Unidad didáctica: Programación 1º Bachillerato

Santiago Gracia Mateo

ÍNDICE

1. Introducción

La unidad didáctica de programación se corresponde con el Bloque 5 de 1º de Bachillerato y con el bloque 1 de 2º de Bachillerato. Nos vamos a centrar en 1º de Bachillerato para la elaboración de esta unidad. Dado que ya conocen los lenguajes de programación existentes y las estructuras/bloques de programación desde 4º de Eso, nos centraremos en un lenguaje de programación concreto y en la aplicación de las estructuras y bloques de programación para la asimilación de los conceptos. Esta unidad didáctica está orientada para que los alumnos de 1º de Bachillerato conozcan el entorno de Unity y tengan una pequeña introducción a la programación de aplicaciones orientadas a la Realidad Virtual, sin olvidarnos de cumplir los objetivos marcados en el currículo de 1º de Bachillerato que mencionaremos en el siguiente punto. La orientación de la unidad didáctica es única, ya que el objetivo es facilitar a los alumnos la introducción al proyecto de innovación de realizar apuntes en VR. La elección del juego a desarrollar, no fue fácil, ya que para realizarlo se tuvieron que tener en cuenta varios factores. El profesor planteo que para que los alumnos llegarán a realizar el proyecto de realizar apuntes en Realidad Virtual de 2º curso, además de conocer todos los elementos básicos de programación (sentencias, funciones, procedimientos, variables…) debían de saber realizar varias funcionalidades que posteriormente se utilizarían en 2º de Bachillerato.

2. Objetivos

Obj.TIC.5.1.1. Desarrolla algoritmos que permitan resolver problemas aritméticos sencillos elaborando sus diagramas de flujo correspondientes.

Obj.TIC.5.2.1. Escribe programas que incluyan bucles de programación para solucionar problemas que implique la división del conjunto en parte más pequeñas.

Obj.TIC.5.3.1. Obtiene el resultado de seguir un pequeño programa escrito en un código determinado, partiendo de determinadas condiciones.

Obj.TIC.5.4.1. Define qué se entiende por sintaxis de un lenguaje de programación proponiendo ejemplos concretos de un lenguaje determinado.

Obj.TIC.5.5.1. Realiza programas de aplicación sencillos en un lenguaje determinado que solucionen problemas de la vida real.

Obj.TIC.Propio. Conoce un entorno de programación y es capaz de desenvolver con soltura es el.

Obj.TIC:Propio. Es capaz de realizar pequeñas funciones y aplicaciones en realidad Virtual.

3. Contribución a la adquisición de las competencias clave

Competencias básicas Además de la competencia en Tratamiento de la información y competencia digital, en esta unidad se desarrollan las siguientes competencias:

Competencia en comunicación lingüística

En los programas complejos es habitual que los programadores incluyan comentarios clarificadores. Es importante destacar este hecho a los alumnos. Esto tiene varias ventajas:

• Por una parte, facilita la posterior modificación del programa.

• Por otra, permite a otros programadores enfrentarse con garantías de éxito a la tarea de actualización o ampliación de un programa o una parte del mismo.

Competencia matemática

En el ejemplo propuesto en esta unidad y en muchos otros ejemplos de programas es necesario aplicar conocimientos de matemáticas. Se realizan diferentes operaciones entre las variables que intervienen, aunque en muchos casos son sumas, restas, multiplicaciones y divisiones simplemente. Además de las operaciones mencionadas se realizan operaciones más complejas con vectores.

Autonomía e iniciativa personal:

Es importante enfrentarse a las tareas de programación sin miedo. Desarrollar una aplicación ofimática o una de retoque fotográfico es difícil, pero poco a poco un programador va adquiriendo destrezas que le permiten aumentar la complejidad de su trabajo. Por todo ello, aparte de un programa común que van a realizar todos los alumnos, en las últimas prácticas se les da la libertad de terminar de programar el juego a su manera, dándole mucha importancia a esta competencia.

Competencia cultural y artística:

Es una de las competencias que probablemente usando otro programa, no se hubiera visto envuelta en la Unidad didáctica, pero al crear objetos en 3D la competencia cultural y artística se ve reflejada , ya que tienen la libertad de poder crear sus propios objetos 3D para introducirlos en el juego.

4. Contenidos

En la unidad didáctica se trabajaran diversos contenidos:

- Creación de proyectos en Unity.
- Modificación de la interfaz de usuario.
- Conocimiento de las variables que se utilizan y de las propiedades de estas.
- Elaborar procedimientos y funciones utilizando las sentencias básicas de programación.

