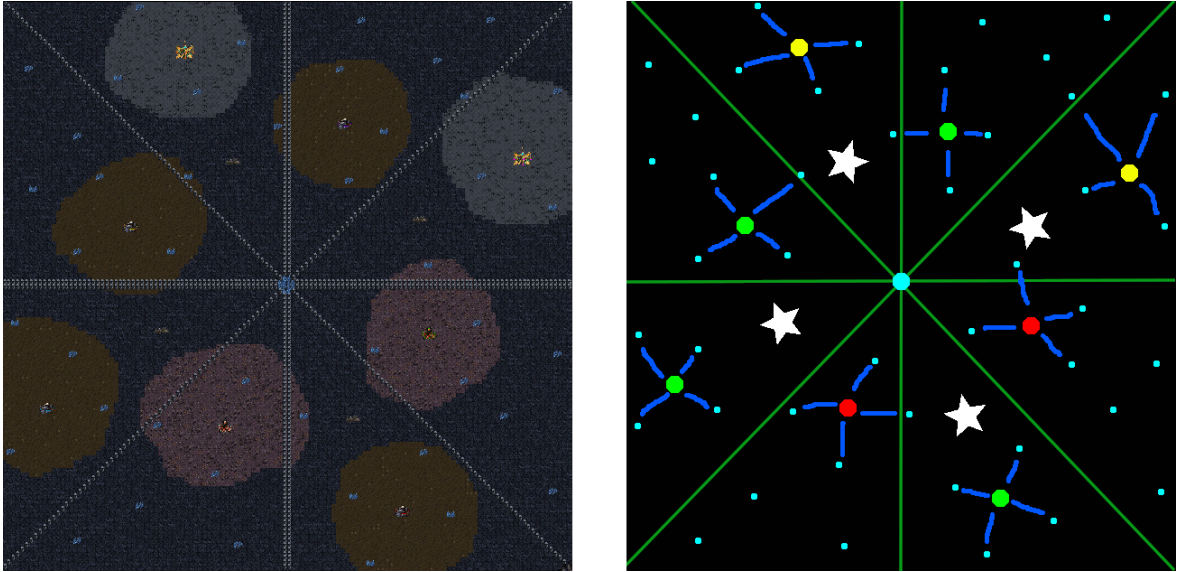


Figura D.11: Creación de ‘OpenStarCraft’ paso 4.

5. Se retiran los muros para señalar las secciones excepto en el centro que se deja una roseta⁶. Ver resultado final en la figura D.12.

La primera de las variantes de este escenario, *OpenStarCraft A*, responde directamente a ser una versión abierta del escenario *StarCraft* (anexo D.1). Es un todos contra todos donde el último ejército vivo es el que gana.

La segunda de las variantes de este escenario, *OpenStarCraft B*, es para introducir el objetivo de investigación (ver M.6.17), y una configuración donde se enfrentan varias facciones con varios ejércitos cada una. Son cuatro facciones compuestas por parejas de ejércitos ubicadas

⁶<https://es.wikipedia.org/wiki/Roseta> (último acceso: 14/8/2023).

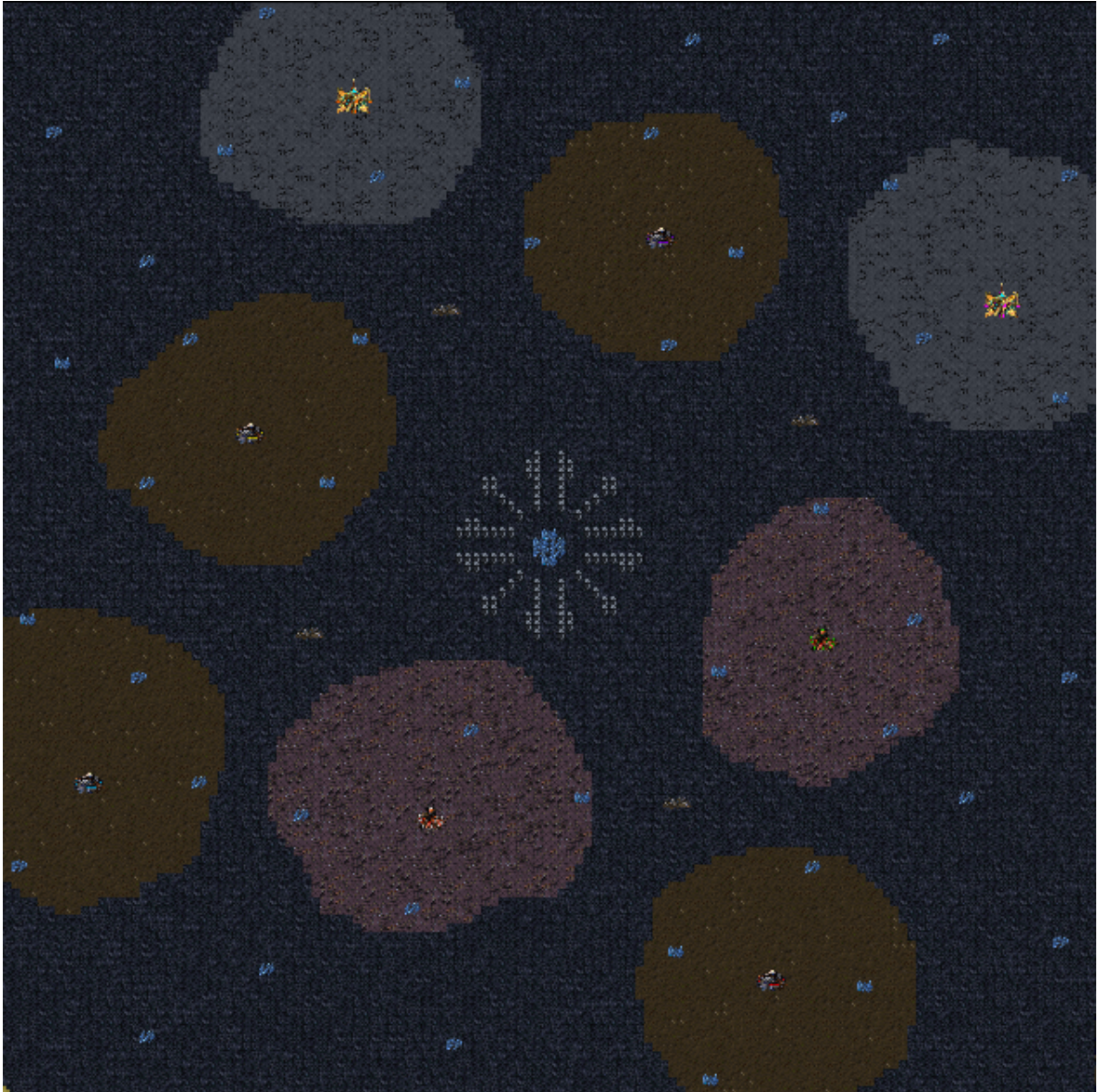


Figura D.12: Escenario de ejemplo ‘OpenStarCraft’.

Anexos E

Pruebas

En este capítulo se exponen las pruebas realizadas sobre Armygeddon para probar sus características.

E.1. Sistema de versiones para depuración

La depuración de un videojuego es crítica¹ por definición, ya que la amplitud de posibilidades de cara al usuario, es decir, su jugabilidad, está relacionada con la dificultad con la que se encuentran sus errores.

Durante una versión los errores que iban apareciendo se escribían en un foro común de Discord² (ver figura E.1) adjuntando en su caso imágenes o vídeos explicativos para poder recrearlo y corregirlo a través de hilos propios para cada uno. Si un error reaparecía, o no estaba del todo arreglado durante una versión, se continuaba dicho hilo.

Todo el proceso recaía en la utilización de un registro de versiones en el que se apuntaban los cambios que aportaba la versión (incluido en el anexo L). Para cada versión se incidía en revisar con mayor énfasis lo implementado en la versión anterior, aunque en la mayoría de casos los errores que se encontraban también eran de versiones anteriores.

En Armygeddon cerca de la mitad del esfuerzo temporal dedicado al proyecto (ver sección 3.7.3) se invierte en la depuración de las versiones. En este proceso también han participado en menor medida otras dos personas y el director del proyecto.

¹https://en.wikipedia.org/wiki/Game_testing (último acceso: 21/6/2023).

²Discord es una red social de mensajería instantánea y de llamadas utilizada en este caso por su capacidad de crear foros llamados servidores en los que es muy fácil aportar imágenes y vídeos, ordenar las publicaciones o incluso compartir la pantalla. <https://discord.com/> (último acceso: 11/8/2023).

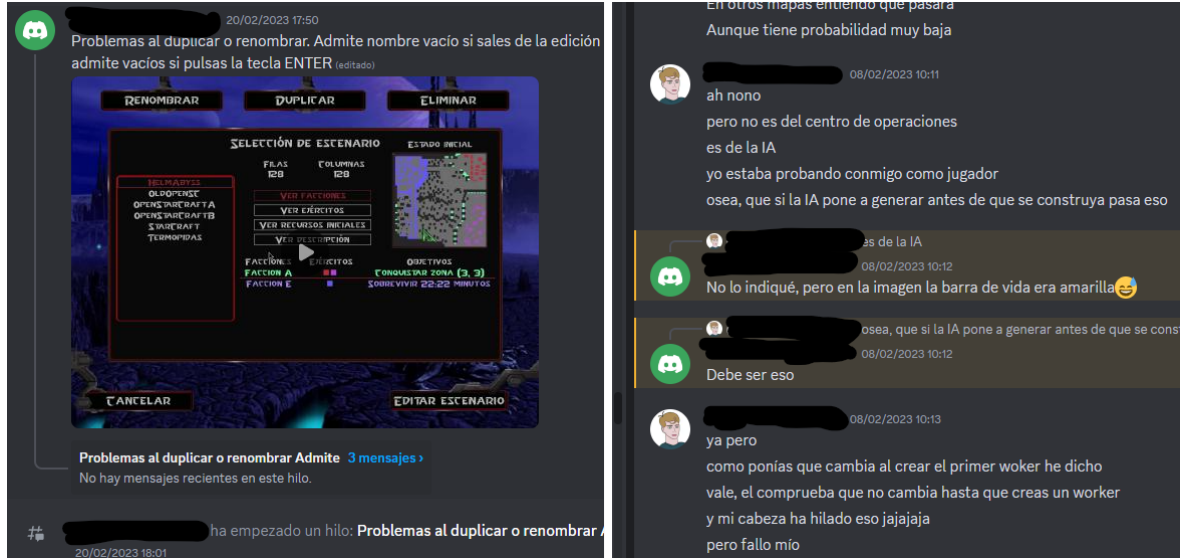


Figura E.1: Captura del foro en Discord para la depuración de Armygeddon.

E.2. Pruebas de balance

Con estas pruebas se intenta medir el balance tanto de los escenarios de ejemplo (anexo D) como de las razas, ya que uno de los escenarios de ejemplo (escenario StarCraft) está balanceado de manera que las razas parten de la misma ventaja. Las pruebas se realizaron con simulaciones continuas hasta que una de las facciones participantes alcanzara las 10 victorias en cada escenario oficial.

E.2.1. StarCraft

En esta sección se tratan los resultados sobre el escenario de ejemplo StarCraft (ver tabla E.1).

Facción	Raza	Victorias
Ejército rojo	Zerg	7
Ejército naranja	Terran	6
Ejército amarillo	Protoss	7
Ejército verde	Terran	7
Ejército azul	Terran	4
Ejército rosa	Zerg	10
Ejército violeta	Zerg	8
Ejército blanco	Protoss	5

Tabla E.1: Resultado de las pruebas de balance sobre StarCraft.

A pesar de que las mecánicas hacen a las razas diferenciadas y enfocadas para ciertas situaciones, haciendo injusto un escenario equitativo, se aprecia cierto balance de poder

entre las razas: los zerg suman 25 victorias, 17 terran y 12 los protoss. Cabe destacar que los protoss tienen tan bajo resultado porque son solo dos ejércitos, y no tres como las otras razas. El porqué los zerg tienen tan buen resultado se debe a la fuerza de la raza al conseguir concentrar unidades, al ser un mapa tan dividido y balanceado por el ejército neutral les permite acumular muchas unidades antes de invadir.

También se aprecia un balance del escenario al haber 26 victorias de los ejércitos situados en las esquinas frente a 28 victorias de los ejércitos de los lados.

E.2.2. Thermopylae

En esta sección se tratan los resultados de las pruebas sobre el escenario de ejemplo Thermopylae (ver tabla E.2).

Facción	Victorias
Griegos	10
Persas	8

Tabla E.2: Resultado de las pruebas de balance sobre Thermopylae.

El resultado ajustado de estas pruebas viene por el continuo balance sobre este. Las victorias de los Persas suelen ocurrir a los 14 minutos, o a los 16 si han decidido exterminar antes a los Griegos.

La *batalla de las Termópilas* (es la batalla que fundamenta este escenario) no es una batalla histórica cuya dinámica encaje mucho con la de StarCraft, por lo que ha tenido que mostrarse en este formato reducido y las pruebas tienen que ajustarse a ello. Son varios los aspectos que condicionan este escenario, como se detallan en el anexo D.3.

E.2.3. HelmAbyss

En esta sección se tratan los resultados de las pruebas sobre el escenario de ejemplo HelmAbyss (ver tabla E.3).

Facción	Victorias
Defensores	8
Invasores	10

Tabla E.3: Resultado de las pruebas de balance sobre HelmAbyss.

Los resultados más allá de las victorias alcanzadas también reflejan equilibrio en este escenario, pues tanto las victorias de los Defensores como de los Invasores suelen ser

ajustadas: si los Invasores ganan es por muy poco, menos de cuatro minutos de tiempo, y si los Defensores ganan también es por un margen muy limitado, pues apenas les quedan defensas.

E.2.4. OpenStarCraft

En esta sección se tratan los resultados sobre el escenario de ejemplo OpenStarCraft variante Exterminio (ver tabla E.4).

Ejército	Raza	Victorias
Rojo	Terran	9
Naranja	Protoss	6
Amarillo	Terran	3
Verde	Zerg	6
Azul	Terran	3
Rosa	Protoss	10
Violeta	Terran	2
Blanco	Zerg	3

Tabla E.4: Resultado de las pruebas de balance sobre OpenStarCraft variante Exterminio.

En este caso los ejércitos más alejados (rojo, naranja, rosa y azul) suman 25 victorias mientras que los ejércitos céntricos (amarillo, verde, violeta y blanco) alcanzan solo 14 victorias, debido a que en este escenario no hay separación ni compensación alguna como ocurre en el escenario StarCraft.

También es destacable que hay mucha variedad en los resultados, aun teniendo en cuenta a los ejércitos si son céntricos o no, o si son de la misma raza. Esto busca probar que los ejércitos difieren de forma que combinaciones de razas, de la ubicación y el momento y ritmo de la partida afectan al resultado. Esto se corrobora también en la tabla E.5.

Facción	Razas	Ubicación	Victorias
A	Terran y Zerg	Céntrica	3
B	Terran y Protoss	Apartada	10
C	Terran y Zerg	Céntrica	2
D	Terran y Protoss	Apartada	5

Tabla E.5: Resultado de las pruebas de balance sobre OpenStarCraft variante Investigación.

Aquí se termina de mostrar que la combinación de razas afecta al resultado de la partida, siendo que la facción B obtiene el doble de victorias que la facción D siendo que ambas son facciones céntricas y compuestas de un ejército terran y de otro protoss.

La razón de esto se debe a la diferencia de rivales vecinos que tienen los componentes de sendas facciones.

E.3. Pruebas del jugador artificial

En esta sección se prueba el cambio de comportamiento sobre el jugador artificial y su repercusión sobre las partidas. Se escoge para las pruebas el mismo método que en las pruebas de balance (ver anexo anterior E.2) pero todas son en el escenario *HelmAbyss*, debido a que es el escenario que más pone a prueba al jugador artificial.

E.3.1. Agresividad al máximo

Para esta prueba se configuran todos los jugadores artificiales con comportamiento “Muy agresivo”, es decir, con la tendencia de la agresividad a crecer el máximo.

El resultado (ver tabla E.6) es parecido al de las pruebas de balance donde los jugadores artificiales tienen una agresividad equilibrada (ver tabla E.3), pero las partidas son radicalmente distintas en ciertos aspectos:

Facción	Victorias
Defensores	10
Invasores	7

Tabla E.6: Resultado de las pruebas del jugador artificial con agresividad máxima.

- Los ejércitos invasores dependen mucho de dónde ataquen primero, pues al tener una mayor agresividad la insistencia en dicha primera invasión es mayor. Esto se traduce en que si atacan en la parte de las murallas perderán muchas más tropas pues es la zona más fortificada por los *Defensores*, al contrario que si atacan el muro inferior que entrarán antes en la fortaleza y es mucho más probable su victoria (ver figura D.5 para distinguir las zonas mencionadas).
- Los ejércitos defensores al tener una alta agresividad hacen cosas atípicas como mandar tropas a las bases enemigas. Esto tiene un impacto positivo si son pocas tropas, pues obliga a los invasores a retroceder y a perder mucho tiempo. Si son muchas tropas acaba siendo negativo para los defensores pues se quedan indefensos y sin poder aprovechar la ventaja estratégica que poseen que es defenderse en la fortaleza.
- Los ejércitos defensores además, hay partidas en las que tras caer los muros inferiores se extienden rápidamente creando estructuras que son muy difíciles de destruir mientras se construyen y ganando mucho tiempo.

En general en esta prueba se demuestra que las razas son distintas y que los jugadores artificiales que controlan una raza zerg son mucho más defensivos, pues sus tropas son más numerosas a pesar de que ponderen igual que las demás para el conteo de bajas, afectando mucho más a la agresividad que con los protoss o los terran.

E.3.2. Agresividad al mínimo

Para esta prueba se configuran todos los jugadores artificiales con comportamiento “Muy defensivo”, es decir, con la tendencia de la agresividad a decrecer lo máximo.

El resultado (ver tabla E.7) es muy distinto a las otras configuraciones, pero no por el efecto que tiene esta configuración de los jugadores sobre la facción defensora, pues este es muy reducido al tener unos ejércitos mucho más débiles y eso lleva a tener la agresividad generalmente baja.

Facción	Victorias
Defensores	2
Invasores	10

Tabla E.7: Resultado de las pruebas del jugador artificial con agresividad mínima.

El cambio importante de esta configuración es sobre la facción invasora, que al tener un comportamiento muy defensivo esta comanda varias invasiones en las distintas zonas del castillo, de forma que no le da tiempo a los defensores de reposicionarse creando así muchas grietas en las murallas que ocasionan este resultado.

E.4. Pruebas de requisitos

Las pruebas de requisitos tratan de demostrar que un ordenador actual podría ejecutar al menos a velocidad “x4” el escenario más costoso. Dicho escenario se ha considerado StarCraft con el truco de *recursos infinitos* activo. Estas pruebas se han realizado de la siguiente manera:

1. Se comienza el escenario StarCraft con la configuración indicada.
2. Se activa el cronómetro.
3. Se activa el modo de velocidad máximo.
4. Se espera a que se acabe la partida y se desactiva el cronómetro.

El equipo que ha ejecutado las pruebas tiene las características indicadas en el manual de usuario como requisitos mínimos (anexo M.2). Las pruebas han resultado triviales

al ver que en el primer resultado se ejecuta una partida de 45 minutos en menos de 10 segundos, casi 300 veces más deprisa que en tiempo de partida.

Como acelerando el juego solo se ejecutan múltiples procesados lógicos pero sigue ejecutándose solo uno gráfico por cada ciclo de juego, se estudia con una herramienta de análisis del uso del procesador cuánto ocupa la carga gráfica respecto de la total. Se utiliza menos de un 15 % de los recursos del procesador utilizados para cuestiones gráficas, así que se entiende también que no es un problema para los requisitos.

Se concluye que para equipos relativamente modernos no hay requisitos mínimos para poder jugar a Armygeddon.

E.5. Pruebas de conectividad multijugador

Se ha probado la conectividad multijugador jugando una partida desde dos nodos, cada uno con cuatro ejecuciones distintas de Armygeddon. Ambos nodos se han conectado a través de una red local virtual utilizando *ZeroTier*³.

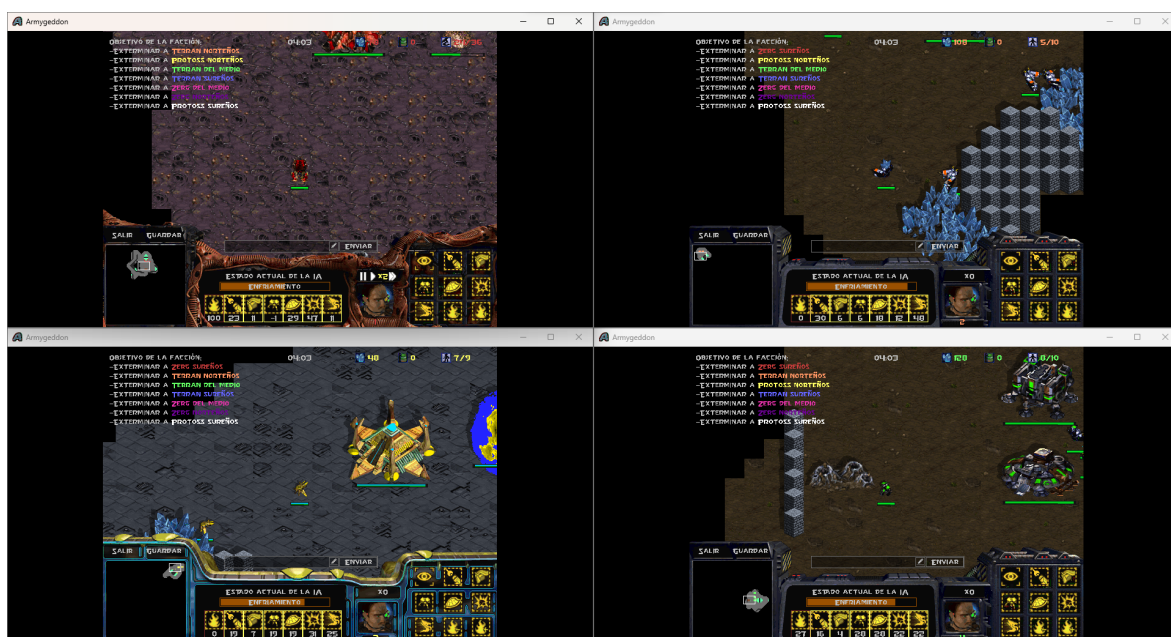


Figura E.2: Captura de pantalla de uno de los nodos durante la prueba multijugador.

La partida era en el escenario StarCraft, donde cada uno de los jugadores era híbrido y por tanto remoto(se generaban órdenes de un jugador artificial en su correspondiente ejecución de Armygeddon). El objetivo era comprobar si a través de la red y debido al protocolo del paso de mensajes utilizado era suficiente para mantener la conexión constante de los ocho clientes.

³<https://www.zerotier.com/> (último acceso: 16/8/2023).

No solo se completó la partida sin incidencias al modo ‘x2’ de velocidad, sino que se hizo un cálculo optimista⁴ del flujo de datos para contrastarlo con el ancho de banda y, comprobar que efectivamente, no se sobrecarga la red con esta arquitectura red.

Se midieron todos los datos que entraban y salían del servidor, que es donde más datos pasan en este diseño centralizado. Algunas observaciones:

- Se recibieron 17959 mensajes, con una longitud media de 240 Bytes. La mayoría al ser órdenes de juego ocupan en torno a 208 y 264 Bytes, siendo este último el valor máximo.
- Se enviaron 103030 mensajes, con una longitud media de unos 1200 Bytes. Existen valores máximos puntuales de unos 15000 Bytes. En algunos casos se están agrupando todos los mensajes a mandar a todos los distintos clientes de una vez, por eso algunos valores son tan elevados.

Siendo la velocidad del juego de 30 ciclos por segundo, y, en este caso, de 60, por estar acelerado a ‘x2’, se entiende que el ancho de banda de bajada es de media $240\text{Bytes} * 60 = 14,4\text{kBps}$ y el de subida de $1200\text{Bytes} * 60 = 720\text{kBps}$. Ambas cifras son muy inferiores al ancho de banda medio en España de la actualidad que es de 135,73 mBps⁵.

⁴No se han tenido en cuenta las cabeceras añadidas por los distintos protocolos de red que hay por debajo del utilizado (la biblioteca *SDLNet*), ya que de ser un problema serían fácilmente sustituibles.

⁵<https://roams.es/companias-telefonicas/blog/internet/velocidad-internet-espana/> (último acceso: 29/8/2023).

Anexos F

Organización de archivos, directorios y mensajes

Los archivos del juego, como se describen organizados en el anexo F.1, pueden ser necesarios e intrínsecos a Armygeddon como los gráficos o, ser generados por el usuario y opcionales. Acto seguido se detallan los tres formatos que se han usado para los distintos tipos de archivos y mensajes que se crean por el usuario con el uso del programa en los anexos F.2, F.3 y F.4.

F.1. Árbol de directorios

A continuación se explica la estructura del proyecto desde el punto de vista de los directorios incluyendo los archivos más relevantes (ver figuras F.1, F.2, F.3 y F.4).

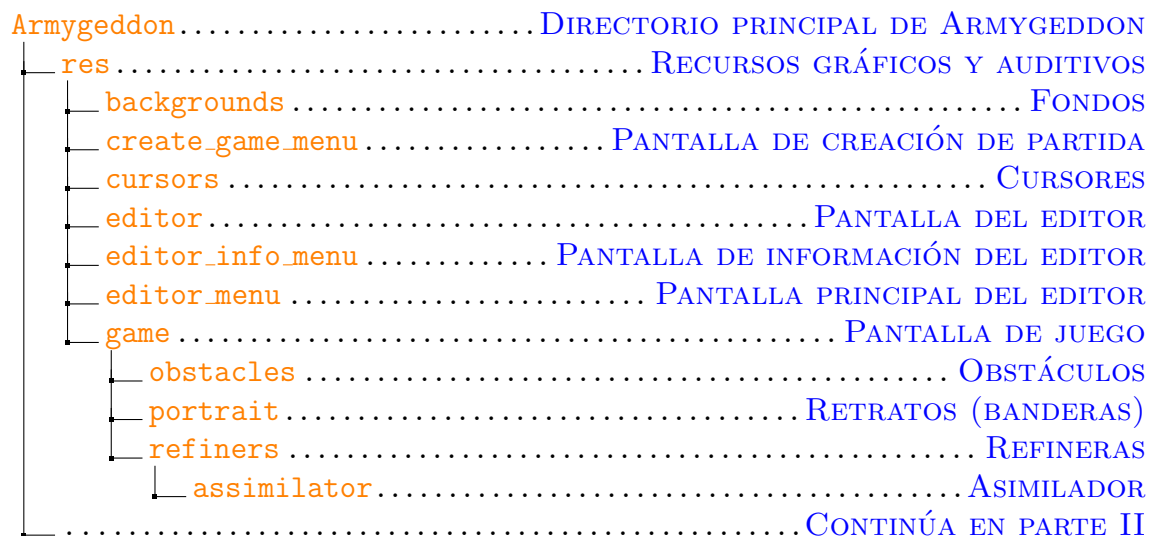


Figura F.1: Árbol de directorios: Parte I.

Armyggedon	CONTINÚA DESDE PARTE I
res	CONTINÚA DESDE PARTE I
game	CONTINÚA DESDE PARTE I
refiners	CONTINÚA DESDE PARTE I
extractor	EXTRACTOR
refinery	REFINERÍA
resources	RECURSOS NATURALES
structures	ESTRUCTURAS
protoss	RAZA PROTOSS
cibernetics_core	CENTRO CIBERNÉTICO
gateway	PORTAL
nexus	NEXO
pylon	PILÓN
terran	RAZA TERRAN
academy	ACADEMIA
barracks	CUARTELES
command_center	CENTRO DE OPERACIONES
defense	DEFENSA
supply_depot	ALMACÉN DE SUMINISTROS
zerg	RAZA ZERG
hatchery	CRIADERO
hydralisk_den	GUARIDA DE HIDRALISCOS
spawning_pool	ESTANQUE DE DESOVE
tiles	BALDOSAS
units	UNIDADES
protoss	RAZA PROTOSS
dragoon	DRAGÓN
probe	SONDA
zealot	FANÁTICO
terran	RAZA TERRAN
firebat	MURFLAMA
marine	MARINE
scv	VCE
zerg	RAZA ZERG
drone	DRÓN
egg	HUEVO
hydralisk	HIDRALISCO
larva	LARVA
overlord	SOBERANO
zergling	ZERGLING
wonder	MARAVILLA
game_info_menu	PANTALLA DE INFORMACIÓN DE JUEGO
game_menu	PANTALLA DE JUEGO
icons	ICONOS
main_menu	PANTALLA PRINCIPAL
editor	BOTÓN DEL EDITOR
.....	CONTINÚA EN PARTE III

Figura F.2: Árbol de directorios: Parte II.

Armygeddon	CONTINÚA DESDE PARTE II
res	CONTINÚA DESDE PARTE II
main_menu	CONTINÚA DESDE PARTE II
exit	BOTÓN DE SALIR
play	BOTÓN DE JUGAR
settings	PANTALLA DE AJUSTES
multiplayer_menu	PANTALLA DE JUEGO EN LÍNEA
music	MÚSICA
online_menu	PANTALLA DE SALA EN LÍNEA
portraits	RETRATOS
protoss	RAZA PROTOSS
Aldaris	ALDARIS
Artanis	ARTANIS
Fenix_Dragon	FENIX (DRAGÓN)
Fenix_Zealot	FENIX (FANÁTICO)
Raszagal	RASZAGAL
Tassadar	TASSADAR
Zeratul	ZERATUL
terran	RAZA TERRAN
Alexei_Stukov	ALEXEI STUKOV
Arcturus_Mengsk	ARCTURUS MENGSK
Edmund_Duke	DUQUE EDMUND
Gerard_DuGalle	GERARD DUGALLE
Jim_Raynor	JIM RAYNOR
Sarah_Kerrigan	SARAH KERRIGAN
Terran_Civilian	CIVIL TERRAN
zerg	RAZA ZERG
Daggoth	DAGGOTH
Hunter_Killer	ASESINO DE CAZADOES
Infested_Kerrigan	KERRIGAN INFESTADA
Kerrigans_Chrysalis	CRISÁLIDA DE KERRIGAN
Samir_Duran	SAMIR DURAN
The_Overmind	LA SUPERMENTE
Zasz	ZASZ
singleplayer_menu	PANTALLA DE JUEGO INDIVIDUAL
sound	SONIDOS
bullet	ALGUNOS ATAQUES
glue	CLIC
misc	ESPECIALES
protoss	UNIDADES PROTOSS
dragoon	DRAGÓN
probe	SONDA
zealot	FANÁTICO
terran	UNIDADES TERRAN
.....	CONTINÚA EN PARTE IV

Figura F.3: Árbol de directorios: Parte III.

Armygeddon	CONTINÚA DESDE PARTE III
res	CONTINÚA DESDE PARTE III
sound	CONTINÚA DESDE PARTE III.
terran	CONTINÚA DESDE PARTE III
firebat	MURFLAMA
marine	MARINE
scv	VCE
zerg	UNIDADES ZERG
drone	DRÓN
egg	HUEVO
hydra	HIDRALISCO
larva	LARVA
overlord	SOBERANO
zergling	ZERGLING
stage_creation_menu	PANTALLA DE CREACIÓN DE ESCENARIOS
Starcraft.ttf	FUENTE DE ARMYGEDDON
saves	PARTIDAS GUARDADAS
stages	ESCENARIOS
Armygeddon.exe	EJECUTABLE
config.txt	CONFIGURACIÓN DEL USUARIO
libjpeg-9.dll	CARGA DE ARCHIVOS .PNG
libpng16-16.dll	CARGA DE ARCHIVOS .PNG
SDL2.dll	SDL BASE
SDL2_image.dll	IMÁGENES CON SDL
SDL2_mixer.dll	SONIDOS CON SDL
SDL2_net.dll	RED CON SDL
SDL2_ttf.dll	TEXTOS CON SDL
zlib1.dll	CARGA DE ARCHIVOS .PNG

Figura F.4: Árbol de directorios: Parte IV.

F.2. Archivo de configuración

El archivo `./config.txt` contiene la configuración de los ajustes de usuario. Al ejecutar Armygeddon se lee este fichero si existe para cargar una configuración personalizada del usuario. En el caso de no existir este fichero se carga la configuración por defecto. Para crear, o sobrescribir si ya existe, dicho archivo de configuración, hay que guardar en la pantalla de ajustes. Este fichero es útil para modificar los ajustes sin necesidad de ejecutar el programa.

Este archivo tiene tres tipos de líneas:

1. *Líneas vacías.* Usadas para distinguir categorías de ajustes.
2. *Líneas comentario.* Útiles para describir los ajustes que vienen a continuación y para señalar los valores esperados. Se distinguen por comenzar con la cadena

“//”.

3. *Líneas de ajuste*. Indican el valor a cargar para un ajuste en concreto. Están compuestas del nombre del ajuste, del operador “=” y del valor a cargar.

Existe cierta prevención de fallos en el tratamiento de este fichero:

- Los ajustes no especificados, en el caso de que se borren sus líneas, son asignados a sus valores por defecto.
- El orden de las líneas no importa, se evalúan de una en una.
- Para los ajustes que no son de teclado:
 - Si el valor introducido no concuerda con el tipo esperado (p.e. se espera un número y se introduce una letra), la línea se ignora.
 - Si el valor introducido excede el rango esperado para dicho ajuste este se normaliza a su término más cercano (p.e. se introduce un 40 donde se esperaba un valor entre 0 y 8 se asigna el valor 8).
- Para los ajustes de teclado, se intenta obtener la tecla introducida.
- Para los ajustes que no existen, se ignora la línea.

Fichero de configuración ejemplo (es el que se genera por defecto):

```
1  //// Controles de teclado
2  CONTROL_PASAR=Return
3  CONTROL_MOVER_SELECCION_IZQUIERDA=Left
4  CONTROL_MOVER_SELECCION_ARRIBA=Up
5  CONTROL_MOVER_SELECCION_DERECHA=Right
6  CONTROL_MOVER_SELECCION_ABAJO=Down
7  CONTROL_MOVER_CAMARA_IZQUIERDA=A
8  CONTROL_MOVER_CAMARA_ARRIBA=W
9  CONTROL_MOVER_CAMARA_DERECHA=D
10 CONTROL_MOVER_CAMARA_ABAJO=S
11 CONTROL_ALTERNAR_CUADRICULA=G
12 CONTROL_SALIR=Escape
13 CONTROL_ACTIVAR_BOTON_1=1
14 CONTROL_ACTIVAR_BOTON_2=2
15 CONTROL_ACTIVAR_BOTON_3=3
16 CONTROL_ACTIVAR_BOTON_4=4
17 CONTROL_ACTIVAR_BOTON_5=5
18 CONTROL_ACTIVAR_BOTON_6=6
19 CONTROL_ACTIVAR_BOTON_7=7
20 CONTROL_ACTIVAR_BOTON_8=8
```

```

21 CONTROL_ACTIVAR_BOTON_9=9
22 CONTROL_ALTERNAR_SILENCIAR_TODO=M
23 CONTROL_MOSTRAR_TABLA_FACCIONES=Tab
24 CONTROL_EJERCITO_ANTERIOR=Q
25 CONTROL_EJERCITO_POSTERIOR=E
26 CONTROL_ALTERNAR_MODO_RECURSOS=R
27 CONTROL_ALTERNAR_MODO_BALDOSAS=T
28 CONTROL_ALTERNAR_MODO_UNIDADES=U
29 CONTROL_ALTERNAR_MODO_EDIFICIOS=I
30 CONTROL_ALTERNAR_MODO_OBSTACULOS=O
31 CONTROL_ALTERNAR_MODO_OBJETIVOS=Y
32 CONTROL_ALTERNAR_PAUSA=Space
33 CONTROL_ALTERNAR_PANTALLA_COMPLETA=F11
34
35 //// Ajustes de sonido
36 // Silenciar todo audio del juego: SI o NO
37 TODO_SILENCIADO=NO
38 // Volumen de la musica: 0 a 8
39 VOLUMEN_MUSICA=6
40 // Volumen de los efectos: 0 a 8
41 VOLUMEN_EFECTOS=6
42
43 //// Ajustes de video
44 // Pantalla completa: SI o NO
45 PANTALLA_COMPLETA=NO
46 // Resolucion seleccionada:
47 // 0 es (640x480)
48 // 1 es (1024x768)
49 // 2 es (1280x960)
50 // 3 es (1440x1080)
51 // 4 es (1600x1200)
52 // 5 es (1920x1440)
53 // 6 es (2880x2160)
54 INDICE_DE_RESOLUCION=0
55 // Velocidad de las animaciones: 0 a 10
56 VELOCIDAD_TRANSICIONES=7
57 // Postergacion de las ventanas emergentes: 0 a 10
58 POSTERGACION_VENTANAS_EMERGENTES=1
59
60 //// Ajustes extra :D
61 // El cursor mueve la camara: SI o NO
62 CURSOR_MUEVE_CAMARA=SI
63 // Avatar para el modo online: 0 a 20
64 ONLINE_PORTRAIT_INDEX=20
65 // Nombre para el modo online:
66 ONLINE_NAME=Jugador

```

F.3. Archivo de escenario

Los escenarios contienen toda la información requerida para jugar una partida en un fichero por escenario en el directorio “./stages/” con ficheros formato “.txt” y cuyos nombres de fichero son el nombre del escenario en sí.

Cada fichero de escenario está compuesto por 5 secciones:

1. *Descripción* del escenario: una línea. Es una cadena de caracteres donde los ‘;’ sirven de separadores para representar un salto de línea dentro de la descripción.
2. *Tamaño* del escenario: una línea. Un número para el número de filas, un guión como separador y otro número para el número de columnas.
3. *Ejércitos*: 9 líneas. Dividiendo entre los ejércitos jugables y el neutral:
 - I. Las 8 primeras líneas representan a los *ejércitos jugables* siguiendo el orden cromático de los ejércitos¹ y tienen el siguiente formato: Una cadena de caracteres alfanuméricos que puede contener espacios siempre que no sea en uno de sus términos, un guión de separación, y 8 números que representan valores del ejército, separados por comas. Dichos números representan:
 - 1) Ejército activo. 0 para indicar que no está activo, 1 para indicar que sí lo está.
 - 2) Cristal inicial. El valor indicado es el valor de cristal inicial de ese ejército al comienzo de la partida.
 - 3) Gas inicial. El valor indicado es el valor de gas inicial de ese ejército al comienzo de la partida.
 - 4) Población máxima inicial. El valor indicado es el valor de la población máxima inicial de ese ejército al comienzo de la partida.
 - 5) Raza del ejército. 0 si el ejército es Protoss, 1 si es Zerg y 2 si es Terran. Este valor sirve para orientar al jugador artificial.
 - 6) Ejército controlado por jugador artificial por defecto. 0 si tiene que inicialmente estar configurado como jugador humano, 1 si tiene que estarlo como artificial.

¹Es el orden indicado en la subsección M.6.4: Rojo, naranja, amarillo, verde, azul, rosa, púrpura y blanco. Este orden se utiliza en distintas partes por seguir un estándar: el editor de escenarios, la selección de escenario, dicho formato de fichero de escenario, y en partida al estar *espiar rivales* activo.

- 7) Velocidad del jugador artificial por defecto. De 0 a 4 para menor o mayor velocidad de respuesta configurada inicialmente.
 - 8) Comportamiento del jugador artificial por defecto. De 0 a 4 para menor o mayor agresividad configurada inicialmente.
- II. La novena línea es para el *ejército neutral* y solo puede contener un 1 para indicar que hay ejército neutral en el escenario o un 0 para el caso contrario.
4. *Facciones*: 8 líneas, una por facción, compuestas de la siguiente forma:
- Nombre de la facción. Cadena alfanumérica que puede contener espacios siempre que no sea en los términos de la cadena.
 - Ejércitos que conforman la facción. Entre guiones y separados por comas los números que representan los ejércitos que conforman la facción, según el orden cromático de los ejércitos mencionado y comenzando por el 0.
 - Tipo de objetivos de facción. Un número para representar el tipo de objetivo que tiene la facción:
 - Facción exterminadora $\rightarrow$ 0.
 - Facción superviviente $\rightarrow$ 1.
 - Facción conquistadora $\rightarrow$ 2.
 - Facción investigadora $\rightarrow$ 3.
 - Parámetros de los objetivos de la facción. Este campo varía en función del tipo de objetivos de facción especificado antes:
 - Facción exterminadora. Números que representan los ejércitos objetivo de esta facción según el orden cromático de los ejércitos ya mencionado, separados por comas.
 - Facción superviviente. Un número que representa los minutos que tiene que sobrevivir, seguido de una coma y seguido de otro número que representa los segundos que tiene que sobrevivir.
 - Facción conquistadora. Un número que representa la X de las coordenadas de la zona de conquista objetivo, seguido de una coma y seguido de otro número que representa la Y de las coordenadas de la zona de conquista objetivo.

- Facción investigadora. Un número que representa la cantidad de maravillas que ha de investigar al completo esta facción para ganar.

5. *Casillas*: una línea por casilla. Son 9 números separados por comas:

- Tipo de baldosa. Escoger entre todos los tipos de baldosas de Armygeddon.
- Textura de baldosa. En sí es la altura a la que se hace el recorte cuadrado en la tira de texturas rectangular del tipo de baldosa correspondiente para obtener la textura que va a tener la casilla en sí.
- Tipo del ocupante. Es el tipo de entidad ocupante de entre todos los posibles (refinera, estructura, unidad, etc.).
- Subtipo del ocupante. El subtipo de entidad, por ejemplo para las unidades sería de qué tipo de unidad se trata.
- Coordenada X del ocupante. La coordenada del eje horizontal empezando por la 0 de izquierda a derecha que tiene la primera casilla del ocupante de esta casilla.
- Coordenada Y del ocupante.
- Ejército del ocupante.
- Coordenada X de la zona de conquista.
- Coordenada Y de la zona de conquista.

F.4. Archivo de guardado y mensajes en línea

Una vez creados los dos formatos de archivos anteriores de forma manual, y al requerir un formato distinto para el paso de mensajes multijugador, se opta por un sistema de serialización automático utilizando la biblioteca de Boost² para la serialización de datos³.