5. Metodología y secuenciación de actividades

Para trabajar en esta unidad didáctica hemos utilizado el aprendizaje basado en tareas. Se les planteará a los alumnos una serie de actividades, y en cada una de las actividades se trabajaran diferentes elementos del mundo de la programación. Las actividades, podrán realizarse independientemente una de las otras, pero el desarrollo del conjunto de todas las actividades planteadas producirá el desarrollo y elaboración de un juego de Star Wars. Cada una de estas actividades tiene su orientación práctica para que en el trascurso del curso siguiente los alumnos puedan manejarse con soltura y puedan realizar el proyecto correctamente. Para la creación de estas actividades, además de tener en cuenta que aprendan los conceptos básicos de programación, se quiso conseguir que los alumnos fueran capaces de realizar una serie de tareas para facilitar la creación del proyecto el siguiente año.

Objetivos a conseguir mediante el juego:

- Movimiento de la cámara
- Colisiones con objetos
- Interacción con la mirada
- Creación de audio
- Introducción a VR

Para todo ello seguimos las siguientes prácticas:

La primera actividad que se planteo fue la de realizar el movimiento de la cámara de Unity. En esta práctica se comienza a conocer el entorno de programación, el tipo de variables existentes en Unity y los conceptos de public y private que tan importantes son en la programación. Todo esto se seguirá usando durante el resto de la práctica, por lo que es fundamental que comprendan su funcionamiento.

La segunda actividad es continuación de la primera, en ella la cámara se moverá a varios puntos. En esta práctica se comienzan a utilizar sentencias "if" y "else" y se conocerá que es una función y del retorno de datos de esta.

Como tercera actividad se ha planteado una tarea algo más compleja. En ella los alumnos van a Destruir Naves mediante un la creación de Rayos que colisionan con estás, además de ello también se introducen las colisiones entre las naves enemigas y nuestra propia nave y la inserción de audios en el programa. Es una actividad que lleva 3 sesiones y es el núcleo del videojuego. En cuanto a conceptos de programación a adquirir en esta actividad es la creación de bucles, utilización de vectores y todas las mencionadas en las actividades anteriores.

Como cuarta actividad se introduce a los alumnos al VR y se realizan las modificaciones del programa necesarias para que este juego pueda trabajar en VR. Quizás es la menos orientada a la programación y que menos está relacionada con la Unidad didáctica, pero si queremos que el juego pueda funcionar en Realidad virtual es necesaria. En esta actividad se interactúa con la mirada, y la destrucción de las naves pasa de ser mediante click del ratón a un click con las gafas virtuales a la posición donde miras con las gafas.

Como quinta actividad se plantean una serie de mejoras al juego que no han sido implementadas ni creadas. Estás mejoras son de libre creación y cada alumno puede optar por desarrollar lo que el crea conveniente. Aun así, se les plantea una serie de

mejoras a realizar para que conozcan un poco la orientación que se le quiere dar a esta actividad.

6. Recursos y materiales.

Se informará en este apartado de las condiciones necesarias tanto en medios materiales: espacios, mobiliario, dispositivos multimedia, equipos informáticos hardware y software, herramientas (montaje de equipos), texto recomendado, biblioteca de aula y de centro, dispositivos de entrenamiento y simulación virtuales, etc., necesarios para llevar a cabo la actividad docente con mínimas garantías de éxito para la unidad didáctica

Recursos didácticos materiales:

Será necesario disponer de un aula de Informática durante todas las sesiones que se trabaje sobre esta unidad didáctica. Esta aula tendrá que estar debidamente equipada y disponer de sillas, mesas, pizarra, proyector y ordenadores. Los ordenadores tendrán que tener la suficiente potencia para trabajar de forma eficiente mediante el programa Unity y el programa Visual Studio. Finalmente mencionar que se necesitará un ordenador por alumno, ya que las tareas serán realizadas individualmente.

7. Temporalización

Las actividades serán realizadas de una forma lineal. Para la realización de las 4 actividades planteadas se invertirán sesiones y se distribuirán de la siguiente forma:

Actividades/Tiempo	Sesión 1	Sesión 2	Sesión 3	Sesión 4	Sesión 5	Sesión 6	Sesión 7	Sesión 8
Actividad Movimiento Cámara								
Actividad Movimiento Cámara (varios puntos).								
Actividad Disparo de Rayos y destrucción de Naves								
Actividad Realidad Virtual								
Actividad: Mejorando el juego y corrigiendo Bugs.								

Como se puede ver en el cuadro se solapan varias actividades, esto es debido a que durante las sesiones 5 y 6 se adelanta la siguiente actividad para aquellos alumnos que no necesiten tanto tiempo como sus compañeros, dejando que cada alumno lleve su propio ritmo de aprendizaje. Durante las 2 primeras sesiones, no se plantea ya que simplemente conocer el entorno de programación e introducirlos al mundo de la programación es capaz de ocupar gran parte de estas sesiones, dejando poco tiempo para que los alumnos sean capaces de avanzar hasta la actividad siguiente. Si se diera el caso de que algún alumno ha sido capaz de llegar a completar las 2 primeras actividades antes de terminar la segunda sesión, se le daría acceso al guíon de la siguiente actividad.

8. Evaluación

8.1. Criterios de evaluación

- Aplicar el lenguaje de programación para desarrollar aplicaciones del ámbito tecnológico.
- Manejar adecuadamente los controles de Visual Basic
- Utilizar adecuadamente los elementos multimedia en las aplicaciones.
- Manejar con soltura las estructuras de control.
- Utilizar correctamente las constantes y variables. · Aplicar las funciones y procedimientos. ·
- Conocer y utilizar adecuadamente el entorno de desarrollo de aplicaciones Visual Basic.
- Utilizar el lenguaje de programación para desarrollar aplicaciones del ámbito científico.

8.2. Calificación del alumnado

Práctica 2: Movimiento de la cámara a varios puntos: 20%.

Práctica 3: Disparo de Rayo y destrucción de Naves: 50%.

Actividad 2: Destrucción Naves Enemigas: 50% (de la práctica 3).

Actividad 3: Colisiones y quitar vida a nuestra Nave: 50% (de la práctica 3).

Práctica 5: Mejorando el juego y corrigiendo Bugs: 30%.

**Los apartados adicionales serán evaluados mediante la rúbrica, y cada uno valdrá 1/3 de la nota de esta práctica (máximo de 3 apartados adicionales).

**Si el apartado adicional requiere de mucho tiempo, puede llegar a valer el 30%, previamente consultar al profesor si consideramos que el apartado realizado puede valer más que el resto.

Todas las actividades se valorarán mediante la siguiente rúbrica:

Categoría	Excelente(3)	Bueno(2)	Mejorable(1)	Deficiente(0)
Funcionamiento (70%)	El programa y compilar y realiza el 100% de la actividad planteada.	El programa compila y al ejecutar haga entre el 80 y 99% del programa.	El programa compila y ejecuta entre el 50 y el 80% del programa.	El programa no compila o al ejecutarse cumple menos del 50% de la actividad.
Claridad y legibilidad del programa (20%)	El 100% del código es claro y legible.	El 80-99% del código es claro y legible.	El 50-79% del código es claro y legible.	El código no es claro ni legible.
Variables y nombres significativos (10%)	Utiliza nombres representativos en el 100% de las variables.	Utiliza nombres representativos en el 80-99% de las variables.	Utiliza nombres representativos en el 50-79% de las variables.	No utiliza nombres representativos.

** Si se detecta plagio en cualquiera de las partes la nota final será un 0.

8.3 Evaluación de la Unidad didáctica.

8.3.1. Métodos de evaluación de la Unidad didáctica

La evaluación de la Unidad didáctica se realizará de 3 formas:

- Implicación y motivación del alumnado durante la realización de las actividades.
- Coloquio al finalizar las actividades
- Encuesta realizada por Google Forms.

El primera forma de evaluar la Unidad didáctica será mediante la observación en las propias clases. Allí veremos si los alumnos realmente se están implicando con la creación del juego y si todos ellos prestan atención y realmente quieren aprender programación. Aspectos como el estar puntuales en clase para comenzar las actividades y que sigan realizando la actividad aunque suene el timbre de cambio de clase pueden ser 2 buenos indicadores de que los alumnos se han involucrado con la actividad y les motiva lo que están realizando.

La segunda forma de evaluar es mediante la realización de un coloquio en clase cuando terminen de realizar las actividades y hayan entregado el trabajo. En esa sesión expondrán su punto de vista de las actividades realizadas, y comentarán que aspectos de las actividades les han resultado más difíciles y cuales les han resultado más útiles para la realización de las actividades.