Las ventajas del uso de la biblioteca son varias:

- Menos esfuerzo dedicado al no tener que definir el formato de los datos serializados.

²<https://www.boost.org/> (último acceso: 11/8/2023).

³https://www.boost.org/doc/libs/1_80_0/libs/serialization/doc/index.html (último acceso: 11/8/2023).

- Mucha mayor facilidad con el manejo de punteros ya que se asignan automáticamente en la carga de los datos. Esto solo se usa en la carga de partidas, en el modo multijugador en línea no es necesario ya que se utilizan identificadores globales.
- Posibilidad de tener varias versiones paralelas de serialización de los datos. Esto es útil para reducir el consumo de red por ejemplo, al no tener que enviar todos los datos como sí hace falta para guardar la partida.
- En caso de haber un fallo, se indica exactamente dónde y de qué tipo.

El funcionamiento se basa en la definición de un método en cada clase en el que se declaran según la versión deseada qué campos se van a guardar, de forma que al cargar los datos serializados se creen objetos nuevos y se les asignen los valores de los campos guardados. Por ejemplo, para el segundo diseño de la arquitectura multijugador (ver sección 2.5.3) solo se enviaban los datos a dibujar como la salud, el escudo, y no datos lógicos como cuándo es el próximo ciclo de regeneración o de ataque.

Anexos G

Ingeniería inversa

En este anexo se detallan brevemente las principales características del proyecto que han requerido de ingeniería inversa sobre StarCraft.

G.1. Tamaño de las casillas

Desde un inicio no se sabía con seguridad el sistema de movimiento de las unidades y su relación con el tamaño de las casillas y se trabajó con casillas de 40 píxeles en la resolución original (640x480). Más adelante se descubrió que eran de 32 píxeles. Fue necesario utilizar un editor de mapas de StarCraft de terceros (ScmDraft2¹) para obtener con certeza el funcionamiento de las casillas y que por debajo utilizaban otro sistema de posicionamiento (las posiciones). Este editor al ser externo utilizaba una interfaz con escala 1:1 con las imágenes del juego y permitió haciendo capturas de pantalla obtener todos los gráficos que no se pudieron obtener desde el propio StarCraft original (obstáculos, algunas baldosas, etc.).

También, aunque no se implementó, se observó que StarCraft usaba un sistema de posicionamiento menor a las casillas y mayor al que actualmente hay en Armygeddon. De esta manera se utilizaban las casillas para definir las baldosas, y las estructuras, pero las unidades podían libremente ocupar varias casillas o parte de estas pues trabajaban con un sistema de posicionamiento menor más vinculado con el tamaño de la unidad.

En Armygeddon se utilizó un mecanismo opuesto, y se escaló las unidades al tamaño de las casillas en algunos casos como es el del soberano (explicado en anexo del manual M.8.3), que ocupaba inicialmente tres veces más.

¹<http://www.stormcoast-fortress.net/cntt/software/scmdraft/> (último acceso: 15/8/2023).

G.2. Nombre ficheros sonido e imagen

La organización de los archivos del juego de StarCraft² es enrevesada, utilizando nombres de fases previas del desarrollo para algunas entidades, o incluso seudónimos. Así que hubo que estudiar los archivos del orden de cientos de opciones, a veces teniendo que recurrir a prueba y error, para encontrar los necesarios para cada entidad en Armygeddon. Por ejemplo, para añadir el estanque de desove se tuvo que averiguar³ que su nombre en clave era completamente distinto: en vez de ser el nombre original (*spawning pool*), era *chrysalis*.

Hay que sumarle además que en el caso de los ficheros de sonido se utilizaba *a veces* una nomenclatura compuesta que hubo que descifrar observando otros archivos y documentación externa de StarCraft. Por poner un ejemplo: *tscwht003.wav* es el cuarto sonido alternativo de respuesta de tipo “what” de la unidad terran VCE (anexo M.9.1). Se traduce de la siguiente manera:

- “sc” es el diminutivo clave de *SCV*, que es el nombre de la unidad en inglés para el VCE.
- “wht” en los archivos de sonido alude a que el tipo de audio es de tipo “What?” (*¿Qué?* en inglés), que es una de las posibles interacciones con las unidades. Al seleccionar una unidad aliada esta te responde con un sonido de este tipo. Hay varios tipos como *pss* para *Pissed*, *yes* para *Yes*, *dth* para *Death*, etc.
- Las letras ‘t’, ‘p’ y ‘z’ aluden a las razas terran, protoss y zerg respectivamente. Es especialmente antiintuitivo pues los archivos se hallan dentro de carpetas que ya distinguían por la raza.
- Los números del final indican que es un sonido con alternativas, que no siempre puede sonar el mismo. Empiezan a contar desde el cero.

G.3. Animaciones entidades

Las ristas de gráficos o *sprites* en Armygeddon son agrupaciones de hasta varios cientos de imágenes pequeñas en las que se hallan todas las posibles imágenes a mostrar de cada entidad según su estado. En el caso de las unidades es aún más extenso pues para cada imagen existen otras 15 alternativas para las 16 direcciones en las que pueden estar dirigidas en una partida. Aunque la realidad es que de cada ciclo de las 16

²Se habla de los archivos una vez extraídos. Estos inicialmente venían comprimidos y cifrados en paquetes en formato MPQ: <https://encyclopedia.pub/entry/37738> (último acceso: 15/8/2023).

³Encontrado en https://starcraft.fandom.com/wiki/Spawning_pool#Development (último acceso: 15/8/2023).

alternativas son solo distintas nueve de ellas y las otras siete son el reflejo vertical de otras siete, como se aprecia en la figura G.1.

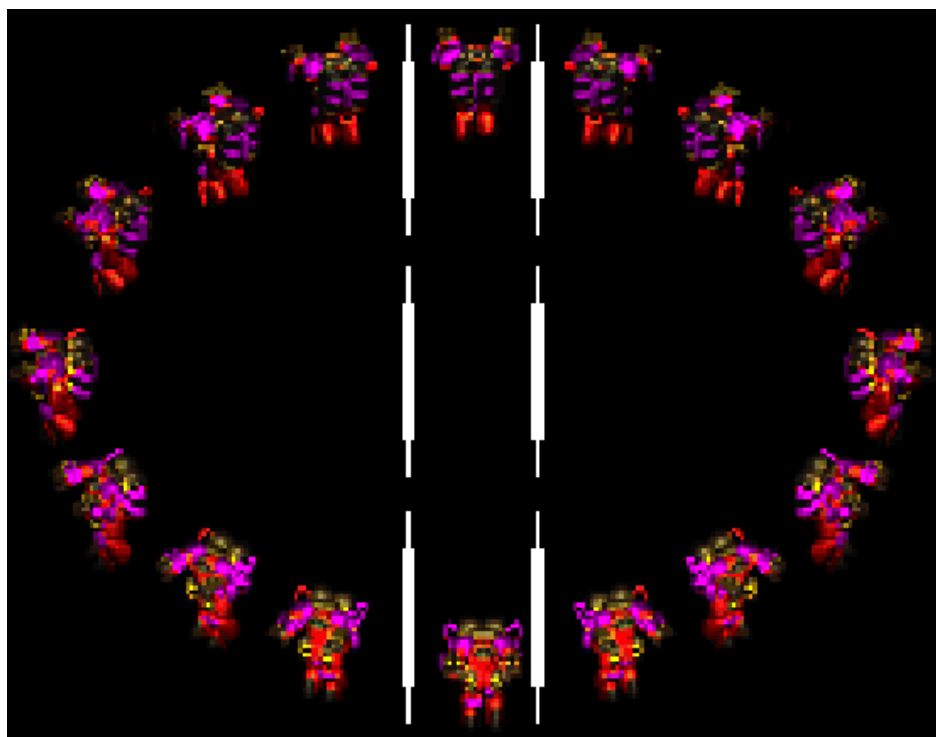


Figura G.1: Murflama ejemplo en todas las direcciones de una unidad.

El problema era al tener que hilar tantas imágenes y darles un sentido y un orden en Armygeddon sin conocer el funcionamiento de la entidad. Se tuvo que seccionar y estudiar los gráficos por separado para conseguir relacionarlos entre sí y crear las animaciones con esquemas como los de las figuras G.2 y G.3.

Se ha intentado lograr una reconstrucción cercana a la original, a pesar de que en muchos casos se reutilizan imágenes en las animaciones y resulta improbable, como ocurre con la animación de construcción/mutación zerg en la que se repiten varias explosiones.

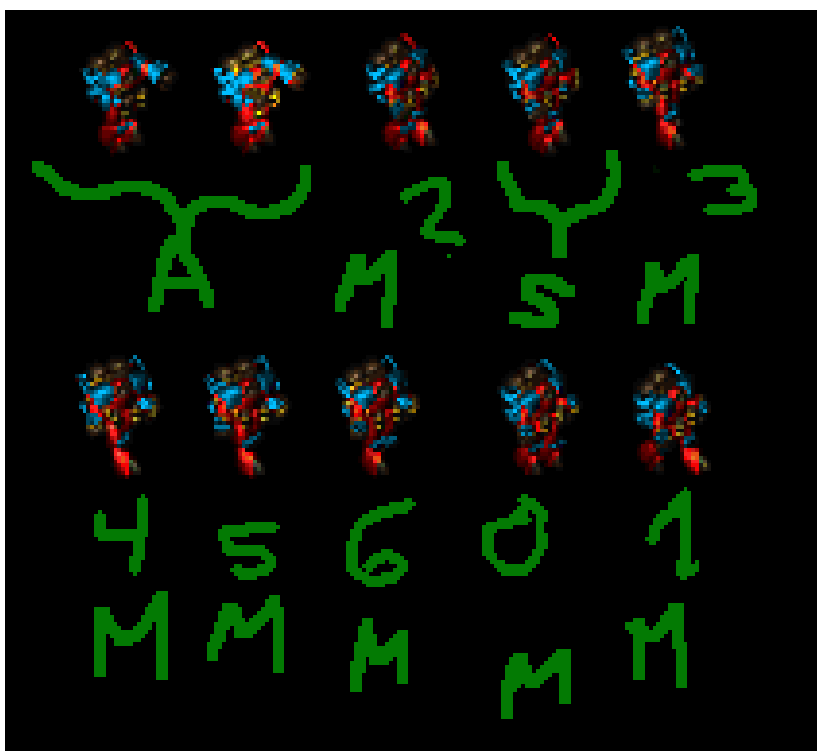


Figura G.2: Intento de obtención de la animación de un murflama.



Figura G.3: Intento de obtención de la animación de un zergling.

Anexos H

Principales dificultades del desarrollo

En este anexo se detallan las principales problemáticas que han surgido durante el desarrollo de Armygeddon y que han estado presentes durante todo el proyecto.

H.1. Sistemas de distancia

Existe un problema al tratar la distancia en una partida: aunque las entidades tengan una posición determinada dentro de un escenario, Armygeddon trabaja con casillas, que son regiones cuadradas del escenario con una matriz de 32x32 posiciones posibles dentro de cada una. Entre otras cosas, una entidad aunque solo tenga una posición, ocupa una o más casillas, ya que todo el sistema tanto en partida como en el editor está orientado a las casillas.

Puesto que todo funciona con casillas y no con posiciones, la unidad de distancia preferida es la propia casilla en sí, y esto puede entenderse (dentro y fuera de la implementación) de dos maneras distintas: Puede no significar lo mismo una determinada cantidad de casillas de distancia (ver anexo H.1.2) y una distancia de una determinada cantidad de casillas (ver anexo H.1.1).

H.1.1. Distancia real

Denominada así al ser la distancia que realmente existe entre dos puntos sin tener desajustes según el eje del escenario. Es una distancia costosa de calcular pues requiere calcular el módulo de la recta entre dos posiciones¹. Por ejemplo, la distancia real d

¹Las posiciones en Armygeddon al ser el escenario plano están compuestas de dos coordenadas, denominadas por convenio 'x' (eje horizontal) e 'y' (eje vertical).

entre las posiciones A y B sería:

$$d_{real} = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$$

Como se aprecia en la figura H.1, se dibuja en dos vistas diferentes un círculo amarillo que marca una distancia real de 3,5 casillas desde la posición de un zumbido^{2 3} en todas las direcciones.

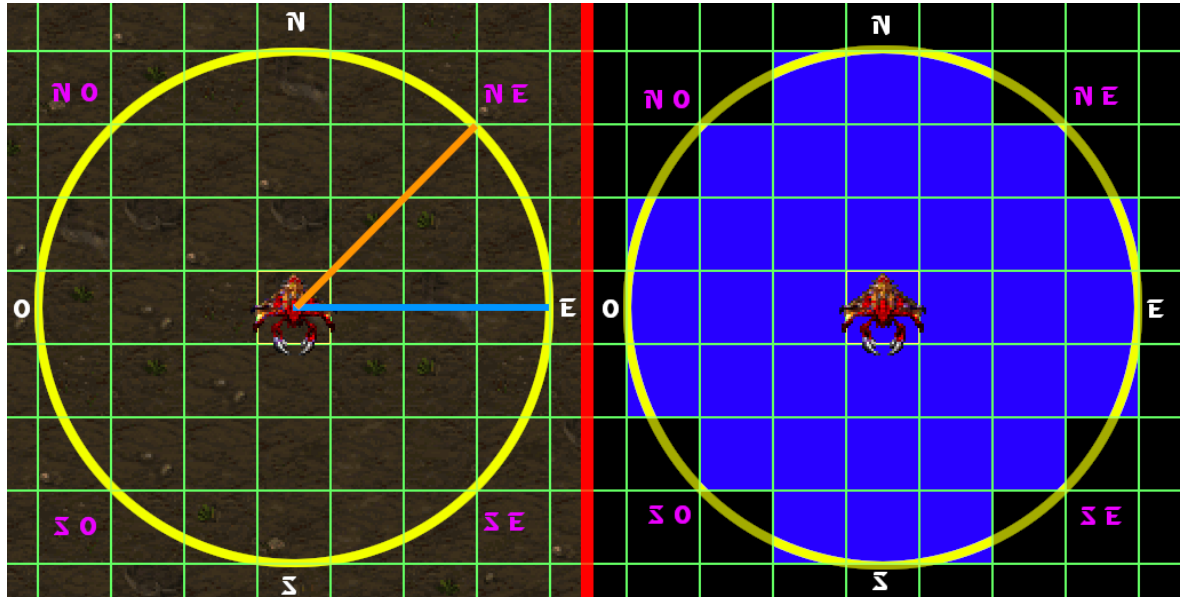


Figura H.1: Ejemplo de distancia real.

Al ser una distancia real, no varía el valor de la distancia entre un eje cardinal⁴ y uno intercardinal⁵, ya que de ser así, la línea azul y la línea naranja (que tienen un módulo ambas de 3,5 casillas) no coincidirían en el círculo amarillo. Es decir: la distancia real independientemente de la dirección en la que se mida es siempre la misma.

En la vista de la derecha se puede apreciar en color azul las casillas que estarían dentro de la distancia de 3,5 casillas respecto de la casilla ocupada por el zumbido.

²Información de lo que es un zumbido: ver subsección M.8.4.

³La posición de las unidades se halla en su centro. En la vista de la izquierda de la figura H.1, el centro exacto del zumbido está en la convergencia de la línea naranja y de la azul.

⁴Siendo desde el punto de vista del zumbido las cuatro direcciones cardinales norte(N), este(E), sur(S) y oeste(O): se entiende por ejes cardinales al eje vertical al eje que pasa por el norte(N) y por el sur(S) y al eje horizontal al eje que pasa por el este(E) y el oeste(O).

⁵Siendo desde el punto de vista del zumbido las cuatro direcciones intercardinales noreste(NE), sureste(SE), suroeste(SO) y noroeste(NO): se entiende por ejes intercardinales los ejes cardinales rotados sobre el plano 45° (un eje que pasa por el noroeste y por el sureste y otro eje que pasa por el noreste y por el suroeste).

H.1.2. Distancia optimizada

Denominada así por englobar todas las posiciones que se encuentren dentro del espacio encerrado por contar una determinada cantidad de casillas en las cuatro direcciones cardinales y en las cuatro intercardinales. Al ser las casillas cuadradas, la distancia real difiere de si se trata una dirección cardinal o de una intercardinal, como se aprecia en la figura H.2. Es una manera optimizada de calcular una distancia pues solo hay que obtener el máximo (función “max”) de las componentes restadas en las coordenadas de dos puntos. Por ejemplo, dados dos puntos ‘A’ y ‘B’ su distancia optimizada ‘d’ será:

$$d_{optimizada} = \max(A_x - B_x, A_y - B_y)$$

En la práctica es más sencillo todavía pues nunca se busca averiguar una distancia, sino obtener lo que está a una distancia dada, así que se recorre la sección cuadrada del escenario que se forma con dicha distancia (sin necesidad de usar la función “max”).

En la figura H.2 se puede ver la distancia optimizada de 3,5 casillas alrededor de un zumbido en todas las direcciones como un cuadrado de color amarillo. Gráficamente cualquier espacio encerrado desde un punto por una distancia optimizada de ‘X’ casillas es un cuadrado de $X * 2 + 1$ casillas de lado con centro en dicho punto (el zumbido en este caso).

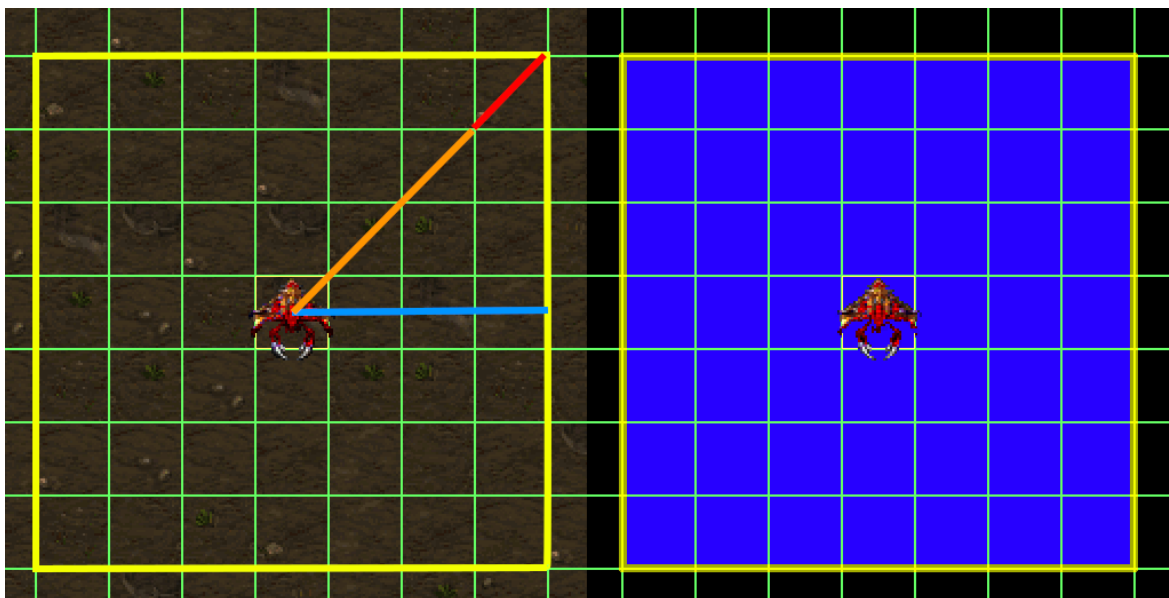


Figura H.2: Ejemplo de distancia optimizada.

Una distancia de 3,5 casillas abarca más en distancia optimizada que en distancia real, ya que cubre más espacio en las direcciones no cardinales. Esto tiene repercusiones dentro de Army of Darkness, por ejemplo, el movimiento de las unidades al moverse en una dirección distinta sigue esta intuición y gráficamente avanzan más espacio al

desplazarse en una diagonal aunque lo que tardan en moverse de una casilla a otra es siempre lo mismo. Al no tener que hacer apenas cálculos para obtener esta distancia es prácticamente inmediato obtener una sección del escenario como la representada en azul en la figura H.2 y se utiliza en muchas tareas como buscar unidades o minerales cercanos.

Esta distancia podría ser completamente opcional, al menos en la mayoría de casos. Ya que siempre que se requiere esta distancia es para analizar zonas cercanas (por lo general por debajo de 10 casillas de distancia), y se podría precalcular y guardar como constantes qué casillas estarían dentro de una distancia real (que es más precisa y costosa) y cuales no. Pero se trata de una vía de optimización que se ha descartado al haber otras en el proyecto con mayor prioridad y peso, y al no jugar un papel tan primario la optimización del proyecto.

H.2. Gestión y acceso de la memoria

El mayor problema del proyecto ha sido con la gestión de memoria, concretamente el tamaño de las texturas usadas por la biblioteca de SDL [7]. Una textura dentro de SDL es un tipo de dato fuertemente encapsulado que de manera optimizada permite renderizar en pantalla una imagen o parte de ella eficientemente. Aunque luego de la textura solo se muestre una parte, la biblioteca está diseñada para cargar las imágenes directamente y convertirlas en texturas, procesándolas y analizando si exceden el tamaño permitido o no al crear la textura.

Es decir, aunque de una textura solo se vaya a mostrar una pequeña parte en pantalla, la biblioteca impide crear texturas cuyo tamaño exceda ciertos límites, como era el caso. Los gráficos de Armygeddon se obtuvieron en sprites⁶ en archivos con formato BMP⁷, donde cada archivo es una imagen en forma de una larga columna con todas las posibles imágenes que podía tener un tipo de entidad. Por ejemplo: una unidad que tuviera una animación de una imagen para mostrar estar quieta, otra de cinco imágenes para mostrar estar atacando y otra cuatro para moverse, dan un total de $(1 + 5 + 4) * 16 = 160$ imágenes distintas⁸.

Lo problemático fue que este error se descubre en una fase avanzada del proyecto al probar Armygeddon en equipos con menores características, una vez que ya se había hecho un sistema de indexación adecuado para generar las animaciones de las entidades

⁶Un *sprite* es el conjunto de imágenes que compone las animaciones de un elemento gráfico de un videojuego.

⁷https://en.wikipedia.org/wiki/BMP_file_format (último acceso: 25/8/2023).

⁸El factor '16' es el número de variaciones de todas las imágenes según hacia qué dirección está orientada una unidad, ya que pueden atacar en todas las direcciones, estar quieto en todas las direcciones, etc.

(ver anexo G.3).

Al ser la biblioteca relativamente avanzada para no considerar sprites tan grandes (si que considera sprites de cierto tamaño, principalmente porque son cuadrados y no en forma de columna), se encuentran dos opciones:

- Dividir todas las imágenes del juego y las que fuesen utilizándose a medida avanzara el proyecto.
- Dividir las imágenes en carga de la misma manera que la biblioteca puede mostrar solo parte de una textura.

Finalmente se utilizó la segunda opción para no descartar el sistema de indexación de los gráficos utilizado, y porque se conocía que en la biblioteca Pygame de Python, la cual está basada en SDL, había una función con este mismo propósito. Se encontraron operativas obsoletas que sorteaban la encapsulación de SDL en su documentación y, que permitían realizar un copiado parcial de los datos de los sprites a tipos de datos intermedios. Estos tipos de datos sí que tendrían cada uno solo una imagen del sprite y con ellos sí se podían crear después correctamente las texturas.

También el manejo de los punteros, identificadores en red para las entidades, y las respectivas serializaciones de ambos, han sido causa de muchos errores y problemáticas que se estiman que la eficiencia del lenguaje escogido (C++) no compensa.

Anexos I

Interfaces

Este anexo se divide en cuatro secciones: una para explicar las diversas pantallas de Armygeddon (ver anexo I.1), otra para detallar los elementos accionables por el usuario en dichas pantallas (ver anexo I.2), otra sección para explicar cómo se ha tenido en cuenta la accesibilidad en la elección de los colores de las interfaces (ver anexo I.3) y finalmente una sección para profundizar en los gráficos y los sonidos utilizados en Armygeddon (ver anexo I.4).

I.1. Pantallas

1. Inicialmente se requerían tres pantallas para la recreación básica de StarCraft (ver figura I.1):

- *Pantalla principal.* Aquí se va a dividir el contenido de Armygeddon en función de su tipo. Por ejemplo, separar ajustes de modo multijugador, del editor o del modo individual; además de una opción para salir.
- *Creación de partidas.* En esta pantalla se espera seleccionar la partida a jugar, y configurarla en su caso.
- *Pantalla de juego.* Donde se juegan las partidas.

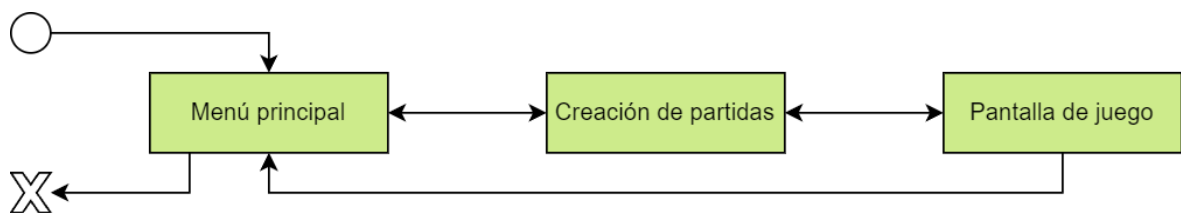


Figura I.1: Primera evolución del modelo de pantallas de Armygeddon.

2. Para añadir el editor de escenarios se tuvieron en cuenta otras dos pantallas (ver figura I.2):

- *Editor de escenarios.* Es el propio editor en sí, donde se van a editar los escenarios.
- *Creación de escenarios.* Pensada para escribir el nombre y la dimensión del escenario.

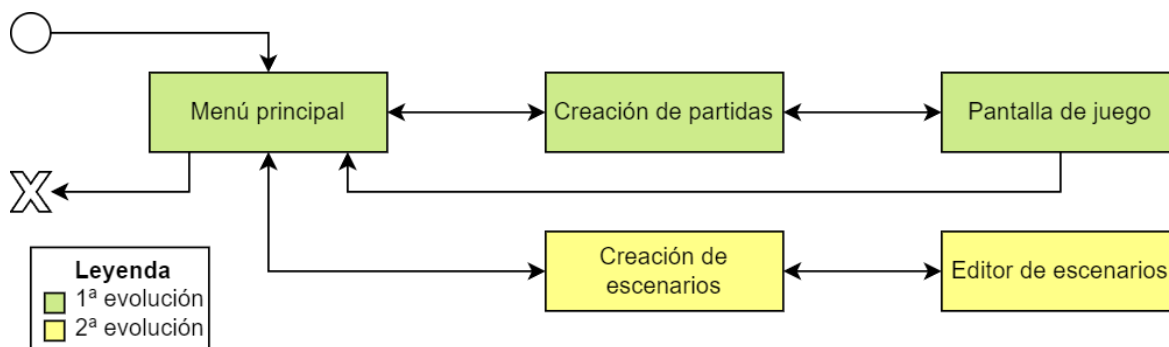


Figura I.2: Segunda evolución del modelo de pantallas de Armygeddon.

3. En este momento aparecen otras dos pantallas que son relevantes (ver figura I.3):

- *Ajustes.* Desde el comienzo del proyecto se pensó que los ajustes podrían englobarse desde una ventana emergente, pero al aumentar el número de controles y al implementar características que son altamente configurables como mover la cámara con el ratón, las animaciones de las pantallas o la aparición de paneles emergentes, se optó por una pantalla dedicada a esta tarea.
- *Fin de partida.* Esta pantalla se añade ya que se quiere generar datos de alguna manera sobre el resultado de las partidas, sobre todo al poder haber más de dos jugadores y ser más complicado saber qué ha pasado al acabar.

4. En vista que la pantalla de creación de escenarios resultó muy sobrecargada y aún había que introducir la selección de escenarios para continuar la edición de un escenario ya creado, se tuvieron que añadir cuatro pantallas adicionales (ver figura I.4):

- *Carga de escenarios.* Aquí se espera gestionar qué escenarios hay creados. Seleccionar cual cargar en el editor, además de previsualizar, eliminar, renombrar y duplicar escenarios.
- *Menú del editor de escenarios.* Las dos pantallas que acceden directamente al editor (creación y carga de escenarios) son consideradas ambas igual de

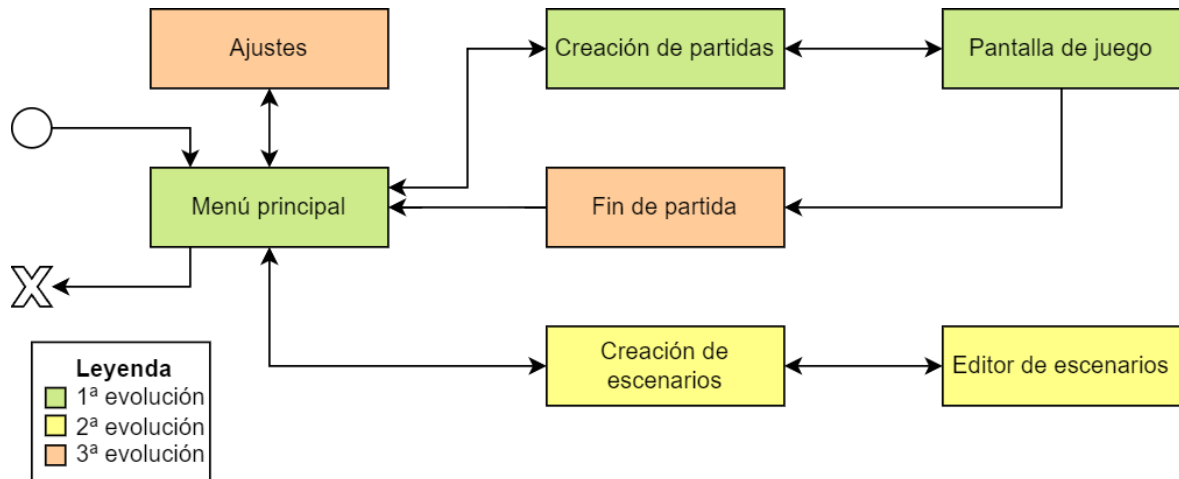


Figura I.3: Tercera evolución del modelo de pantallas de Armygeddon.

importantes conceptualmente (no tienen prioridad una sobre otra). Por ello, no se debería acceder desde una a la otra sino desde fuera y por eso se crea esta pantalla. La alternativa era acceder desde la pantalla principal pero este estaba muy sobrecargado.

- *Carga de partidas.* Había que añadir la carga de partidas (partidas ya empezadas sin acabar) y por la misma razón de evitar sobrecargar la pantalla de creación de partidas, y por conservar la disociación entre creación y carga, se incluye esta pantalla.
- *Elección de juego.* Motivada por lo mismo que al menú del editor de escenarios.

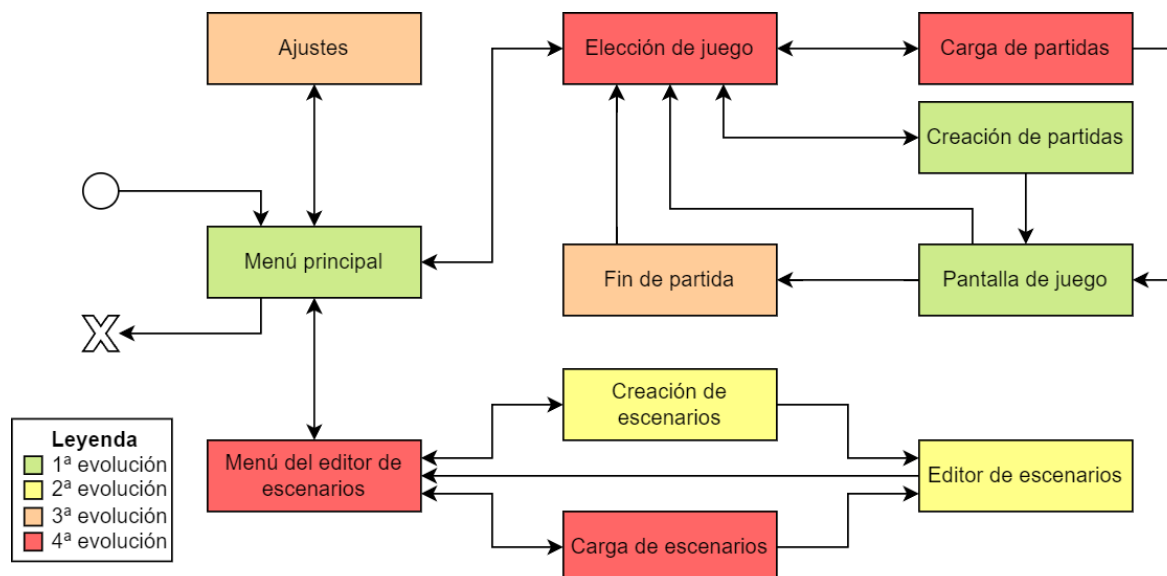


Figura I.4: Cuarta evolución del modelo de pantallas de Armygeddon.

5. En este último paso se añadieron seis pantallas para cubrir algunos aspectos

opcionales de Armygeddon (ver figura I.5):

- *Pantalla inicial.* Conforme avanzaban las versiones de Armygeddon, la espera inicial mientras se cargan todos los recursos gráficos y de sonido estaba aumentando por encima de un par de segundos. Es por esto que mejor que una ventana en blanco se opta por una pantalla inicial de carga en la que además poder poner el título.
- *Menú de juego multijugador.* La modalidad multijugador¹ se basa en la conexión con otras instancias de Armygeddon (que pueden estar o no en la misma máquina) y para ello se busca un diseño de “sala de jugadores”. En vez de emparejar a los usuarios entre sí al mismo nivel, se intenta que uno de los usuarios tenga que crear una “sala” donde esperar al resto de usuarios con los que se va a jugar² la partida. Se busca que en esta pantalla se escoja y se configure precisamente eso.
- *Información del editor.* Al tratarse Armygeddon de un videojuego complejo, en vez de añadir un tutorial se optó por añadir una lista de conceptos con información explicativa para facilitar la comprensión de las características del editor.
- *Información del juego.* Por lo mismo a la pantalla anterior, se hizo una pantalla paralela con información sobre conceptos del funcionamiento de las partidas en Armygeddon.
- *Menú de juego individual.* Al añadir el menú de juego multijugador, y la información del juego, se decide desplazar la carga y creación de partidas individuales a otra pantalla por claridad y no sobrecargar más la de elección de juego.
- *Sala multijugador.* Una vez aclarado el rol del usuario dentro de la sala multijugador como anfitrión o como huésped en la pantalla de elección multijugador, inundar la misma pantalla con las funcionalidades de la propia sala multijugador, y con las de creación de partida individual se estimaba peor idea que crear esta pantalla extra para dicho propósito.

¹En realidad el concepto es “multijugador en línea”, pero por claridad y convenio se utiliza a secas “multijugador” para referirse al modo en línea e “individual” para referirse a partidas que solo ocurren en una instancia de Armygeddon. Aun a pesar de que en ambos modos pueda haber hasta ocho jugadores artificiales, solo en el modo multijugador puede haber más de un jugador humano o híbrido.

²No tiene porqué ser un juego de usuarios. Por ejemplo, puede crearse una partida para que la jueguen ocho jugadores artificiales ejecutándose cada uno de ellos en las distintas instancias de los hasta ocho usuarios conectados.

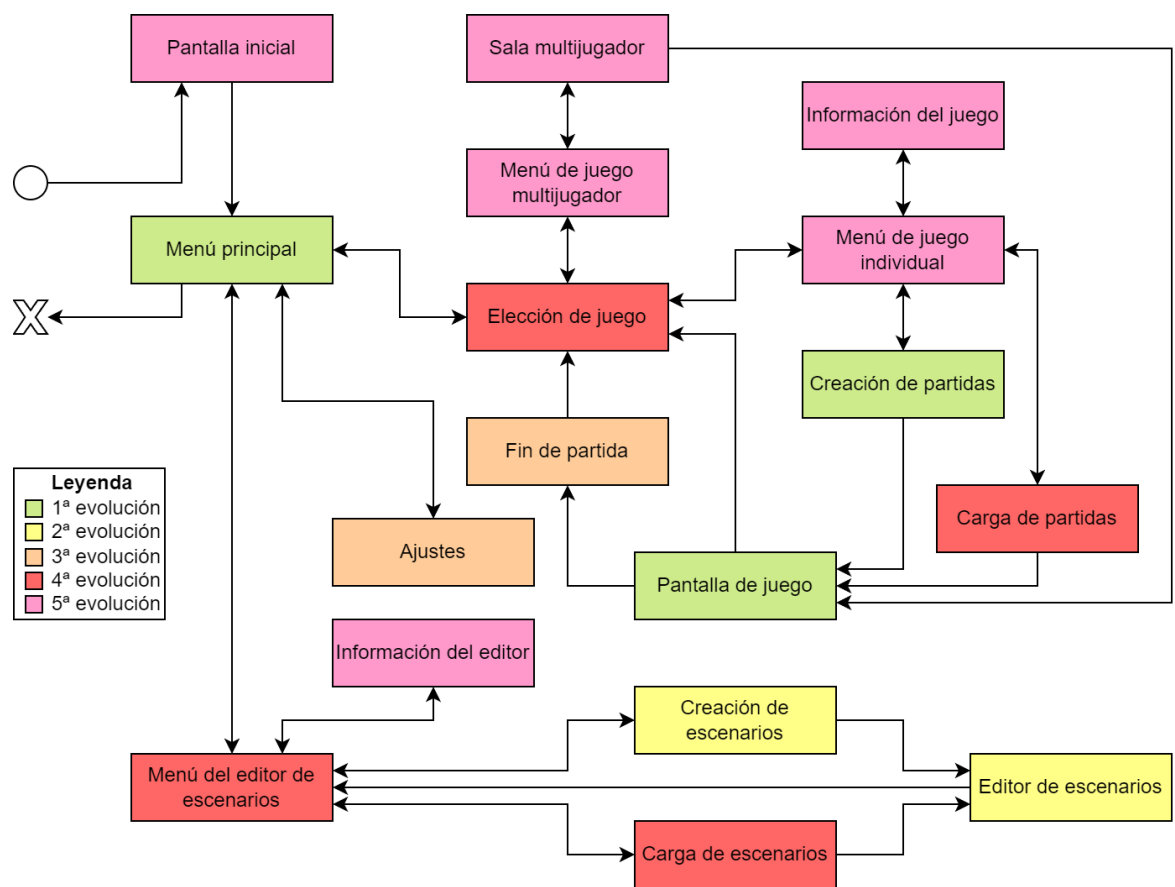


Figura I.5: Quinta evolución del modelo de pantallas de Armygeddon.

Existe una pantalla más que no se ha terminado de implementar que es la *carga de partida multijugador*. La razón de su rechazo al final fue que la funcionalidad de carga de partida para varios usuarios se estimó repetitiva y como un aglomerado de otras funcionalidades ya existentes: carga de partidas individual, lectura y configuración de partidas guardadas, y sala multijugador.

I.2. Elementos reactivos

Un elemento reactivo es un elemento que reacciona a la entrada de los usuarios. Por ejemplo, el elemento reactivo más complejo sería el tablero de una partida (la parte tangible de un escenario).

Los elementos reactivos son de índole muy variada, pues se trata de un videojuego futurista, y para notificar al usuario de cuales son los elementos reactivos y de qué tipo son, o qué comportamiento ocasionan, se proceden toda una serie de estándares.

I.2.1. Implementación

Dichos elementos reactivos a la entrada del usuario están implementados de la siguiente manera en su mayoría:

- Son seleccionables. La selección³ es una característica informativa de cara al usuario que representa el alcance del elemento que está siendo apuntado por los dispositivos de entrada (teclado o ratón) y que en el caso de hacer clic o pulsar el control *presionar*, dicho elemento se va a activar.
- Son activables. Se activan de las maneras ya mencionadas, por teclado o ratón. A su vez pueden ser activables de las siguientes maneras:
 - Botón simple (ver figura I.6). Las acciones dentro de una partida, o las herramientas del editor son ejemplos de este tipo de elementos reactivos. Producen una acción y por lo general se bloquean o desaparecen. Su aspecto es un dibujo representando el evento que ocasionan o un rectángulo con un texto que lo explique. Suelen venir acompañados de un panel emergente con información relevante si se posa el cursor lo suficiente encima de estos.
 - Marca (ver figura I.7). Similar a un interruptor con dos estados, siendo este activable y desactivable. Su forma es la de un cuadrado pequeño con fondo uniforme.

³No confundir con la *selección de entidades* que se puede realizar en partida, la cual puede coexistir con esta selección.



Figura I.6: Posibles estados de un elemento reactivo simple: botón 'Guardar' de ajustes.

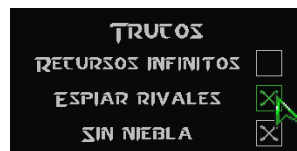


Figura I.7: Ejemplo de marcas: ajustes de trucos al crear una partida.

- Botones de radio (ver figura I.8). Vienen en grupos, para permitir ajustes en los que solo una opción esté activada, y siempre al menos esté activa una. Su apariencia es de varios círculos con fondo uniforme en una misma línea, columna o zona. A veces se muestran como rectángulos de texto en los que se ilumina de rojo⁴ el texto al estar activos, este caso es una variación de botones de radio que intenta expresar *interfaces de paso* más que un *conjunto de marcas excluyentes entre sí*.

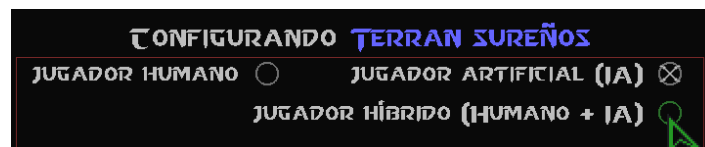


Figura I.8: Ejemplo de botones de radio: tipo de jugador al crear una partida.

- Transición. Producen cambios en la información que se muestra al usuario. Estos a su vez pueden ser de dos tipos:
 - Menor (ver figura I.9). Cambian parte de la pantalla, ya sea el interior de un panel, o el contenido de un campo. Suelen ser rectángulos con texto o un icono (siempre que no sea un lápiz) dentro.

⁴Las categorías de la pantalla de ajustes (*Teclado*, *Sonido*, *Vídeo* y *Extra*) son un ejemplo de este tipo especial de botones de radio.



Figura I.9: Ejemplo de botones de transición menor: Ajustes de avatar en línea.