Finalmente la tercera forma de evaluar la unidad didáctica será mediante un formulario creado con Google form en el cual se realizarán varias preguntas de respuesta corta y otras de ellas relacionadas con la dificultad de la actividad y otros factores.

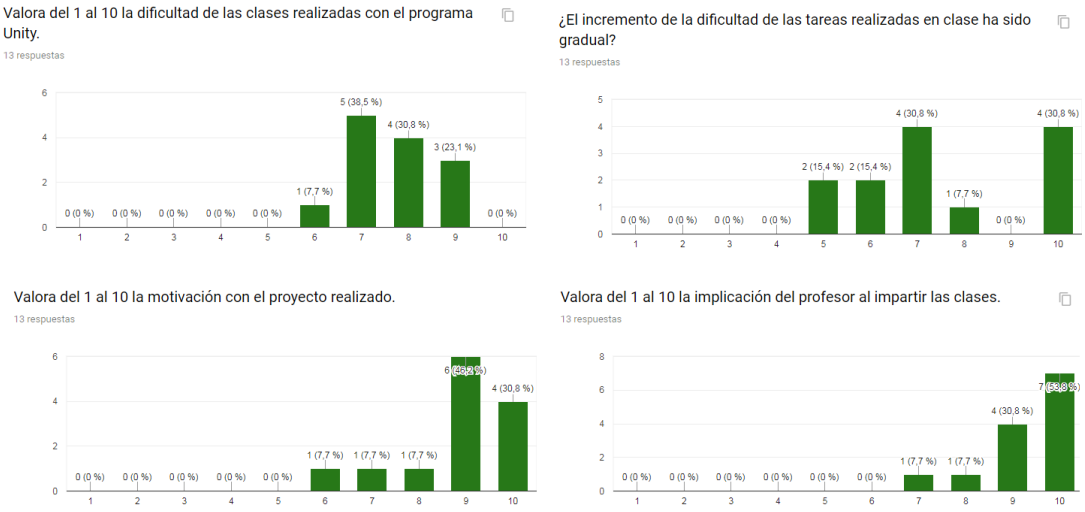
8.3.2. Resultados y conclusiones de la evaluación de la Unidad didáctica.

Las impresiones al realizar la evaluación de esta Unidad didáctica han sido muy satisfactorias. Por un lado se ve que la implicación de los alumnos durante la Unidad didáctica ha sido realmente alta, y la motivación en la realización del proyecto también ha seguido la misma línea. Por otro lado, aunque se ha conseguido enganchar a los alumnos y motivarles desde el inicio, viendo los comentarios recogidos en el coloquio de la última clase y de los resultados obtenidos en el Google Form, habría que plantear muchos cambios a la hora de volver a realizar esta Unidad didáctica.

Por un lado, los materiales y el proyecto a realizar en la Unidad didáctica probablemente se mantendrían, ya que la temática de Star Wars ha gustado mucho y las prácticas realizadas han seguido un incremento gradual de dificultad. Los principales cambios vendrían a la hora de dar las explicaciones y el guiado de la actividad. En vez de pasar directamente en la práctica 1 de mover la cámara, hubiera sido más fructífero

realizar primero un análisis del "Roll a ball" que hicieron antes de comenzar con esta unidad. Esto es debido a que mediante ese paso a paso no llegaron a comprender lo que realmente estaban haciendo y se dedican a copiar y pegar código y ver si funciona o no. Por otro lado, también habría que cambiar la orientación de estas prácticas en cuanto a la programación. Durante todas ellas, solo nos hemos centrado en el código y se les da dado todos los objetos 3D y los audios insertados en el programa, provocando que no salieran de ver código. Aunque sea una Unidad para aprender programación, descargar un poco añadiendo elementos visuales e interactuando con el programa podría haber ayudado a hacerles las clases más amenas.

Mediante la herramienta Google Form hemos obtenido los siguientes resultados:



Como podemos ver mediante estos resultados corroboramos lo visto en el punto anterior.

Algunos de los comentarios más relevantes sacados del Google Form, son los siguientes:

Me habría gustado que se hubiera podido avanzar según el ritmo de cada estudiante pero sé que es difícil o casi imposible.

En mi opinión creo que la explicación ha sido complicada, porque era un tema nuevo y nos costaba asimilar la información.

Habría empezado por algo más fácil

Igual tendríamos que a ver explicado un poco más antes de realizar las prácticas, pero tampoco ha estado mal porque no teníamos muchas clases para hacer mucho más.

Ha sido algo nuevo para nosotros ya que unity no lo hemos utilizado hasta ahora y a mi parecer usted lo ha hecho muy bien para ayudarnos en el nuevo programa, haciendo un juego conocido, fácil y práctico.

En general me han gustado bastante las prácticas, el profesor ha manejado bastante bien la presión del grupo con tantas dudas y he aprendido que para hacer algo en unity tienes muchas opciones y lo mejor es buscar en la página la función que necesitamos y ver cómo funciona.

Por todo expuesto anteriormente, concluimos diciendo que la Unidad didáctica ha sido un éxito, pero eso no implica que la próxima ver que se imparta no vuelva a tener una reestructuración completa para corregir los fallos y dificultades que los alumnos se han encontrado en la Unidad didáctica.

9. Anexos

Práctica 1: Movimiento de la cámara

Actividad 1: Movimiento de la cámara a un punto determinado.

1º Descargar la carpeta completa del siguiente enlace:

Proyecto 1º Bachillerato.

2º Creamos un Objeto vacío en la posición (0, 0, 1000). Se llamará Punto1

3º Completar el código de la función moverCamara del ScriptNavePrincipal .

El código tendrá que ser del estilo al siguiente:

```
Vector3 direccion = posicionCamara – posicionObjetoVacio;

direccion = direccion.normalized;

posicionCamara = posicionCamara + direccion * velocidad;
```

4º Se llamará a la función moverCamara() desde la función Update().

El script "ScriptNavePrincipal" esta añadido a la cámara por lo que si queremos acceder a sus componentes (Por ejemplo posición), tendremos que escribir **this.transform.position o transform.position.

Si queremos acceder a las variables (por ejemplo la posición) del punto vacío que hemos creado (Punto1), tendremos que pasarlo declarando una variable como public GameObject objetoVacio. Esta vez, en vez de utilizar "this." tendremos que utilizar la variable "objetoVacio"→ **objetoVacio.transform.position

**velocidad es una variable que se tendrá que crear como public y float que podrá ser modificada desde fuera del código.

Práctica 2: Movimiento de la cámara a varios puntos

Actividad 1:

En esta actividad vamos a conseguir que la cámara se mueva a varios puntos.

1º Se crean 3 Objetos vacíos como en la práctica anterior. Los llamamos Punto2, Punto3 y Punto 4.

2º Se cambian las coordenadas de los puntos.

Punto1 a la posición (0,0,250).

Punto2 a la posición (0,0,500).

Punto3 a la posición (0,0,750).

Punto4 a la posición (0,0,1000).

3º Vamos a modificar la función moverCamara de Script "`ScriptCamara`" para que pase por diferentes puntos.

Utilizaremos un if, para saber si se ha llegado al siguiente punto.

El código sería el siguiente:

```
private bool moverCamara(){
     if (posicionCamaraEnZ       >=       posicionDelPuntoEnZ) {
          sumamos 1 a un contador de puntos.
```

```
            Cogemos el siguiente punto y se lo pasamos al
            objetoVacio, para que ahora nuestro objetoVacio sea
            el Punto2, luego el Punto3;
      }
      //Cógido práctica 1 funcion moverCamara().
}
```

4º Si llegamos al último punto, queremos que la Nave se detenga.

Para que no se mueva, tenemos que hacer que la función moverCamara() no se ejecute más veces. Para ello creamos una variable bool (que representa un booleano) y cuando moverCamara nos devuelva un false, pararemos el programa.

```
void Update () {
        if (movemosCamara==true) {
            movemosCamara=moverCamara();
        }
   }
```

5º Hacer que moverCamara() devuelva false(para que se pare el movimiento de la cámara.

```
private bool moverCamara(){
      if(No              hay              más              Puntos){
           return false;
      }else{
            Código punto 3 de esta práctica.
      }
}
```

Práctica 3: Disparo de Rayos y destrucción de Naves

Actividad 1: Disparo de rayos

El código esta hecho en la práctica, así que vamos a intentar explicar que hacen las diferentes funciones.

Physics.Raycast:

Returns

bool True when the ray intersects any collider, otherwise false.

Description

Same as above using `ray.origin` and `ray.direction` instead of `origin` and `direction`.