- Mayor (ver figura I.10). Cambian notoriamente la información que se muestra, ya sea mostrando una ventana emergente importante o porque provoque un cambio de una pantalla a otra. Son los más variados, pueden ser rectángulos con texto conectados por un soporte metálico hacia fuera de la pantalla, animaciones grandes de satélites, estrellas, etc. acompañadas de texto, o bien los botones rectangulares con “Salir” escrito.



Figura I.10: Ejemplo de botón de transición mayor: Botón principal del editor.

- De escritura (ver figura I.11). Activo, y mientras no se seleccione otro elemento, transforma parte⁵ de la entrada por teclado a texto en vez de activar las órdenes oportunas. Tienen el aspecto de cuadrados pequeños con un lápiz dentro y están posicionados cerca del texto que modifican.

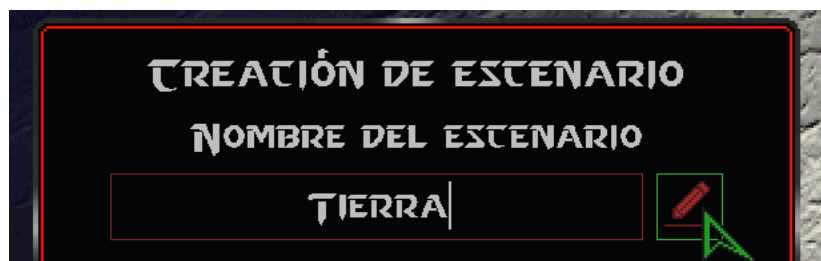


Figura I.11: Ejemplo de botón de escritura: Nombre del escenario en la creación de escenarios del editor.

⁵Las órdenes de *presionar* o *salir* no se ven afectadas.

- Deslizante (ver figura I.12). Es un elemento desplazable mientras se mantenga activo, permitiendo al usuario afectar a otra parte de la interfaz con dicho desplazamiento.



Figura I.12: Ejemplo de botón deslizante: Barra deslizante en ajustes de teclado.

- Pueden estar bloqueados, y a pesar de ser seleccionables no son activables según se den las circunstancias.

Como ya se ha mencionado el *tablero de un escenario en una partida o en el editor* es un elemento reactivo complejo que no se puede detallar aquí en su totalidad. Sin embargo se entiende a través de la *información conceptual en Armygeddon*⁶ y del manual de usuario en el anexo M.6.13 qué tipo de órdenes se pueden ejercer en una partida.

I.2.2. Respuesta audiovisual

Los elementos reactivos por lo general provocan respuestas sonoras con solo seleccionarlos (más detalle en anexo I.4.3). Se utiliza un sonido largo y lejano si el elemento reactivo es de transición mayor, y un sonido breve y nítido para casi todos los demás casos. En cuanto a la activación, también suelen producir una respuesta sonora de la misma manera: si el elemento reactivo es una transición mayor se escuchan sonidos distintos a los demás casos. Por ejemplo, se escucha un sonido de *intentar abrir un portón cerrado* cuando se intenta activar un elemento reactivo de transición mayor bloqueado. Esto es además de por inmersión, principalmente porque el usuario tiene que saber que puede perder parte de la información introducida al activar ese elemento reactivo, ya que cambiaría de pantalla en el interfaz.

Finalmente destacar la importancia del coloreado de los elementos reactivos, para

⁶Dicha información se puede consultar en las pantallas de Armygeddon *Información del editor* e *Información del juego* (ver manual M.3).

señalizar el tamaño del elemento reactivo, si está bloqueado o no, y también a veces pistas sobre su acción asociada. Como se aprecia en la figura I.13 no solo está iluminado el *botón simple* con el ‘-’ teniendo el cursor encima, sino que también se ilumina la última celda de la barra de volumen pues es la que se va a desactivar si se hace acciona dicho elemento reactivo.



Figura I.13: Ejemplo de respuesta visual sobre elemento reactivo.

I.2.3. Organización

Los elementos reactivos están organizados para intentar seguir el orden de lectura⁷. Esto se refleja en la navegabilidad por las pantallas del programa que se muestran a la izquierda los botones de “Salir” o “Cancelar”⁸ y a la derecha los botones que ahondan más en la interfaz del programa. También se puede apreciar en los propios elementos reactivos donde si se puede se coloca el texto encima o a la izquierda de un elemento reactivo.

Además los elementos reactivos, y en general toda la información que se muestra al usuario, está agrupada por funcionalidad y por importancia de la misma manera. Por ejemplo, como se aprecia en la figura I.14, en las pantallas de elección de escenarios se sitúan a la izquierda la lista de escenarios, y a la derecha la información de estos escenarios. Y encima de ambas cosas el título que indica que se está en una selección de escenarios.

I.3. Estándares cromáticos

Existen varios estándares que se utilizan para ayudar al usuario entender y poder disfrutar Armygeddon que se basan solamente en las paletas de colores utilizadas. Un claro ejemplo es el coloreado de la cuadrícula en el editor de escenarios (más detalle

⁷De izquierda a derecha y después, de arriba a abajo. En este caso.

⁸La pantalla de ajustes es el único caso donde esto no se cumple pues al igual que con el oscurecimiento de la fuente en esta pantalla, o del fondo, se intenta conseguir que el usuario sienta que esta pantalla no es parte del videojuego para no obstaculizar la inmersión del propio usuario.

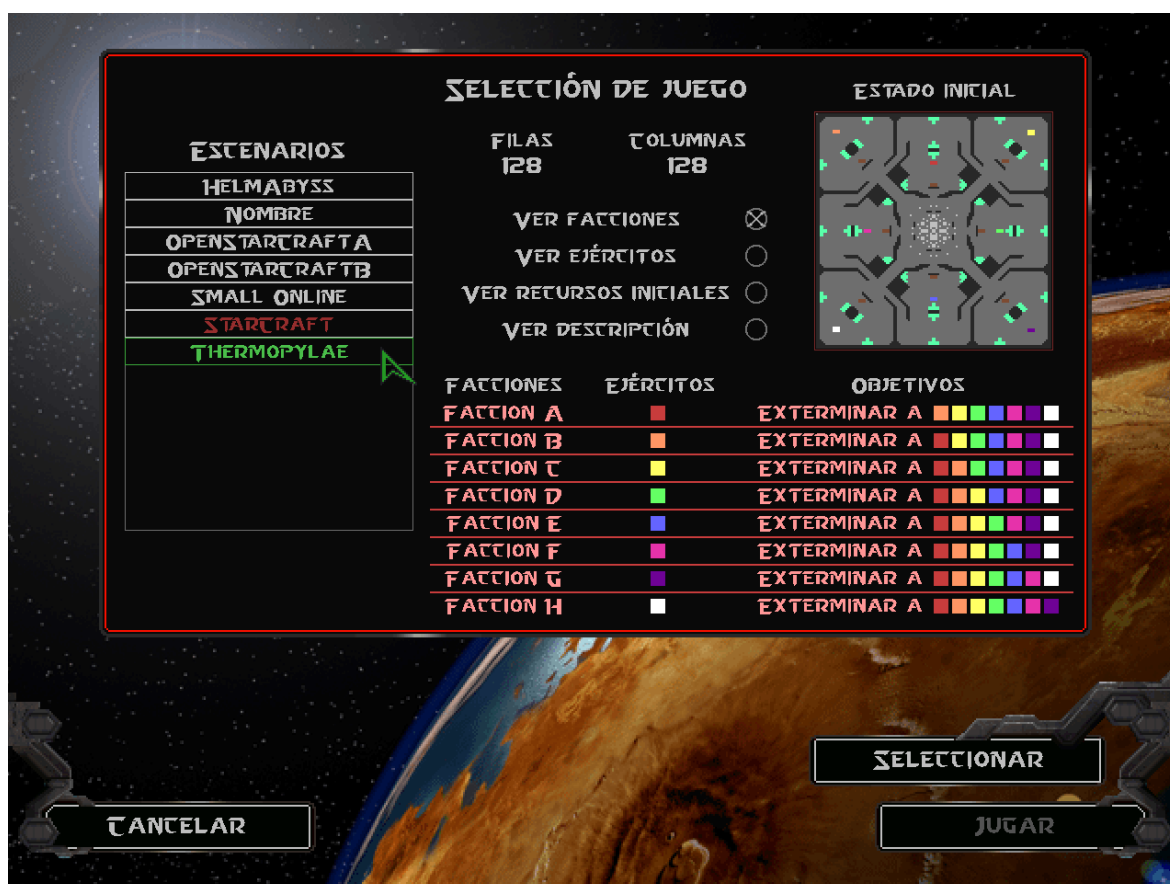


Figura I.14: Ejemplo de organización de los elementos reactivos: Creación de partidas.

en el anexo M.49). Para todos ellos se suele tener en cuenta el verde y el rojo, en consonancia por ejemplo con los colores de los semáforos.

Al igual que StarCraft, Armygeddon es de temática oscura, predominando letras claras sobre fondo oscuro en su totalidad. Hay varias situaciones en las que se tienen que reflejar la información de hasta ocho ejércitos, y eso incluye también el color del propio ejército. El programa limita bastante la información que se puede mostrar en pantalla al ser la resolución mínima tan baja, de hecho, la propia fuente⁹ como más se utiliza es en su tamaño mínimo (15 píxeles), a partir del cual dejan de reconocerse algunas letras¹⁰.

La solución, entre otras cosas, fue poner el color de la fuente del color del propio ejército. Pero la fuente era de color blanco inicialmente, de verde en algunas selecciones, y de rojo si se activa, y eso son tres colores que ya estaban en uso por los colores de los ejércitos generados. Así que se cambiaron los colores de la fuente a gris claro, verde oscuro y rojo oscuro respectivamente en un ejercicio de *responsabilidad cromática* a partir de la versión 2.0.3 del proyecto (ver anexo L).

En cuanto a los ejércitos, se escogieron los del arcoíris para poder ser diferenciados fácilmente. A estos había que sumarles el blanco (color claro y distinguido) ya que se requerían ocho colores para los ejércitos de los usuarios. Para el ejército neutral se escogió el gris oscuro para poder distinguirse de todos los demás ejércitos que tenían colores claros, ya que no puede estar controlado por ningún usuario y siempre va a tratar de obstaculizar a los demás ejércitos.

En cuanto a los colores de los nombres de las facciones, que tienen el mismo propósito que el de fusionar el color del ejército con el nombre, dependen del objetivo que tenga esa facción. Se escogieron colores menos saturados para diferenciarlos de los demás colores de fuente.

Los elementos reactivos siguen el siguiente código:

- Blanco. Se trata de una zona seleccionable del elemento reactivo.
- Verde. Se está seleccionando dicho elemento reactivo, delimita la zona de selección y la de cambio. También se suele iluminar algo más de verde si el elemento reactivo

⁹Solo se utiliza la fuente de títulos de StarCraft, a sugerencia del director. Pero se utiliza en tres tamaños diferentes: 15 píxeles para texto general, 18 para encabezados o texto destacado y 21 para títulos o botones grandes.

¹⁰Y eso que la fuente se genera a un tamaño mucho mayor, 66 puntos, para tener mayor definición en según qué casos y lo que se hace es escalarla a tamaños menores para mostrarla luego en pantalla. Se evita así un *efecto de pixelado* apreciado sobre todo en los conceptos de Armygeddon (pantallas de información, ver manual M.3) que es texto en tamaño real.

es activable.

- Rojo. El elemento reactivo está actualmente activo y no puede activarse más.
- Gris. El elemento reactivo no se acciona directamente desde aquí, pero puede alterarse eventualmente.
- Gris oscuro. El elemento reactivo está bloqueado y no importa cuánto se intente activar que no va a reaccionar.

I.4. Contenido audiovisual

En esta sección se detallan brevemente los gráficos (anexo I.4.1), la música (anexo I.4.2) y los efectos de sonido (anexo I.4.3) utilizados en Armygeddon.

I.4.1. Gráficos

En Armygeddon se utilizan en su mayoría gráficos del videojuego original StarCraft (anexo I.4.1), pero también se utilizan gráficos nuevos creados a partir de modificaciones de los originales (anexo I.4.1) o creados desde cero (anexo I.4.1).

Gráficos originales

Prácticamente toda la carga gráfica, incluida la fuente del programa, corresponde a los gráficos utilizados en el videojuego StarCraft original en 1998 obtenidos a través de tutoriales para crear modificaciones del juego [6]. También gran parte de las figuras de este documento que muestran gráficos o bocetos de las entidades son en su defecto del videojuego original StarCraft o de la página de documentación citada en la bibliografía [9].

El título de Armygeddon (anexo M.1), y su icono (ver figura I.15) se han creado utilizando un efecto sobre la fuente de StarCraft en una página web¹¹ y posteriormente haciendo modificaciones utilizando Gimp¹² [4].

Gráficos modificados

Los gráficos de los ejércitos (exceptuando el ejército rosa) se generan utilizando los del ejército rosa originales y con Gimp cambiando su mapa de colores para obtener ejércitos de colores distintos. Ver ejemplo en la figura I.16.

¹¹El efecto se llama “Style-GlitchA” y la página es: <https://fontmeme.com/starcraft-font/> (último acceso: 14/8/2023).

¹²Gimp es una herramienta de edición de imágenes de código abierto: <https://www.gimp.org/> (último acceso: 14/8/2023).



Figura I.15: Icono de Armygeddon.

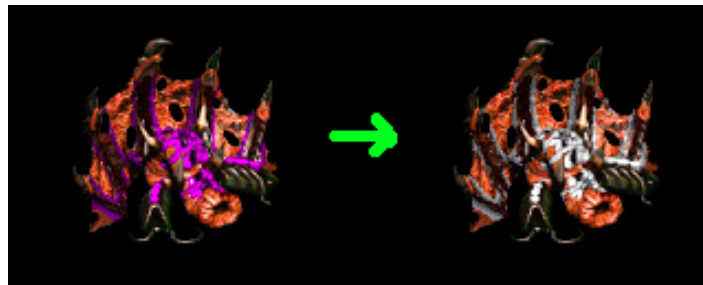


Figura I.16: Ejemplo de cambio de color de una guarida de hidraliscos.

En la figura I.17 están señalados con verde los ocho colores que se alteran para generar los distintos ejércitos.

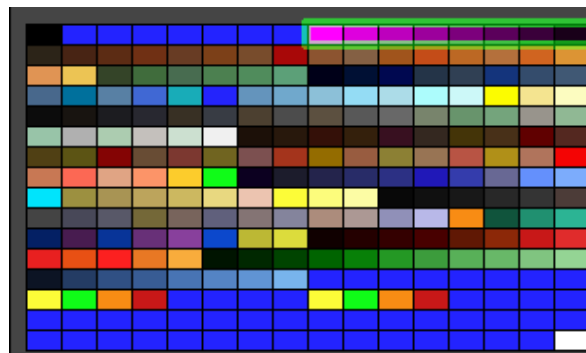


Figura I.17: Mapa de colores de una entidad de StarCraft.

También se han editado una amplia variedad de elementos de la interfaz, como por ejemplo la interfaz inferior del editor de escenarios, que es una fusión de tres interfaces distintas (ver figura I.18).



Figura I.18: Interfaz de las tres interfaces de razas combinadas del editor de escenarios.

Gráficos nuevos

Los iconos de los modos de edición, así como algún dibujo de entidad o los iconos de algunos elementos reactivos han tenido que hacerse desde la nada. Ver la figura I.19.

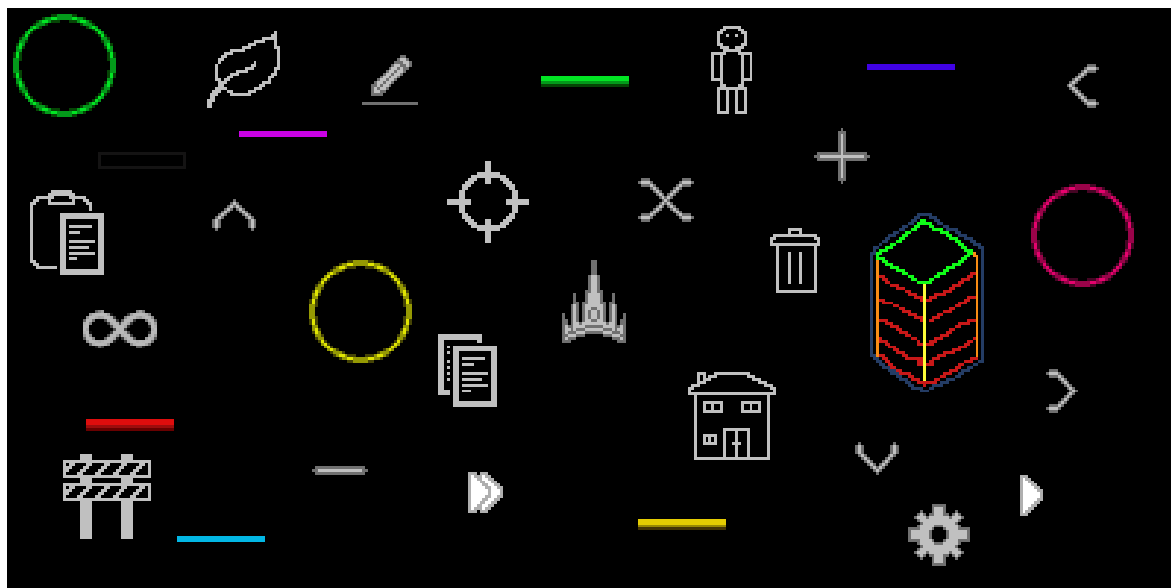


Figura I.19: Algunos gráficos que se han requerido crear desde cero y que eran costosos de generar por el programa.

Existen casos como las barras de ajustes del volumen de la música (ver figura anterior I.13) que se han podido generar directamente desde código.

I.4.2. Música

En todas las pantallas del interfaz de usuario hay música salvo en el menú inicial. Esto es deliberado pues se busca conseguir un efecto doble: que impacte más en el usuario el título de Armygeddon, y que se asocie la música a lo divertido, que es cuando el programa ya ha terminado de cargar y se está en el menú principal.

En el editor de escenarios suena *Terran One*, pues además de ser una de las piezas más icónicas de StarCraft al ser de las primeras que se escucha, tiene un ritmo calmado que favorece la creatividad requerida para creación de escenarios. Se había descartado con anterioridad para este menú y para otros todo el set de canciones de la raza protoss, al tener este un aire trágico e impersonal, asociado a la raza. También se descarta el set de canciones de la raza zerg, al considerarse como se pretende de la raza, oscuro y violento.

Dentro de partida existe un abanico de posibilidades en función del tipo de partida y de la raza que se esté jugando donde todas las pistas tienen la misma probabilidad de salir (tabla I.1).

Tipo de partida y raza	Alternativas de pistas de música
Jugador humano o híbrido protoss	<i>Protoss One</i> <i>Protoss Two</i> <i>Protoss Three</i>
Jugador humano o híbrido zerg	<i>Zerg One</i> <i>Zerg Two</i> <i>Zerg Three</i>
Jugador humano o híbrido terran	<i>Terran One</i> <i>Terran Two</i> <i>Terran Three</i>
Simulación	Cualquiera de las anteriores

Tabla I.1: Posibles pistas de música para las partidas

Las pistas de música de la pantalla de fin de partida también dependen de la raza jugada o de si es una simulación y además de si el jugador gana o pierde la partida (tabla I.2).

Tipo de partida	Resultado	Pista de música
Jugador humano o híbrido protoss	Victoria	<i>Protoss Victory</i>
	Derrota	<i>Protoss Defeat</i>
Jugador humano o híbrido zerg	Victoria	<i>Zerg Victory</i>
	Derrota	<i>Zerg Defeat</i>
Jugador humano o híbrido terran	Victoria	<i>Terran Victory</i>
	Derrota	<i>Terran Defeat</i>
Simulación	Victoria o derrota	<i>Terran One</i>

Tabla I.2: Posibles pistas de música para fin de partidas

En todas las demás pantallas se reproduce la pista *StarCraft Main Title*.

Cada pista de música se repite un número indeterminado¹³ de veces, y tienen una longitud de dos a cinco minutos salvo las pistas de fin de partida que tienen una longitud entre medio minuto y un minuto.

Las transiciones de pistas de música son inmediatas para acompañar las transiciones entre pantallas en las que cambia la música que también lo son.

¹³Se intenta que se repita indefinidamente pero el límite de SDL mixer es de aproximadamente 65000 veces, según la documentación de la especificación.

I.4.3. Efectos

Los efectos de sonido están en su mayoría ligados a los elementos reactivos, su comportamiento y su estado (más detalle en el anexo previo I.2). Hay tres tipos de efectos de sonido en Armygeddon:

1. Sonidos de selección. Suenan cuando el usuario selecciona un elemento reactivo bien mediante cursor o bien mediante teclado.
2. Sonidos de respuesta. Suenan cuando el usuario acciona un elemento reactivo. Estos a su vez tienen dos variantes:
 - Sonidos de activación. Suenan cuando se ha conseguido accionar un elemento reactivo, esto ocurre si el elemento seleccionado no está bloqueado.
 - Sonidos de error. Suenan cuando se intenta accionar un elemento reactivo bloqueado.
3. Sonidos de evento. Suenan cuando ocurre un evento no relacionado directamente a la entrada / salida del usuario. Ejemplos: Unidades siendo seleccionadas, muriendo, obedeciendo una orden o apareciendo en partida. Ataques de unidades.

Anexos J

Posibles expansiones

En este anexo se explican algunas de las características que se iban a introducir en el proyecto o con las que se había teorizado un diseño en algún momento, mencionando su coste temporal aproximado y el impacto que se estima que tendrían en Armygeddon.

J.1. Fusión de terrenos

- Coste temporal: Un día.
- Impacto de la expansión: Aumento de la inmersión.

En Armygeddon las baldosas están limitadas a las casillas a las que se les asigna, dejando marcados los límites de las casillas en cuanto se mezclan dos tipos de baldosas. Esto no se aprecia en los cambios de zona horizontales o verticales, pero sí en los diagonales, ver figura J.1.

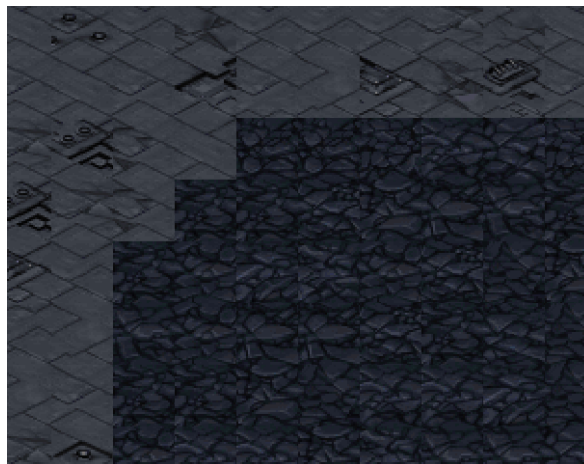


Figura J.1: Unión de terrenos en Armygeddon.

Aprovechando que se dispone de los gráficos necesarios para ello¹, la expansión consistiría en que al colindar dos terrenos distintos la baldosa no sea completamente cuadrada, sino que varíe en función de las casillas vecinas.

En la figura ejemplo J.2 se indica con la cuadrícula activa la línea que dividiría los terrenos de estar implementada la fusión de terrenos. Esto sería así suponiendo que el suelo que primero se ha colocado fuese el gris que hay en la parte superior izquierda y que luego se hubiera añadido el suelo de piedra en las mismas casillas que en la figura J.1.

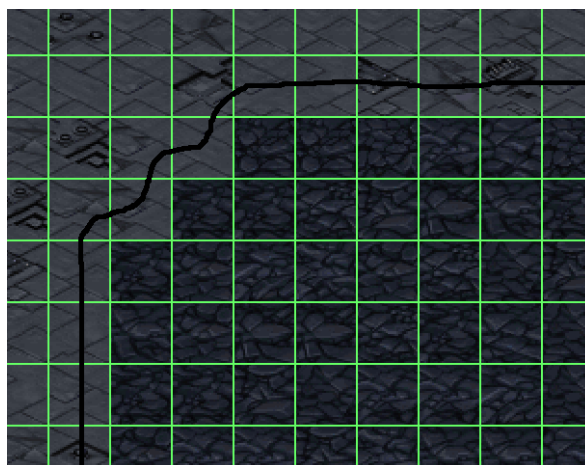


Figura J.2: Fusión de terrenos.

La idea consiste en que cada baldosa de un terreno colocada encima de otro terreno distinto “salpique” a las casillas vecinas haciendo una fusión de terrenos menos brusca. Luego para dibujar estas nuevas baldosas mixtas (las que tienen la línea negra encima en J.2) habría que dibujar primero la baldosa del terreno base y encima la baldosa del terreno “salpicado”.

J.2. Mejora del objetivo investigación

- Coste temporal: Un día.
- Impacto de la expansión: Mejora de la jugabilidad.

El objetivo de facción *investigación* es un objetivo que mezcla supervivencia con tener que obtener recursos, pues las maravillas (detalladas en el anexo M.6.17) requieren tiempo además de altas cantidades de gas y de cristal. Además de que las maravillas tienen que permanecer intactas hasta que se completa la investigación del número de maravillas objetivo.

¹Esta expansión se obtiene por ingeniería inversa estudiando los gráficos del videojuego original, pues tenían *casillas incompletas* con formas de esquinas o similares, que se estiman que eran para esto.

Aquí aparece un problema y es que en Armygeddon (al igual que en StarCraft), los recursos que se pueden obtener en partida están limitados. No hay problema por parte del gas, pues si se coloca un géiser ya hay en potencia gas ilimitado, pero los minerales sí que tienen un límite finito.

Es decir, la problemática va más allá de la responsabilidad del usuario que diseña el escenario sobre si ha puesto suficientes recursos para completar una partida de investigación. Puede ocurrir que se utilice todo el cristal disponible para otras tareas y “bloquearse” así a los ejércitos de investigación, ya que no tienen alternativa posible de progreso.

Se consideran dos soluciones a esta mejora del objetivo investigación que no tienen porqué ser excluyentes entre sí:

- Añadir una generación pasiva del cristal. De esta manera se obtendría el cristal necesario con el tiempo.
- Implementar mecanismos que calculen el cristal y el gas restante, y que terminen la partida para el jugador investigador si no le salen los números para lograr su objetivo. También puede implementarse un aviso en el editor para que no se permita crear escenarios de investigación sin los recursos suficientes.

J.3. Reproductor de música

- Coste temporal: Un día.
- Impacto de la expansión: Mejora de la jugabilidad, posible reducción o aumento de la inmersión.

Un reproductor de música dentro de partida permitiría cambiar entre las numerosas pistas de música que se utilizan en el proyecto o incluso reproducir pistas externas del usuario, además de permitir pausar la pista actual o ponerla desde el principio. Esta característica se ha visto en otros videojuegos como es el caso de *Metin2*².

El único problema sería que al no tener más espacio en pantalla dentro de partida habría que por ejemplo, introducir un panel nuevo donde tener espacio. Se podría sustituir el actual botón de “Salir” por uno de “Menú”, y que al activarse se muestre un panel en medio de la pantalla e introducir ahí tanto el reproductor de música como la opción para salir. También podría gestionarse las órdenes del reproductor mediante atajos de teclado y mostrar información en medio de la pantalla como mensajes temporales de por ejemplo, qué pista se está reproduciendo ahora.

²<https://gameforge.com/en-US/play/metin2> (último acceso: 15/8/2023).

Para cargar pistas propias y poder escucharlas en partida bastaría con designar una carpeta en el directorio de Armygeddon de la cual leer los archivos en formato *MP3*³ que el usuario introdujera.

J.4. Efectos en partida del terreno

- Coste temporal: Una semana.
- Impacto de la expansión: Mejora de la jugabilidad, aumento de la inmersión.

El tipo de terreno es actualmente una característica estética, pero los distintos tipos de baldosas podrían tener características pasivas. Por ejemplo, según el tipo de terreno podría hacerse que ciertas unidades se desplazasen más despacio, sufrieran daño, se recuperasen, atacaran más lejos, etc. Este tipo de característica es fundamental en videojuegos famosos como *Divinity 2*⁴.

El problema de esta expansión es que requeriría alterar muchas mecánicas y tendría muchas interacciones, sobre todo si se trata junto con la expansión de fusión de terrenos (ver anexo previo J.1). Habría que implementar una rutina más en la actualización de las entidades que tuviera en cuenta el terreno en el que está ubicada y habría que decidir si afecta a las entidades que ocupan más de una casilla como ocurre con casi todas las estructuras.

Además, habría que rebalancear los escenarios de ejemplo y repetir las pruebas.

J.5. Otras mecánicas de las razas

- Coste temporal: Un mes las listadas, aproximadamente una semana cada una.
- Impacto de la expansión: Mejora de la jugabilidad, aumento de la inmersión.

Se han implementado algunas mecánicas de las razas pero hay muchas que no, por mencionar algunas:

- *Mejoras de las unidades*. Esta mecánica varía según la raza pero es común a las tres razas. Consiste en añadir opciones a las estructuras que apliquen aumentos permanentes a las estadísticas de todas las unidades de un tipo. Habría que, al aplicar la mejora, aumentar las estadísticas de las unidades ya existentes y tenerse en cuenta para la creación de unidades futuras.

³<https://en.wikipedia.org/wiki/MP3> (último acceso: 15/8/2023).

⁴<https://divinity.game/> (último acceso: 15/8/2023).

- *Matriz de energía protoss*. Los protoss tienen un alcance limitado a la hora de construir estructuras: estas tienen que estar cerca de otras estructuras propias para conectarse de manera inalámbrica (simulando enlaces de la energía psiónica) entre sí hasta alcanzar la base para que se activen las funcionalidades de cada estructura. Esta mecánica permite que no se abuse de otras posibles mejoras como el *teletransporte protoss*, que permite a las unidades viajar inmediatamente entre un *portal* y otro (definición en el anexo M.7.6). Para implementarlo habría que tener en cuenta la destrucción de estructuras para ver si se pierde una conexión, y mostrar el flujo de energía al usuario a la hora de construir una estructura.
- *Rango de las unidades*. Las unidades suben de nivel al eliminar enemigos y ganan estadísticas por ello. De hecho, podría combinarse con la mecánica de *mejoras de las unidades* para aumentar aún más la profundidad en una unidad. La parte costosa sería el encontrar una manera de reflejar a simple vista el nivel de una unidad a todos los jugadores, por ejemplo, podría hacerse con una insignia sobre la barra de vida.

J.6. Sistema manual de creación de elementos

- Coste temporal: De una semana a varios meses, en función de la complejidad.
- Impacto de la expansión: Mejora de la jugabilidad, posible aumento o reducción de la inmersión.

Esta expansión trataría en un profundizar un nivel más en el *metajuego* que proporciona el editor de escenarios, permitiendo la edición de nuevos elementos para los escenarios. Serviría para aprovechar los sistemas genéricos de las entidades y reutilizarlos para crear entidades nuevas (siempre que se dispusiera de los gráficos) desde el propio editor. La complejidad iría desde crear nuevos tipos de suelo que solamente requeriría de los gráficos, hasta la creación de nuevas unidades que requeriría definir un formato de los gráficos (lectura de *sprites*⁵) o reutilizar los gráficos de otra unidad preexistente y variar sus colores desde el propio videojuego.

También se considera editar las entidades ya existentes en Armygeddon, de forma que se pueda alterar sus estadísticas, alcance, o sus gráficos. Pero se estima más apropiado y mucho menos costoso implementar esta característica a través de la *customización* (ver J.10).

⁵[https://es.wikipedia.org/wiki/Sprite_\(videojuegos\)](https://es.wikipedia.org/wiki/Sprite_(videojuegos)) (último acceso: 15/8/2023).

J.7. Carga multijugador

- Coste temporal: Una semana.
- Impacto de la expansión: Mejora de la jugabilidad.

Esta expansión es simplemente una mezcla de la carga de escenarios, la configuración de escenarios y la sala multijugador que no se ha implementado por ser solamente eso.

Si la dinámica de las partidas fuese más lenta y las partidas durasen más tiempo sí que sería más relevante esta característica, pues es más probable que no se llegue a acabar una partida y haya que continuarla en otro momento.

El diseño sería similar al de la sala multijugador que se requirió hacer un panel más comprimido para mostrar la información de los escenarios, y hacer lo mismo para el panel de carga de partidas.

J.8. Navegación por teclado completa

- Coste temporal: Menos de una semana.
- Impacto de la expansión: Mejora de la accesibilidad.

El sistema de navegación alternativo (mediante teclado) no engloba a todas las pantallas de Armygeddon. Dentro de una partida o en el editor de escenarios dicho sistema no está implementado debido a su complejidad.

Una posible implementación pasaría por mostrar una marca en el centro de la cámara y por añadir controles que simulen los dos botones del ratón en teclas, así como unos controles para poder también ajustar el zum que no sean la ruleta del ratón.

De este modo, acciones como la ‘selección de una zona’ funcionarían así:

1. Desplazar la cámara hasta situar la marca central en el inicio de la zona a seleccionar.
2. Pulsar y mantener el control alternativo al clic izquierdo del ratón.
3. Desplazar la cámara hasta situar la marca central en el final de la zona a seleccionar.
4. Soltar el control alternativo al clic izquierdo del ratón.

Para hacer este proceso menos complejo se podrían añadir más controles que realicen

atajos como un control que seleccione las unidades que se ven en la pantalla.

J.9. Rehumanización del jugador artificial

- Coste temporal: Una semana.
- Impacto de la expansión: Aumento de la inmersión.

Se puede aumentar aún más la naturalidad del jugador artificial haciéndole unas modificaciones:

- Haciendo que pueda fallar. Basta con que todas las órdenes generadas tengan una probabilidad muy baja de que puedan no enviarse o sufrir alteraciones.
- Cambiando el intervalo entre sus ciclos de actualización de manera aleatoria (dentro de unos márgenes).
- Introduciendo manías. Modificando los protocolos para que muestre cierta tendencia a realizar un tipo de acciones, por ejemplo, crear todas las invasiones sobre el mismo sector hasta que se resuelva, o ir a atacar a los trabajadores ignorando a los soldados.

J.10. Customización

- Coste temporal: Una semana.
- Impacto de la expansión: Aumento de la jugabilidad, posible aumento o reducción de la inmersión.

Existen muchos valores constantes en el código que definen y limitan el funcionamiento de Armygeddon. De exportarse dichos valores a un archivo que se leyera al ejecutar el programa, se permitiría fácilmente al usuario el poder modificarlos.

Ejemplos de dichos valores serían:

- Las estadísticas de las unidades, estructuras, minerales ...
- Parámetros internos del jugador artificial como el tamaño de los sectores lógicos, ponderación de las entidades para el cálculo de la amenaza, prioridades de los protocolos, velocidad de respuesta ...
- Límites como las dimensiones del escenario.
- Temporizadores como el tiempo de minado o de recolección.

J.11. Nuevo sistema de casillas y posiciones

- Coste temporal: Tres semanas.
- Impacto de la expansión: Aumento de la jugabilidad, posible aumento o reducción de la inmersión.

El sistema de casillas y el sistema interno de las posiciones (ver figura anterior 3.6) no resuelve el problema que tienen las unidades ya que no todas tienen el mismo tamaño y no caben en una casilla, siempre ocupan parte de la casilla que tienen encima. Esto no sería ningún problema salvo porque según el orden de dibujado del tablero (explicado en sección 3.2.1) se dibujan las casillas y sus ocupantes en el orden de izquierda a derecha y de arriba a abajo. Con este orden, y al cambiar las unidades de casilla en el momento en el que empiezan el movimiento hacia dicha casilla, estas a veces se dibujaban debajo de otra entidad que no debían. Además, de que la selección funciona con las casillas y las unidades al empezar el desplazamiento de una casilla a otra no están donde se les dibuja sino en la casilla objetivo.

Todo esto se solucionaría de la misma manera: asociar las unidades a un sistema de posiciones que no dependa de las casillas, de esta manera el tamaño de las unidades dependería de cada tipo de unidad y se permitiría además unidades que sobrepasaran el tamaño que da una casilla. Entre otras cosas esto permitiría por ejemplo el movimiento en las 16 direcciones y no solo en las ocho actuales.

Se requeriría un nuevo sistema de almacenamiento para acceder a las unidades, y cambiaría gran parte de la lógica del programa que se basara en la abstracción por casillas. Además de adaptar la búsqueda de caminos para que trabaje con grupos de posiciones de tamaño variable.

Anexos K

Resumen del trabajo práctico StarCraft

El trabajo práctico StarCraft [1] de la asignatura de Videojuegos se realizó durante los meses de marzo, abril y mayo del curso 2021-2022 por Alexandre Zheng, Alonso del Rincón, Juan Plo y el autor de este Trabajo de Fin de Grado (Antonio Gallén Recio). Ver figura K.1.



Figura K.1: Captura del trabajo práctico StarCraft.

Aquí se muestra una tabla comparativa de las características de este trabajo que están también presentes en Armygeddon y sus diferencias (ver tabla K.1):

A continuación se muestran las dos características que el trabajo práctico tenía y que

Trabajo práctico StarCraft	Armygeddon
Se permitían partidas de dos jugadores	Se permiten partidas de hasta ocho jugadores
Se podían jugar dos razas con mecánicas mínimas	Se pueden jugar las tres razas con grandes diferencias entre ellas
Solo había unos pocos escenarios, todos hechos a mano en el código	Se permite crear todo tipo de escenarios con el editor de escenarios
El jugador artificial tenía tácticas muy simples y solo un parámetro	El jugador artificial tiene varios perfiles configurables que se adaptan al tipo de partida con muchos parámetros
Las partidas consistían en destruir la base del ejército rival	Los ejércitos se agrupan en facciones y cada facción tiene sus propios objetivos
Los escenarios difícilmente tenían más de 1000 casillas	Los escenarios pueden tener hasta 16.384 casillas

Tabla K.1: Comparativa de las características del trabajo práctico de videojuegos que están en Armygeddon.

se han decidido no implementar en Armygeddon:

- Se podían mejorar los ejércitos dentro de la partida, afectando a las estadísticas de todas las unidades de un jugador. No se ha implementado pues para la poca profundidad de las partidas no se creía apropiado por el momento, aunque se ha listado en una expansión opcional un diseño que suple esta característica (ver anexo J.5).
- Había niebla blanda, que ocultaba a los contrincantes al no haber ninguna entidad cercana. Debido a que Armygeddon es un juego muy volátil y centrado en muchos ejércitos (puede haber ejércitos aliados que cubren más escenario), se entiende que la niebla blanda es una característica que se podría implementar pero tendría poco impacto en la jugabilidad.

La implementación se realizó en Python, utilizando la biblioteca para crear juegos Pygame, y utilizando PyInstaller para generar el ejecutable para *Windows* (en formato .exe). Al ser Python un lenguaje interpretado y no compilado, el uso de los recursos hardware era muy ineficiente y eso inmediatamente se traducía en notables problemas de rendimiento.

Había problemas en el diseño, debido a la falta de tiempo para el proyecto, y las malas prácticas como los *imports cíclicos* eran más que frecuentes ya que el lenguaje además los permite y resultan en más problemas de rendimiento. Además, no se consiguió resolver uno de los grandes problemas que tuvo el equipo de desarrollo de *StarCraft* en

su momento, que fue la búsqueda de caminos de las unidades¹. Para ejemplificar:

- Era complicado terminar una partida puesto que el programa se cerraba frecuentemente.
- Si en una partida se generaban más de una decena de unidades, el programa comenzaba a ralentizarse y a sufrir parones.
- Las unidades si se movían varias de vez también bloqueaban brevemente el programa.
- Ni las estadísticas de las entidades de una partida (edificios, unidades, mejoras), ni los escenarios en sí, estaban balanceados.
- Existía un alcance muy limitado de la búsqueda de caminos, produciendo que las unidades se bloquearan fácilmente ante la presencia de obstáculos, esto incluye a las unidades mismas desplazándose.

¹Hay varios ejemplos de que tenían problemas con la búsqueda de caminos, el más notorio es que en el videojuego las unidades apenas podían sortear obstáculos. Otro ejemplo sería: [https://starcraft.fandom.com/wiki/Draagoon_\(StarCraft\)#Development](https://starcraft.fandom.com/wiki/Draagoon_(StarCraft)#Development) (último acceso: 11/8/2023).

Anexos L

Registro de cambios

Las 65 versiones del proyecto están acompañadas de un registro de cambios (*changelog*) en el que se distinguen el contenido añadido, mejorado, o arreglado respecto a la versión anterior. Se aprecian tres grupos de versiones en función del estado del proyecto:

- Recreación de *StarCraft* para obtener un núcleo básico para el proyecto: versiones de la 0.1.0 a la 1.0.2. Constituye la obtención de las características básicas comunes con el propio videojuego original *StarCraft*. Un total de 12 versiones.
- Evolución de ese núcleo básico a Armygeddon: versiones de la 1.1.0 a la 2.0.7. Contenido para llegar a un resultado mínimo que se considere que ha alcanzado los objetivos propuestos. Un total de 40 versiones.
- Expansiones opcionales sobre Armygeddon: versiones 2.1.0 a 2.3.6. Adicionalmente se incluyen todas las ideas que parten de propuestas opcionales como un uso avanzado de inteligencia artificial o el modo multijugador en línea. Un total de 13 versiones.

Recreación de StarCraft básica

Versiones 0.1.X

- 1 v0.1.0.
- 2 - Añadido:
- 3 + Menú inicial con opciones de jugar y salir.
- 4 + Mapa prediseñado básico con solo suelo.
- 5 + Zealots (unidades protoss básicas).
- 6 + Gateways (estructuras protoss básicas).
- 7 + Sistemas de ataque entre unidades, ataque a estructuras, movimiento y muerte.

- 8 + Juego completamente escalable a la ventana, aunque no es recomendable usar una
 ↪ resolución que no cumpla con la proporción de 640x480 por el alliasing.
- 9 + Círculos de selección.
- 10 + Tipos de cursores.
- 11 + Barras de vida y de escudo.
- 12 + Las entidades tienen regeneración para compensar la falta de unidades médicas
 ↪ y como factor balanceador.
- 13 + Autoataque de las unidades al eliminar a un objetivo pasar a atacar a otro
 ↪ cercano si lo hay.