This example draws a line along the length of the Ray whenever a collision is detected:

```
using UnityEngine;

public class ExampleClass : MonoBehaviour
{
    void Update()
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;

        if (Physics.Raycast(ray, out hit, 100))
            Debug.DrawLine(ray.origin, hit.point);
    }
}
```

RaycastHit

struct in UnityEngine / Implemented in:UnityEngine.PhysicsModule

Leave feedback Other Versions

Description

Structure used to get information back from a raycast.

See Also: Physics.Raycast, Physics.Linecast, Physics.RaycastAll.

Properties

barycentricCoordinate	The barycentric coordinate of the triangle we hit.
collider	The Collider that was hit.

Código práctica:

```
private void shoot(){

        if(Input.GetMouseButtonDown(0)){

            RaycastHit hit;
            Ray ray= oculusCam.GetComponent<Camera>().ScreenPointToRay(Input.mousePosition);

            lanzaRayo ();
            sonidoLaser ();

            if(Physics.Raycast(ray,out hit,range)){
                if (hit.collider.gameObject.GetComponent<ScriptNaveEnemiga> () != null) {
                    hit.collider.gameObject.GetComponent<ScriptNaveEnemiga> ().hit (1);
                }
            }
        }
    }


   private void lanzaRayo(){
           Vector3 posicionLaser = oculusCam.GetComponent<Camera>().transform.position;
           GameObject g =    Instantiate (rayoLaser, posicionLaser,Quaternion.identity);
    }




private void movimientoLaser(){
        Vector3 direccion = GameObject.Find ("CenterEyeAnchor").GetComponent<Camera> ().ScreenPo
intToRay (Input.mousePosition).direction;
        direccion = direccion.normalized;
        this.transform.position = this.transform.position + (direccion * 10);
    }
```

¿Que hemos conseguido con la función “lanzaRayo”?

¿Qué hemos conseguido con la función “shoot”?

¿Qué hace la función “movimientoLaser” ? (Esta en el Script “CrearRayos”).

Actividad 2: Destrucción Naves Enemigas.

En esta actividad vamos a destruir las naves Enemigas.

1º Para destruir Naves, primero tenemos que crearlas.

Para ello, vamos a descomentar (quitar “//”) la parte del código que va a crear las naves Enemigas.

Entramos en el Script: “Spawn” y quitamos “//” de la línea: `StartCoroutine (rutinaSpawn());`

2º Vamos a modificar el Script asociado a las Naves Enemigas que se crean.

Para ello vamos a buscar en el Script que está asociado a las Naves Enemigas.(Las naves Enemigas están guardadas en Prefabs) y abrimos el Script.

Vemos que en el Script, nos encontramos la función “hit”, la cual se va a ejecutar, cada vez que a una NaveEnemiga le alcance un rayo.

```
public void hit(int shoot){
    updateLive(live - shoot);
}
```

```
private void updateLive(int l){
  live= l;//Cada vez que se le dispara a una Nave enemiga, entra en updateLive(esta función)
          //y directamente se le resta uno a live (por lo que live valdrá live = live -1 ;


    if (true) {//¿Cuando queremos que entre por este if?  Cuando  su  vida  sea  0!
        //Aquí se tiene que escribir un código para destruir la Nave cuando se queda con la
        vida  a 0!
    } else if (true) {//¿Cuando queremos que entre por este if?  Cuando  su  vida  sea  2.
        //Aquí se tiene que escribir un código para cambiar el color de las naves enemigas
        //cuando su vida sea 2!
    }


}
```

Actividad 3: Colisiones y quitar vida a nuestra Nave

En esta actividad vamos a destruir las naves Enemigas cuando colisionen con nuestra Nave. Para ello utilizaremos la función "OnTriggerEnter" que será añadida al Script de nuestra Nave Principal.

1º Comprobar que se han añadidos los Box Collider:

Para realizar esta actividad comprobaremos que las NavesEnemigas y nuestra NavePrincipal, tengan creado el componente "Box Collider".

- Las naves Enemigas están en la carpeta Prefabs.
- Nuestra Nave está asociada a la Camara y su nombre es "y-wing".
-

2º Modificación de la función para destruir las Naves Enemigas.

Para destruir las Naves Enemigas, introduciremos el código en el Script " ScriptNavePrincipal" y modificaremos la función OnTriggerEnter (Collider other).

```
  void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag ("NaveEnemiga")){
    //Aquí se tiene que crear un código para destruir las Naver enemigas que colisionen con
    nuestra nave.
```

```
                GameObject explosion = Instantiate (prefabExplosion);

                explosion.transform.position = posicionNaveEnemiga;

                Destroy (explosion, 3);

        }
    }
```

3º Quitar vida a nuestra Nave por cada colisión

La vida de nuestra Nave, la vamos a representar con un contador y unas imágenes que están añadidas en el Canvas. Cada vez que colisionemos con una Nave Enemiga, restaremos 1 al contador y deshabilitaremos una de las imágenes.

El contador empezará en 5, ya que tenemos 5 imágenes.

Para recuperar las imágenes del Canvas usamos la siguiente función:

```
GameObject go = GameObject.Find("Canvas");
imagenes  =  go.GetComponentsInChildren<Image>();
```

La función para deshabilitar las imágenes es la siguiente:

imagenes [contador].enabled = false;

Así que nuestro código final quedaría así:

```
void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag ("NaveEnemiga")){


                //Destruyo Nave Enemiga que colisiona

                GameObject explosion = Instantiate (prefabExplosion);

                explosion.transform.position = posicionNaveEnemiga;

                Destroy (explosion, 3);

                //Deshabilito la imagen que se corresponda con el contador.

                //Resto vida a nuestra nave(contador).
```

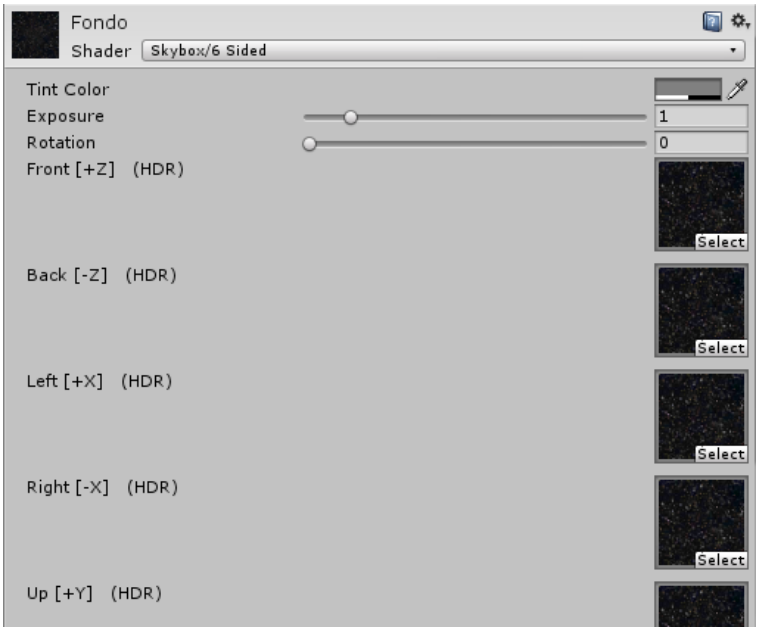
```
        }
    }
```

Práctica 4: Realidad Virtual

¿Qué elementos de nuestro juego incorporan Realidad Virtual? ¿Cuáles de ellos no funcionarían?

Cámara de las Gear → Es el primer elemento para introducir nuestro juego en la realidad virtual. (Es un prefabs que viene en el package "OculusUtilities.unitypackage").

SkyBox→ Es un material formado por 6 imágenes



Actividad 1: Modificar elementos que no funcionarían en las Gafas de VR.

El resto de elementos no mencionados, corresponden a un juego 3D normal. Vamos a modificar los que no funcionen en nuestras gafas de realidad virtual!

1º Modificación creación de Rayo Laser

El Rayo que se crea para 3D, en las gafas de VR no funciona, habría que cambiarlo.

El rayo que creamos en la práctica anterior (RaycastHit) depende de donde pulsemos con el rato, y de la función ScreenPointToRay, que no parece funcionar en VR).

Código antiguo:

```
RaycastHit hit;
Ray ray= oculusCam.GetComponent<Camera>().ScreenPointToRay(Input.mousePosition);
```

Ahora vamos a trabajar con la rotación de la cabeza (y recuperaremos la dirección donde miremos con las gafas).