Versiones 0.2.X

- 14 v0.2.0.
- 15 - Añadido:
 - 16 + Generación de unidades desde los edificios.
 - 17 + Menú de selección de juego, aunque solo está el fondo y se pasa con un clic.
- 18 - Mejora:
 - 19 + Al eliminar un objetivo la unidad buscará también estructuras enemigas
 ↪ cercanas.
 - 20 + Barras de vida y escudo de estructuras mejor colocadas.
- 21 - Arreglo:
 - 22 + Animación de ataque corregida, no se queda la unidad quieta tras un ataque.
 - 23 + Si a una unidad se le da la orden de moverse a otra casilla mientras está ya
 ↪ yendo a una casilla distinta, primero termina ese desplazamiento entre 2
 ↪ casillas antes de cambiar de rumbo.
 - 24 + Las unidades miran hacia el centro de la estructura que estén atacando, además
 ↪ de atacar a una casilla cercana que sea parte de la estructura y no siempre
 ↪ a una esquina.

Versiones 0.3.X

- 25 v0.3.0.
- 26 - Añadido:
 - 27 + Interfaz inferior dentro del juego.
 - 28 + Animaciones de explosiones para las destrucciones de edificios.
 - 29 + Limite a la distancia a la que puede generar unidades un edificio.
 - 30 + Botones de interfaz en función del edificio seleccionado.
 - 31 + Se pueden generar unidades desde la estructura pulsando el botón de la opción
 ↪ desde teclado.
- 32 - Arreglo:
 - 33 + Las unidades ya no buscan un enemigo si este ha muerto.
 - 34 + Los edificios desaparecen una vez destruidos.
 - 35 + Si una unidad no encuentra un camino para atacar a una estructura porque se ha
 ↪ ocupado la última casilla, busca otra casilla accesible desde la que atacar.
 - 36 + Animación que crasheaba al atacar una unidad en ciertas condiciones.
 - 37 + No se quedan las casillas de una estructura como ocupadas tras ser esta
 ↪ destruida.
 - 38 + No se puede controlar una unidad enemiga, atacarse a sí misma y convertirse en
 ↪ una unidad inmortal.

39 + No se permite generar unidades enemigas desde edificios enemigos.

Versiones 0.4.X

40 v0.4.0.

41 - Añadido:

42 + Marine. Soldado básico terran.

43 + Zergling. Soldado básico zerg.

44 + Centro de operaciones. Estructura base terran.

45 + Con la tecla 'P' se puede poner el tamaño del juego al doble de lo original,
↪ ya que este tamaño era muy pequeño, y así evitar líneas de aliasing.

46 - Mejora:

47 + La regeneración no va a trompicones, es bastante más agradable y uniforme.

48 + Las barras de escudo están en la parte de arriba de la barra de estado de las
↪ entidades y no en la de abajo.

49 + Rehecho el sistema de estados de las unidades, mover, atacar, etc.

50 + Cambiado el tamaño del juego a potencias de 2, el tamaño de una casilla es de
↪ 32 píxeles, un alma descarriada me aseguró que el tamaño de las casillas del
↪ juego era 40x40. El juego aparentará ser más pequeño pero será mucho más
↪ realista.

51 - Arreglo:

52 + No redimensiona mal el juego si cambiabas el tamaño de la pantalla durante el
↪ menú de creación de juego.

Versiones 0.5.X

53 v0.5.0.

54 - Añadido:

55 + Hatchery. Estructura base zerg.

56 + SCV. Trabajador terran.

57 + Drone. Trabajador zerg.

58 + Probe. Trabajador protoss.

59 + Animaciones de muerte de las unidades.

60 + Animaciones de hatchery y centro de operaciones, obviamente no están hechas
↪ para tener ese ritmo, se irá ajustando.

61 - Mejora:

62 + Las opciones de la interfaz pueden ser clicadas con ratón.

63 + Cambiados los colores. Al entrar se escoge un color aleatorio.

64 - Arreglo:

65 + Las unidades no cambian a caminos que empeoran la distancia al objetivo.

66 + Las unidades que están siendo atacadas si no consiguen alcanzar su objetivo
↪ porque hay otras entidades enemigas en medio, les ataca a estas.

67 + Algunas unidades no terminaban de alcanzar caminos.

68 + Al seleccionar unidades tras seleccionar un edificio se seguían mostrando las
↪ opciones del edificio seleccionado previamente.

69 + No se actualiza el tipo de ratón en función del tipo de entidad que hay en la
↪ casilla apuntada por el ratón a través de la interfaz.

70 + No se muestra la selección al hacer click en la interfaz.

71 + Se puede jugar con una GPU pobre: se ha disminuido del todo el requerimiento
↪ de tamaño de memoria gráfica al dividir todas las texturas a su expresión
↪ mínima: un sprite por textura.

Versiones 0.6.X

72 v0.6.0.

73 - Añadido:

74 + Minerales. Recurso desde el cual obtener cristal y con este obtener casi todo.
75 + Pilares. Estamos de pilares, tocaba.
76 + Los trabajadores pueden minar cristales.
77 + Las unidades tienen posición de ataque entre ataques.

78 - Mejora:

79 + Los ataques tardan el doble ya que el juego va el doble de rápido que en el
↪ original.
80 + Se muestra en información de la interfaz de estado, el tipo de la entidad
↪ seleccionada (en mono-selección solamente).
81 + Mejoras visuales a todos los edificios y a los círculos de selección.
82 + Los minerales van desapareciendo visualmente, ya no desaparecen de golpe.

83 - Arreglo:

84 + Se muestra correctamente ciertas entidades en lo alto de la pantalla.
85 + Hacer un redimensionado en la pantalla principal funciona correctamente.
86 + No falla por ciertas condiciones en el trazado de caminos.
87 + No se quedan unidades inmortales en ciertas condiciones.
88 + Las unidades no se "marean" en rutas que mantenían exactamente la misma
↪ distancia al objetivo durante varias casillas.
89 + No se pierden los trabajadores de camino a devolver el mineral a veces por la
↪ búsqueda de caminos.
90 + No se cierra el juego al mandar muchas unidades a una misma mena de cristal.
91 + Los trabajadores pierden el cristal al entablar combate.
92 + Atacar a un trabajador que está minando hace que deje de minar.
93 + La búsqueda de caminos en determinadas ocasiones ya no olvida el camino
↪ original si no encuentra un nuevo camino.
94 + El juego no se cierra al atacar una unidad que no tiene casillas disponibles
↪ desde las cuales atacarle.
95 + Las animaciones de destrucción de edificios no son invisibles a veces o hacen
↪ que se cierre el juego.
96 + El juego no se cierra a veces con múltiples unidades atacando a un mismo
↪ objetivo.
97 + A veces se impedía minar dos veces el mismo mineral quedándose este bloqueado
↪ indefinidamente.
98 + El juego no se cierra a veces al mandar varias unidades al mismo sitio.
99 + Las unidades muestran correctamente que transportan minerales si es el caso.

Versiones 0.7.X

100 v0.7.0.

101 - Añadido:

- 102 + Botón para el editor de mapas.
- 103 + Pantalla de creación de mapas para el editor de mapas, con ya algunos botones,
↪ textos y números.
- 104 + Opción de volver al menú principal desde el menú de creación de mapa pulsando
↪ la tecla 'Esc'.
- 105 + Atajo de salir del juego pulsando en el menú principal la tecla 'Esc'.
- 106 + Botones en el menú de creación de juego.
- 107 + Ventana emergente para confirmar salir de una partida.
- 108 + Menú de configuración de la partida en el menú de creación de juego.
- 109 + Final del juego con su correspondiente pantalla de victoria o derrota.
- 110 - Mejora:
 - 111 + Aspecto visual del menú principal.
 - 112 + La interfaz de estado del juego incluye equipos, además de mostrar mejor el
↪ tipo de entidad.
- 113 - Arreglo:
 - 114 + Aparece en todos los casos el cursor en el menú inicial, antes a veces no.
 - 115 + Los textos generados no ocupan cada vez mas memoria hasta el colapso del
↪ juego.

Versiones 0.8.X

- 116 v0.8.0.
- 117 - Añadido:
 - 118 + Estructuras protoss básicas de verdad: Nexus. Por error estaban hasta los
↪ Gateways.
 - 119 + Dibujos a la interfaz de estado.
 - 120 + Indicadores para el cristal, gas y población (la población tiene un indicador
↪ por raza: psi, control y supply).
 - 121 + Control de la población máxima con estructuras que lo otorgan: la población
↪ máxima aumenta y disminuye según se crean y se destruyen según
↪ qué estructuras.
 - 122 + Funcionalidad al botón de trucos.
 - 123 + Pop-ups de las opciones de la interfaz ingame.
 - 124 + Control de la población actual con unidades que la consumen: aumenta y
↪ disminuye la población en uso según se generan y se destruyen según
↪ qué unidades.
 - 125 + Mecánica de ataques dobles de los Zealots.
 - 126 + Refineras, una por cada raza: Asimiladores, Extractores y Refinerías.
 - 127 + Géiseros. Introduciendo también por ende el recurso que ofrecen, el gas.
 - 128 + Mapa adaptado a la raza, base y color especificados en el menú de creación de
↪ juego.
 - 129 + Los trabajadores pueden recolectar gas de las refineras.
 - 130 + Los trabajadores muestran el gas que cargan en orbes, sacos o tanques según su
↪ raza.
 - 131 + Jugador artificial con capacidades mínimas, para hacer pruebas de todo un
↪ poco.
- 132 - Mejora:

133 + Desaparecen los botones en el menú de creación de mapa si no se pueden
 ↪ accionar por límite inferior o superior (dependiendo) de las columnas o las
 ↪ filas introducidas.

134 + No se muestra el equipo al que pertenece una entidad con un "Equipo <color
 ↪ equipo>", se elimina dicho campo y se muestra el nombre de la entidad con el
 ↪ color correspondiente.

135 + La pantalla se centra sola al pulsar la tecla para cambiar la resolución.

136 + No se puede cabrear al menú de creación de juego para que te eche, adiós
 ↪ easter egg.

137 + Mapa más balanceado y decorado.

138 - Arreglo:

139 + No se bloquea el mineral, impidiendo volver a minar un mineral si se
 ↪ interrumpía este antes el proceso de minado a mitad.

140 + No se conservan las filas y las columnas introducidas en el menú de creación
 ↪ de mapas una vez sales y vuelves a entrar en dicha pantalla.

141 + Las unidades (también los trabajadores) no tienen salud negativa a veces al
 ↪ morir.

142 + Las unidades (también los trabajadores) mueren al llegar a 0 de salud, en vez
 ↪ de sobrevivir hasta que reciben otro golpe más.

143 + Aparece el fondo de derrota Protoss al perder con dicha raza, en vez de el
 ↪ fondo de victoria.

Versiones 0.9.X

144 v0.9.0.

145 - Añadido:

146 + Pilones.

147 + Larvas.

148 + Mecánica de evolución Zerg: Las unidades zerg no salen directamente sino que
 ↪ tienen que evolucionar de larvas primero.

149 + Huevos. Como estado intermedio entre larva y unidad.

150 + Estanques de desove.

151 + Almacenes de suministros.

152 + Cuarteles.

153 + Portales (los Gateways que se tenían como Nexus y luego se quitaron).

154 + Soberanos.

155 + Mecánica de construir más edificios base.

156 + Mecánica de construcción Terran.

157 + Ajuste de espionaje: permite ver los recursos de todos los jugadores.

158 + El jugador artificial se defiende.

159 - Mejora:

160 + Mapa balanceado.

161 + Nuevo autómatas para las unidades, concretamente atacar y moverse.

162 + Buscador de caminos: intenta 2 caminos por sitios distintos para sortear
 ↪ obstáculos.

163 + Más información a la barra de estado, se muestra el estado de la estructura o
 ↪ unidad, además de la acción que está realizando.

164 + Los pop-ups de los botones de las opciones de dentro del juego distinguen
 ↪ claramente qué recursos resta y cuales suma.

165 + El buscador de caminos, de nuevo.
 166 + El jugador artificial es algo más inteligente.
 167 + El ajuste de espiar solo muestra los jugadores vivos, no 8 siempre.
 168 + El cristal inicial es de 150, por acelerar las cosas.
 169 + El jugador artificial construye con más sentido, no se pone a construir
 ↪ edificios sin parar.

170 - Arreglo:

171 + El jugador artificial no cierra el juego al eliminar la base antes que a sus
 ↪ unidades.

172 + La pantalla de fin de juego se muestra con el fondo correcto.
 173 + El jugador artificial deja de poder generar Zealots de los Nexus.
 174 + El jugador artificial deja de poder construir edificios sin coste alguno.
 175 + Los estanques de desove dejan de tener escudo.
 176 + El juego deja de cerrarse al mandar a minar una unidad que ya estaba minando.
 177 + La búsqueda de caminos con varias unidades no crea ciclos (parecen bailes
 ↪ provenzales).

178 + Vencer al jugador artificial no cierra el juego.
 179 + Deja de permitirse el moverse un trabajador mientras está dentro de una
 ↪ refinera.

180 + Animación de los CommandCenter.
 181 + Animación de los Cuarteles.
 182 + Más bugs menores que en todas las versiones anteriores juntas.

Versiones 1.0.X

183 v1.0.0.

184 - Añadido:

185 + Minimapa dentro del juego.
 186 + Música principal para las pantallas que no son ni el propio juego o fin de
 ↪ partida.

187 + Música para cada raza dentro de partida, hay 3 posibilidades para cada raza.
 188 + Construcción Protoss: no se requiere de trabajadores, se crea un portal y se
 ↪ teletransporta solo el edificio.

189 + Construcción Zerg: tampoco requiere de trabajadores durante la construcción,
 ↪ es automática.

190 + Música para el final de la partida, hay para cada raza y para ambos casos:
 ↪ victoria o derrota.

191 + Información de estado de las refineras.
 192 + Posición de la cámara dibujada en el minimapa.
 193 + Interfaces dentro del juego dependientes de la raza.
 194 + Dragoon.
 195 + Hydralisk.
 196 + Firebat.
 197 + Cybernetics Core.
 198 + Academy.
 199 + HydraliskDen.

200 - Mejora:

201 + Cursor dentro del juego: incluye animaciones de click, y cursores compuestos.

202 + Nuevo sistema de búsqueda de caminos.

203 + Nuevo sistema de comportamiento global de las unidades (quieto, moviéndose,
↳ minando, atacando, extrayendo gas).

204 - Arreglo:

205 + El juego no se cierra al perder el jugador humano.

206 + Se juega con el color que has seleccionado aunque sea distinto al
↳ predeterminado.

207 + Las unidades dejan de poder entregar mineral a edificios sin completar.

208

209 v1.0.1.

210 - Añadido:

211 + Posibilidad de silenciar y desilenciar la música con la tecla 'M'.

212 + Interfaz del editor de mapas.

213 + Salida directa: el juego finaliza si se pulsa la 'X' de la ventana del juego.

214 - Mejora:

215 + Menú del editor de mapas tiene todas las funcionalidades.

216 + Los caminos se optimizan tras ser creados, ya no se anda dos veces por la
↳ misma casilla y la casilla origen no se tiene en cuenta como ocupada para
↳ crear el camino.

217 - Arreglo:

218 + Al presionar la tecla de salir mientras se está escogiendo dónde construir una
↳ estructura no deselectiona a la unidad además de cancelar la construcción.

219 + El juego permite pasar al modo de instalación de refineras directamente desde
↳ el modo de construcción de edificios.

220 + Las larvas dejan de poder atacar y de cerrar con ello el juego.

221

222 v.1.0.2.

223 - Añadido:

224 + Animación de ataque de los Probe.

225 + Información en la barra de estado de la multiselección.

226 + Animación de los Huevos.

227 + Titulares al final de cada partida.

228 + Fase de construcción a las refineras Terran.

229 + Cursor en el editor de mapas.

230 + Terreno en el editor de mapas.

231 + Posibilidad de salir del editor de mapas con la tecla de salir.

232 + Movimiento de cámara en el editor de mapas.

233 - Arreglo:

234 + Regeneración de los Huevos, estaba a valor por defecto y los hacía inmortales.

235 + Al mover la cámara mientras se está seleccionando no se selecciona todo lo
↳ barrido.

236 + Se muestran los fondos correctos en el final de partida según raza.

237 + Los constructores terranos construyen haciendo la animación y no posando.

238 + Los constructores terran van directos a trabajar la construcción tras mandarse
↳ y no se desplazan aunque estén al lado.

239 + El Dragoon no cierra el juego al atacar.

Evolución a Armygeddon

Versiones 1.1.X

- 240 v1.1.0.
- 241 - Añadido:
- 242 + Botón para salir en el editor de mapas.
 - 243 + Botón para guardar en el editor de mapas.
 - 244 + Botón para editar casillas ('T' de Terreno por defecto) en el editor de mapas.
 - 245 + Set 'vacío' (hollowed) de casillas para el terreno de un mapa.
 - 246 + Mecánica de editar en el editor el terreno clicando sobre este.
 - 247 + Set 'triste' (sad) de casillas para el terreno de un mapa.
 - 248 + Modo de bases al editor de mapas.
 - 249 + Modo de recursos al editor de mapas.
 - 250 + Pantalla de selección de mapa
 - 251 + Con la tecla de salir se puede salir también de la pantalla de fin de juego.
 - 252 + Carga de mapa esencial (terreno).
- 253 - Mejora:
- 254 + Mecánica de editar de manera sostenida: mientras se mantenga el clic y se
 - ↪ desplace el cursor de una casilla a otra, pudiendo volver a editar la misma
 - ↪ si se sale y vuelve.
 - 255 + Mecánica de editar de manera sostenida: también desplazando la cámara y no
 - ↪ solo cursor.

Versiones 1.2.X

- 256 v1.2.0.
- 257 - Añadido:
- 258 + Menú de juego implementado, para escoger entre cargar o crear partida.
 - 259 + Menú de mapa implementado, para escoger entre cargar o crear mapa.
 - 260 + Borrar casillas en el editor: las casillas borradas se muestran como una cruz.
 - 261 + Menú de carga de mapa implementado, carga los mapas que se almacenan en la
 - ↪ carpeta \maps.
 - 262 + Borrar recursos en el editor (minerales y géiseres).
 - 263 + Borrar bases en el editor (de todos los equipos).
 - 264 + Poner obstáculos en el editor.
 - 265 + El editor de mapas permite en general borrar elementos de todo tipo.
 - 266 + Botón en el editor para vaciarlo de unidades y de estructuras.
- 267 - Mejora:
- 268 + Nuevo sistema de menús.
 - 269 + Menú principal cambiado, ligeramente, para adaptarse al resto.
 - 270 + Menú de crear mapa rehecho: permite entre otras cosas crear mapas usando
 - ↪ plantillas, sobrecribir mapas (lo cual avisa), y ojear los mapas
 - ↪ disponibles si los hay para ser usados como plantillas mostrando sus
 - ↪ respectivas filas y columnas.
 - 271 + En el editor cuando se pinta una casilla del mismo tipo del que ya está pasa a
 - ↪ la siguiente en el orden en el que están almacenadas las casillas de ese
 - ↪ conjunto en vez de a una aleatoria.

Versiones 1.3.X

- 272 v1.3.0.
- 273 - Añadido:
- 274 + Opción de habilitar o deshabilitar un equipo en el editor de mapas para que
↪ figure en el o no. En caso de deshabilitar un jugador se convertirán todas
↪ sus unidades y estructuras (si las hay) a las del equipo neutral.
- 275 + Nuevos obstáculos.
- 276 + Diseñados nuevos interfaces para el editor de mapas.
- 277 + Simulaciones de partidas sin el jugador humano.
- 278 + Se pueden jugar mapas creados.
- 279 + Sonidos de muerte, selección, generación y ordenar ataque o movimiento.
- 280 - Mejora:
- 281 + Actualizados todos los menús.
- 282 - Arreglo:
- 283 + Se permite renombrar, eliminar y cargar mapas sin problema.
- 284 + El ajuste de espiar a los jugadores artificiales.

Versiones 1.4.X

- 285 v1.4.0.
- 286 - Añadido:
- 287 + Pantalla de carga.
- 288 + Ajuste de volumen con las teclas '-' y '+'.
- 289 + Sonidos al pasar por encima de un botón en el menú principal, menú de edición
↪ de escenarios o menú de juego.
- 290 + Botón de ajustes en el menú principal.
- 291 + Zum en el editor de escenarios.
- 292 + Poder editar los recursos iniciales de cada equipo.
- 293 + Modo de edición 'x3' de baldosas en el editor de mapas.
- 294 - Mejora:
- 295 + Todo pasa a escenarios, y es editor de escenarios y no de mapas.
- 296 + La cuadrícula del editor de escenarios se coloca debajo de los elementos.
- 297 + La pantalla de carga.
- 298 - Arreglo:
- 299 + Zum del editor de escenarios (iba mal).
- 300 + La cámara en el juego (iba mal por meter el zum).
- 301 + Añadido zum también dentro del juego.
- 302 + Añadida una contramedida para evitar rasgaduras de dibujo en el suelo con el
↪ cambio de zum.
- 303 + Se cargan correctamente los recursos iniciales configurados en el editor
↪ dentro del juego.
- 304 + Recursos iniciales en el editor, además se muestra correctamente los ceros.
- 305 + Zum de los cristales de los trabajadores.
- 306 + El juego no se cierra si se introduce un escenario sin nombre en la carpeta de
↪ escenarios.
- 307 + El juego no se cierra si se hace clic en siguiente equipo en editor escenarios
↪ desde el último equipo.

Versiones 1.5.X

308 v1.5.0.
309 - Añadido:
310 + Opciones de configuración predeterminadas a cada equipo en el editor de
↪ escenarios.
311 + Desplegable para editar la facción de cada equipo en el editor de escenarios.
312 + Poder renombrar facciones en el editor de escenarios.
313 + Poder renombrar equipos en el editor de escenarios.
314 + Modo de edición "infinito" para las baldosas en el editor de escenarios.
315 + El editor de escenarios indica la casilla que se está señalando.
316 + En la creación de juego se reflejan los nuevos datos (nombre equipos y valores
↪ predefinidos).
317 + Primer escenario oficial: StarCraft.
318 - Mejora:
319 + Apariencia en selección de equipo en el editor de escenarios.
320 + Los jugadores artificiales desplazan a los trabajadores hasta que encuentran
↪ cristal.
321 - Arreglo:
322 + El editor de escenarios no cierra el juego si se hace clic en la zona de
↪ configuración de los demás equipos desde el equipo neutral.
323 + El editor de escenarios no cierra el juego si se coloca una unidad del equipo
↪ neutral.
324 + El juego ya no se cierra con el espiar IA activo.
325 + La creación de unidades Zerg no duplica el coste en población.
326
327 v1.5.1.
328 - Añadido:
329 + Poder mover la cámara en partida desde el minimapa.
330 - Mejora:
331 + Disminuido el zum en el juego, a x1.5 de lejanía.
332 + Aumentada la velocidad de la cámara, un 33%.
333 - Arreglo:
334 + Los jugadores artificiales manejaban al jugador en las partidas con jugador.
335
336 v1.5.2.
337 - Añadido:
338 + Menú de ajustes, aunque no hace nada.
339 - Mejora:
340 + Los obstáculos se reflejan también en el minimapa.
341 - Arreglo:
342 + Se actualiza bien el nombre de un escenario si se dejaba vacío en el menú de
↪ crear escenarios.
343 + No se suman 2 veces el coste de población de las unidades en el juego.
344 + Se muestra en partida correctamente la interfaz correspondiente y no siempre
↪ la Protoss.
345
346 v1.5.3.
347 - Añadido:

348 + Opción de quitar o activar la cuadrícula también en el editor de escenarios.
 349 + Ventana emergente de aviso al intentar salir del editor de escenarios con
 ↪ cambios sin guardar.
 350 + Sombras dentro del juego.

351 - Mejora:

352 + Aspecto gráfico de la apertura.
 353 + Panel equipos / facciones del editor de escenarios.
 354 + Los paneles emergentes del editor de escenarios.
 355 + El volumen inicial del juego vuelve a estar al 50%.

356 - Arreglo:

357 + No se cierra el juego ya a veces al perder un jugador artificial.
 358 + Se actualizan correctamente los minerales.
 359 + Los trabajadores no van a intentar entregar cristal a un edificio sin
 ↪ construir.
 360 + No suenan los avisos de unidad lista sin estar dicha unidad visible en
 ↪ pantalla.
 361 + El botón de guardado que no aparece iluminado inicialmente en cargas del
 ↪ editor de escenarios.

362

363 v1.5.4.

364 - Añadido:

365 + Defensas Terran.
 366 + Piedras.
 367 + Set de casillas oscuro (dark).
 368 + Panel emergente de las facciones en el editor de escenarios.

369 - Mejora:

370 + La barra de carga se atasca en otro sitio.
 371 + El desplegable de las facciones del editor de escenarios.
 372 + Empiezan a ajustarse los textos al tamaño real y no al tamaño por número de
 ↪ letras: Menú principal, menú ajustes, menú escenarios.
 373 + Menú de ajustes tiene más ajustes.

374 - Arreglo:

375 + Botón de guardado del editor de escenarios responde bien.
 376 + Al borrar un escenario se muestra correctamente la información del siguiente
 ↪ escenario.
 377 + No se cierra el juego al guardar en el editor de escenarios sin haber usado
 ↪ todos los equipos.
 378 + Se muestra el previo de edición de casillas "x3".
 379 + Marca cambios sin guardar al cambiar de raza en el editor de escenarios.
 380 + Vuelve a aparecer el icono del juego.

Versiones 1.6.X

381 v1.6.0.

382 - Añadido:

383 + Categorías en el menú de ajustes.
 384 + Límites a los recursos dentro de partida para evitar que desborden, que dudo
 ↪ que ocurra.

385 - Mejora:

386 + Sonidos de las unidades limitados: solo suena uno si hay varias unidades
↳ seleccionadas.

387 + El jugador artificial mejora la ruta de los trabajadores mandándoles a
↳ cristales más cercanos si se encuentran, una vez terminan su ruta.

388 - Arreglo:

389 + No se escucha varias veces un sonido simultáneamente si se forzaba.

390 + El botón de crear escenario se quedaba en verde al salir una ventana
↳ emergente.

391 + Los soberanos no dan el doble de la población que deberían.

392 + No se cierra el juego al intentar cargar un escenario irregular.

393 + Las tiles oscuras se muestran correctamente y todas las versiones de estas.

394

395 v1.6.1.

396 - Añadido:

397 + Posibilidad de pausar el juego, despausar y acelerarlo a x2 y a x4.

398 + Botón de salir dentro de partida.

399 + Botón de guardar dentro de partida, no hace nada.

400 - Arreglo:

401 + Las piedras nuevas se cargan sin cerrar el juego en partida.

402 + Las defensas terranas se cargan sin cerrar el juego en partida.

403 + No se cierra el juego si se activan trucos sin seleccionar un jugador humano.

404

405 v1.6.2.

406 - Añadido:

407 + Se puede mover la cámara con el ratón en el juego y en el editor de
↳ escenarios.

408 - Mejora:

409 + Marca "humano" en vez de "jugador" en el editor de escenarios.

410 + El jugador artificial ataca y no solo defenderse.

411 - Arreglo:

412 + Se ilumina el botón de guardado al cambiar una raza en el editor de
↳ escenarios.

413 + Se renderiza bien la configuración de jugador artificial predeterminada al
↳ seleccionar otro equipo en el editor de escenarios.

414 + No se puede indicar más de un jugador humano en el editor de escenarios.

415 + El jugador artificial ocasiona que se cierre el juego al intentar salir de la
↳ partida.

416 + Varias causas relacionadas con las larvas y los huevos dejan de hacer que se
↳ cierre el juego.

417

418 v1.6.3.

419 - Añadido:

420 + Sonido en los botones de configuración.

421 - Mejora:

422 + El jugador artificial genera más edificios si los necesita.

423 - Arreglo:

424 + Se muestran correctamente el icono de las poblaciones de las razas en partida.

425 + El jugador artificial no controla al jugador en según qué circunstancias.
 426 + Los nuevos edificios no cierran el juego al intentar construirlos.
 427
 428 v1.6.4.
 429 - Añadido:
 430 + Paneles emergentes fuera del editor.
 431 + Las unidades aparecen inicialmente mirando a una dirección aleatoria.
 432 + Modo turbo: desata el poder del juego ahorrando todos los ciclos posibles,
 ↪ obviamente funciona más deprisa con partidas poco complejas y va más lento a
 ↪ medida aumentan el número de jugadores, unidades, etc.
 433 - Mejora:
 434 + El minimapa lo actualiza un hilo (~30% de aumento de velocidad general).
 435 + Los recursos en partida se actualizan cada menos y más organizadamente (~10%
 ↪ de aumento de velocidad general en partida, ~25% si se usa espiar recursos).
 436 + Jugador artificial: es mucho más peligroso.
 437 + El jugador artificial ya no usa siempre el mismo hilo, usa distintos.
 438 + Aspecto visual del cursor.
 439 + Reducidos los FPS a la mitad: 60 -> 30. El juego lo aprovecha mejor.
 440 + El texto de tipo de entidad seleccionada se prerenderiza y se escala bien.
 441 - Arreglo:
 442 + No se cierra el juego en el fin de partida con según qué teclas.
 443 + No se cambia de canción si la que está sonando es la misma que se va a poner.
 444 + No se cierra el juego al instanciar una estructura del equipo neutral en el
 ↪ editor de escenarios.
 445 + El juego no se cierra con "Tried to reserve an unexistent tile" al no
 ↪ encontrar el jugador artificial un hueco para construir.
 446 + No se cierra el juego con los worker minando en según qué circunstancias.
 447 + Las pantallas de fin de partida para las simulaciones.

Versiones 1.7.X

448 v1.7.0.
 449 - Añadido:
 450 + Mecánica de velocidad: cada tipo de unidad tiene su velocidad de movimiento.
 451 + Trucos en simulación: se activan trucos a todos los jugadores artificiales si
 ↪ es simulación.
 452 - Mejora:
 453 + Se han dividido todas las entidades por funcionalidad, lo cual acelera
 ↪ bastante el juego.
 454 + Se ha reducido la heurística de caminos y su alcance a la mitad, la cual aún
 ↪ es mayor que la de StarCraft original, acelerando hasta el doble de ciclos
 ↪ en partida.
 455 + Mejorado el jugador artificial: va mandando refuerzos eventualmente a las
 ↪ zonas de conflicto hasta que se resuelven.
 456 + Se prerenderiza más texto dentro del juego (los campos 'vida' y 'escudo').
 457 + Escenario StarCraft.
 458 + Aumentada la velocidad de la cámara.
 459 - Arreglo:

460 + Los trabajadores no van a dar una vuelta tonta una vez han devuelto el cristal
↳ en un edificio.

461 + Las unidades ya no aparentan estar giradas hacia donde no deben en ciertas
↳ circunstancias.

462 + Espiar recursos funciona bien.

463 + La muestra de algunos colores en el minimapa.

464 + No se cierra el juego al matar una unidad del equipo neutral.

465

466 v1.7.1.

467 - Añadido:

468 + Mecánica de mutación a estructuras Zerg: el dron se transforma en el edificio
↳ que construye.

469 + Mecánica de generación: en función del tipo de unidad, esta tardará más o
↳ menos en generarse.

470 + Mecánica de búsqueda alternativa de minerales: si un trabajador es mandado a
↳ un mineral que está ocupado este buscará otro disponible.

471 + Mecánica de ataques de murflama: Los murflama tienen en teoría ataque en área
↳ doble, y como es mucha mecánica los daños en área para una sola unidad, para
↳ compensar, los murflama tienen ataques individuales triples.

472 - Mejora:

473 + Unidades rebalanceadas. En resumen, hacer más débiles a los fanáticos.

474 + El escenario StarCraft está más balanceado.

475 + El jugador artificial puede crear refineras y explotaras, construir edificios
↳ avanzados y generar todo el nivel avanzado de tropas.

476 - Arreglo:

477 + Los Zerg pueden volver a evolucionar.

478 + El jugador artificial vuelve a poner huevos.

479 + La mejora de ruta de los trabajadores antes no aplicaba a pesar de estar
↳ implementada.

480 + El jugador artificial vuelve a crear soberanos.

481 + El jugador artificial vuelve a crear zerglings.

482 + Las refineras vuelven a funcionar y no estar bloqueadas.

483 + Los sonidos solo suenan dentro de la pantalla.

484 + Los minerales vuelven a ser bloqueantes: solo se puede minar uno por un
↳ trabajador por vez.

485

486 v1.7.2.

487 - Añadido:

488 + Activación automática en el editor de escenarios: para añadir un ejército
↳ basta con crear una unidad o estructura de tal ejército para que se active,
↳ y para desactivarla vale con eliminar todas las unidades y estructuras de
↳ dicho ejército.

489 + Sonidos de selección de unidades en partida.

490 + Funcionalidad al botón de salir de partida.

491 - Mejora:

492 + Duplicada la velocidad de la animación del menú de selección de juego.

493 + Se pasa de poner en todas partes "equipo" a poner "ejército".

494 + Duplicada la velocidad de la animación del menú de creación de juego.

495 + Duplicada la velocidad de la animación del menú de creación de escenario.
 496 + Duplicada la velocidad de la animación del menú de selección de escenario.
 497 + Limitado el sonido de selección múltiple a una sola unidad.
 498 + Se prerrenderizan las hp, y los escudos, además de los nombres de las
 ↪ entidades seleccionadas.
 499 + Bajón a la velocidad de todas las unidades.
 500 - Arreglo:
 501 + Cierre inesperado del juego.
 502
 503 v1.7.3.
 504 - Añadido:
 505 + Escalado de cifras de los recursos iniciales del editor de escenarios.
 506 + Autoincremento y autodecremento manteniendo el clic en los recursos iniciales
 ↪ en el editor de escenarios.
 507 + Menú de ajustes de teclado: este permite guardar y restablecer dichos
 ↪ controles. Se guarden entre distintas ejecuciones del juego.
 508 - Mejora:
 509 + Aumentado el tiempo de animación del cursor de escritura en la creación de
 ↪ escenario.
 510 - Arreglo:
 511 + La iluminación de los botones de cambio de recursos iniciales en el editor de
 ↪ escenarios.
 512 + Deja de poderse pulsar algo sin querer en el menú de creación de juego, carga
 ↪ de juego, creación de escenario, carga de escenario, mientras
 ↪ está animándose.
 513
 514 v1.7.4.
 515 - Añadido:
 516 + Menú de ajustes de sonido.
 517 + Control de activar / desactivar pantalla completa, por defecto con F11.
 518 + Menú de ajustes de video.
 519 - Mejora:
 520 + Se ha eliminado el control de subir y bajar volumen, ya que van a meterse los
 ↪ ajustes para dichas funcionalidades.
 521 + Se ha eliminado el control de eliminar y se ha asignado siempre a la tecla
 ↪ backspace (retroceso o borrar).
 522 + Se ha eliminado el control de redimensionado, ya que se puede editar desde
 ↪ ajustes.
 523 - Arreglo:
 524 + Las teclas de derecha y abajo no están intercambiadas en los ajustes por
 ↪ defecto.
 525 + Si se deselecciona modo de edición y se vuelve a seleccionar por teclado la
 ↪ casilla apuntada se actualiza correctamente.
 526

Versiones 1.8.X

527 v1.8.0.

528 - Añadido:

529 + Más información al menú de selección de juego: facciones y vista previa del
↪ escenario.

530 + Categorías de visibilidad dentro del editor de escenarios para todas las
↪ categorías de elementos.

531 + Opción visible para activar / desactivar la cuadrícula en el editor de
↪ escenarios.

532 + El botón de borrado del editor de escenarios elimina solo las categorías
↪ visibles.

533 + El menú inicial es más informativo e intuitivo, y está escalado.

534 - Mejora:

535 + El menú de juego tiene los textos escalados.

536 + El menú de selección de juego tiene todos los textos escalados.

537 + El menú de selección de juego, responde mejor al usuario en diversos aspectos.

538 + Los botones seleccionar y cancelar están separados en el menú de selección de
↪ juego para una mejor orientación.

539 + Rehecho el menú de selección de escenario del editor.

540 + Aumentado el zum mínimo en el editor de escenarios para que se pueda ver
↪ cualquier mapa al completo: x1:5 -> x1:13.

541 + En el hueco faltante de la barra de estado del editor de escenarios la facción
↪ del ejército seleccionado si es que tiene facción dicho ejército.

542 + El menú de selección de juego tiene todos los textos prerrenderizados

543 + El menú de selección de escenario del editor tiene todos los textos
↪ prerrenderizados

544 + Todo está prerrenderizado, lo cual ha provocado notables fallos en el menú de
↪ juego y en el de creación de juego. Esto se irá arreglando y haciendo que
↪ desaparezcan esos "?" molestos en cuanto se trace cada uno.

545 - Arreglo:

546 + No se cierra el juego al presionar la tecla de salir en los menús de cargar
↪ escenario y cargar juego.

547 + Se actualizan correctamente los valores de ejércitos en los ficheros de los
↪ escenarios.

548 + El estado del "silenciar todo" en ajustes de sonido se refleja correctamente
↪ según esté activo o no.

549 + No se cierra el juego con "Editing without an editing mode?" cuando se
↪ mantiene el click en el editor de escenarios y se mueve el ratón.

550 + El texto de la ventana emergente de eliminar lo visible del editor de
↪ escenarios está estandarizado.

551 + El juego no se cierra al pulsar una tecla dentro del juego. Que ha ocurrido en
↪ esta versión, pero quería reflejarlo.

552

553 v1.8.1.

554 - Añadido:

555 + Configuración para las facciones exterminadoras en el editor de escenarios.

556 + Minerales en la vista previa del escenario.

557 + Géiseres en la vista previa del escenario.

558 + Neutralidad: está muy muy experimental pero ya están las alianzas, no se puede
↳ atacar a miembros de la misma facción y se ven diferentes.

559 + Tolerancia de ciertos fallos en ficheros de escenario: todos los escenarios
↳ con mala previa del escenario con cosas que no cuadran, que eran porque se
↳ hicieron hace varias versiones, se corrigen automáticamente si se editan y
↳ se vuelven a guardar.

560 - Mejora:

561 + El editor de escenarios está remodelado.

562 + El escenario StarCraft.

563 + Todos los textos del editor de escenarios están estandarizados.

564 + El editar nombre de facción se hace desde el menú de la propia facción en el
↳ editor de escenarios.

565 + El color de las unidades neutrales en la previa del escenario es más
↳ diferenciable que antes.

566 + Restaurar ajustes funciona en las 3 categorías implementadas.

567 - Arreglo:

568 + Al hacer clic en objetivo previo de facción sin haberlo no pasa al siguiente.

569 + Los nombres de los equipos no pueden salirse de su rectángulo.

570 + Arreglada la iluminación de los botones de sobreescritura de la ventana
↳ emergente del menú de creación de escenarios al crear un escenario con un
↳ nombre ya en uso.

571

572 v1.8.2.

573 - Añadido:

574 + Selección en el editor de escenarios: borrar, copiar y pegar mediante trazado
↳ de patrones que se conservan entre mapas.

575 + Deseleccionar el escenario en la selección de escenario pulsando la tecla de
↳ salir.

576 + Duplicar escenarios.

577 + Botón de truco de niebla, sin implementar.

578 + Minerales y géiseres aparecen en el minimapa dentro del juego.

579 + Presionando TAB (por defecto), se puede ver un panel de facciones dentro de la
↳ partida.

580 -Mejora:

581 + El menú de configuración de juego muestra las facciones y el nombre del
↳ escenario a jugar.

582 + No se muestran los recursos de los jugadores que no participan en la partida
↳ al activar espiar recursos, pero sí se muestran los recursos de los
↳ jugadores eliminados.

583 + Centrado el indicador de páginas de opciones del modo de edición en el editor
↳ de escenarios.

584 + La interfaz de creación de partida: los botones de los trucos aparecen arriba
↳ distinguidos.

585 + Remodelado el menú de configuración de juego acorde a los nuevos estándares.

586 + Los botones de guardado y de salir dentro del juego están normalizados.

587 - Arreglo:

588 + Se ve el número de páginas de opciones del modo de edición en el editor de
↳ escenarios.

589 + Eliminar escenario en la selección de escenario.
 590 + Se reestablecen correctamente los ajustes de audio.
 591 + No se ven paneles emergentes vacíos en el editor de escenarios cuando el
 ↪ ejército actual no está activo.
 592 + No se mueve la cámara por el ratón al activar el desplegable de facciones en
 ↪ el editor de escenarios.
 593 + Se actualiza correctamente la facción de un ejército al activarlo y al
 ↪ desactivarlo.
 594 + Se actualizan correctamente las texturas de cada ejército al activarlos.

Versiones 1.9.X

595 v1.9.0.
 596 - Añadido:
 597 + Se muestran los objetivos dentro del juego para las facciones exterminadoras.
 598 + Reloj dentro de partida.
 599 + Una facción exterminadora ya gana si se cumplen sus objetivos aunque no sea
 ↪ exterminar a todas las facciones.
 600 + El botón de eliminar lo visible que es redundante estando el "eliminar lo
 ↪ visible" pasa a ser el botón de modo de edición de objetivos.
 601 + Zona de conquista.
 602 + Las refineras se reflejan en el minimapa.
 603 + Fondo de fin de juego para las simulaciones.
 604 + Se deselecciona la selección en el editor de escenarios haciendo un clic
 ↪ suelto en el escenario.
 605 - Mejora:
 606 + Se quita la sombra a los 3 tipos de grietas ya que no tienen.
 607 + No se muestran cruces en las casillas no declaradas, para poder tener
 ↪ escenarios irregulares.
 608 + Al presionar el control de salir se deselecciona la selección en el editor de
 ↪ escenarios.
 609 + La selección en el editor de escenarios se crea solo si se mueve el ratón
 ↪ mientras se selecciona.
 610 + La interfaz superior del editor de escenarios está extendida para espaciar más
 ↪ sus elementos.
 611 - Arreglo:
 612 + Se puede volver a borrar caracteres en el nombre de un nuevo escenario en el
 ↪ menú de creación de escenarios.
 613 + El fondo no cambia a blanco al mostrar la tabla de facciones.
 614 + Al entrar a veces en partida no se muestra el icono de pausa activado sin que
 ↪ corresponda.
 615 + La "H" de Humano en la pantalla de creación de partida se refleja en el
 ↪ ejército correcto.
 616 + No se cierra el juego al ganar el jugador humano.
 617 + No se cierra el juego al perder el jugador humano.
 618 + No se borra el nombre del ejército actual o de la facción si se pulsa el botón
 ↪ de borrar sin que se esté editando dicho nombre.
 619 + Se actualiza correctamente el número de página en la selección de juego.

620 + No se bloquea el juego al hacer clic derecho en el editor de escenarios al
↳ cambiar de casilla sin ningún modo de edición.

621 + Se quita la facción de un ejército deshabilitado en el editor de escenarios.

622 + Si se cambia de modo en el editor de escenarios mientras el cursor no está en
↳ el escenario, no se muestra el elemento que actualmente se está editando por
↳ el escenario.

623 + Al no tener un escenario seleccionado se desactiva correctamente el botón de
↳ editar escenario en el menú de carga de escenario.

624 + No se copia si no hay selección en el editor de escenarios.

625 + No se cierra el juego si se intenta copiar sin que haya nada seleccionado en
↳ el editor de escenarios.

626 + No se cierra el juego si se pega sin seleccionar donde en el editor de
↳ escenarios.

627 + No se cierra el juego si se pega sin tener contenido que pegar en el editor de
↳ escenarios.

628 + No hay dificultades de jugador artificial raras como "A". Esto estaba ya en la
↳ versión anterior pero no lo notifiqué.

629 + No empieza a seleccionar cuando se mantiene el clic izquierdo editando en el
↳ editor de escenarios con cualquier modo de edición.

630 + Las refineras pueden ser destruidas.

631 + Las refineras pueden ser atacadas.

632

633 v1.9.1.

634 - Añadido:

635 + Nuevo estado en las unidades: instalando. Para disociar de una vez por todas
↳ edificios y refineras. Sirve para aludir a una unidad que está instalando
↳ una refinería en un géiser.

636 + Nuevo modo de construcción: ya no más creaciones espontáneas, la unidad tiene
↳ que ir hasta donde se vaya a realizar la construcción para comenzarla.

637 - Mejora:

638 + Cambiado el motor interno del juego: procesa todas las órdenes internamente
↳ una vez por ciclo y sin distinciones entre jugadores humanos o artificiales.

639 + Multi-hilado a todos los jugadores artificiales, para mayor equidad.

640 + El formato del indicador de objetivos dentro de partida.

641 + El formato de la tabla de facciones dentro de partida, indica el nombre de los
↳ ejércitos y si están destruidos o no.

642 + Las unidades muestran el estado de manera estandarizada.

643 + El truco de espiar recursos tiene un formato diferente y solo muestra los
↳ ejércitos vivos.

644 - Arreglo:

645 + El juego no se cierra al dejar vacío el nombre para un nuevo escenario.

646 + Se detectan correctamente como aliados a todos los ejércitos cuando se
↳ está haciendo una simulación (se ven las barritas y el cursor en amarillo en
↳ vez de en rojo).

647 + No pone "Equipo", pone "Ejército", en el editor de escenarios.

648 + El juego acaba bien detectando correctamente la raza del jugador humano (sin
↳ cerrarse).

649 + Se puede hacer clic en el modo de edición de objetivos en el editor de
↳ escenarios.

650 + Se ilumina el botón de guardado al cambiar un nombre de facción o de ejército
↳ en el editor de escenarios.

651 + Solo detecta clic en los trucos al crear una partida al hacer clic en el
↳ cuadrado asignado y no en toda la textura del texto.

652 + No está activado por defecto el botón de "editar escenario" en la selección de
↳ escenarios.

653 + Se deselecciona las entidades que estén seleccionadas correctamente (sin dejar
↳ puesto el círculo de selección).

654 + Los botones de las ventanas emergentes en el juego no son "?".

655 + Los trabajadores vuelven a mostrar el cristal o el gas que cargan.

656 + Se ilumina el botón de guardado en el editor de escenarios al cambiar el tipo
↳ de jugador por defecto de uno de los ejércitos.

657 + Las unidades vuelven a mostrar su estado.

658 + Solo aparecen los objetivos en exterminio que están jugando en la partida.

659 + Se actualiza correctamente la facción de un ejército deshabilitado no
↳ seleccionado como sin facción al quitar todos sus elementos del escenario en
↳ el editor de escenarios.

660 + No se muestra el elemento en edición en el editor de escenarios si el cursor
↳ está en la interfaz superior o en el desplegable de las facciones.

661 + Se cargan correctamente los nombres de los ejércitos y de las facciones por
↳ defecto al crear un nuevo escenario en el editor de escenarios.

662 + No se cierra el juego al intentar copiar sin selección en el editor de
↳ escenarios.

663 + El icono de copia en el editor de escenarios aparece correctamente desactivado
↳ al inicio de una edición ya que no se tiene selección.

664 + El icono de copia y el de pegado se actualizan correctamente en diversas
↳ situaciones en el editor de escenarios.

665

666 v1.9.2.

667 - Añadido:

668 + Se deshabilita el paso por las casillas sin suelo.

669 + Orden de recolectar en el nuevo núcleo del juego.

670 - Mejora:

671 + Más paneles emergentes, y menos donde no deberían.

672 + La animación de fin de partida va el doble de rápido.

673 + Mejorado el camino de vuelta de las unidades mineras, ya no hace un movimiento
↳ antes de volver como si estuviera mareada, tanto al mineral como al
↳ edificio.

674 + Nombre para un nuevo escenario escalado y limitado correctamente en los
↳ diferentes casos.

675 + No hay filtrados de memoria (memory leaks), al menos por las texturas de
↳ texto.

676 + Centrado el nombre del nuevo escenario a crear.

677 + Por defecto está la cuadrícula desactivada en el editor de escenarios.

678 + Se ha quitado el multi-hilado en los jugadores artificiales, no resultaba
↳ eficiente.

679 - Arreglo:

680 + Se muestran correctamente los objetivos en el editor de escenarios al crear un
↳ escenario.

681 + No se puede mandar órdenes a las unidades a través de la interfaz dentro de
↳ una partida.

682 + No se activa la animación completa del cursor del ratón al hacer clic derecho
↳ si no se ejecuta una orden dentro de la partida, se muestra el estado del
↳ clic derecho igualmente como parte de la animación.

683 + No da error el mandar a una unidad fuera del escenario: irá a las coordenadas
↳ dentro de lo posible más cercanas.

684 + Se vuelve a permitir copiar sin selección en el editor de escenarios.

685 + No se desactiva el icono de copiar en el editor de escenarios en ningún caso,
↳ ya que no debería.

686 + Se activa correctamente el icono de pegar en el editor de escenarios al
↳ seleccionar algo si hay contenido en el portapapeles

687 + Funciona la edición de nombres de ejércitos en el editor de escenarios.

688 + Se marca correctamente el modo de edición de objetivos, sin desactivarse por
↳ pasar el ratón en el editor de escenarios.

689 + Se muestran las facciones por defecto en el editor de escenarios correctamente
↳ al crear un escenario.

690 + Se trata bien el nombre a la hora de crear un escenario y se refleja bien
↳ también el cursor.

691 + No se borra sin empezar a editar el nombre en la creación de escenario.

692 + Se puede renombrar correctamente en la pantalla de selección de escenario a
↳ editar.

693 + No se puede realizar acciones a través del cartel de facciones dentro del
↳ juego ni se refleja el estado de lo apuntado.

694 + Se cree haber arreglado la línea azul extraña que aparece en el editor de
↳ escenarios sin motivo aparente, pero como no se consigue replicar no es
↳ seguro.

695 + Se restablecen los ajustes de sonido correctamente.

696 + Se crea un escenario correctamente al duplicar uno en la selección de
↳ escenarios.

697 + No se cierra el juego al pulsar borrar selección sin selección en el editor de
↳ escenarios.

698

699 v1.9.3.

700 - Añadido:

701 + Niebla de guerra dura (la permanente, no la temporal).

702 + Puede volver a generarse unidades en los edificios con el nuevo núcleo del
↳ juego.

703 + Se refleja en el minimapa lo visible.

704 + Mutación de la mecánica de construcción Zerg: ahora tal el dron irá como los
↳ demás trabajadores a la zona designada de construcción a diferencia de que
↳ el dron se transformará en el edificio en sí y el resto de unidades no
↳ (antes tenía que estar el dron dentro de la ubicación desde un principio, ya
↳ no).

705 - Mejora:

706 + Se ha bajado la velocidad del dragón, se le ha aumentado la animación y parece
 ↳ más o menos correcta la animación de moverse (es un error heredado de la
 ↳ propia unidad, según internet).

707 + Los colores ahora son más orientativos para el minimapa (gris claro ejército
 ↳ neutral, gris para fondo de mapa y gris oscuro para obstáculos).

708 + Se indica como rojo la construcción que se esté intentando hacer dentro del
 ↳ juego si esta sobresale del mapa.

709 - Arreglo:

710 + No se cierra el juego al mandar a minar a una unidad dentro de una partida.

711 + Se actualizan los ejércitos activos también si se deshabilitan por pegado en
 ↳ el editor de escenarios.

712 + El juego no detecta brevemente algo a través del cartel de facciones al
 ↳ activarlo.

713 + En la selección de escenarios al renombrar un escenario y dejarlo sin nombre
 ↳ muestra correctamente el nombre que hubiera antes.

714 + En el editor de escenarios se borra correctamente el contenido del
 ↳ portapapeles si se copia una selección vacía, y no solo de manera visual.

715 + No se pueden colocar elementos en el editor de escenarios en casillas sin
 ↳ suelo.

716 + Aparecen correctamente los objetivos en todas las partidas y no solo en la
 ↳ primera.

717 + No se cierra el juego al haber ciertos conflictos de movimiento en una unidad.

718 + El juego no se cierra al intentar crear un edificio fuera del escenario.

719

720 v1.9.4.

721 - Añadido:

722 + Modo de juego supervivencia: solo hay que resistir el reloj y se gana. También
 ↳ añadida su correspondiente configuración dentro del editor.

723 + Retratos: aunque son solo banderas, sirven para representar el ejército que
 ↳ está siendo jugado en partida.

724 - Mejora:

725 + Más o menos traída de vuelta la evolución: es la última mecánica que no
 ↳ terminaba de encajar en el nuevo núcleo.

726 + Más optimizado el editor de escenarios (reduce el consumo de procesador aunque
 ↳ no importa porque no tiene turbo speed ni nada pero oye ahí está).

727 - Arreglo:

728 + No se pierde la capacidad de la función del clic derecho si por ejemplo se
 ↳ clicca con el clic izquierdo en una zona distinta dentro del juego (ej: clic
 ↳ en minimapa y luego mover unidades).

729 + De nuevo, no peta el juego al construir fuera del escenario.

730 + Si una unidad muere no se queda como seleccionada dentro del juego.

731 + Los ejércitos se deshabilitan correctamente o se habilitan correctamente
 ↳ también al usar el modo de edición de baldosas.

732 + Se borra los elementos grandes en un escenario aunque se seleccione para
 ↳ eliminar solo una parte pequeña de estos.

733 + Fuera del escenario también se refleja si las facciones han sido eliminadas
 ↳ correctamente al seleccionar todo y borrarlo.

734 + Se dibujan correctamente las zonas de conquista en el editor de escenarios.

735 + No se corrompe el suelo alrededor de una unidad o estructura neutral dentro
 ↪ del juego.

736

737 v1.9.5.

738 - Añadido:

739 + Jugador artificial: variante exterminio. En este nuevo núcleo del juego con
 ↪ las órdenes uniformes y pseudoinmediatas tiene las siguientes mecánicas:

740 * Invadir: Todos los soldados o bien exploran o bien atacan a enemigos
 ↪ cercanos de haberlos

741 * Defenderse: Si hay trabajadores siendo atacados, o edificios o refineras
 ↪ con enemigos cercanos, o en última instancia soldados siendo atacados y
 ↪ hay soldados disponibles, estos acuden, y con esas prioridades, a ayudar
 ↪ en la lucha.

742 * Generar trabajador: La ecuación que sigue para decidir si generarlo o no
 ↪ es bastante enrevesada, pero siempre intentará tener al menos uno
 ↪ creado.

743 + Las configuraciones para las modalidades de conquista e investigación en el
 ↪ editor.

744 + La zona de conquista dentro del juego.

745 + La maravilla como editable dentro del editor.

746 - Mejora:

747 + 40% del código del juego de la lógica de la partida depurado (casi 1500 líneas
 ↪ eliminadas, unas 30 funciones completamente eliminadas, la mitad de las
 ↪ funciones restantes pasadas a privado, etc.), entre otras zonas.

748 + Al no haber más opciones en los menús con flechas aparece una flecha
 ↪ deshabilitada.

749 - Arreglo:

750 + No se muestran los obstáculos fuera de visión en el minimapa dentro del juego.

751 + Las larvas correctamente evolucionan dentro del juego; sin que una larva
 ↪ naranja por ejemplo se transforme en huevo y aparezca un dron azul.

752 + No se quedan seleccionadas ni las larvas ni los huevos ni los drones al
 ↪ evolucionar o mutar.

753 + Las unidades de los huevos ya no ocupan doble población.

754 + No se puede dar órdenes a cualquier ejército, solo al tuyo.

755 + No se ve el tipo de lo apuntado (enemigo, aliado o neutral) a través de la
 ↪ niebla jugando.

756 + El copiar y pegar de baldosas vacías no hace que se transformen en baldosas
 ↪ sin declarar pero con la textura (se ven también en el minimapa).

757 + No se quedan las unidades muertas como selecciones fantasma que puedes
 ↪ escuchar si les das órdenes.

758 + Si se manda una unidad a construir no irá si no está inicialmente el espacio
 ↪ disponible a no ser que sea un trabajador Terran y vaya a terminar de
 ↪ construir una estructura.

759 + No se puede colocar algo encima de una zona de conquista en el editor.

760

761 v1.9.6.

762 - Añadido:

763 + Sonidos de ordenación de generación de unidades (al mandarlos generar, no al
↳ generarlos).

764 + Jugador artificial: variante de exterminio. Nuevas mecánicas:

765 * Construir: Si hay trabajadores, termina las obras pendientes si se es
↳ Terran, y después en todos los casos va comprobando categorías de
↳ edificios, y en caso de encontrar una que tenga la necesidad de
↳ construir más, y se lo pueda permitir, buscará trabajadores dispuestos,
↳ un sitio apto y tratará de construirlo. Solo construye una vez por vez
↳ que se activa esta mecánica.

766 * Generar soldado: En función de los soldados T1 que se pueda permitir, y de
↳ si no hay soldados a secas en el ejército, se mandan crear de 1 a hasta
↳ un tercio de los permitidos + 1 limitado por el número de generadores
↳ que existan (esto es larvas o cuarteles o portales).

767 * Trabajar: Manda a los trabajadores a buscar cristales, o si ha encontrado
↳ los pone a minar. De tener refinería mantiene hasta 3 recolectando gas.

768 - Mejora:

769 + Se refleja el estado de las estructuras de nuevo.

770 - Arreglo:

771 + No se puede atacar con unidades a las propias unidades o a otras unidades
↳ aliadas.

772 + Se puede atacar a las estructuras enemigas.

773 + Se deselecciona las estructuras al ser destruidas.

774 + No se cierra a veces el juego al mover unidades (era por el minimapa: estaba
↳ mal la concurrencia, se vuelve al juego monoproceso por ahora).

775 + Las razas Protoss y Terran vuelven a consumir recursos al generar unidades.

776 + Se realizan varias órdenes que por adaptarse al nuevo núcleo no se hacían (ej:
↳ construir refinerías terranas).

777 + No se termina el juego con "Can't occupy blabla" a veces cuando una unidad
↳ está construyendo o instalando.

778 + No se termina el juego a veces al eliminar a un soldado neutral.

779 + No se dibuja la sombra de los trabajadores al estar recolectando gas.

780 + No se quedan las unidades atascadas si quieren atacar y no llegan a su
↳ objetivo.

781 + Se copia y se pega correctamente las baldosas vacías en el editor.

782 + Las órdenes de los jugadores artificiales ya no producen sonido.

783 + El juego no se cierra al intentar instalar una refinería en el suelo.

784 + Si se activa el truco de niebla ya se refleja lo apuntado en el cursor.

785

786 v1.9.7.

787 - Añadido:

788 + Las refinerías pasan a ser restrictivas: solo puede recolectar un trabajador en
↳ una refinería por vez. Si un trabajador intenta entrar estando esta ocupada,
↳ este esperará fuera.

789 + Modo de conquista: Consiste en alcanzar 1000 puntos de conquista para ganar.
↳ Estos decaen por si solos y se incrementan con cada casilla ocupada de la
↳ zona de conquista por algún elemento de la facción conquistadora (unidades,
↳ edificios, etc.). También se pierden puntos de conquista si hay elementos de
↳ facciones enemigas en la zona a conquistar.

790 + Maravilla dentro del juego como opción construible para las 3 razas, estético
↳ por ahora.

791 + Mis escenarios de depuración y pruebas, para algo pueden servir.

792 - Mejora:

793 + Carga de órdenes de los jugadores artificiales en cada turbociclo (mejora la
↳ equidad con el modo turbo).

794 + El autómata de la unidad minera no busca otro mineral si no encuentra cómo
↳ llegar al que inicialmente está intentando alcanzar. Pero sí que busca un
↳ nuevo mineral libre en el caso de que el que esté intentando minar
↳ esté ocupado.

795 - Arreglo:

796 + El jugador artificial no manda órdenes inútiles de generar trabajadores a
↳ edificios que no tienen dicha capacidad.

797 + Los minerales son restrictivos: solo se pueden minar con un trabajador por
↳ vez.

798 + Se pueden simular partidas de nuevo.

799 + Las unidades que mueren de una selección grupal se retiran de la selección
↳ visualmente.

800 + Termina la partida si la facción del jugador humano, si lo hay, pierde.

801 + Se construyen correctamente las refineras terranas creadas por el jugador
↳ artificial.

802 + Copiar y pegar vuelve a copiar y pegar los elementos ocupantes y no solo el
↳ suelo en el editor.

803 + El jugador humano también puede construir refineras Terran con más de un
↳ trabajador (el que procesaba la orden de construcción era el único que
↳ podía).

804 + Los trabajadores Terran pueden ir a su refinera a por gas si los manda el
↳ jugador humano.

805 + Se detiene el modo turbo al pulsar la tecla de salir o al hacer clic en salir.

806 + Al estar construyendo o instalando no se refleja lo que hay a través de la
↳ niebla en las marcas de construcción dentro de una partida.

807 + No se cierra el juego con "Coordinates doesn't exist" construyendo con Terran
↳ edificios sin acabar en partida.

808

809 v1.9.8.

810 - Añadido:

811 + Nueva información dentro de partida:

812 * Autómata para los minerales: Estado "partiéndose" a falta de un término
↳ mejor.

813 * Indicador de gas: marcando cuando está agotado y solo se puede sacar gas
↳ de baja calidad.

814 * Cola de generación de unidades.

815 - Mejora:

816 + El escenario StarCraft está ahora más balanceado (para hacer pruebas, aún no
↳ me meto en tema de escenarios).

817 + Normalizado y rehecho prácticamente todo el sistema de información dentro de
↳ partida antiguo.

818 + Intercambiados el indicador de velocidad con el del botón de despausar, para
↳ mejor estética.

819 + Al alcanzar el modo turbo se deshabilita el botón de acelerar y se ve mejor el
↳ indicador de velocidad.

820 + Los colores del botón de acelerar siguen más el estándar.

821 + Paneles emergentes para los botones del juego de estructuras, refineras y
↳ unidades: lo que había antes era un intento bastante pobre.

822 - Arreglo:

823 + El escenario HelmAbyss no cierra el juego al intentar jugarlo, pero no
↳ está terminado.

824 + El duplicar escenario en la selección de escenario si se renombra con un
↳ nombre cuyo código numérico esté ordenado antes de "A" en el primer
↳ carácter, esto es, ya da igual el nombre que se introduzca que se va a
↳ renombrar correctamente.

825 + Se reinicia correctamente el icono de visibilidad de objetivos al cambiar de
↳ escenario en el editor de escenarios.

826 + La selección de escenarios en otra página que no sea la primera en la
↳ selección de juego.

827 + No se cierra el juego con "Coordinates doesn't exists" cuando la IA pasa cerca
↳ de un límite del escenario.

828 + Los trabajadores van mirando hacia donde deben cuando van a instalar una
↳ refinera o a recolectar gas dentro de una partida.

829 + Al ir a construir con los Terran no se sustituye el área de construcción por
↳ la inmediatamente inferior o derecha, o inferior derecha si el clic no
↳ coincide con la primera baldosa de la construcción resultando en
↳ construcción a distancia.

830 + Las unidades no cambian de objetivo al ser atacadas si han empezado a cargar
↳ el ataque, esto es, que han llegado a rango del objetivo que tienen. En caso
↳ contrario si.

831 + El indicador de las refineras se muestra correctamente y no invisible a veces.

832 + Los trabajadores recolectando ya no dan una vuelta innecesaria al salir de una
↳ refinera.

833 + El jugador humano puede atacar a refineras enemigas.

834 + No se muestra como pulsable el botón de página anterior en la primera página
↳ de las opciones de edición de los modos de edición del editor de escenarios.

835 + Se comprueba al borrar una selección o al pegarla que se habiliten o
↳ deshabiliten las facciones correspondientes en el editor de escenarios.

836

837 v1.9.9.

838 - Añadido:

839 + Información extra en la pantalla de final de partida.

840 + Modo de juego de investigación: Investiga maravillas hasta su último nivel
↳ para completar una. Si entre toda la facción se consigue mantener
↳ simultáneamente un número objetivo de maravillas al máximo nivel se gana la
↳ partida.

841 + Refineras en el editor de escenarios.

842 + Nuevo estado en las refineras: Lista. Refleja cuando la refinera
↳ está construida pero vacía.

843 - Mejora:

844 + Al destruir una refinera el trabajador que esté dentro sufre una muerte
 ↳ terrible y dolorosa.

845 + Disminuido el enfriamiento de regeneración de vida Protoss: 200 -> 120 ciclos.

846 + Aumentado el enfriamiento de regeneración de escudo Protoss: 50 -> 60 ciclos.

847 + Disminuido el enfriamiento de regeneración de vida Terran: 200 -> 80 ciclos.

848 + Aumentado el enfriamiento de regeneración de vida Zerg: 20 -> 50 ciclos.

849 + Rebalanceado el escenario StarCraft.

850 - Arreglo:

851 + La barra de generación no se extiende por encima del límite si no consigue
 ↳ generar la unidad una estructura dentro del juego por falta de espacio.

852 + El jugador artificial detecta correctamente los trabajadores recolectando y
 ↳ solo manda hasta 3, no más, en partida.

853 + Las refineras al explotar no vacían el gas del géiser.

854 + La selección de una partida a otra se conservaba.

855 + Los trabajadores terranos dejan de ir a la muerte periódicamente, lo cual
 ↳ aumenta significativamente sus posibilidades de supervivencia en el juego.

856 + Las refineras se reflejan correctamente como neutrales en simulaciones.

857 + Las refineras Terran dejan de ser consideradas enemigos para los de su propia
 ↳ facción dentro del juego.

858 + Se aplican algunos balances que se habían introducido con anterioridad, ya que
 ↳ por despiste no estaban siendo aplicados:

859 * Marines: atacan el doble de rápido, regeneración de vida 100 -> 80 ciclos.

860 * VCE: atacan el doble de rápido, velocidad 4 -> 3 ciclos.

861 * Zergling: atacan el doble de rápido, regeneración de vida 20 -> 50 ciclos.

862 * Sondas: atacan el doble de rápido.

863 * Zumbidos: atacan el doble de rápido, regeneración de vida 20 -> 50 ciclos.

864 * Larva: regeneración de vida 20 -> 50 ciclos.

865 * Huevo: vida máxima 200 -> 100 puntos.

866 * Academia: regeneración de vida 20 -> 80 ciclos.

867 + No se cierra el juego con "Can't empty an already empty tile" a veces porque
 ↳ no se permite evolucionar una larva muerta.

868 + El jugador artificial termina de construir los almacenes de suministros.

869 + No se suma el suministro extra nada más colocar un almacén de suministros.

870 + La selección de escenarios para editarlos funciona correctamente a partir de
 ↳ la primera página.

871 + La selección de escenarios para jugarlos funciona correctamente a partir de la
 ↳ primera página.

872 + No se rompe el juego al jugar un escenario vacío, simplemente termina.

873 + Los trabajadores no tienen que terminar su camino actual si se les da una
 ↳ orden a mitad al construir.

874 + Los trabajadores se desplazan mirando donde deben mirar.

Versiones 2.0.X

875 v2.0.0.

876 - Añadido:

877 + Jugador artificial: variante superviviente. Sin capacidad de invadir, y
 ↳ capacidades defensivas mejoradas:

878 * Desplaza las unidades recién salidas de los edificios para no molestar a
 ↳ los trabajadores.

879 * Los soldados mantienen las posiciones al no haber movimientos invasivos.

880 * Se añade defensa indirecta a los soldados, de forma que actúan como
 ↳ vigilantes.

881 * Prioridades de defensa nuevas (en resumen: primero defiende trabajadores,
 ↳ luego soldados, luego edificios).

882 + Jugador artificial: variante conquistadora. En resumen se han cambiado las
 ↳ facultades defensivas y ofensivas para priorizar el intentar llegar hasta la
 ↳ zona de conquista.

883 + Paneles emergentes descriptivos: pasa el cursor por el nombre de lo
 ↳ seleccionado dentro del juego. Por lo general he sintetizado la historia del
 ↳ juego.

884 + Sonidos de huevo: No sé describirlos, pero tiene sonidos.

885 + Añadido control de pausar / despausar el juego, con la barra espaciadora por
 ↳ defecto.

886 - Mejora:

887 + Bottleneck actualizado: escenario oficial de supervivencia contra exterminio
 ↳ (1 vs 1).

888 + Al hacer clic mientras el juego espera la pulsación para un control en el
 ↳ menú de ajustes se desactiva dicha espera.

889 + Las defensas ya no muestran barra de vida (da más realismo).

890 + HelmAbyss actualizado: escenario oficial de conquista contra supervivencia (2
 ↳ vs 1).

891 - Arreglo:

892 + El juego no se cierra al cargar una partida con facciones que no estén en las
 ↳ primeras posiciones.

893 + El editor de escenarios marca como que hay cambios al actualizar los
 ↳ parámetros de una facción superviviente.

894 + El indicador en final de partida muestra correctamente la duración, ej: 12:03
 ↳ y no 12:3.

895 + Los murflama y los dragones tienen tiempo de generación.

896 + Más o menos tratado el problema de los escenarios con minúsculas o mayúsculas
 ↳ que se consideraban distintos pero se guardaban igual.

897 + Las refineras no se quedan inmortales si se destruyen justo al acabar una
 ↳ extracción.

898 + El nivel 1 del aura de las maravillas se dibuja correctamente.

899 + Al cambiar un control en el menú de ajustes dejando el cursor sobre el control
 ↳ que se está cambiando se ilumina correctamente la nueva tecla ajustada una
 ↳ vez se cambia sin tener que mover el cursor otra vez.

900 + No aparece una zona de conquista extraña en las coordenadas (-1, -1) dentro
 ↳ del editor de escenarios.

901 + Vuelve a seleccionarse correctamente todas las unidades a partir del huevo
 ↳ (zerg y terran) en el editor de escenarios y no se cierra el juego al
 ↳ intentar editar zumbidos.

902 + Se ajusta correctamente el nombre de facción al vaciarlo en el editor de
↳ escenarios.

903 + Las unidades no se atacan en ciertas circunstancias siendo de la misma facción
↳ en partida.

904 + No termina el juego a veces al terminar un mineral a destiempo.

905 + No termina el juego si se borra la zona de conquista asignada a una facción y
↳ sin que esté vacía la lista de zonas de conquista se edita de nuevo la
↳ facción en el editor de escenarios.

906 + Los almacenes de suministros destruidos sin terminar de construir no restan
↳ población máxima.

907 + No se cierra el juego en determinadas circunstancias al reentrar en el menú de
↳ selección de escenarios para el editor tras salir y haber dejado
↳ seleccionado un escenario renombrado.

908 + Se actualiza correctamente la lista de escenarios para el editor de escenarios
↳ al renombrar.

909 + Se actualiza el estado habilitado o inhabilitado de cada facción correctamente
↳ al añadir o al borrar una refinera en el editor de escenarios.

910 + Las refineras Zerg ya se construyen.

911 + Las refineras ya inician correctamente al finalizar su construcción en el
↳ estado de "Lista".

912 + No aparece una página vacía según el número de escenarios en la pantalla de
↳ selección de juego.

913 + La selección de un trabajador que explota en un refinera se limpia
↳ correctamente.

914 + El juego ya no acaba si el espiar recursos está activo en un escenario vacío.

915 + La partida termina al perder la facción del jugador humano correctamente.

916

917 v2.0.1.

918 - Añadido:

919 + Animación de instalación del extractor.

920 + Animación de muerte a la larva.

921 + Animación pequeña de destrucción de estructuras protoss.

922 + Animaciones de todo tipo para la destrucción de estructuras Terran.

923 + Animaciones de todo tipo para la destrucción de estructuras Zerg.

924 + Contador de muertes para los vencedores en la pantalla de fin de partida.

925 + Animación de muerte de Hidralisco.

926 + Animación de muerte de Soberano.

927 + Animación de muerte de Huevo.

928 - Mejora:

929 + Las unidades atacan a la refinera si están intentando atacar al trabajador que
↳ esté dentro en partida.

930 + El escenario de StarCraft tiene por defecto los jugadores artificiales a
↳ dificultad Inhumana.

931 + El campo de ganador se oculta en la información de final de partida si se ha
↳ perdido.

932 + El icono del juego ahora es más fantástico.

933 + Más agradable el tiempo de actualización de las animaciones de las
↳ estructuras: 10 -> 4 ciclos.

934 + Ajuste en el jugador artificial conquistador: 33% -> 50% de probabilidades de
 ↪ ayudar a unidad/estructura/etc. en vez de intentar ir a la zona de conquista
 ↪ (en todas las situaciones).

935 + Reducido el consumo de procesador del modo turbo en un 15% añadiendo una
 ↪ "espera de razonamiento" a los soldados de forma que si no encuentra camino
 ↪ a un objetivo, no cambia inmediatamente de objetivo. Evita también los
 ↪ colapsos del juego cuando varias unidades se traban en un sitio cerrado.

936 + Balanceado Bottleneck.

937 + Balanceado HelmAbyss.

938 + Balanceado StarCraft.

939 + Ventana emergente de salida del juego.

940 + Animación de construcción Zerg mejorada.

941 - Arreglo:

942 + Hacer clic dentro del juego en el minimapa desplaza correctamente la cámara.

943 + El minimapa refleja correctamente la cámara dentro del juego (el rectángulo
 ↪ blanco).

944 + El juego no conserva el estado del ratón al acabar una partida haciendo cosas
 ↪ raras como dejar el clic izquierdo pulsado y así al volver a entrar a otra
 ↪ partida.

945 + La refinera Terran vuelve a progresar al estar extrayendo gas en partida.

946 + Los constructores Terran miran correctamente al centro de la refinera al
 ↪ instalarla en partida.

947 + Los centros de operaciones ya aparecen correctamente contruidos al empezar
 ↪ una partida.

948 + Al borrar caracteres en los parámetros de facción de supervivencia en el
 ↪ editor de escenarios se actualiza correctamente el botón de guardado como
 ↪ pulsable.

949 + Se permite recolectar a los trabajadores Zerg en partida.

950 + Ahora sí, deshabilita una facción correctamente al borrar una refinera en el
 ↪ editor.

951 + Cambiar dificultad del jugador artificial por defecto en el editor sí se
 ↪ refleja el botón de guardado.

952 + Se pierde correctamente al jugar en una partida si tu facción ha sido
 ↪ derrotada al completo.

953 + Se desactiva el círculo de selección a las unidades que están muriendo.

954 + No se queda a veces un trabajador en modo "invisible" tras ser mandado a hacer
 ↪ ciertas acciones mientras estaba recolectando, pasando a ser inmune y
 ↪ organizando una avalancha detrás.

955 + Destruir un huevo con un soberano en su interior ya no disminuye población que
 ↪ no existe.

956

957 v2.0.2.

958 - Añadido:

959 + Sonidos de ataques (zumbidos y VCEs usan los de su minado, los zergling el att
 ↪ de los broodling y los hidraliscos, marines y murflamas usan su fir, no se
 ↪ han encontrado otros mejores).

960 + Asesinatos en el panel dentro del juego.

961 + La tecla enter permite terminar de editar un campo de texto.

962 + Jugador artificial: variante investigadora.

963 + Anunciado Unizar y autor en el comienzo del juego.

964 - Mejora:

965 + Escenario OpenStarCraft.

966 + Autómata del jugador artificial para los trabajadores, ahora no manda a minar
 ↪ a todos sus trabajadores si hay minerales visibles, solo manda uno por
 ↪ mineral.

967 + Calibrado cómo se muestra la cámara en el minimapa de nuevo.

968 - Arreglo:

969 + Las refineras del escenario aparecen construidas al comenzar una partida.

970 + Las miniaturas de las estructuras generando indican los tipos de unidad que
 ↪ generan correctamente.

971 + Los criaderos aparecen correctamente contruidos y generando larvas al
 ↪ comienzo de partida.

972 + Se comprueba las facciones y no los ejércitos a la hora de ver si una facción
 ↪ exterminadora ha ganado en partida.

973 + Los trabajadores no se quedan en "forma fantasmal" si les explota la refinera
 ↪ en la que están.

974 + En los objetivos de facción exterminadora también se refleja ahora los
 ↪ ejércitos aunque se evalúe por facción (al mostrarlos dentro de partida).

975 + La selección de un ejército a otra partida no se posterga si se después se
 ↪ hace una simulación.

976 + El panel emergente de las refineras se muestra correctamente.

977

978 v2.0.3.

979 - Añadido:

980 + Menú información de juego: Tiene 5 conceptos añadidos para su disfrute por
 ↪ ahora.

981 + Menú información del editor de escenarios: No se añaden conceptos aquí hasta
 ↪ que se de el visto bueno a ambos menús de información.

982 + Paneles emergentes para los trucos en el menú de creación de partida.

983 - Mejora:

984 + Disminuida la regeneración de vida Terran: 80 -> 120 ciclos.

985 + Los paneles emergentes del ratón ahora se desplazan en las 4 direcciones
 ↪ (esquivando convenientemente al cursor si se da el caso) para verse en todo
 ↪ momento en pantalla.

986 + Actualizado HelmAbyss.

987 + El juego es más oscuro, al cambiar el color de la fuente de blanco a gris
 ↪ claro.

988 + OpenStarCraft tiene su versión alternativa de exterminio.

989 - Arreglo:

990 + Las unidades no tienen ataques en falso dentro de la partida.

991 + El juego no se cierra al cargar una partida con las facciones desordenadas y
 ↪ con algunas facciones deshabilitadas.

992 + Corregido el escenario OldOpenSC.

993 + Se muestra correctamente sí que sí la cámara en el minimapa dentro de partida.

994 + Los trabajadores vuelven a atacar mirando hacia dónde atacan.

995 + Editar el número de maravillas a construir por una facción en el editor de
 ↪ escenarios refleja cambios a guardar.

996 + Editar la zona de conquista objetivo de una facción en el editor de escenarios
 ↪ refleja cambios a guardar.

997 + La tabla de facciones refleja correctamente los datos dentro del juego.

998

999 v2.0.4.

1000 - Añadido:

1001 + La cámara apunta inicialmente a donde esté el jugador humano si lo hay.

1002 + Inteligencia artificial en la cámara del juego en las simulaciones.

1003 + Información del editor de escenarios: un nuevo concepto añadido.

1004 + Ventana emergente de salida en la pantalla inicial.

1005 + Opción de editar la descripción de un escenario en el editor de escenarios. Se
 ↪ refleja en el formato de los escenarios y está actualizado en los escenarios
 ↪ oficiales.

1006 + Reescalado de la ventana del juego a 4:3 cuando se cambia de tamaño de la
 ↪ ventana fuera de la pantalla completa.

1007 + Adiós pantalla completa. Ocasionaba muchos problemas que no se pueden tratar
 ↪ sin tocar sistema operativo. Se pueden sortear pero si eso se hará más
 ↪ adelante.

1008 + Ajuste de velocidad de transiciones: para configurar la velocidad de las
 ↪ transiciones de menús y cosas así.

1009 + Ajuste de postergación de ventanas emergentes.

1010 - Mejora:

1011 + Rehecho el menú de creación de partida, se ve lo que he hecho, no voy a
 ↪ listarlo aquí.

1012 + Rehecho el menú de selección de juego, más de lo mismo.

1013 + Los sistemas de paginación ahora están ocultos si no se supera una página de
 ↪ resultados.

1014 + Hacer clic no termina de editar los campos de texto.

1015 + Zum en partida disminuido, la escala máxima de visión es: x1.5 -> x2.

1016 + Zum máximo en general aumentado, la escala mínima de visión es: x1 -> x0.4.

1017 + Interfaz del editor de escenarios.

1018 + Descoloridos los iconos para darles animación de control, esto es: flechas a
 ↪ gris, lapicerito a gris, engranaje a gris...

1019 + Botón de desplegable de facciones del editor de escenarios.

1020 + Suavizado a todas las baldosas para evitar las líneas feas de cosido en
 ↪ cualquier momento y con cualquier zum.

1021 + Más reactivo el menú de ajustes.

1022 - Arreglo:

1023 + Asimiladores muestran su panel emergente descriptivo correctamente.

1024 + Según qué ciertas combinaciones de navegación por los menús ya no se ilumina a
 ↪ veces el fondo de colores atípicos momentáneamente.

1025 + La cámara no se mueve al activar el desplegable de facciones en el editor de
 ↪ escenarios.

1026 + Los menús de información no dejan el concepto como seleccionado al presionar
 ↪ la tecla de salir.

1027 + Los géiseros no se quedan seleccionados al desaparecer porque se les ha
↳ instalado una refinera encima.

1028

1029 v2.0.5.

1030 - Añadido:

1031 + Sistema de daño casi nuevo:

1032 * Los atacantes a rango no infligen daño a las defensas a penas, solo los de
↳ corto alcance. Daño a las defensas de las unidades que no ataquen cuerpo
↳ a cuerpo: x1 -> x0.2.

1033 * Animaciones de recibir ataques excepto de los producidos por los
↳ trabajadores ya que son animados de otra manera.

1034 * Las unidades que están atacando, sin cargar el ataque, se giran a quien
↳ les ataque y le atacan de estar a rango.

1035 + El jugador artificial no va a minerales aleatorios de los conocidos, va a los
↳ cercanos primero.

1036 + Al cancelar una edición de nombre pulsando Esc, vuelve a un estado inicial.

1037 + Los géiseros se desgastan visualmente.

1038 - Mejora:

1039 + Los nombres de los escenarios se calculan con tamaño dinámico y no fijo.

1040 + Los rectángulos de los escenarios en la lista de escenarios en cualquier
↳ menú de selección están correctamente centrados (se me habían ido un píxel
↳ con las mates jeje).

1041 + Se pueden crear escenarios con nombres con comas, puntos, espacios y números
↳ además de letras.

1042 + Nombre de escenario oficial: Bottleneck -> Thermopylae.

1043 + Rehecho el menú de creación de escenario:

1044 * Posicionado de botones donde toca para seguir el estándar.

1045 * Escalado de textos.

1046 * Cambiado el tamaño de la interfaz.

1047 * Cambiado a campos de texto las filas y las columnas.

1048 * Estandarizadas las separaciones de textos.

1049 * Etc, etc, etc.

1050 + Ajustada la animación del menú de información del juego.

1051 - Arreglo:

1052 + El juego no revienta al crear un escenario y guardarlo sin descripción.

1053 + Cabe en la creación de escenarios el nombre de una plantilla dentro de los
↳ límites.

1054 + Se muestran correctamente los recursos de los escenarios seleccionados de
↳ manera posterior al primero.

1055 + Se muestran correctamente los objetivos de las facciones de exterminio en la
↳ selección de escenarios.

1056 + Se portan correctamente los paneles emergentes de cambiar de ejércitos en el
↳ editor de escenarios al estar estos inactivos.

1057 + Se deselectiona correctamente el último concepto seleccionado al pulsar ESC en
↳ el menú de información de juego y en el menú de información del editor de
↳ escenarios.

1058 + Se activa el botón de guardado al borrar caracteres de la descripción del
↳ escenario en el editor.

1059

1060 v2.0.6.

1061 - Añadido:

1062 + Animación de instalación de extractores fiel a las estructuras Zerg (tenía el
↳ modelo antiguo).

1063 + Control de selección: por defecto es la tecla enter.

1064 + Sistema de navegación por teclado, sonido de transición, más sonidos de
↳ flotado (son los sonidos de "hover" por flotar el ratón sobre un elemento) y
↳ sonidos de error en:

1065 * Menú principal (el de escoger entre jugar, editor de escenarios, ajustes y
↳ salir).

1066 * Menú de ajustes.

1067 * Menú de juego (el que hay tras el menú principal).

1068 * Menú de editor de escenarios (el que hay tras el menú principal).

1069 + Los edificios sin terminar de construir no tienen regeneración.

1070 + Mecánica de progresión de vida y escudo en una construcción: se podía
↳ construir una estructura y que esta nada más aparecer tuviera el 100% de
↳ vida y de escudo; la nueva mecánica irá aumentando ambos campos conforme se
↳ vaya construyendo (partiendo con una pequeña base de vida y escudo). Esto no
↳ solo facilita la destrucción de edificios recién ordenados construir sino
↳ que también dificulta destruir un edificio que esté construyéndose en
↳ general puesto que la cura total es superior a los valores máximos de dicha
↳ estructura (en instalación de refineras también todo).

1071 + Interfaz de botones escondida en el editor de escenarios sin modo de edición,
↳ para poder añadir la mecánica de editar la dimensión del escenario desde
↳ dentro.

1072 + Mecánica para editar las dimensiones del escenario desde el editor de
↳ escenarios.

1073 + Se permite añadir guiones a los escenarios.

1074 - Mejora:

1075 + Rediseñados casi todos los iconos del editor de escenario para que se vean
↳ mejor (esto está desde la versión anterior, perdón que se me pasó ponerlo).

1076 + Los paneles emergentes con costes de dentro del juego solo muestran los costes
↳ o cambios de verdad, no los que valen por ejemplo 0 de algún recurso o que
↳ no repercuten en la población actual.