Código nuevo:

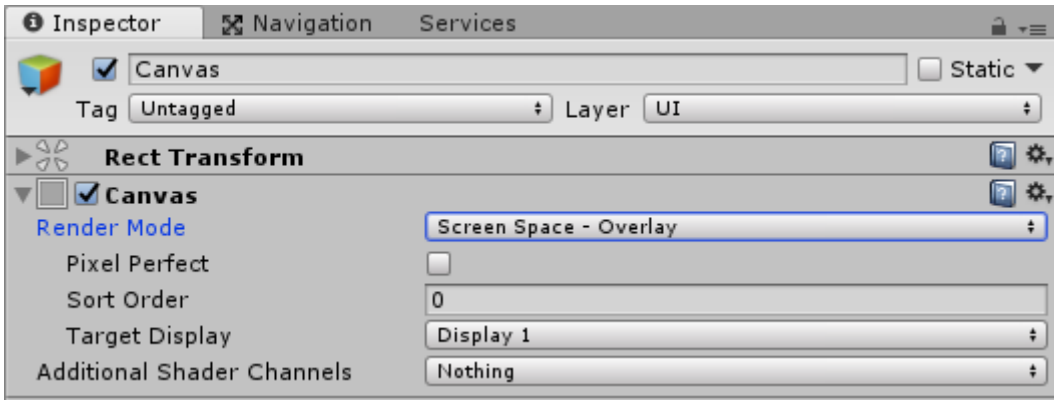
```
RaycastHit hit;

Quaternion headRotation = InputTracking.GetLocalRotation (XRNode.Head);//Rotacion de la cabeza

Vector3 dir eccion= (posicionDeLaCamara + (headRotation * Vector3.forward) * 2000) – posicionDeLaCamara

Ray ray = new Ray ();

ray.origin = posicionDeLaCamara.

ray.direction = dir;
```

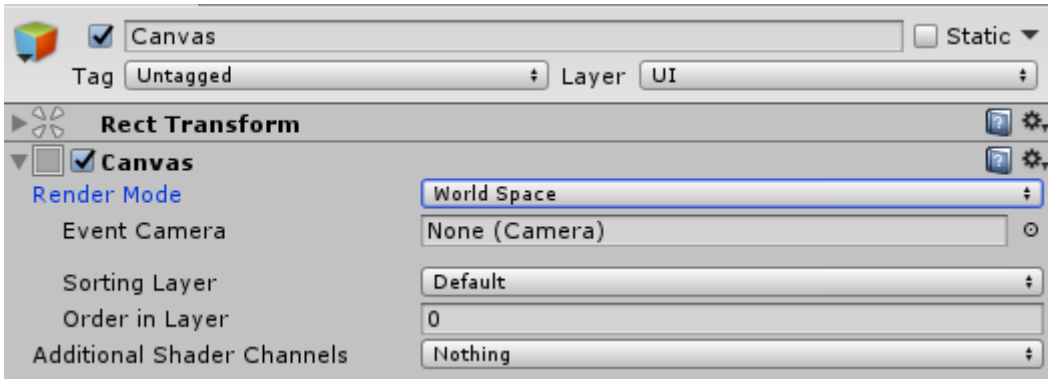
2º Modificación movimiento del laser:

Para modificar el movimiento del laser, tenemos que cambiar el código que hemos creado en el Script "CrearRayos" por un código similar al utilizado en el apartado anterior. La función a modificar es: `movimientoLaser();`

3º Modificación del Canvas.

El canvas que contiene las imágenes, tampoco funciona en VR. Para conseguir que funcione, buscamos el objeto y cambiamos el Render Mode de "Screen Space – Overlay" a "World Space". Tras ello lo modificamos(cambiamos su posición) para que se vea en nuestra pantalla.





Práctica 5: Mejorando el juego y corrigiendo Bugs.

Esta actividad es libre, cada uno modificara y mejorará lo que crea conveniente del juego. Antes de empezarla, guardar una copia del juego realizado, por si al quitar/poner objetos y funciones, rompemos algo que estaba funcionando.

Algunas ideas de mejora del juego:

- Modificar el movimiento de la cámara para que vaya a los puntos mediante la función Vector3.MoveTowards y que cambie de punto al llegar a ese sin depender solo de Z.
- Añadir al Canvas un contador de Naves Destruidas.(Recuerda el Roll a Ball).
- Corregir el error por el cual se siguen deshabilitando imágenes de la vida de la Nave. (Provocando un NullPointException).
- Añadir una nueva Escena cuando se Destruya nuestra Nave y poner el texto "Game Over"
- Crear una escena final en la que se destruya nuestra Nave y escribir "Game Over".

- Añadir un botón de Reinicio al juego. (Botón volver de las gafas).
- Introducir al juego otros objetos 3D (Por ejemplo la Estrella de la muerte), podéis utilizar la página https://free3d.com/3d-models/ .
- Cambiar las explosiones de las Naves.