1077 + Los paneles emergentes ahora reflejan correctamente el tipo de población
↳ actual en función de la raza.

1078 + Se enfoca a un trabajador si no hay unidades, edificios o refineras al empezar
↳ a jugar una partida.

1079 - Arreglado:

1080 + Los edificios se terminan de destruir.

1081 + La animación del menú de información de juego escala con la velocidad de
↳ transiciones.

1082 + La mecánica de borrar del editor de escenarios elimina correctamente todo lo
↳ seleccionado.

1083 + Las pantallas de selección de escenarios no marcan a uno mismo como objetivo
↳ si exterminan.

1084 + El juego no se cierra por las últimas mecánicas del jugador artificial cuando
↳ esta no tiene edificios.

1085 + Se muestra correctamente los objetivos si juegas en una facción exterminadora.

1086 + Se muestra correctamente en la selección de escenarios los objetivos de las
↳ facciones investigadoras si hay que investigar solo una maravilla.

1087 + Corregido el panel emergente de compra del centro cibernético.

1088

1089 v2.0.7.

1090 - Añadido:

1091 + Ajuste extra para habilitar / deshabilitar el control de la cámara con el
↳ cursor.

1092 + Sistema de navegación por teclado y sonidos nuevos:

1093 * Menú de selección de juego.

1094 * Menú de selección de escenario.

1095 * Menú inicial.

1096 * Menú de información de juego.

1097 * Menú de información de editor de escenarios.

1098 * Menú de creación de escenarios.

1099 + Concepto del juego: Exterminar.

1100 - Mejora:

1101 + Concepto del juego: Facción.

1102 + Concepto del juego: Protoss.

1103 + Concepto del juego: Terran.

1104 + Formato de fichero de configuración: se pueden borrar líneas, ajustar al
↳ gusto, leer y entenderlo, reordenar líneas, etc.

1105 + Puede escribirse comas, puntos, espacios y números en el nombre del escenario
↳ también en la pantalla de carga de escenarios.

1106 - Arreglo:

1107 + Navegación en el menú del editor de escenarios (con acceso a info, carga o
↳ creación de escenarios), no le faltan cosas.

1108 + Algunos aspectos visuales, fallos que cerraban el juego y demás del menú de
↳ ajustes.

1109 + No se puede crear un escenario sin filas o sin columnas.

1110 + El editor de escenarios refleja correctamente cuando desaparece un ejército
↳ por borrado de filas o columnas.

1111 + El editor de escenarios activa el botón de guardado al alterar la dimensión
↳ del escenario.

1112 + No se puede crear un escenario con el nombre vacío (o aparentemente vacío).

1113 + No se muestran los paneles emergentes de los botones desactivados en el editor
↳ de escenarios.

1114 + Se cargan los objetivos del jugador correctamente en una partida evitando
↳ cambios de objetivo o que se cierre el juego.

1115 + La animación del menú de información del editor.

1116 + Las refineras y estructuras que están al empezar una partida aparecen con la
↳ vida y el escudo al máximo.

1117 + La ventana emergente de los botones se actualiza correctamente al presionar o
↳ clicar los botones dentro del juego.

- 1118 + No se quedan partidas bloqueadas por muertes indebidas de los jugadores
↪ artificiales que aún siguen vivas.
- 1119 + No hay condiciones de carrera en la carga de escenarios del editor.
- 1120 + No se procesan las órdenes hasta que se sale de la pausa de una partida.

Expansiones opcionales

Versiones 2.1.X

- 1121 v2.1.0.
- 1122 - Añadido:
 - 1123 + Guardado y carga de partidas.
 - 1124 + Animación de ataque de los VCE, la última que quedaba.
 - 1125 + Texto de "escenarios" a la selección de escenarios en el editor.
- 1126 - Mejora:
 - 1127 + Menú de cargado de partidas.
 - 1128 + Cambios en el motor para ahorrar uso de memoria (optimizaciones varias,
↪ limpieza, etc.).
- 1129 - Arreglo:
 - 1130 + Suenan las músicas ingame cuando tocan, no aleatoriamente (aleatoriamente si,
↪ pero cuando se juega partida se escoge entre una de las canciones posibles
↪ de la raza).
 - 1131 + Se ejecuta el juego sin fichero de configuración.
 - 1132 + Se puede silenciar el juego desde el menú inicial (el de la pantalla de carga
↪ y tal).
 - 1133 + Algunos edificios o unidades aún funcionaban con la regeneración antigua: se
↪ han estandarizado y balanceados todos por raza.
 - 1134 + Suena el control de salir en el menú inicial al mostrar la ventana emergente
↪ de salida.
 - 1135 + Opacidad correcta en el concepto Protoss en información del juego.
 - 1136 + Mover el ratón deselecta correctamente en el menú de información de juego.
 - 1137 + No carga siempre un concepto en el menú de información de juego, solo
↪ cargará uno si es distinto al seleccionado. Al intentarlo sonará el
↪ característico sonidito de error.
 - 1138 + Las extractoras tienen animaciones de sufrir daño.
 - 1139 + La visión con cuadrícula dentro de partida que por las medidas del suavizado
↪ se había fastidiado.
 - 1140 + Se muestra el indicador de edición en los escenarios en la selección de
↪ escenarios.
 - 1141 + Se deja de escribir correctamente un escenario al cambiar de zona.
 - 1142 + Actualiza correctamente el número de filas y columnas al borrar un escenario.
 - 1143
- 1144 v2.1.1.
- 1145 - Añadido:
 - 1146 + Los guardados se almacenan por orden cronológico.
- 1147 - Mejora:
- 1148 - Arreglo:

1149 + Se puede borrar todos los escenarios a cargar sin que pete el juego.
 1150 + Se puede cargar correctamente las partidas.
 1151 + Se puede guardar correctamente las partidas.
 1152 + Refineras que no sean Terran se terminan de construir.
 1153 + Centros cibernéticos no se regeneran cuando no deben.
 1154 + Se puede cargar una partida siendo el jugador.
 1155 + El editor no tiene una posibilidad de cerrar el juego (altísima probabilidad).
 1156 + Se cargan correctamente los colores de las velocidades a cargar partidas.
 1157 + No se puede duplicar un escenario y ponerle nombre vacío.
 1158 + Se actualizan correctamente las filas y columnas al borrar un escenario en la
 ↪ selección de escenarios para editarlos.
 1159 + Los VCE miran donde toca al construir.
 1160 + El botón de borrado de guardados se desactiva correctamente al no quedar más
 ↪ guardados.

Versiones 2.2.X

1161 v2.2.0.
 1162 - Añadido:
 1163 + Ajuste de elección de retrato para las partidas en línea: 21 en total, 7 por
 ↪ raza.
 1164 + Ajuste de elección de nombre para las partidas en línea.
 1165 + Menú multijugador.
 1166 + Menú un jugador.
 1167 + Indicador de carga en la pantalla inicial (si, otro más).
 1168 + Menú de selección de escenario multijugador.
 1169 + Tabla de casos online en el documento asociado.
 1170 + Agresividad del jugador artificial en el editor.
 1171 + Juego multijugador en línea.
 1172 - Mejora:
 1173 + Mejor fondo para el menú de creación de escenario.
 1174 + Menú de selección de escenario individual.
 1175 - Arreglo:
 1176 + No parpadea el cursor al entrar en algunos menús por primera vez.
 1177 + No se cierra seleccionar visión de facción con algún escenario en una
 ↪ selección de escenario.
 1178
 1179 v2.2.1.
 1180 - Añadido:
 1181 + Mensaje echar del sistema en sala multijugador.
 1182 + El pulsar la tecla de salir deselectiona escenario en sala multijugador.
 1183 - Arreglo:
 1184 + No se cierra el juego al darle a seleccionar como cliente.
 1185 + No se cierra al sustituir un color por otro en sala multijugador.
 1186 + El juego trata bien los nuevos clientes en la sala multijugador.
 1187 + El juego no se cierra al entrar como cliente en juego multijugador.
 1188 + Los colores no se van al seleccionar escenario en juego multijugador.
 1189 + Los colores no se quedan en el servidor al cambiar de controlador de un
 ↪ ejército.

1190 + No se marca el control del ejército anterior si se asigna el jugador a
 ↪ controlar otro ejército.

1191 + La indexación de páginas en el multijugador funciona sin cerrar el juego.

1192

1193 v2.2.2.

1194 - Añadido:

1195 - Mejora:

1196 + Modo multijugador en línea: se ha reducido el número de caracteres enviados
 ↪ por ciclo de actualización del escenario (Ejemplo, en StarCraft: 660000 ->
 ↪ 12000 char).

1197 + Se puede mover la cámara en partida en línea.

1198 + Se puede aumentar o disminuir el zum en partida en línea.

1199 + Carga el escenario a los clientes en partida en línea.

1200 + Se refleja lo ocurrido en el servidor en el cliente en el modo en línea.

1201 + Reducido el consumo de procesador en el modo en línea al no hacer duplicados
 ↪ de los datos del escenario.

1202 + Reducido el consumo de datos en modo en línea al no pasar datos de cálculo ya
 ↪ que en los clientes solo se dibuja.

1203 + Refactorizadas todas las entidades, ahora tienen menos atributos lo cual
 ↪ mejora la lógica, reduciendo el consumo de procesador y de datos a guardar o
 ↪ compartir en el modo en línea (lo cual reduce la latencia).

1204 + Rehecho el sistema de las estructuras, todo lo que es el autómata, para
 ↪ igualarlo al de las unidades y así poder cargar textura en cualquier
 ↪ momento, permitiendo que se vean en el modo en línea y en carga de partida
 ↪ nada más entrar.

1205 + Se carga el reloj en los clientes en partida en línea.

1206 + Se carga el minimapa en los clientes en partida en línea.

1207 - Arreglo:

1208 + Modo de crear partida individual, que cerraba el juego con los cambios hechos
 ↪ para introducir el modo multijugador.

1209 + Guardado de partidas individual, por lo mismo.

1210 + Carga de partida individual, por lo mismo.

1211 + El juego en línea ya no entra con el modo cuadrícula activado.

1212 + Al entrar en una partida cargada ya aparecen las texturas de todo sin tener
 ↪ que pasar ciclo.

1213

1214 v2.2.3.

1215 - Añadido:

1216 + Tras finalizar una partida en línea se va a la pantalla de fin de partida y
 ↪ luego se vuelve a la sala inicial.

1217 + Chat en partida en línea.

1218 + Retrato dentro de partida en línea.

1219 + Nombre de jugador dentro de partida en línea.

1220 + Tabla facciones en partida en línea siendo cliente o servidor.

1221 + Botones de pegado de texto para el menú en línea (solo gráficamente por
 ↪ ahora).

1222 - Mejora:

1223 + Se puede atacar al servidor con otros ejércitos en partida en línea.

1224 + No se cierra el juego con el fin de partida en el modo en línea.
 1225 + Se muestra el fondo correctamente en el final de las partidas en línea.
 1226 + Chat en partida en línea permite enviar varios mensajes presionando la tecla
 ↪ de selección.
 1227 + Chat en partida en línea notifica cuando hay mensajes no leídos.
 1228 + El enviar en la sala en línea aparece centrado.
 1229 + No se muestran los controles de velocidad a los clientes.
 1230 + Radio botones, se explican mejor viéndolos.
 1231 + En el menú en línea aparece ya inicialmente una opción seleccionada.
 1232 + Se ha eliminado la opción de "crear partida desde guardado".
 1233 - Arreglo:
 1234 + Soberano no suma dos veces la población y quita una.
 1235 + Se actualiza en ejércitos externos si hay muertes externas en partida en
 ↪ línea.
 1236 + Todos los jugadores pasan al fin de partida tras una partida en línea.
 1237 + El juego ya no se cierra por muchos, muchos motivos que no voy a listar aquí.
 1238 + El escenario ya no marca agujeros nuevos siendo cliente en el minimapa en
 ↪ partida en línea.

Versiones 2.3.X

1239 v2.3.0.
 1240 - Añadido:
 1241 + Nuevo modo en línea. No se pasa el estado del escenario, se simula con los
 ↪ cambios suponiendo máquinas perfectas en todos los casos y con la misma
 ↪ semilla para la generación de valores aleatorios.
 1242 + Jugador artificial en remoto.
 1243 + Mecánica de error adaptable (los errores se auto-ajustan para caber en las
 ↪ ventanas).
 1244 + Pegar en campos modo en línea.
 1245 + Sonidos en partida de clientes en línea.
 1246 + El cursor se actualiza en partida de clientes en línea.
 1247 + Partidas en línea con velocidad ajustable: modo turbo incluido.
 1248 - Mejora:
 1249 + Menús modo en línea (muchas cosas como para listarlas).
 1250 + Sombreado de nombre en partida en línea para verlo mejor.
 1251 + Núcleo de las entidades.
 1252 + El panel del menú multijugador en línea está ahora ajustado.
 1253 - Arreglo:
 1254 + Vuelven los sonidos dentro de partida.
 1255 + No explota con diversas combinaciones de teclas en el juego sin tener nada
 ↪ seleccionado en partida.
 1256
 1257 v2.3.1.
 1258 - Añadido:
 1259 + Sonidos de acierto, error y flotado en el editor.

1260 + Salir una vez se pierde la partida individual como humano: antes te quedabas
 ↪ en partida hasta que apareciera un ganador, con el cambio se ofrece la
 ↪ posibilidad de acabar la partida si has perdido y pasar directamente a las
 ↪ estadísticas.

1261 + Sonido al salir de partida.

1262 + Los botones en el editor son reactivos visualmente.

1263 - Mejora:

1264 + Código de lectura de movimientos de ratón, afecta poco al rendimiento pero
 ↪ ayuda mucho en la depuración.

1265 + Ventana emergente de salida dentro del juego.

1266 + Se han desactivado los mensajes de depuración sobre la recepción de mensajes,
 ↪ menos retardo.

1267 + La textura de la tabla de las facciones y ejércitos del editor es más acertada
 ↪ (tiene la fila de los colores en gris en vez de en blanco ya que no se
 ↪ pueden activar esas casillas).

1268 - Arreglo:

1269 + No se cierra el juego a veces al fallar a conectar.

1270 + Las refineras actualizan correctamente su textura al cargar partida o unirse a
 ↪ partida multijugador.

1271 + No peta al cargar directamente una partida: se ha cambiado el formato de
 ↪ guardado y cargado de partida así que guardados anteriores a esta versión no
 ↪ tienen que funcionar, lo siento.

1272 + Se actualiza correctamente dentro de las refineras cuánto queda de extracción,
 ↪ lo cual permite que se vea el progreso en la barra de extrayendo.

1273 + Los extractores muestran el panel emergente correcto en el dibujo.

1274 + La maravilla se construye correctamente (maravillosamente).

1275 + Paneles de estadísticas funcionando desde hace una versión o dos.

1276 + Se vuelve a poder jugar un ejército en partida individual, si seleccionabas un
 ↪ ejército humano se activaban todos como humanos y solo te dejaba jugar el
 ↪ último de ellos.

1277 + No se cierra el juego al destruir una refinera en construcción.

1278 + Se termina correctamente una partida en local.

1279 + Se termina correctamente una partida sin ganador.

1280 + No se cierra el juego a veces al recolectar con un trabajador.

1281 + La animación de fin de partida funciona correctamente en todos los casos.

1282 + Las facciones se muestran correctamente en la selección de escenario del
 ↪ editor, del modo individual y del modo multijugador en línea.

1283 + Thermopylae no cierra el juego al entrar.

1284 + Las piedras vuelven a cargar bien las texturas.

1285 + HelmAbyss no cierra el juego al entrar.

1286 + Los trabajadores ya no construyen mirando donde no deben.

1287 + Chispitas al construir de los trabajadores Terran.

1288 + Cargan los nombres y los iconos (dibujos) de las entidades al cargar partida o
 ↪ al entrar en una partida multijugador en línea.

1289 + En la selección de escenario para el editor, o de juego para el modo en línea
 ↪ o individual se muestran correctamente las zonas de conquista o los tiempos
 ↪ de supervivencia.

1290 + No se quedan cursores activos de un acceso a otro en el editor.

1291 + La edición de un campo en el editor se desactiva si se abandona su zona de
↳ edición.

1292 + Renombrar un escenario a vacío y pulsar la tecla de salir también restablece
↳ el nombre.

1293

1294 v2.3.2.

1295 - Añadido:

1296 + Jugador artificial en remoto (no solo en menús, ahora se puede utilizar).

1297 + Mecánica de rechazo directa a los jugadores con el mismo nombre en la sala en
↳ línea.

1298 + Se refleja el estado de confirmando configuración al estar reuniendo votos en
↳ sala multijugador en línea.

1299 + Se cancela la aprobación de una configuración si se echa a un jugador en la
↳ sala en línea.

1300 + Desactivada la opción de abandonar la partida en línea si se pierde siendo el
↳ servidor.

1301 + La ventana del juego se inicializa con el tamaño mínimo y luego se ajusta al
↳ requerido.

1302 - Mejora:

1303 + El cierre de las conexiones es más robusto, evitando así algunos cierres
↳ inesperados del servidor.

1304 - Arreglo:

1305 + La selección de un control para un ejército en una sala en línea ya no es
↳ intermitente al mover el ratón.

1306 + Se ilumina el botón de selección al pulsar la tecla de salir en la sala en
↳ línea en el estado de configuración.

1307 + Aparecen los nombres de los jugadores ya coloreados correctamente coloreados a
↳ los nuevos jugadores que se unen a una sala en línea.

1308 + No se recolorea el botón de seleccionar en sala en línea en estado de
↳ selección si no eres servidor y sacas el ratón de dicho botón o lo colocas
↳ en dicho botón.

1309 + No se recolorea el botón de jugar en sala multijugador en línea en estado de
↳ selección si no eres el servidor y colocas el ratón en dicho botón.

1310 + Los escenarios seleccionados en una sala en línea que estén en una página que
↳ no sea la primera ya se tratan correctamente en el envío y se muestran bien
↳ a los clientes.

1311 + Los escenarios seleccionados en una sala en línea que estén en una página que
↳ no sea la primera no hacen que el juego reviente dentro de partida.

1312 + No se cierra el juego en según qué pestañas de según qué escenarios tanto en
↳ sala multijugador en línea como en selección de escenario de modo
↳ individual.

1313 + Se muestran correctamente las filas y las columnas del escenario a los
↳ clientes de la sala multijugador en línea.

1314 + Adiós ventana emergente de conectando: ahora aparece un error si no se
↳ consigue conectar a la IP y puerto introducidos.

1315

1316 v2.3.3.

1317 - Añadido:

1318 + Funcionalidad a los campos de edición de recursos iniciales en el editor.
1319 + Lapiceros en editables del editor.
1320 + Controles de movimiento de cámara disociados de los de mover selección,
↳ así que más teclas.
1321 + 6 conceptos del editor.
1322 + Jugador híbrido en partida multijugador en línea.
1323 + Jugador híbrido en modo individual.
1324 + Ajustes en vivo de jugador artificial / híbrido de partida multijugador en
↳ línea.
1325 + Mensajes del sistema al ser derrotado un ejército en partida multijugador en
↳ línea.
1326 + Se muestra el estado del jugador artificial al no seleccionar nada al jugar un
↳ jugador artificial en partida multijugador en línea.
1327 + Documento sobre los jugadores artificiales adjunto.
1328 + Reactivada la visión en todos los modos y configuraciones. Estaba quitada para
↳ introducir todo este contenido.
1329 + Cámara automática solo muestra lo visible en el modo híbrido.
1330 + Los trabajadores de los jugadores con inteligencia artificial van a explorar a
↳ zonas no visibles para estos cuando no tienen más recursos que explotar.
1331 + Las larvas y soberanos de los jugadores con inteligencia artificial se van
↳ recolocando hasta no estorbar (no tener ni edificios ni unidades pegadas).
1332 + Añadida Meta inteligencia artificial: se forma grupos de invasiones en función
↳ de amenazas, distancia... etc.

1333 - Mejora:
1334 + Editor más reactivo.
1335 + Los indicadores de velocidad son más estándar.
1336 + La cámara automática ahora apunta sólo a tu propio ejército al jugar como
↳ jugador híbrido.
1337 + La cámara automática ahora enfoca mejor a los edificios.
1338 + El rectángulo del ejército actual en el editor es más estándar.
1339 + Botones que antes eran demasiado oscuros en el editor ya no lo son.
1340 + Se muestran las hitbox de los botones de la ventana emergente de salida del
↳ editor.
1341 + Se indica qué elemento de las opciones del editor se está seleccionando y cual
↳ está activo.
1342 + Marca error si intentas reelegir un concepto en el menú de información
↳ del editor.
1343 + Se selecciona automáticamente al entrar el primer concepto en el menú de
↳ información del editor.
1344 + Los jugadores con inteligencia artificial construye mas expansivamente.
1345 + Escenario StarCraft.

1346 - Arreglo:
1347 + No se marca la velocidad como x-1 a veces en partida.
1348 + Algunos botones en el editor se iluminan correctamente y no con otras zonas
↳ distintas.
1349 + No se generan texturas que luego no se limpiaban en el editor al crear un
↳ escenario.
1350 + Los trabajadores vuelven a atacar.

1351 + El menú de selección de escenario individual responde correctamente a todo y
↳ permite ajustar las dos configuraciones de los jugadores con inteligencia
↳ artificial.

1352 + Si se entra en partida individual se activa correctamente el truco de la
↳ niebla en modo simulación.

1353 + Se muestra correctamente la hitbox de los botones de la ventana emergente de
↳ salir de partida.

1354 + Botón de siguiente raza en el editor se ilumina cuando debe.

1355 + Hay cruces en el editor que ahora se iluminan cuando pones el cursor encima.

1356 + La selección de escenario en modo individual no tiene fallos con las partidas
↳ con un humano configurado de base.

1357 + Las refineras dejan de matar a todos los trabajadores que entran en ellas.

1358 + Las refineras muestran bien su círculo de selección y su información en
↳ partida.

1359 + Las refineras no se quedan inmortalmente vivas.

1360

1361 v2.3.4.

1362 - Añadido:

1363 + Modo huida: si una unidad está combatiendo cuando recibe la orden de huir
↳ dicho modo se activa y hasta que no reciba otra orden o alcance su destino
↳ no dejará de intentar huir ignorando todos los ataques que esta reciba.

1364 + Los jugadores con inteligencia artificial aprenden a retirarse: si una unidad
↳ invasora tiene poca vida intentará volver a la base.

1365 + Los jugadores con inteligencia artificial aprende a patrullar: las unidades no
↳ invasoras darán vueltas por la zona.

1366 + Los jugadores con inteligencia artificial mejoran su percepción de derrotas:
↳ si una unidad muere evita el doble la zona que si por ejemplo una unidad se
↳ retira, además de tener en cuenta también las tropas que ha enviado allí.

1367 + Barras negras cubren lo no cubierto por la ventana por el ratio del juego.

1368 + Pantalla completa de nuevo, y su control (F11 por defecto).

1369 + Se quita la terminal y se introducen ventanas de Windows de error.

1370 + No se sale del menú inicial (a no ser que cierres el juego o pulses la tecla
↳ de salir): se sale desde el menú principal y en el menú inicial solo hay un
↳ botón de entrar al menú principal.

1371 + 4 conceptos de información del juego.

1372 + La selección se mueve con la cámara en el juego.

1373 - Mejora:

1374 + Los jugadores con inteligencia artificial reescalan sus acciones ponderando el
↳ doble de probables invadir, trabajar y defender (haciendo que ocurran con el
↳ doble de frecuencia).

1375 + La agresividad evoluciona con todas las acciones.

1376 + El comportamiento de los jugadores con inteligencia artificial define cómo
↳ evoluciona la agresividad.

1377 + Si la agresividad alcanza un mínimo, los jugadores con inteligencia artificial
↳ retiran invasiones. Conservando la retirada individual.

1378 + Escenario StarCraft.

1379 + La inteligencia artificial genera muchos más trabajadores.

1380 + Aumentado el límite de reducir zum en partida x1:2 -> x1:13.

1381 + Los jugadores con inteligencia artificial empiezan a mandar exploradores a lo
↳ que hay 3 trabajadores minando.

1382 + El movimiento del minimapa con el ratón ahora es más suave al trabajar por la
↳ posición y no por casillas.

1383 + Si un jugador con inteligencia artificial vence un cuadrante (no hay más
↳ enemigos en el) se vacía su registro de bajas de ese cuadrante.

1384 + El texto de "Jugador humano como jugador artificial (IA)" es ahora "Jugador
↳ híbrido (Humano + IA)".

1385 + "Trucos" se ve mejor y se distingue mejor.

1386 + El cursor del ratón sigue al ratón real (no se desplaza por las barras
↳ negras).

1387 - Arreglo:

1388 + El texto del truco "espiar a la ia" es "espiar rivales".

1389 + No se mutea el juego mientras se escribe en el chat en sala en línea porque
↳ escribes una M.

1390 + No se cierra el juego al hacer clic sobre vacío en una simulación.

1391 + No se cierra el juego al jugar un jugador con inteligencia artificial actual
↳ en mapas no regulares (misma altura y anchura).

1392 + La refinera Terran no aparece como "sin construir" mientras se
↳ está extrayendo.

1393 + La refinera Terran no se queda bug en extrayendo.

1394 + Los jugadores con inteligencia artificial remanejan correctamente las
↳ invasiones existentes.

1395 + Las nuevas capacidades de los jugadores con inteligencia artificial no
↳ provocan que se cierre el juego al intentar analizar escenarios sin sectores
↳ múltiples de 16.

1396 + No se cierra el juego con escenarios como HelmAbyss porque no carga bien los
↳ jugadores ni en color ni forma.

1397 + El jugador con inteligencia artificial en remoto funciona.

1398 + No se crean indefinidamente refineras por parte de un jugador con inteligencia
↳ artificial.

1399 + Los jugadores con inteligencia artificial provocan menos cierres del juego.

1400

1401 v2.3.5.

1402 - Añadido:

1403 + 5 nuevos conceptos del editor.

1404 + 6 nuevos conceptos del juego.

1405 + Pulsar el control de silenciar sonido o el de pantalla completa en el menú de
↳ ajustes activa el botón de guardado.

1406 + Botones de '+' y '-'.

1407 + Ajuste de pantalla completa también en ajustes de vídeo.

1408 + Radio botones también en el editor.

1409 + Un jugador con inteligencia artificial va a explorar con soldados si no tiene
↳ sectores de peligro.

1410 + Jugador artificial: variante conquistadora.

1411 * Prioriza sectores más cercanos a la zona a conquistar.

1412 * Si está invadiendo el sector de la zona a conquistar y lo logra va
↳ colocando soldados en la zona de conquista.

1413 + Jugador artificial: variante superviviente:

1414 * La defensa no permite que los patrulladores se vayan muy lejos de los
 ↪ edificios.

1415 + Jugador artificial: variante investigadora de nuevo, con nueva defensa y nueva
 ↪ invasión.

1416 - Mejora:

1417 + Un jugador con inteligencia artificial defiende mejor.

1418 + HelmAbyss ahora es un monstruoso escenario de 5 ejércitos.

1419 + Los nombres de las facciones están centrados en el desplegable del editor.

1420 + Mejor colocado el mensaje de "Jugador híbrido".

1421 + Ventana emergente de configuración de facción en el editor.

1422 + Un jugador con inteligencia artificial por lo general invade más y tiene más
 ↪ en cuenta el comportamiento configurado.

1423 + Menú multijugador en línea.

1424 + El menú de ajustes de sonido.

1425 + Ahora solo se permite en el campo de contraseña de salas multijugador en línea
 ↪ letras, números, puntos, comas, espacios, interrogante de cierre y
 ↪ exclamación de cierre.

1426 + Los criaderos Zerg tienen más control por criadero, para darles más libertad y
 ↪ balancear así la raza: 2 -> 4.

1427 - Arreglo:

1428 + Se ven bien los botones de los conceptos del editor.

1429 + Edición de recursos iniciales en editor funciona.

1430 + Cursor de edición de ejércitos en editor se marca.

1431 + Se ilumina la ruleta de configuración de facción en el editor.

1432 + Lápiz en edición de facción se ilumina.

1433 + No aparece cursor de edición al editar facción en el nombre del ejército.

1434 + Edición de facción en el editor ahora es reactivo y no está todo en verde.

1435 + Las facciones se colorean correctamente según su objetivo en el editor.

1436 + Al volver de configuración a selección en selección individual de escenario se
 ↪ oculta correctamente la vista de control.

1437 + La selección de comportamientos de los jugadores artificiales va bien en el
 ↪ editor.

1438 + En la selección de escenarios se reflejan bien las partidas con ejércitos
 ↪ deshabilitados.

1439 + "Espiar rivales" se muestra correctamente en modo individual.

1440 + Los retratos no se ven en blanco en según qué zonas.

1441 + Salir del juego en multijugador no cierra el juego en algunos casos.

1442 + No ocasiona el cierre del juego un jugador con inteligencia artificial por
 ↪ salirse de los sectores posibles.

1443 + Modificar los recursos iniciales afecta al botón de guardado en el editor.

1444 + Los segundos y los minutos de una facción de supervivencia no se quedan en
 ↪ blanco en el editor.

1445 + Escala bien el nombre de la facción en el editor de escenarios.

1446 + Va bien la tabla de facciones en partida en individual.

1447 + Cámara del minimapa en cliente multijugador en línea.

1448 + Visibilidad del minimapa en cliente multijugador en línea.

1449 + La cámara en cliente multijugador en línea empieza donde empieza su ejército.

1450 + No se queda la cruz en verde en algún ajuste de sonido.
1451 + No aparece un rectángulo verde en las partidas al comenzar.
1452 + No se selecciona lo que no esté visible.
1453 + No sale construcción del cliente al editor.
1454 + Soberano y Hidralisco tienen tiempo de aparición.
1455 + Los muros no sirven ya para dejar el mineral.
1456
1457 v2.3.6.
1458 - Añadido:
1459 + Carga individual de partidas con configuración: se puede activar trucos,
↪ modificar el comportamiento y la velocidad de los jugadores con inteligencia
↪ artificial, y el propio control del jugador.
1460 + Se muestra un signo de infinito al activar el truco de los recursos.
1461 + Un jugador con inteligencia artificial en conquista intenta conquistar la zona
↪ incluso si no hay enemigos.
1462 + Dibujo de las defensas.
1463 + Dibujo de la zona de conquista.
1464 - Mejora:
1465 + Un jugador artificial mide mejor su agresividad en función del tamaño de las
↪ tropas en invasión respecto del de las que hay en defensa.
1466 + Ahora se guardan las estadísticas de los jugadores con inteligencia artificial
↪ y básicamente todos los datos en un guardado.
1467 + Menú de carga de escenarios.
1468 + Menú de selección de escenarios individual.
1469 + Colores de ejércitos más claros.
1470 + Escenario HelmAbyss.
1471 + Escenario Thermopylae.
1472 + Ahora se puede editar el nombre de un ejército más caracteres hasta el espacio
↪ gráfico de dicho campo.
1473 + Ahora se permiten espacios en los nombres de los ejércitos.
1474 + Un jugador con inteligencia artificial va un 33% más deprisa, tiempo entre
↪ acciones base: [1, 2] seg -> [0.5, 1.5] seg (esto se multiplica en función
↪ lo baja que es la velocidad hasta un máximo de [2, 6] seg).
1475 + La música no se repite en el menú de fin de partida.
1476 + Añadida información del ejército neutral en el formato de escenarios.
1477 - Arreglo:
1478 + Ahora se activa solo el truco sin niebla en simulaciones y se puede
↪ seleccionar todo correctamente y se refleja en el cursor.
1479 + El truco de recursos infinitos ahora sí actúa como tal en vez de elevarlos a
↪ una alta cantidad.
1480 + Aparece el título de la ventana emergente de salida del juego al pulsar la
↪ tecla de salir y no solo al hacer clic en el botón de salir.
1481 + Un jugador con inteligencia artificial calcula correctamente los sectores en
↪ todos los casos.
1482 + El servidor de las partidas multijugador en línea tiene bien la cámara del
↪ minimapa, ya no solo el cliente.
1483 + El cliente ve bien la lista de ejércitos en la sala multijugador en línea en
↪ el caso de ejércitos deshabilitados.

1484 + No se rompe el juego al borrar el primer archivo de guardado en la carga
↪ individual en según qué circunstancias.

1485 + No aparece el campo "Enfriamiento" de un jugador híbrido en otras partes del
↪ juego.

1486 + Marca bien los botones de cambio de página de los modos de edición en el
↪ editor.

1487 + La selección de control de teclado no se vuelve intermitente al desplazar el
↪ cursor sobre el mismo elemento.

1488 + Deja de poderse activar o desactivar la cuadrícula dentro de partida.

1489 + Los Zerg del jugador artificial construyen correctamente las estructuras
↪ avanzadas, ya que al solo costar esta estructura avanzada gas se
↪ imposibilitaba a sí misma construir una refinera primero por el orden de
↪ prioridades de construcción.

1490 + Las opciones de modo en el editor se seleccionan bien a partir de la segunda
↪ página

1491 + No se cierra el juego con ciertas destrucciones del ejército neutral.

1492 + No se cierra el juego al intentar jugar un escenario seleccionado pero no
↪ configurado en individual.

Anexos M

Manual de usuario



Figura M.1: Título de Armygeddon.

“Guía definitiva para ‘Armigedón’, que suena a dinosaurio, pero así lo pronuncia todo el mundo.”

M.1. Resumen

Armygeddon es un videojuego de estrategia en tiempo real basado en el universo de StarCraft (1998) donde tres grandes razas espaciales luchan por el control de los distintos planetas (ver figura M.2).



Figura M.2: Una simulación en Armygeddon.

Se permiten hasta ocho jugadores en cada partida, las cuales pueden ser en local o en línea. Pueden ser partidas entre distintos usuarios, contra la máquina, o simulaciones donde la máquina se enfrenta a sí misma.

Las partidas consisten en la lucha de varias facciones de ejércitos donde cada facción combate por sus propios objetivos. Existen cuatro tipos de objetivos para cada facción que determinan el modo de juego de los participantes: Exterminio, Supervivencia, Conquista e Investigación.

Un jugador tiene que comandar uno de los ejércitos y alcanzar sus metas haciendo equipo con los demás jugadores de su facción para resultar victoriosos.

Además, Armygeddon incluye un *editor de escenarios* que permite la creación de niveles completamente personalizados que después poder jugar.

M.2. Requisitos

“Lo sentimos, solo se puede salvar la galaxia desde un Windows, las versiones para Linux y Mac se las comieron los zerg.”

Para poder jugar Armygeddon se requiere un sistema Windows de 64 bits.

Puede ser necesario desactivar el antivirus para ejecutar el programa.

Puede solicitarse el acceso por el cortafuegos al usar las características en red.

Solo se permite el uso de Armygeddon con fines académicos.

Requisitos mínimos (son los mínimos en los que se ha probado Armygeddon, pero los requisitos son mucho menores):

- Procesador: AMD Ryzen 5 4500U (6 núcleos, 2,38 GHz cada núcleo).
- Gráfica: Radeon HD 6700.
- RAM: 8 GB.
- Pantalla: se ha utilizado una pantalla con resolución 1920x1080 (el mínimo requerido por el programa es de 640x480).

M.3. Navegación

La navegación de Armygeddon tiene las siguientes 17 pantallas de interfaz de usuario (ver figura M.3), cada una asociada a un propósito:

- Pantalla inicial. Es la pantalla en la que se abre el juego. Aquí se cargan los recursos del juego y permite ir al *Menú principal*.
- Menú principal. Permite escoger entre ir a la *Elección de juego*, a los *Ajustes*, o al *Menú del editor de escenarios*. También permite salir del juego.
- Ajustes. En este menú se pueden configurar algunos aspectos básicos del juego.
- Elección de juego. Aquí se elige entre juego en línea o el juego local (individual). Además está el acceso a la *Información del juego*.
- Información del juego. En esta pantalla se muestran los conceptos fundamentales sobre las reglas del juego.
- Menú de juego multijugador. Permite configurar el perfil en línea del usuario para



crear o unirse a una sala multijugador.

- Sala multijugador. Aquí se preparan los usuarios y se configura un escenario para jugar una partida en línea.
- Pantalla de juego. En esta pantalla se desarrolla una partida.
- Fin de partida. Muestra los resultados y algunas estadísticas de una partida.
- Menú de juego individual. Aquí se escoge entre crear o cargar una partida individual.
- Creación de partidas. Permite visualizar la lista de escenarios e información sobre cada uno, escoger un escenario y, configurar algunos aspectos de este para comenzar una partida individual.
- Carga de partidas. En esta pantalla se muestra la lista de partidas guardadas, las cuales se pueden borrar o configurar para ser jugadas.
- Menú del editor de escenarios. Aquí se elige entre acceder a la *Información del editor*, a la *Creación de escenarios* o a la *Carga de escenarios*.
- Carga de escenarios. Permite consultar la lista de escenarios disponibles, ver su información y escoger uno para editarlo. Además se permite renombrar, duplicar o eliminar escenarios.

M.4. Controles

“Me negué en rotundo a implementar los 252 controles en total que tenía StarCraft, que a su vez podían tener combinaciones entre sí, así que ‘solo’ hay 34 controles.”¹

La lista de controles de Armygeddon es la siguiente:

- Pulsar el elemento seleccionado. Por defecto la tecla *Return* (Intro). Intenta accionar el elemento activable seleccionado, esté bloqueado o no. También se acciona con el clic izquierdo del ratón. Durante una partida o la edición de un escenario solo está activo este control mediante el ratón.
- Dar una orden. Solo se activa con el clic derecho del ratón. Sirve para comandar las unidades durante una partida.
- Ajustar el zum. Solo se puede alejar o acercar el zum con la ruleta del ratón. Sirve para ampliar o reducir el campo de visión en partida o en la edición de un

¹<https://starcraft.fandom.com/wiki/Hotkey> (último acceso: 17/5/2023).

escenario.

- Mover la selección hacia una dirección (cuatro controles). Por defecto las teclas de dirección *Left*, *Up*, *Right* y *Down* (Flecha izquierda, Flecha arriba, Flecha derecha y Flecha abajo respectivamente). Si no hay ningún elemento seleccionado el pulsar una de estas teclas selecciona uno, y de haber uno seleccionado, desplaza la selección al elemento más cercano en la dirección deseada si lo hay. Estos controles no tienen efecto en el juego ni en el editor.
- Mover la cámara hacia una dirección (cuatro controles). Por defecto las teclas ‘A’ (izquierda), ‘W’ (arriba), ‘D’ (derecha) y ‘S’ (abajo). Permite desplazar la cámara tanto en el editor como en partida siempre que haya espacio en la dirección en la que se quiere desplazar. Estos controles se pueden mantener para continuar moviendo la cámara de forma continua. Permite movimientos combinados siempre que no sean opuestos.
- Activar o desactivar cuadrícula. Por defecto la tecla ‘G’. Activa o desactiva la visión de cuadrícula dentro del editor.
- Salir. Por defecto la tecla ‘Escape’. Al activarlo intenta salir de la pantalla actual, dirigiéndose siempre hacia la pantalla principal donde si se presiona este control de nuevo sale del juego. En algunas circunstancias este control deselectiona (al estar configurando un escenario justo antes de entrar a una partida o al tener selección en partida por ejemplo).
- Botones del panel de la derecha (nueve controles). Se refiere al panel de acciones de la derecha presente tanto en el editor como dentro de partida. Por defecto está vinculado a las teclas de los números del uno al nueve (afectan a los botones del ‘1’ al ‘9’ respectivamente ver figura M.4).
- Activar o desactivar sonido global del juego. Por defecto la tecla ‘M’. Activa o desactiva todo el sonido (música y efectos) según se pulse, haciendo que el juego produzca sonido o no independientemente de los niveles de volumen del juego (ver ajustes de sonido en el anexo M.5.2).
- Mostrar la tabla de facciones durante partida. Por defecto la tecla ‘TAB’(tabulador). Mientras este control esté presionado dentro de partida se muestra la tabla de facciones en medio de la pantalla.
- Seleccionar ejército anterior. Por defecto la tecla ‘Q’. Salvo si se tiene seleccionado el primer ejército (el ejército rojo) en el editor de escenarios este control selecciona el ejército anterior al actualmente seleccionado.

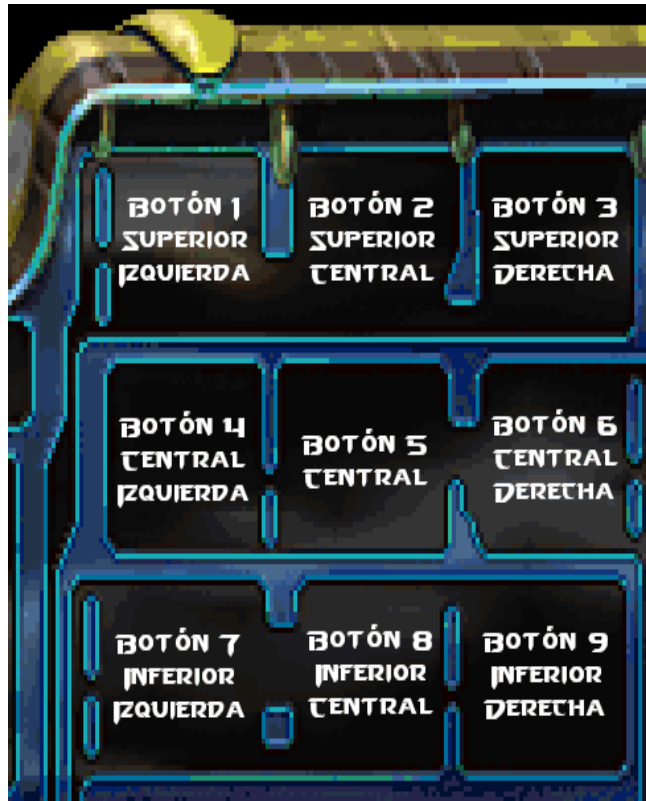


Figura M.4: Panel derecho de acciones protoss señalado.

- Seleccionar ejército posterior. Por defecto la tecla ‘E’. Salvo si se tiene seleccionado el último ejército (el ejército neutral, de color gris) en el editor de escenarios este control selecciona el ejército anterior al actualmente seleccionado.
- Activar o desactivar modo edición de recursos. Por defecto la tecla ‘R’. En el editor de escenarios este control desactiva o activa el modo de edición de recursos en función de si el modo de edición actual es el de recursos o no.
- Activar o desactivar modo edición de baldosas. Por defecto la tecla ‘T’. En el editor de escenarios este control desactiva o activa el modo de edición de baldosas en función de si el modo de edición actual es el de baldosas o no.
- Activar o desactivar modo edición de unidades. Por defecto la tecla ‘U’. En el editor de escenarios este control desactiva o activa el modo de edición de unidades en función de si el modo de edición actual es el de unidades o no.
- Activar o desactivar modo edición de estructuras. Por defecto la tecla ‘I’. En el editor de escenarios este control desactiva o activa el modo de edición de estructuras en función de si el modo de edición actual es el de estructuras o no.
- Activar o desactivar modo edición de obstáculos. Por defecto la tecla ‘O’. En el editor de escenarios este control desactiva o activa el modo de edición de

obstáculos en función de si el modo de edición actual es el de obstáculos o no.

- Activar o desactivar modo edición de objetivos. Por defecto la tecla ‘Y’. En el editor de escenarios este control desactiva o activa el modo de edición de objetivos en función de si el modo de edición actual es el de objetivos o no.
- Activar o desactivar el modo pausa. Por defecto la tecla *Space* (barra espaciadora). En partida de estar el juego pausado cuando se usa este control se cambia la velocidad a ‘x1’, en caso contrario el juego se pausa.
- Activar o desactivar el modo pantalla completa. Por defecto la tecla ‘F11’. Intercambia el estado de la ventana del juego de normal a pantalla completa y viceversa.

M.5. Ajustes

“Seguro que nadie ha notado hasta la letra es algo más oscura en la pantalla de ajustes para resaltar más que no es un menú de juego y por lo tanto no es divertido.”

En la pantalla de ajustes se permite configurar algunos aspectos del juego, estando estos agrupados en 4 categorías distintas: ‘Teclado’, ‘Sonido’, ‘Vídeo’ y ‘Extra’. Además, hay una opción para restablecer los ajustes a por defecto y otra para guardar los ajustes en el fichero de configuración para que se carguen automáticamente la próxima vez que se ejecute el juego.

El fichero de configuración² está en la ruta `.\config.txt` dentro del directorio del juego. Los ajustes especificados en este fichero se intentan cargar al ejecutar el juego si es que los hay. El contenido del fichero se puede modificar libremente aun sin estar el juego en ejecución y viene acompañado de comentarios para hacer su edición más sencilla.

M.5.1. Teclado

“Dedicado en especial a los “odio los juegos con WASD” o los “odio los juegos con las flechas”.”

En la categoría de ajustes *Teclado* (ver figura M.5) están listados todos los controles asignados a las órdenes que se pueden accionar por teclado.

Para *cambiar un control* hay activar el botón de tecla asociado a este. Al hacerlo, el botón se pondrá en rojo y desaparecerá la tecla que estuviera asociada antes. En este momento si se pulsa una tecla que no esté en uso ya en el resto de la lista de ajustes de

²Ver formato de ficheros en el anexo F.



Figura M.5: Lista de controles en ajustes de teclado.

teclado, el control pasará a estar ligado a dicha tecla. De introducir una tecla que ya esté en uso, el control volverá a estar asociado a la tecla que hubiera antes del cambio.

M.5.2. Sonido

“Cuando te atacan 80 enemigos, y los 80 hacen su desagradable sonido cada poco, estos ajustes vienen bien.”

El sonido del juego se divide en dos categorías (ver figura M.6): música y efectos. La música suena constantemente, se repite cuando se acaba (excepto en el fin de partida) y cambia según cambian las pantallas. Los efectos son todos los sonidos generados por el juego (sonidos de las unidades, botones, selección, etc.), breves, que suenan una vez y que responden a un evento que no sea cambio de pantalla.

El ajuste de “Silenciar todo”, activable también por teclado³, desactiva todo el sonido generado por el juego, tanto la música como los efectos.

Los ajustes de “Volumen de la música” y “Volumen de los efectos” permiten ajustar las barras de ocho niveles que representan el volumen de cada una de las dos categorías de sonido. Al estar uno de los dos volúmenes al mínimo (completamente desactivadas) se desactiva la correspondiente categoría de sonidos. Estos niveles de volumen valen todos lo mismo, un 12,5 % del volumen total posible, estando el volumen al 100 % al

³El ajuste de “Activar o desactivar sonido global del juego”. Ver ajustes de teclado en anexo M.5.1.



Figura M.6: Ajustes de sonido.

estar los ocho niveles de volumen activos.

M.5.3. Vídeo

“Debería permitir más ajustes gráficos, como por ejemplo quitar o poner las sombras, pero perdería la gracia de juego de los 90.”

En esta categoría de los ajustes se permite cambiar todas las cuestiones gráficas del juego, que son cuatro en total (ver figura M.7):

- *Pantalla completa:* Al activarse cambia el tipo de la ventana del juego de normal a pantalla completa y viceversa. Pantalla completa en este caso supone que el juego deja de estar rodeado por la interfaz de ventanas de Windows y ocupa toda la pantalla pero respetando el ratio del juego de 4:3, por lo que pueden aparecer aun así las dos “barras negras” de relleno para tapar lo que no ocupe el juego. y viceversa.
- *Resolución:* Permite cambiar la resolución actual del juego dentro de unas posibilidades. Estas van desde la resolución mínima 640x480 hasta la máxima soportada por la pantalla que cumpla con el ratio 4:3 dentro de las resoluciones más utilizadas. Aunque siempre se puede ajustar con más precisión modificando el tamaño de la ventana a través de la interfaz de Windows. Si se modifica con la interfaz de Windows, los cambios no se reflejarán en la resolución aquí indicada.



Figura M.7: Ajustes de vídeo.

- *Velocidad de transiciones*: Permite configurar la velocidad de las animaciones de transición entre pantallas⁴, siendo ‘x1’ la velocidad normal, ‘x2’ el doble de rápido, y así sucesivamente. Si el ajuste está a ‘x0’, se desactivan las transiciones entre pantallas.
- *Postergación de las ventanas emergentes*: Aquí se configura cuánto tardan en aparecer los paneles emergentes que genera el cursor en determinadas zonas para otorgar más información.

M.5.4. Extra

“En algunas entregas de la saga de videojuegos de “Silent Hill” en los ajustes extra podías cambiar el color de la sangre de los enemigos. Aquí puedes cambiar algo igual de desagradable que es que la cámara se mueva con el ratón.”⁵

En la categoría extra de ajustes se pueden personalizar ciertos aspectos del juego secundarios (ver figura M.8).

⁴Solo en las pantallas que las tienen, por ejemplo, la pantalla de elección multijugador.

⁵Se permitía en Silent Hill (1999): https://silenthill.fandom.com/wiki/Silent_Hill_Secrets_and_Unlockables#Extra_options_menu (último acceso: 17/5/2023).

Se permitía en Silent Hill 3 (2003): <https://www.ign.com/wikis/silent-hill-3/Basics> (último acceso: 17/5/2023).

Se permitía en Silent Hill 4 (2004): https://www.silenthillmemories.net/sh4/versions/sh4_manual_pc_eu.pdf (último acceso: 17/5/2023).



Figura M.8: Ajustes extras.

Si la opción *El cursor desplaza la cámara* está activada, al acercar el cursor a los bordes del espacio no ocupado por interfaces dentro de partida o del editor, la cámara se desplazará en esa dirección respecto del centro de la pantalla⁶.

El *Retrato para las partidas en línea* permite seleccionar qué retrato se va a llevar durante las partidas multijugador en línea, tanto en la sala multijugador como dentro de la propia partida.

El *Nombre para las partidas en línea* es el nombre con el que identificarse dentro de una partida multijugador en línea. Este ajuste permite letras, números y espacios siempre que no quede vacío o empiece por espacio.

M.6. Reglas del juego

“Zerg gana a terran, terran gana a protoss y protoss gana a zerg. El problema es que también ocurre al revés.”

En esta sección se explican una serie de conceptos sobre el funcionamiento del juego. Existe igualmente dentro del juego una lista de paneles con información básica sobre las reglas del juego en el menú de información de juego.

⁶Solo permite desplazamientos en las direcciones cardinales o en las intercardinales, al igual que con los controles de teclado.

M.6.1. Partida de Armygeddon

“Algo de ‘StarCraft’, un poco de ‘Total War’, un poquito de ‘World of Warcraft’, una pizca de ‘League of Legends’, un atisbo de ‘Divinity 2’ y, mucha, mucha imaginación.”

Una *partida* de Armygeddon consiste en la lucha entre una o más facciones de ejércitos por conseguir sus objetivos. La partida termina cuando todas las facciones han sido derrotadas o cuando una facción resulta victoriosa al alcanzar sus objetivos. En una partida puede haber hasta ocho ejércitos. Las partidas se crean con escenarios, los cuales contienen toda la configuración sobre el mapa, los ejércitos y las facciones (ver más información de la creación de escenarios en el anexo M.10).

M.6.2. Facciones

Una *facción* es un grupo de hasta ocho ejércitos con unos objetivos comunes.

Existen cuatro tipos de objetivos de facción: Exterminar, Sobrevivir, Conquistar e Investigar. En función de su tipo de objetivo, el nombre de la facción será de un color o de otro como se ve en la figura M.9.



Figura M.9: Variación gráfica de las facciones en función de su tipo de objetivo.

Una facción es derrotada cuando todos sus ejércitos son destruidos o cuando una facción rival ha completado sus objetivos.

M.6.3. Jugador

Un *jugador* de la partida controla a un ejército de los ocho jugadores que puede haber como máximo simultáneamente. El objetivo del jugador es conseguir los objetivos de la facción a la que pertenece su ejército con la ayuda de los demás miembros de la facción si los hay. Un jugador puede ser de tres tipos:

- Jugador humano. No se generan órdenes automáticamente, el jugador controla un ejército con las órdenes que genere un usuario.
- Jugador artificial. El jugador genera las órdenes automáticamente calculadas por el propio juego y en función de unos parámetros:

- Velocidad de respuesta. Configura a qué velocidad genera órdenes el jugador artificial.
- Comportamiento. Define la estrategia del jugador artificial, si más o menos agresiva.
- Jugador híbrido. Las órdenes son generadas automáticamente por el propio juego de la misma forma que con un jugador artificial pero un usuario puede añadir sus órdenes propias e incluso ordenarle al juego qué tipo de órdenes automáticas tiene que generar.

M.6.4. Ejércitos

Un *ejército* es el conjunto de entidades que controla un jugador en una partida. Las entidades de un ejército pueden ser unidades, estructuras y refineras.

Cada ejército tiene asociado un color, que se muestra en todas sus entidades. Los ejércitos por lo general se muestran en el siguiente orden de colores (ver de izquierda a derecha la figura M.10).

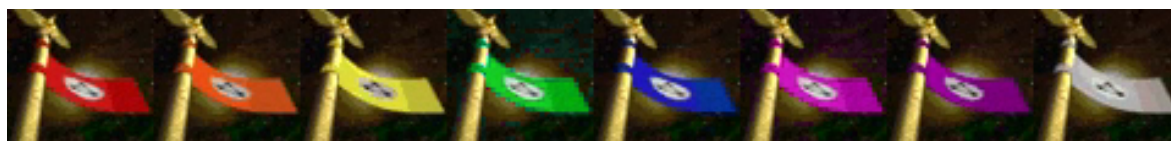


Figura M.10: Variación gráfica de los ejércitos en función de su color.

Un ejército es derrotado en el momento en el que en el escenario no hay ninguna unidad, estructura o refinería que le pertenezca, es decir, de su color, o cuando una facción rival completa sus objetivos. Un ejército tiene que pertenecer a una facción y solo a una.

Las entidades de un ejército son de una misma raza⁷ de las tres razas disponibles.

Existe un ejército especial que no está controlado por ningún jugador, que es el ejército neutral. Este ejército de color gris oscuro no progresa, y su único objetivo es dificultar los avances de los jugadores de la partida.

El ejército neutral atacará a cualquier entidad que encuentre cerca de su territorio, pero no le perseguirá si esta sale de dicha zona⁸.

⁷El juego está preparado y balanceado solo para que los ejércitos estén compuestos de una sola raza, pero se permite al usuario crear escenarios con ejércitos interraciales.

⁸Más información sobre el ejército neutral en el anexo C.2.



Figura M.11: Ejército neutral en el escenario oficial StarCraft.

M.6.5. Raza

Todas las entidades pertenecen a una de las tres razas del juego⁹: Protoss (anexo M.7), zerg (anexo M.8) y terran (anexo M.9).

Aunque que las razas comparten mecánicas básicas, estas difieren entre sí y tienen rasgos únicos, como por ejemplo la creación de edificios.

A pesar de que se permite al usuario utilizar las tres razas en un mismo ejército, esto no es recomendable.

M.6.6. Cristal

El cristal es uno de los tres recursos principales del juego. Se utiliza para casi cualquier compra o mejora durante una partida.

Este se obtiene de los minerales con las unidades trabajadoras. Es un recurso limitado pues cada mineral tiene una cantidad fija de cristal hasta que se agota, desapareciendo el mineral.

M.6.7. Gas

El gas es uno de los tres recursos principales del juego. Se utiliza para las compras y mejoras avanzadas durante una partida.

Este se obtiene de los géiseres tras instalar una refinería encima haciendo uso de las unidades trabajadoras. Aunque cada géiser dispone de una cantidad limitada de gas

⁹Hay un par de entidades excepcionales que se pueden usar en todas las razas: la maravilla (anexo M.6.17) y la defensa (anexo M.6.15).

de *alta calidad*, puede seguir extrayéndose gas de *baja calidad* indefinidamente tras agotarse el primero.

M.6.8. Población

La población es uno de los tres recursos principales del juego. En sí representa la capacidad del ejército de suministrar todo lo necesario para mantener dispuestas para el combate a las unidades de las que dispone. Si una unidad consume más población de la que se dispone, esta no se generará hasta que haya suficiente población.

Existe una *población máxima* que es la capacidad de suministrar máxima y, una *población activa* que es la población actualmente en uso con las unidades de las que se dispone.

El tipo de población y los métodos para obtenerla varía también según la raza.

M.6.9. Unidad

Es el tipo de entidad más complejo y frecuente del juego. Tienen una cantidad de vida determinada y por tanto, pueden morir.

Una unidad está siempre en uno de 10 estados posibles:

- *Quieta*. La unidad conserva la posición sin realizar ninguna acción. Es el estado inicial y el de por defecto.
- *Moviéndose*. La unidad está desplazándose a una casilla objetivo sin ningún objetivo más que el de reposicionarse. Este estado responde a la orden de moverse.
- *Atacando*. La unidad está atacando o desplazándose hacia un objetivo para conseguir estar al alcance y llegar a atacarlo. Se hace una transición a este estado bien cuando se le ordena atacar a un enemigo o bien cuando la unidad es atacada (normalmente). La larva (anexo M.8.1), el huevo (anexo M.8.2) y el soberano (anexo M.8.3) no pueden atacar y por eso no entran en este estado.
 - Mecánica de conservación *Autoataque*: Cuando una unidad acaba con un enemigo, sea este cualquier tipo de entidad enemiga (unidad, estructura o refinería), busca otro enemigo cercano al que atacar.
- *Huyendo*. Este estado surge para permitir al usuario comandar a una unidad lejos del conflicto a pesar de que la estén atacando. Si se ordena moverse a una unidad que está luchando y esta es atacada, la unidad volvería automáticamente al estado *Atacando*. Para evitarlo se crea este estado temporal que hace que la

unidad ignore todos los ataques que esté recibiendo para centrarse en llegar a la ubicación designada.

- *Evolucionando.* Resume la mecánica zerg de la evolución (anexo M.6.13). En este estado la entidad evoluciona en otra distinta transformándose primero en un huevo (anexo M.8.2) hasta llegar a eclosionar y surgir la nueva entidad. Este proceso obliga a la unidad a permanecer inmóvil y sin poder cancelarse la transformación salvo que la unidad sea destruida.
- *Instalando.* Una unidad trabajadora entra en este estado cuando se le ordena instalar una refinera (anexo M.6.11) en un géiser. Este estado consiste en el desplazamiento de la unidad hasta el géiser y una vez allí instalar la refinera. Si bien no encuentra camino para llegar al géiser designado, no se disponen recursos para la instalación, o se ocupa con anterioridad el géiser por otra instalación, la unidad se detiene.
 - Mecánica de conservación *Autoinstalación terran*: Al requerirse unidades trabajadoras durante todo el proceso de la instalación terran (anexo M.9), al acabar dicha instalación se busca otra sin acabar cercana si la hay para continuarla.
- *Construyendo.* Análogamente al estado anterior, una unidad trabajadora entra en este estado cuando se le ordena construir una estructura en una ubicación determinada. El estado resume el desplazamiento hasta la ubicación designada y su construcción. Si no se encuentra camino hasta la ubicación de la construcción, no se disponen de recursos, o está ocupada la ubicación por otra entidad, la unidad se detiene.
 - Mecánica de conservación *Autoconstrucción terran*: Análogamente a la instalación, la construcción terran (anexo M.9) al requerir de unidades trabajadoras durante todo el proceso de la construcción, se diseña al trabajador para que al acabar dicha construcción este busque otra cercana si la hay para continuarla.
- *Minando.* En este estado la unidad está minando cristal de minerales, llevando una unidad de cristal ya minado a una estructura del mismo ejército, o desplazándose de vuelta a un mineral para repetir el proceso. Hay que destacar que la unidad de cristal conseguida no se pierde mientras se lleva a una estructura al cambiar de estado salvo si esta unidad entra en combate.
 - Mecánica de conservación *Autominado*: Tras cada unidad de cristal traída a un edificio, la unidad intenta volver al cristal designado a por más. Si la

unidad agota el cristal designado, buscará otro cercano que no esté en uso por nadie y, en caso de no encontrar ninguno, la unidad se detiene.

- *Recolectando*. La unidad está yendo a una refinera de su ejército a por gas, esperando en la puerta si la refinera está en uso, recolectando el gas dentro de la refinera en una unidad de gas (en un bote, orbe o saco dependiendo de la raza), o volviendo a una estructura a dejar la unidad de gas. Mientras la unidad está dentro de la refinera no es posible cambiar de estado ya que la operación no se puede cancelar, salvo si la refinera es destruida. Si se destruye la refinera o bien no hay estructuras en las que dejar el gas, la unidad pasa a estar quieta.
- Mecánica de conservación *Autorrecolectado*: Tras cada unidad de gas entregada a una estructura, el trabajador intentará volver a la misma refinera a por más aun si el gas restante en el géiser es de baja calidad.

M.6.10. Estructura

La estructura es un tipo de entidad que pertenece a un ejército y que por lo general ocupa más de una casilla. Si su vida llega a cero, esta se destruye.

Las estructuras las construyen los trabajadores, de una u otra manera, dependiendo la raza. Una vez que se construye una estructura, esta puede desbloquear la construcción de otras estructuras distintas, permitir la generación de unidades o aumentar la población.

Las estructuras actúan también de punto de recogida de cristal para los jugadores.

M.6.11. Refinera

La refinera es un tipo de entidad que se instala sobre los géiseres. Su objetivo es permitir la recolección de gas por parte de los trabajadores. Si se destruye una refinera se conserva el géiser con la misma cantidad de gas restante.

M.6.12. Obstáculo

Un obstáculo es un tipo de entidad inmóvil e indestructible cuya única finalidad es la de obstaculizar a los ejércitos ocupando casillas. Aparecen fijos desde el comienzo de la partida y forman parte del decorado del escenario.

M.6.13. Órdenes del jugador

La interacción del usuario en partida con el escenario es a través de las siguientes órdenes de juego:

- **Seleccionar.** Hacer clic izquierdo sobre una entidad (salvo los obstáculos), esta se selecciona y se muestra más información sobre ella. Se pueden seleccionar múltiples unidades aliadas si se hace clic y se mantiene mientras se abarca un área del tablero de juego. Las entidades seleccionadas que pertenezcan al ejército del usuario son susceptibles de recibir otras órdenes de juego.
- **Moverse.** Una unidad puede desplazarse en las ocho direcciones a través de casillas que estén libres. Al seleccionar una o varias unidades y hacer clic derecho sobre el tablero de juego sin que sea sobre un enemigo o una entidad sobre la que se active otro tipo de orden, hará que las unidades seleccionadas se desplacen hacia allí.
- **Atacar.** Si se hace clic derecho sobre una entidad enemiga se marcarán a las unidades aliadas seleccionadas que esa entidad es su objetivo y se dirigirán a atacarle.
- **Recolectar.** Al hacer clic derecho con uno o varios trabajadores seleccionados en una refinería aliada, estos irán a recolectar gas de la refinería para posteriormente traerlo de vuelta a una estructura y poder usarlo.
- **Minar.** Haciendo clic derecho sobre un mineral mientras se tiene uno o varios trabajadores seleccionados les ordenará ir a extraer cristal de dicho mineral y de otros cercanos para traerlo de vuelta a una estructura aliada y poder utilizarlo.
- **Generar.** Al tener seleccionada una unidad o estructura, según el tipo, se muestran opciones en el panel de botones que permiten generar una unidad por un coste.
- **Evolucionar.** Una unidad de la raza zerg puede transformarse en otra unidad o en una estructura si se dan los requisitos necesarios. A veces se solicita tener suficientes recursos además de poseer construidas determinadas estructuras. Según el tipo de unidad zerg seleccionada se muestran las opciones en el panel de botones con más información al respecto.
- **Mejorar.** Las maravillas (anexo M.6.17) pueden ser mejoradas por un precio para alcanzar objetivos de investigación. Para ello hay que tener una maravilla aliada seleccionada y se mostrará las mejoras (si las tiene) en el panel de botones.

M.6.14. Objetivo de facción: exterminar

“Es exactamente lo que parece, nada de vecinos felices que se prestan azúcar.”

Para que una facción exterminadora resulte victoriosa tiene que eliminar todas las entidades de las facciones marcadas como objetivos (ver figura M.12). Eso incluye

todas las refineras, estructuras y unidades enemigas de cada ejército componente de una facción.

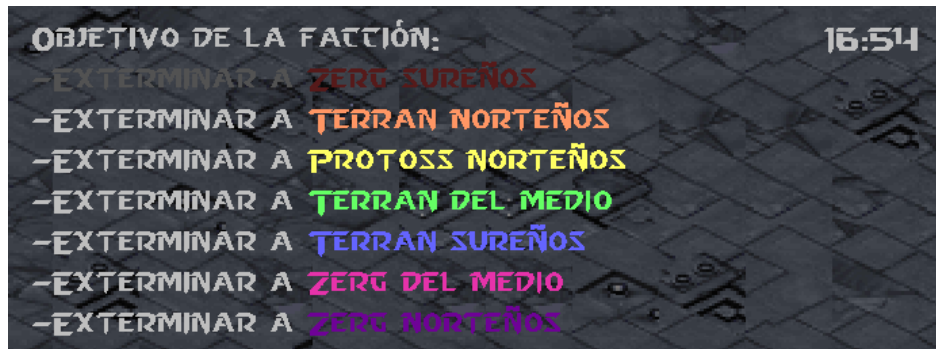


Figura M.12: Progreso de objetivos de exterminio dentro de partida.

Cuando un ejército es aniquilado de esta manera, deja de marcarse como objetivo de partida (figura M.12) y de mostrarse en la tabla de facciones (ver figura M.13).

STARCRRAFT		
FACCIONES	EJÉRCITOS	ASESINATOS
FACCION A	ZERG SUREÑOS	26
FACCION B	TERRAN NORTEÑOS	18
FACCION C	PROTOSS NORTEÑOS	24
FACCION D	TERRAN DEL MEDIO	79
FACCION E	TERRAN SUREÑOS	52
FACCION F	ZERG DEL MEDIO	22
FACCION G	ZERG NORTEÑOS	13
FACCION H	PROTOSS SUREÑOS	24

Figura M.13: Tabla de facciones dentro de partida en el escenario StarCraft.

M.6.15. Objetivo de facción: sobrevivir

“Aquí sientes que tu mayor enemigo es el tiempo. Mi día a día haciendo este proyecto.”

El objetivo de supervivencia consiste en mantener al menos una entidad de la facción viva hasta un determinado momento de la partida objetivo.

Para aumentar la capacidad de defensa de los ejércitos se crea una estructura-obstáculo denominada ‘Defensa’ (ver figura M.14).

La *defensa* es una estructura especial que no tiene mayor funcionalidad que la de obstaculizar a los rivales. No tiene regeneración de salud y tampoco puede crearse

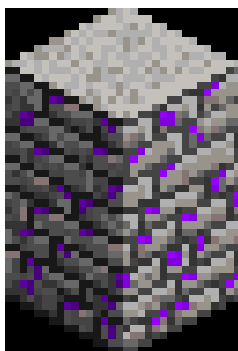


Figura M.14: Defensa del ejército violeta.

durante la partida. Además, no indica la salud restante salvo si esta se selecciona. Tiene la característica especial de recibir un 80 % menos de daño de las unidades que atacan a distancia.

M.6.16. Objetivo de facción: conquistar

Una facción con este objetivo alcanzará la victoria al obtener 1000 puntos de conquista.

Los puntos de conquista se consiguen a través del control de una *zona de conquista* (ver figura M.15).



Figura M.15: Zona de conquista.

Una zona de conquista es un espacio de cuatro casillas de ancho y dos de alto que otorga puntos de conquista según el ejército que esté ocupando dichas casillas.

Los puntos de conquista subirán por cada ciclo de partida y por cada casilla de la zona de conquista que esté siendo ocupada por una entidad de la facción que la tenga como objetivo. Los puntos de conquista, sin embargo, disminuirán con el paso del tiempo y a mayor velocidad si hay entidades enemigas en la zona de conquista.

Cuenta cualquier tipo de entidad de un ejército, eso incluye a las estructuras, tanto para conquistar como para defender la zona de conquista.

M.6.17. Objetivo de facción: investigar

“La maravilla intenta ser mi granito de arena a la historia de StarCraft (ya que no existen en el videojuego original).”

Las facciones investigadoras tienen como objetivo completar las mejoras de un determinado número de estructuras especiales denominadas *maravillas* (ver figura M.16).

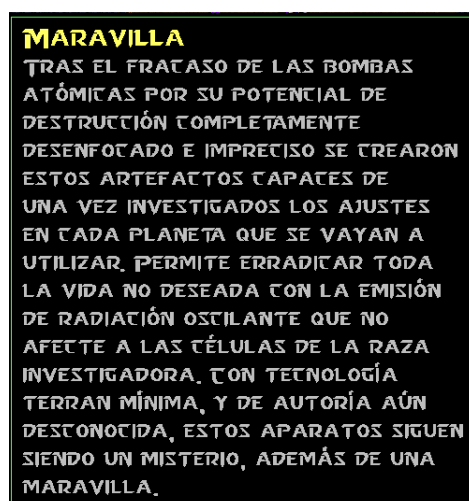


Figura M.16: Panel de información de la maravilla dentro de partida.

Una maravilla es una estructura especial común a todas las razas que requiere de la construcción previa de los edificios avanzados para poder crearse. Una vez construida se pueden realizar hasta cuatro mejoras, tras las cuales se considera la investigación finalizada y se aumenta el progreso de la facción investigadora que la construya. Como se ve en la figura M.17, de izquierda a derecha y de arriba a abajo, se muestran los seis distintos aspectos de una maravilla: primero siendo construida, segundo estando operativa y las otras cuatro formas según se terminan de aplicar las mejoras.



Figura M.17: Distintas apariencias de la maravilla del ejército rojo.

M.7. Protoss

“Parecen momias, manejan tecnología avanzada, y de sociables poco. Dirán que son ‘egipcios galácticos’, pero yo creo que son informáticos.”

La raza protoss (ver figura M.18) dispone de las siguientes facultades únicas:

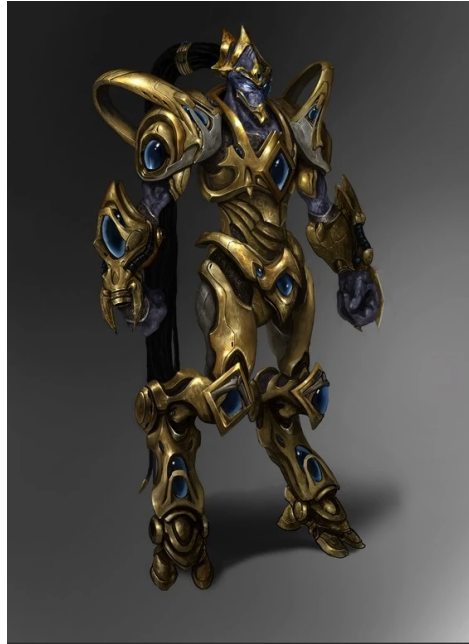


Figura M.18: Un protoss.

- Tienen escudo. Este amortigua el daño antes de que se pueda dañar a la salud. Además el escudo se regenera a un ritmo mayor que la salud y pueden recuperarse ambos valores en paralelo.
- Su método de construcción de estructuras y refineras consiste en teletransportar dichas entidades desde su planeta natal, reduciendo la tarea del trabajador a simplemente activar el portal y quedando así libre para otras acciones durante el proceso de construcción.
- Utilizan la energía ‘psi’. Esto los convierte en la raza con mejores estadísticas ya que mejoran así sus armas y estructuras, acompañado sin embargo de unos mayores costes. Su población se cuantifica precisamente en la cantidad de energía *psi*.

M.7.1. Trabajador: Sonda

La *sonda* es la unidad trabajadora protoss. Utiliza un rayo de partículas para las tareas de minado pero también puede servirle para intentar defenderse. Aunque puede volar su velocidad es limitada, así como su salud y su escudo.



Figura M.19: Cartel: Sonda.

- Se genera en: Nexo (anexo M.7.4).
- Tiempo de generación: 20 segundos.
- Coste de cristal: 50 unidades.
- Coste de gas: 0 unidades.
- Psi: Consume 1.

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
20	20	3	1	22	5

Tabla M.1: Estadísticas: Sonda.

M.7.2. Soldado básico: Fanático

“Se le iba a llamar ‘Simpático’, pero como se dedica a perseguir a los demás con dos espadas no terminaba de cuajar.”

Un *fanático* es el soldado básico protoss. Cada fanático hace uso de sus dos hojas psiónicas para combatir. Su resistencia es bastante superior a la de sus homólogos de las otras razas, aunque también lo es su coste.

- Se genera en: Portal (anexo M.7.6).
- Tiempo de generación: 40 segundos.
- Coste de cristal: 100 unidades.



Figura M.20: Cartel: Fanático.

- Coste de gas: 0 unidades.
- Psi: Consume 2.
- Característica especial *Fanatismo*: Cada vez que consigue atacar, se ensaña realizando un segundo corte que infringe el 100 % del daño.

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
100	60	2	1	22	8

Tabla M.2: Estadísticas: Fanático.

M.7.3. Soldado avanzado: Dragón

“Hasta cierto punto del desarrollo lo llamaron ‘Templario’, pero sigue siendo igual de anticlimático para un ente con cuatro patas que lanza bolas de energía.”¹⁰

Un *dragón* es el soldado avanzado protoss. Su equipamiento consta de un disruptor de fase que dispara proyectiles de energía. Es una unidad muy pesada, haciendo que tanto sus ataques como su movimiento sean lentos, pero su capacidad destructiva fácilmente lo compensa.

- Se genera en: *Portal* (anexo M.7.6), pero requiere *Centro cibernético* (anexo M.7.7).
- Tiempo de generación: 50 segundos.
- Coste de cristal: 125 unidades.

¹⁰[https://starcraft.fandom.com/wiki/Draagoon_\(StarCraft\)#Development](https://starcraft.fandom.com/wiki/Draagoon_(StarCraft)#Development) (último acceso: 22/5/2023).



Figura M.21: Cartel: Dragón.

- Coste de gas: 50 unidades.
- Psi: Consume 2.

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
100	80	2	4	30	20

Tabla M.3: Estadísticas: Dragón.

M.7.4. Edificio base: Nexo



Figura M.22: Cartel: Nexo.

El *nexo* es una gigantesca estructura piramidal que actúa de edificio base protoss y, que además supone la fuente principal de la energía psiónica.

- Tiempo de construcción: 120 segundos.

- Coste de cristal: 400 unidades.
- Coste de gas: 0 unidades.
- Psi: Otorga 10.
- Característica especial *Generador de trabajadores protoss*: Permite crear sondas (anexo M.7.1).

Salud	Escudo	Anchura en casillas	Altura en casillas
500	500	4	3

Tabla M.4: Estadísticas: Nexo.

M.7.5. Edificio poblacional: Pilón

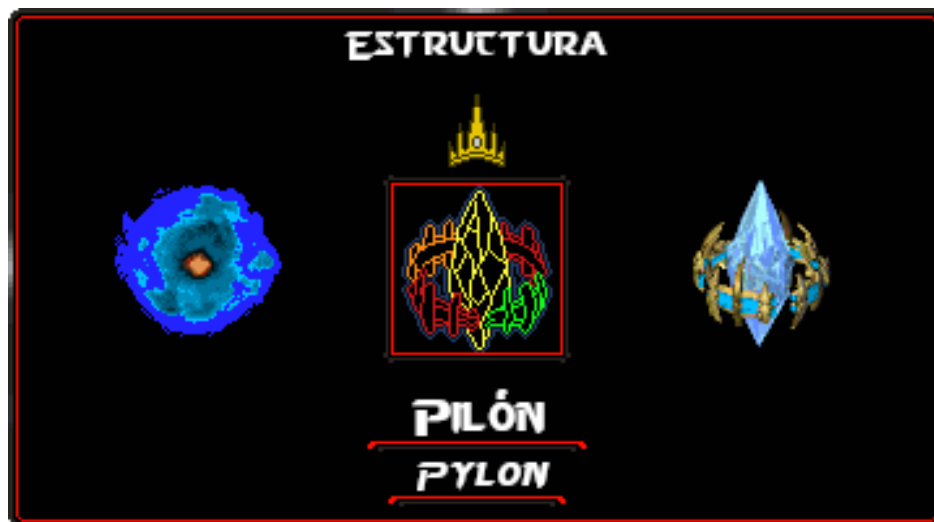


Figura M.23: Cartel: Pilón.

El *pilón* es un cristal flotante que permite la amplificación de energía psi. Los pilones suponen el grueso del recurso poblacional protoss.

- Requisito: Nexo (anexo M.7.4).
- Tiempo de construcción: 30 segundos.
- Coste de cristal: 100 unidades.
- Coste de gas: 0 unidades.
- Psi: Otorga 8.

Salud	Escudo	Anchura en casillas	Altura en casillas
300	300	2	2

Tabla M.5: Estadísticas: Pílon.



Figura M.24: Cartel: Portal.

M.7.6. Edificio reclutamiento: Portal

Con un *portal* los soldados protoss pueden ser invocados desde su planeta natal para luchar en la partida.

- Requisito: Nexo (anexo M.7.4).
- Tiempo de construcción: 60 segundos.
- Coste de cristal: 150 unidades.
- Coste de gas: 0 unidades.
- Psi: No consume psi.
- Característica especial *Generador de soldados protoss*: Permite la generación de soldados protoss básicos y avanzados.

Salud	Escudo	Anchura en casillas	Altura en casillas
500	500	4	2

Tabla M.6: Estadísticas: Portal.

M.7.7. Edificio avanzado: Centro cibernético

“Lo que he entendido es que aquí se genera, además de la cibernética, una especie de zumo para descargar el espíritu de los protoss caídos y subirlo, por ejemplo, a un dragón. Solo si eres un soldado de élite, si no, no hay zumito.”¹¹

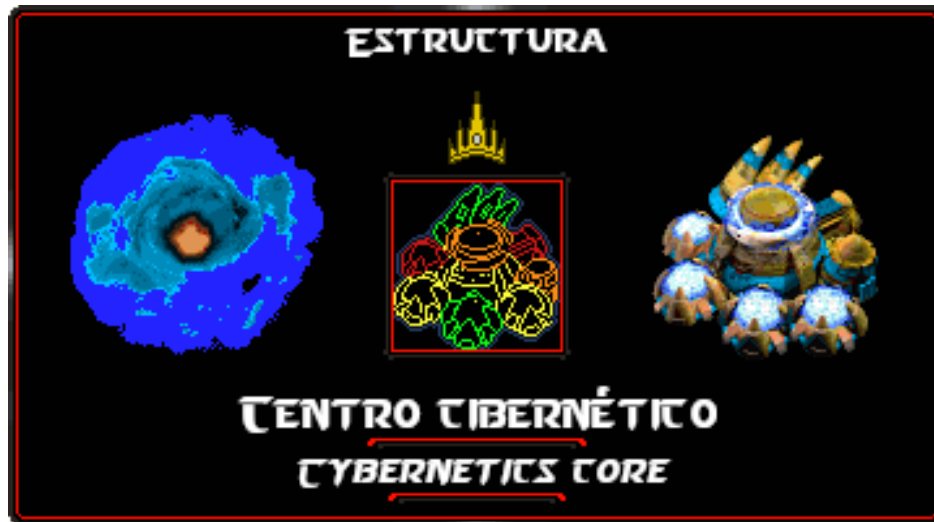


Figura M.25: Cartel: Centro cibernético.

En el *centro cibernético* se fabrica el armamento protoss más avanzado, además de recuperarse los espíritus de los soldados de élite caídos.

- Requisito: Nexo (anexo M.7.4) y Portal (anexo M.7.6).
- Tiempo de construcción: 60 segundos.
- Coste de cristal: 200 unidades.
- Coste de gas: 0 unidades.
- Psi: No consume psi.

Salud	Escudo	Anchura en casillas	Altura en casillas
500	500	3	2

Tabla M.7: Estadísticas: Centro cibernético.

M.7.8. Instalación refinera: Asimilador

“Nadie tiene claro lo que hace más allá de ¿asimilar? gas, y meterlo en botecitos.”

¹¹https://starcraft.fandom.com/wiki/Cybernetics_core#Overview (último acceso: 24/5/2023).



Figura M.26: Cartel: Asimilador.

El *asimilador* es la refinera protoss que permite la recolección de gas a los trabajadores mediante botes de cristal.

- Tiempo de instalación: 40 segundos.
- Coste de cristal: 100 unidades.
- Coste de gas: 0 unidades.
- Psi: No consume psi.

Salud	Escudo	Anchura en casillas	Altura en casillas
450	450	4	2

Tabla M.8: Estadísticas: Asimilador.

M.8. Zerg

“Devoran todo sin piedad, asimilan lo que les interesa y descartan lo que no. Algo así como una gran multinacional.”

La raza *zerg* dispone de las siguientes facultades únicas:

- Poseen una mayor regeneración de vida. Debido a su dominio genético y facilidad a la hora de absorber nutrientes para mutar.
- Su método de construcción de estructuras y refineras se basa en la mutación de la propia unidad trabajadora hasta transformarse en la construcción deseada. La generación de las unidades también es a través de mutación y sacrificio de



Figura M.27: Un hidralisco zerg.

unidades especiales conocidas como larvas (anexo M.8.1).

- No se tienen capacidad de suministro como tal, sino que su población se mide por el *control* de las unidades zerg que ejerce la mente colmena de la que todos los zerg del ejército son parte.

M.8.1. Unidad especial: Larva

“También conocida como ‘gusano valiente explorador’.”



Figura M.28: Cartel: Larva.

La *larva* es la unidad especial zerg desde la que mutan todas las demás unidades. No tienen capacidades ofensivas, por lo que tampoco ningún arma. Se generan sin coste alguno aunque existe un límite de larvas en partida.

- Se genera en: *Criadero* (anexo M.8.7).
- Requiere: Espacio de larva (anexo M.8.7).
- Tiempo de generación: 15 segundos.
- Coste de cristal: 0 unidades.
- Coste de gas: 0 unidades.
- Control: No consume control.
- Característica especial *Mutación a unidad*: Pueden mutar a cualquier otra unidad zerg (siempre que se cumplan los requisitos), tras una transformación dentro de un huevo (anexo M.8.2).

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
25	0	2	-	-	-

Tabla M.9: Estadísticas: Larva.

M.8.2. Unidad especial: Huevo



Figura M.29: Cartel: Huevo.

El *huevo* es una unidad especial zerg que actúa como estado intermedio entre la larva (anexo M.8.1) y otra unidad zerg distinta a la que la larva esté evolucionando. No tiene forma de atacar o de desplazarse, por lo que queda expuesto durante su incubación. Se crea instantáneamente al ordenar a una larva evolucionar.

- Se genera en: Larva.
- Requiere: Varios edificios en función de la unidad incubando.

- Coste de cristal: Varía en función de la unidad incubando.
- Coste de gas: Varía en función de la unidad incubando.
- Control: Varía en función de la unidad incubando.

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
100	0	-	-	-	-

Tabla M.10: Estadísticas: Huevo.

M.8.3. Unidad poblacional: Soberano

“El equipo de desarrollo se debió quedar sin ideas cuando juntó un oso hormiguero, un cangrejo, una araña y un zepelín y al resultado le pusieron sonidos de una persona y un caballo.”¹²



Figura M.30: Cartel: Soberano.

El *soberano* es una unidad especial sin capacidades ofensivas cuya principal funcionalidad es la de aumentar el control mental sobre las masas zerg.

- Se genera en: Larva (anexo M.8.1).
- Tiempo de generación: 40 segundos.
- Coste de cristal: 100 unidades.
- Coste de gas: 0 unidades.
- Control: Otorga 16.

¹²[https://starcraft.fandom.com/wiki/Overlord_\(StarCraft\)#Notes](https://starcraft.fandom.com/wiki/Overlord_(StarCraft)#Notes) (último acceso: 22/5/2023).

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
200	0	2	-	-	-

Tabla M.11: Estadísticas: Soberano.

M.8.4. Trabajador: Zumbido

“Versión encantadora y amistosa del ‘abrazacaras’ de Alien.”¹³



Figura M.31: Cartel: Zumbido.

El *zumbido* es la unidad trabajadora zerg. Sus garras le permiten partir minerales y enemigos por igual.

- Se genera en: Larva (anexo M.8.1).
- Tiempo de generación: 20 segundos.
- Coste de cristal: 50 unidades.
- Coste de gas: 0 unidades.
- Control: Consume 2.
- Característica especial *Mutación a estructura*: Un zumbido es capaz de transformarse en una masa biológica hecha de tentáculos y órganos dispersos que a través de explosiones genéticas muta hasta convertirse en la estructura zerg deseada.

¹³Más información sobre el “abrazacaras”: <https://avp.fandom.com/wiki/Facehugger> (último acceso: 17/5/2023).

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
40	0	3	1	22	5

Tabla M.12: Estadísticas: Zumbido.

M.8.5. Soldado básico: Zergling

“Resulta que este em, ¿perro alien rabioso con brazos? (si, eso que parecen antenas son sus brazos) ha estado en el Smithsonian.”¹⁴



Figura M.32: Cartel: Zergling.

Los *zergling* son los soldados básicos zerg. Aunque su constitución es débil, sus dientes y garras le convierten en una arma muy dañina si se le deja atacar.

- Se genera en: Larva (anexo M.8.1).
- Requiere: Estanque de desove (anexo M.8.8).
- Tiempo de generación: 28 segundos.
- Coste de cristal: 25 unidades.
- Coste de gas: 0 unidades.
- Control: Consume 2.
- Característica especial *Cuadrúpedo*: Su instinto y genética animal le otorgan una velocidad superior a cualquier otra unidad.

¹⁴<https://news.blizzard.com/es-es/starcraft/20719767/dias-de-rock-and-roll-de-starcraft-mirada-retrospectiva-del-desarrollo> (último acceso: 17/5/2023).

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
30	0	4	1	10	5

Tabla M.13: Estadísticas: Zergling.

M.8.6. Soldado avanzado: Hidralisco

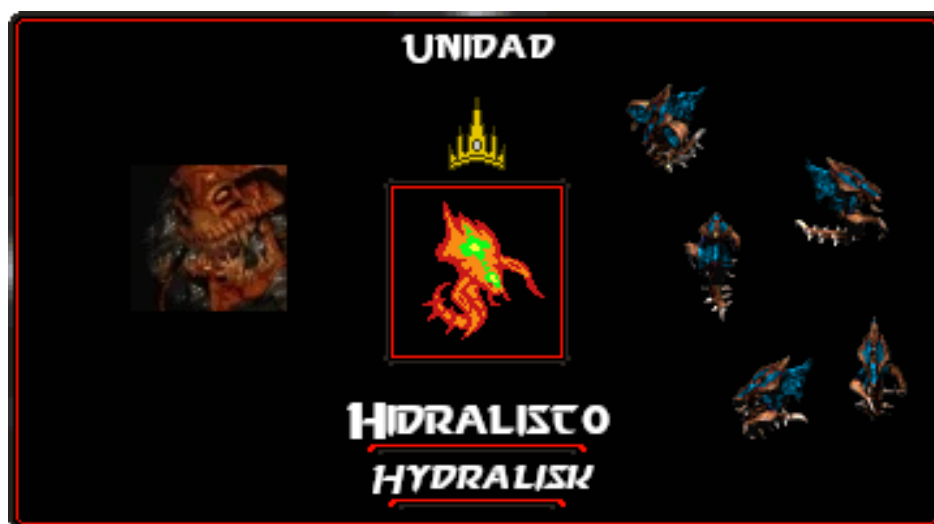


Figura M.33: Cartel: Hidralisco.

El *hidralisco* es el soldado avanzado zerg. A pesar de ser producto de la absorción genética de una especie herbívora pacífica, han evolucionado hasta ser una peligrosa fuerza zerg a distancia gracias a la secreción de cientos de agujas venenosas que utilizan como arma.

- Se genera en: Larva (anexo M.8.1).
- Requiere: Estanque de desove (anexo M.8.8) y guarida de hidraliscos (anexo M.8.9).
- Tiempo de generación: 28 segundos.
- Coste de cristal: 75 unidades.
- Coste de gas: 25 unidades.
- Control: Consume 2.

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
70	0	2	4	18	10

Tabla M.14: Estadísticas: Hidralisco.



Figura M.34: Cartel: Criadero.

M.8.7. Edificio base: Criadero

El *criadero* es el corazón de todo ejército zerg, pues de este salen las larvas (ver M.8.1) y de ellas todo lo demás.

- Se genera en: Zumbido (anexo M.8.4).
- Tiempo de construcción: 300 segundos.
- Coste de cristal: 300 unidades.
- Coste de gas: 0 unidades.
- Control: Otorga 4.
- Característica especial *Generación acotada de larvas*: El criadero está en constante generación de larvas sin coste alguno, pero solo se generan larvas mientras haya menos de tres larvas por criadero en todo el ejército.

Salud	Escudo	Anchura en casillas	Altura en casillas
1250	0	4	2

Tabla M.15: Estadísticas: Criadero.

M.8.8. Edificio especial: Estanque de desove

“Un ‘estanque’, de ‘pringue primordial’, con el ‘material genético’ que ... Bueno. He visto suficiente.”

El *estanque de desove* acumula todo el material genético necesario para crear las diferentes formas de vida zerg. Permite a las larvas evolucionar a zergling (anexo M.8.5).



Figura M.35: Cartel: Estanque de desove.

- Requiere: Criadero (anexo M.8.7).
- Se genera en: Zumbido (anexo M.8.4).
- Tiempo de construcción: 80 segundos.
- Coste de cristal: 200 unidades.
- Coste de gas: 0 unidades.
- Control: No consume control.

Salud	Escudo	Anchura en casillas	Altura en casillas
750	0	3	2

Tabla M.16: Estadísticas: Estanque de desove.

M.8.9. Edificio avanzado: Guarida de hidraliscos

La *guarida de hidraliscos* conserva la esencia de la especie absorbida por los propios hidraliscos, permitiendo así su generación por la evolución de las larvas (anexo M.8.1).

- Se genera en: Zumbido (anexo M.8.4).
- Requisito: Estanque de desove (anexo M.8.8).
- Tiempo de construcción: 40 segundos.
- Coste de cristal: 100 unidades.
- Coste de gas: 50 unidades.



Figura M.36: Cartel: Guarida de hidraliscos.

- Control: No consume control.

Salud	Escudo	Anchura en casillas	Altura en casillas
850	0	3	2

Tabla M.17: Estadísticas: Guarida de hidraliscos.

M.8.10. Instalación refinera: Extractor



Figura M.37: Cartel: Extractor.

El *extractor* es la refinera zerg. Esta permite la recolección de gas a los trabajadores mediante orbes de carne.

- Tiempo de instalación: 40 segundos.
- Coste de cristal: 100 unidades.

- Coste de gas: 0 unidades.
- Control: No consume control.

Salud	Escudo	Anchura en casillas	Altura en casillas
750	0	4	2

Tabla M.18: Estadísticas: Extractor.

M.9. Terran

“Algo así como vaqueros moto-metaleros espaciales. Juran, beben y fuman deseando matar cosas con sus trajes, armas y vehículos hiper pesados de metal.”¹⁵

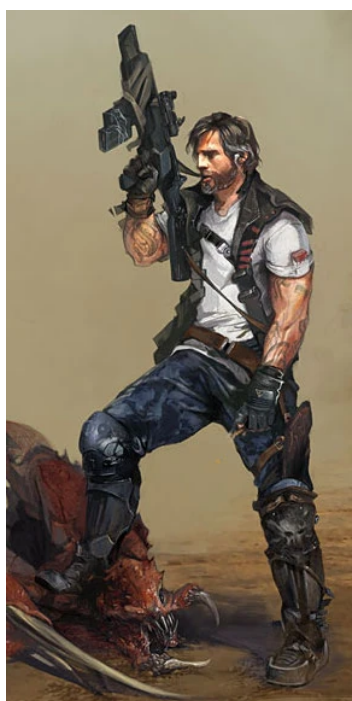


Figura M.38: Un marine terran.

La raza *terran* dispone de las siguientes facultades únicas:

- Tienen mayor alcance de ataque: todos sus soldados atacan a distancia. Los terran son débiles y por eso diseñan armas de fuego para eliminar a sus objetivos antes de tener que enfrentarse cuerpo a cuerpo con ellos.
- El proceso de construcción de estructuras y refineras no es automático, requiere de trabajadores que realicen dicha construcción. La ventaja está en la unión de trabajadores, pues a más trabajadores participen, más rápidamente se terminará la obra.

¹⁵<https://starcraft.fandom.com/wiki/Terran#Notes> (último acceso: 23/5/2023).

- Requieren alimento, munición, y a veces, de gasolina. Por eso su recurso poblacional son los *suministros*. Como lado positivo está que todas las unidades son baratas, rápidas de generar y consumen la misma cantidad de suministros: una unidad.

M.9.1. Trabajador: VCE



Figura M.39: Cartel: Vehículo de construcción espacial.

El *Vehículo de Construcción Espacial (VCE)* es un técnico cualificado para hacer las labores de unidad trabajadora terran utilizando un *exotraje* mecanizado. Los extremos de sus brazos robóticos pueden transformarse en taladros que sirven para perforar el mineral y, en una situación extrema, para defenderse.

- Se genera en: Centro de operaciones (anexo M.9.4).
- Tiempo de generación: 20 segundos.
- Coste de cristal: 50 unidades.
- Coste de gas: 0 unidades.
- Suministros: Consume 1.

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
60	0	3	1	15	5

Tabla M.19: Estadísticas: VCE.

M.9.2. Soldado básico: Marine

“Los intelectuales terran (no, son idiotas con rifles y tengo pruebas).”¹⁶

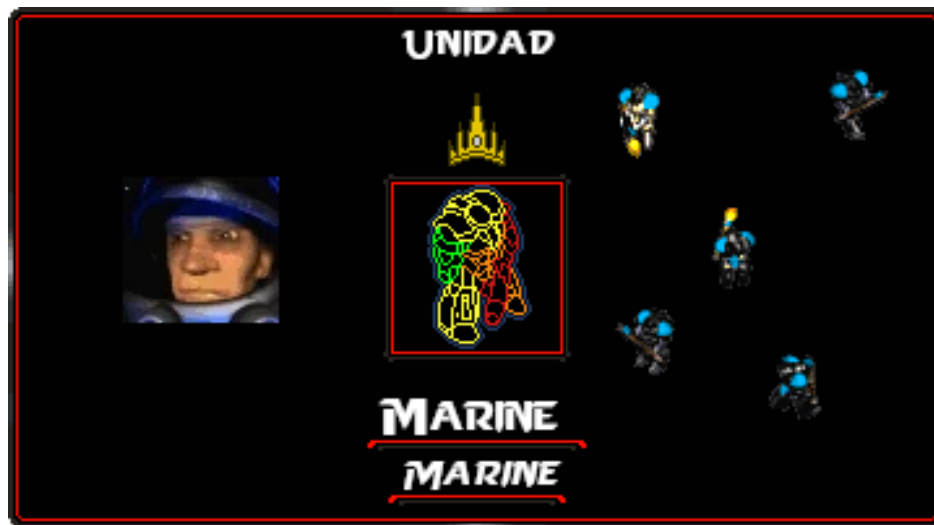


Figura M.40: Cartel: Marine.

Los *marines* son el grueso de las tropas terran, ya que son los soldados básicos, su coste es mínimo, y por si fuera poco tienen la ventaja de poseer un rifle de asalto con el que matar a sus enemigos a distancia.

- Se genera en: Cuarteles (anexo M.9.6).
- Tiempo de generación: 24 segundos.
- Coste de cristal: 50 unidades.
- Coste de gas: 0 unidades.
- Suministros: Consume 1.

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
40	0	2	4	15	6

Tabla M.20: Estadísticas: Marine.

M.9.3. Soldado avanzado: Murflama

“El esfuerzo requerido para traducir de ‘Firebat’ a ‘Murflama’ ha pasado por cosas tan horrendas como ‘Llamálogo’, ‘Ignibate’ o ‘Maldito pirómano amargado’.”

¹⁶[https://starcraft.fandom.com/wiki/Marine_\(StarCraft\)#Development](https://starcraft.fandom.com/wiki/Marine_(StarCraft)#Development) (último acceso: 23/5/2023).

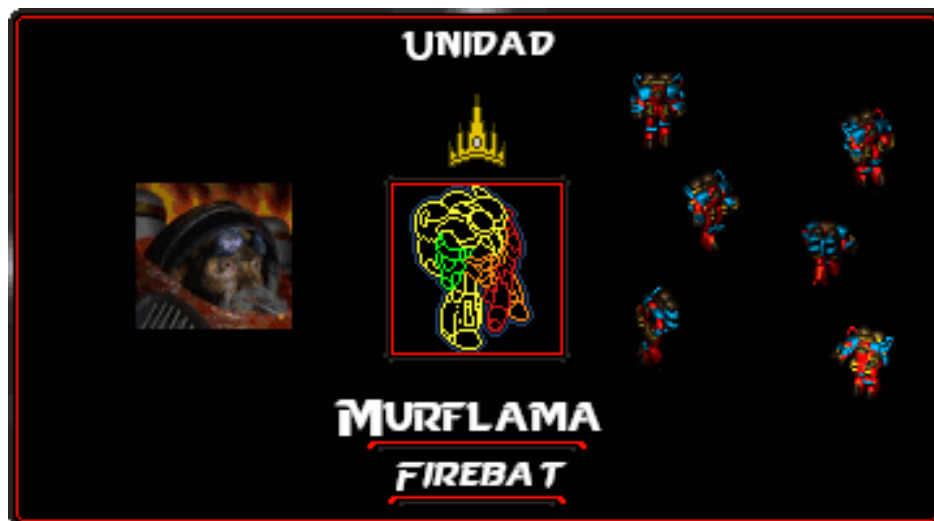


Figura M.41: Cartel: Murflama.

Los *murflama* son la infantería avanzada terran. Van equipados con lanzallamas en cada brazo.

- Se genera en: Cuarteles (anexo M.9.6).
- Requiere: Academia (anexo M.9.7).
- Tiempo de generación: 24 segundos.
- Coste de cristal: 50 unidades.
- Coste de gas: 25 unidades.
- Suministros: Consume 1.
- Característica especial *Por si acaso*: Los murflama mantienen el gatillo un poco más tras cada ataque y realizan los ataques de tres en tres.

Salud	Escudo	Velocidad	Alcance	Enfriamiento	Daño
50	0	2	2	22	8

Tabla M.21: Estadísticas: Murflama.

M.9.4. Edificio base: Centro de operaciones

*“Me recuerda mucho a la nave de ‘E.T.’ y ni siquiera sé porqué.”*¹⁷

El *centro de operaciones* es la base terran donde los altos cargos dan las órdenes a los reclutas.

¹⁷<https://ettheextraterrestrial.fandom.com/wiki/Spaceship> (último acceso: 23/5/2023).



Figura M.42: Cartel: Centro de operaciones.

- Tiempo de construcción: 120 segundos.
- Coste de cristal: 400 unidades.
- Coste de gas: 0 unidades.
- Suministros: Otorga 10.
- Característica especial *Generador de trabajadores terran*: Permite crear VCE (anexo M.9.1).

Salud	Escudo	Anchura en casillas	Altura en casillas
500	0	4	2

Tabla M.22: Estadísticas: Centro de operaciones.

M.9.5. Edificio poblacional: Almacén de suministros

“Aquí guardan las cosas. También es donde se escaquean los soldados a veces para beber . Me alegra ver que en el siglo XXVI la despensa sigue siendo una buena idea.”^{18 19}

El *almacén de suministros* es la única forma de hacer llegar a tiempo los suministros que mantienen el orden en las fuerzas terran.

- Tiempo de construcción: 40 segundos.
- Coste de cristal: 100 unidades.

¹⁸https://starcraft.fandom.com/wiki/Supply_depot#Overview (último acceso: 23/5/2023).

¹⁹<https://starcraft.fandom.com/wiki/Timeline> (último acceso: 23/05/2023).



Figura M.43: Cartel: Almacén de suministros.

- Coste de gas: 0 unidades.
- Suministros: Otorga 8.

Salud	Escudo	Anchura en casillas	Altura en casillas
500	0	3	2

Tabla M.23: Estadísticas: Almacén de suministros.

M.9.6. Edificio reclutamiento: Cuarteles

“Según los desarrolladores parece una tostadora, al fin coincidimos en algo.”²⁰



Figura M.44: Cartel: Cuartel.

²⁰<https://starcraft.fandom.com/wiki/Barracks#Development> (último acceso: 23/5/2023).

En los *cuarteles* se adiestran todas las tropas terran. Son edificios tan resistentes como el centro de operaciones (anexo M.9.4).

- Requiere: Centro de operaciones.
- Tiempo de construcción: 80 segundos.
- Coste de cristal: 150 unidades.
- Coste de gas: 0 unidades.
- Suministros: No consume suministros.
- Característica especial *Generador de soldados terran*: Permite la generación de soldados terran básicos y avanzados.

Salud	Escudo	Anchura en casillas	Altura en casillas
1500	0	4	3

Tabla M.24: Estadísticas: Cuarteles.

M.9.7. Edificio avanzado: Academia

“Dice mucho de las personas que una academia sea edificio avanzado y no básico.”



Figura M.45: Cartel: Academia.

La *academia* permite instruir a los reclutas en el uso de armamento complejo. Permite la creación de los murflamas (anexo M.9.3).

- Requisito: Cuarteles (anexo M.9.6).
- Tiempo de construcción: 80 segundos.

- Coste de cristal: 150 unidades.
- Coste de gas: 0 unidades.
- Suministros: No consume suministros.

Salud	Escudo	Anchura en casillas	Altura en casillas
1500	0	3	2

Tabla M.25: Estadísticas: Academia.

M.9.8. Instalación refinera: Refinería

“Está clarísimo: la refinera ‘Refinería’ refina gas.”



Figura M.46: Cartel: Refinería.

La *refinería* es la refinera terran que permite la recolección de gas a los trabajadores.

- Tiempo de instalación: 40 segundos.
- Coste de cristal: 100 unidades.
- Coste de gas: 0 unidades.
- Suministros: No consume suministros.

Salud	Escudo	Anchura en casillas	Altura en casillas
750	0	4	2

Tabla M.26: Estadísticas: Refinería.

M.10. Editor de escenarios

“Podía haberlo hecho más complicado, pero me detuvieron. Por ahora...”

En el editor de escenarios (ver figura M.47) se permite modificar los escenarios en los que se va a jugar una partida. Existen diversas herramientas para ayudar a visualizar y editar el escenario seleccionado que se comentan más adelante.

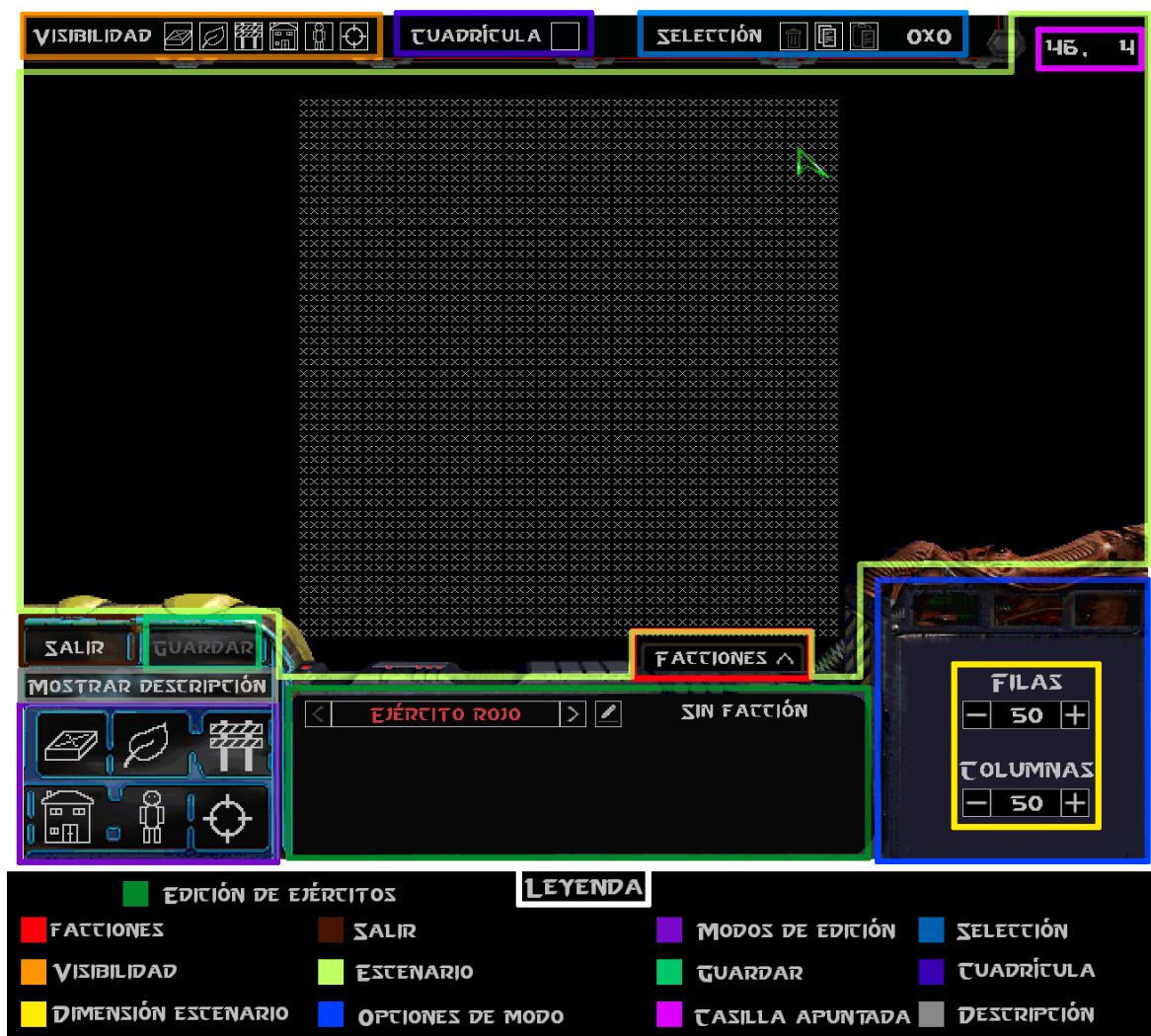


Figura M.47: Captura general del editor de escenarios con señalizaciones.

Ante cualquier cambio el editor activará el botón de *Guardar* permitiendo, al accionarlo, guardar el estado del escenario en un fichero asignado a este en la carpeta dentro del directorio del juego *stages*, sobrescribiendo una versión anterior si esta existiera. Para salir, presionando el control de salir o pulsando la opción de *Salir*, habiendo cambios sin guardar, será necesario confirmar que se descartan los cambios en una ventana emergente que aparece al realizar dicha acción (ver figura M.48).

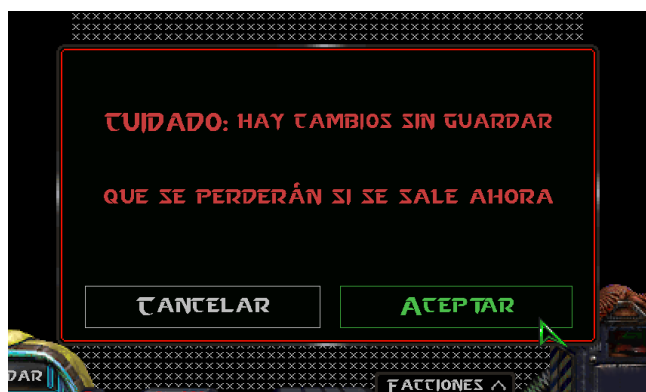


Figura M.48: Ventana emergente del editor de cambios sin guardar.

M.10.1. Modos de edición

“Los controles hacen línea en el teclado: ‘R’ de ‘Recursos’, ‘T’ de ‘Tejas’ (Tiles), ‘U’ de ‘Unidades’, ‘I’ de ‘Infraestructuras’, ‘O’ de ‘Obstáculos’, e ‘Y’ de ... ‘Y objetivos’.”

Los modos de edición alteran la manera de interactuar con el escenario. Cuando no hay ningún modo de edición activado la edición solo es posible a través de las herramientas de selección (anexo M.10.3).

Al activar un modo de edición aparece en la esquina derecha inferior las opciones de modo, desde las cuales se permite escoger qué elemento se quiere añadir al escenario. Los elementos mostrados dependen del modo de edición seleccionado, al igual que los elementos que se pueden borrar en dicho modo. Ambos aspectos están ligados a la categoría de elementos que agrupa el modo de edición.

Los modos de edición ofrecen las siguientes opciones en función de cual se tenga activo:

- Baldosas. Agrupa todas las baldosas del escenario y además dispone de variaciones de cada tipo de baldosa para facilitar si se alteran de una en una, de tres en tres o todas a la vez. También permite editar casillas para que no tengan baldosa y actúen de *casillas sin declarar* (anexo M.10.2).
- Recursos. En este modo se engloban las entidades que ofrecen indirecta o directamente recursos naturales (cristal y gas).
- Obstáculos. Aquí están todos los obstáculos indestructibles.
- Estructuras. Se incluye también la estructura-obstáculo, defensa (anexo M.6.15).
- Unidades. Todas las unidades, sean soldados, trabajadores o unidades especiales.
- Objetivos. La zona de conquista (anexo M.6.16) y la maravilla (anexo M.6.17). Ya que son vitales para completar algunos objetivos de facción.

Independientemente del elemento seleccionado, siempre que se tenga un modo de edición activo se podrán borrar elementos colocados en el escenario si son de la misma categoría.

M.10.2. Cuadrícula

Cuando la cuadrícula está activa se muestran los bordes de las casillas del escenario. El color de los bordes depende del tipo de ocupante que tenga la casilla en ese momento (ver figura M.49):



Figura M.49: Escenario con la cuadrícula activa y el control de cuadrícula activo.

- **Verde - Casilla vacía.** Puede incluir una zona de conquista ya que las zonas de conquista no ocupan el espacio como tal.
- **Amarillo - Casilla con unidad.** Da igual el tipo, soldado o trabajador o unidad especial entra dentro de esta categoría.
- **Naranja - Casilla con estructura.** Todas las estructuras del modo de edición de estructuras, incluyendo las defensas.
- **Rojo - Casilla con obstáculo.** Cualquier obstáculo del modo de edición de obstáculos produce este color.
- **Violeta - Casilla con refinería.** Las refineras al ser instalaciones y no estructuras tienen un color distinto.

- **Azul - Casilla con géiser.** Los géiseres tienen una mecánica única además de ser fijos al escenario y no pertenecer a ningún ejército, por lo que también tienen un color distintivo.
- **Gris claro - Casilla con mineral.** Los minerales tienen una mecánica única además de ser volátiles y no pertenecer a ningún ejército, por lo que tienen también un color distintivo.
- **Gris oscuro - Casilla sin declarar.** La *casilla sin declarar* es una casilla que no podrá ser ocupada y que no tiene dibujo de baldosa. Al crear un escenario, este comienza con todas las casillas vacías y sin declarar. Este tipo de casilla permite que se creen escenarios que no sean de aspecto completamente cuadrados. Al borrar la baldosa de una casilla, esta pasa a ser sin declarar. En el editor, estas casillas se muestran con una cruz blanca.

M.10.3. Herramientas de selección

Con las herramientas de selección (ver figura M.50) se permite editar de una manera más generalizada el escenario que con los modos de edición.

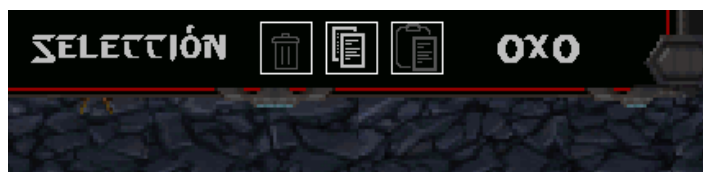


Figura M.50: Herramientas de selección.

Al seleccionar con el cursor una región en el escenario sin tener un modo de edición activo esta pasa a estar señalada con un rectángulo que abarca las casillas seleccionadas (ver figura M.51). Con una selección se pueden realizar tres acciones:

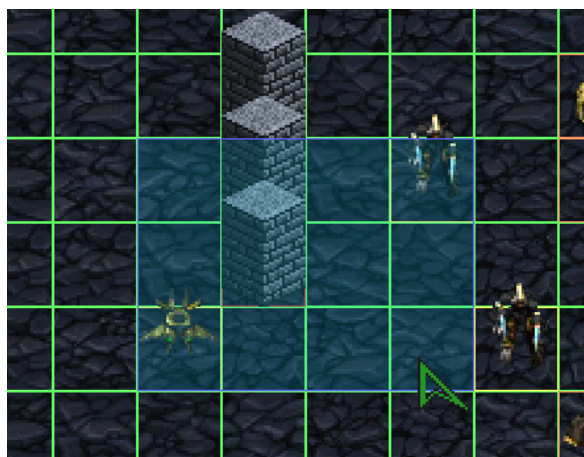


Figura M.51: Selección en el editor.

1. Eliminar la selección (ver figura M.52). Quita al ocupante de la casilla si lo tiene y la deja sin declarar, es decir, sin baldosa si la tiene.

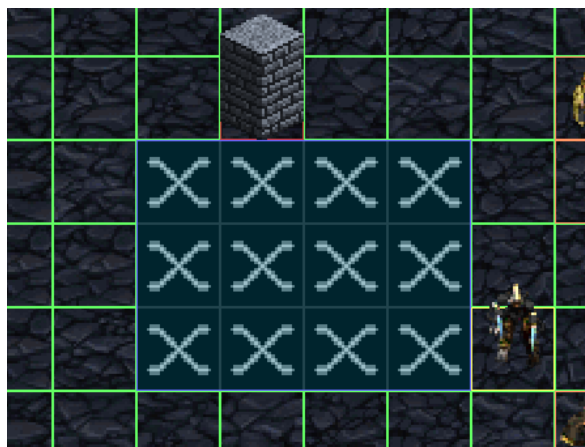


Figura M.52: Selección en el editor tras pulsar en eliminar.

2. Copiar la selección. Guarda en el portapapeles la configuración de los elementos de la selección con el tamaño de esta incluyendo ocupantes de las casillas y el tipo de baldosa de estas. En el indicador se muestra el tamaño de la configuración guardada. Esta configuración es exportable entre escenarios hasta que se sobrescriba o se cierre el juego. En el caso de copiar una selección vacía se vacía el contenido del portapapeles.
3. Pegar la selección (ver figura M.53). Repite la configuración guardada en el portapapeles en bucle hasta cubrir toda la selección actual sustituyendo lo que hubiese antes.

Cabe indicar que la selección está limitada eliminando y copiando solo lo visible según los ajustes de visibilidad.

M.10.4. Ajustes de visibilidad

“Por si el que edita el escenario no es a la vez coordinador logístico, estratega militar, decorador de exteriores y, como me llamó una vez mi tutor, experto en baldosas.”

Los *ajustes de visibilidad* (ver figura M.54) permiten al usuario decidir qué categorías de elementos se muestran y cuales no durante la edición. Estos ajustes afectan a las herramientas de selección (anexo M.10.3), de forma que al borrar o copiar la selección solo se tendrá en cuenta lo visible, aunque haya más elementos que no se están visualizando en dicho espacio.

Como se muestra en la figura M.54, gracias a tener la cuadrícula activa, hay una estructura en medio que no se está dibujando al haber casillas **naranjas**. También se



Figura M.53: Ejemplo de pegar una selección de 2x3 en un espacio de 7x8.

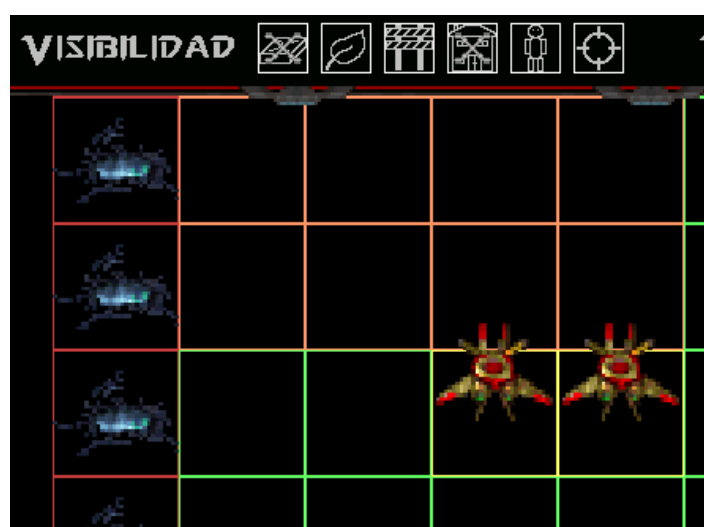


Figura M.54: Escenario siendo afectado por los ajustes de visibilidad en el editor.

aprecia que las baldosas están declaradas ya que si no serían todas de color **gris oscuro** con una cruz verde, y no podrían tener ninguna en ellas.

M.10.5. Edición de ejércitos

“Ni un fichero ‘ZIP’ está tan comprimido.”

Los recursos iniciales de los ejércitos, así como su configuración predeterminada sobre el tipo de jugador que lo va a controlar se editan desde el panel inferior del editor. La edición de entidades que pertenezcan a un ejército se crean en función del ejército actualmente seleccionado en dicho panel.

M.10.6. Edición de facciones

Encima del panel de edición de los ejércitos (anexo M.10.5) está el acceso al desplegable de las facciones. En este desplegable se pueden editar qué ejércitos componen las facciones y acceder a sus configuraciones para poder editar los nombres y objetivos de cada facción.

M.10.7. Descripción del escenario

“Por si, por lo que fuera, la vista de ejércitos, la vista de facciones, la vista de objetivos, la vista de recursos iniciales, el nombre del escenario, la vista previa del escenario y la dimensión del escenario no dieran suficiente información.”

La descripción del escenario (ver figura M.55) es un campo de texto de ocho líneas como máximo que tiene cada escenario y que permite por ejemplo identificar o explicar el propio escenario.

Esta descripción se puede cambiar desde el editor de escenarios pulsando en el botón *Mostrar descripción* (ver figura M.56).

En la descripción se permite el uso de letras²¹, números, espacios, puntos, comas y guiones. El campo de texto cambia automáticamente de línea cuando se supera la anchura máxima dividiendo la frase en el último espacio.

M.11. Problemas frecuentes

Aquí se muestra una lista de algunos de los problemas más frecuentes que pueden ocurrir utilizando Armygeddon y el cómo resolverlos:

²¹Letras ASCII solamente, la letra ‘Ñ’ no se ha introducido. <https://es.wikipedia.org/wiki/ASCII> (último acceso: 12/6/2023).

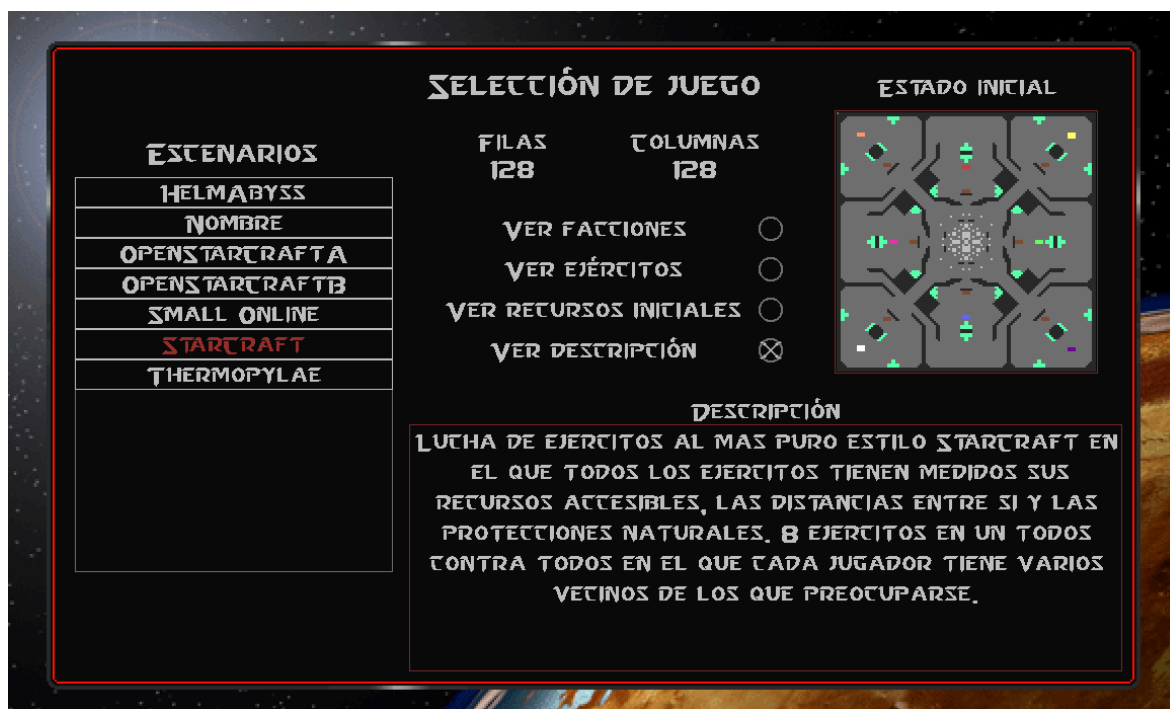


Figura M.55: Descripción de un escenario vista desde la selección de escenario individual.

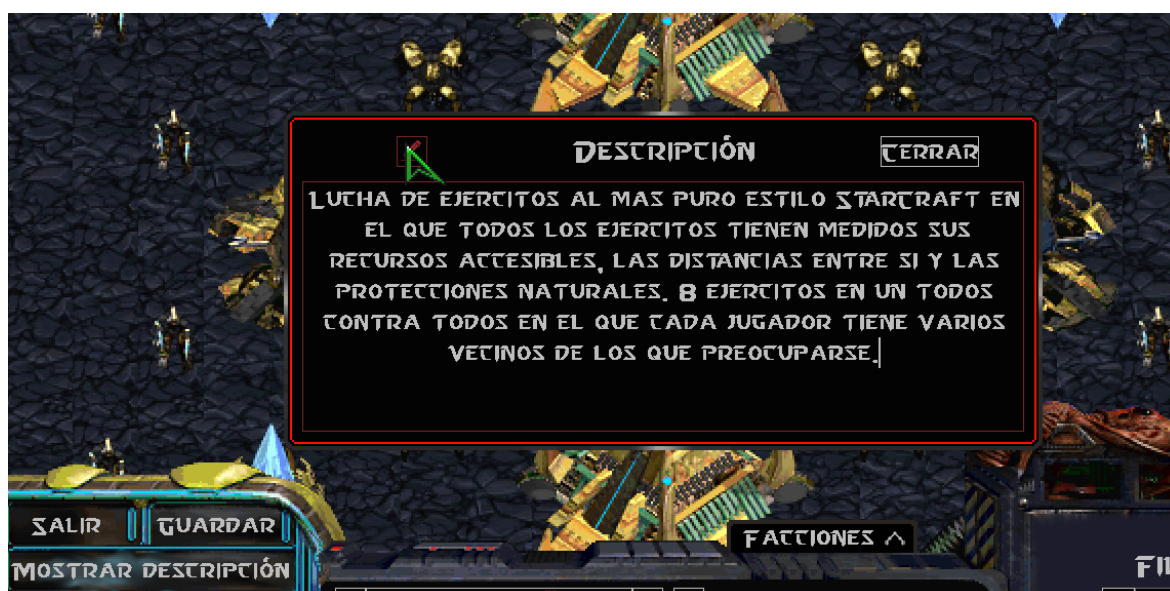


Figura M.56: Ventana emergente de edición de la descripción de un escenario vista desde el editor de escenarios.

- Falta de DLL: Al ejecutar Armygeddon faltando un archivo con formato .dll se mostrará un error similar al de la figura M.57. Se soluciona reinstalando el programa.

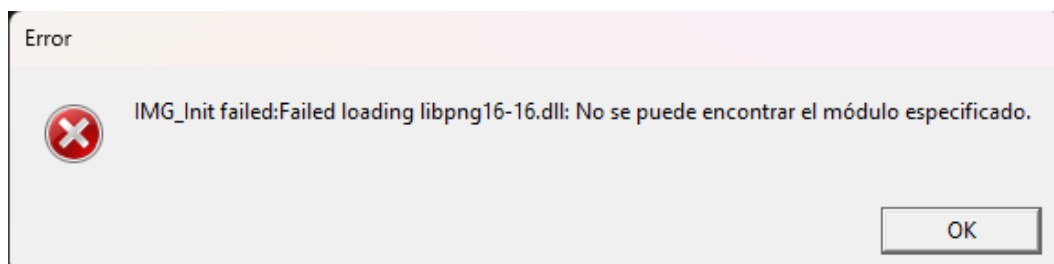


Figura M.57: Error al intentar ejecutar Armygeddon sin todos los archivos DLL.

- Falta de recurso: Al faltar un archivo recurso de Armygeddon se mostrará un error durante la carga inicial como el de la figura M.58. Se soluciona reinstalando el programa.

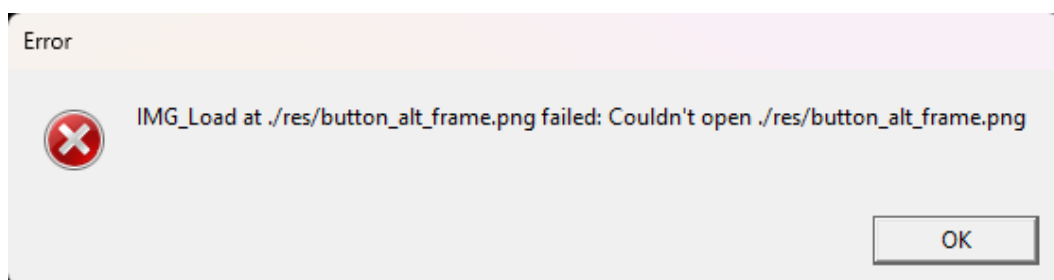


Figura M.58: Error al intentar ejecutar Armygeddon sin todos los recursos.

- Falta de carpeta base: Si falta una carpeta, aunque esta esté vacía, puede presentarse un error como el indicado en la figura M.59. Se soluciona reinstalando el programa.

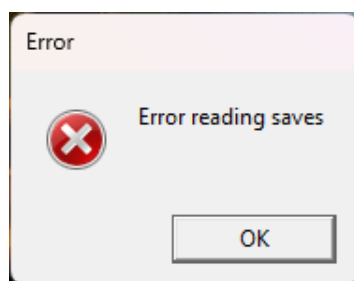


Figura M.59: Error al intentar cargar partida sin existir la carpeta './saves'.

- Se cierra el programa por el antivirus: Es posible que durante una partida en línea un antivirus lo considere una potencial amenaza y lo cierre. Se soluciona creando una excepción en el antivirus para Armygeddon.