Universidad Zaragoza

Trabajo Fin de Grado

Exploración de superficies cuádricas como primitivas de renderizado

Exploring quadric surfaces as rendering primitives

Autor

Adrián Samatán Alastuey

Director

Adolfo Muñoz Orbañanos

Grado en Ingeniería Informática

ESCUELA DE INGENIERIA Y ARQUITECTURA

2020

Exploración de superficies cuádricas como primitivas de renderizado

Resumen

La malla de triángulos es la estructura de datos más utilizada para representar geometrías arbitrarias en informática gráfica, ya que es muy sencilla y además cuenta con algoritmos sofisticados y soporte hardware. Sin embargo, no está exenta de problemas, ya que al ser una discretización de la geometría real, si no se utilizan suficientes triángulos es posible apreciar el contorno de éstos en las figuras renderizadas y en las sombras de las mismas. No obstante, su principal limitación es que cada triángulo, por ser una superficie plana, tiene una única normal a todos los puntos de su superficie, y dado que el cálculo para generar la apariencia de la figura depende de esa dirección normal, aparecen discontinuidades de iluminación entre triángulos contiguos con diferente normal. Para evitar esto, se utilizan normales de *shading*, calculadas mediante interpolación entre normales asignadas a cada uno de los vértices de los triángulos. De esta manera se consigue una apariencia suave, más realista y ajustada a lo que la malla de triángulos estaría representando. Al no ser esta normal de *shading* la normal geométrica real de la figura, pueden aparecer artefactos en ángulos rasantes (*grazing angles*) en la superficie. Esto sucede al utilizar una normal distinta a la normal geométrica para calcular la apariencia de una figura utilizando técnicas basadas en el algoritmo de trazado de rayos (*ray tracing*), pudiendo los rayos entrar en la geometría y causar artefactos.

En este trabajo se presentan tres métodos para sustituir los triángulos presentes en una malla de triángulos por una nueva primitiva de renderizado, los triángulos cuádricos. Éstos están modelados como superficies de segundo grado, por lo que la normal varía en toda su superficie sin necesidad de interpolar. En su construcción se garantiza mantener la estructura original de la malla de triángulos, consiguiendo también generar una superficie con una normal geométrica proporcional a las normales de shading utilizadas en el triángulo original. De esta manera se eliminan artefactos causados por el uso de una normal de *shading* distinta a la geométrica real de las figuras, consiguiendo un contorno suave por utilizar superficies de segundo grado y manteniendo la apariencia otorgada a la malla de triángulos por las normales de *shading*.

Índice general

Capítulo 1

Introducción

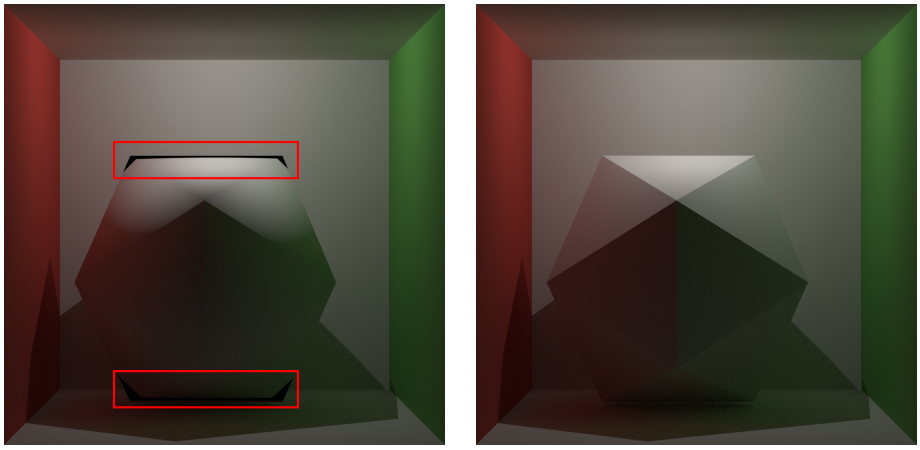
1.1. Motivación

Tradicionalmente los algoritmos de generación de imágenes sintéticas (renderizado) han representado la geometría de los objetos principalmente mediante mallas de triángulos (colecciones de triángulos y vértices que aproximan una superficie en 3D). El uso de estas mallas conlleva muchas ventajas, ya que permiten modelar geometrías arbitrarias fácilmente, son eficientes por utilizar el polígono más simple posible, cuentan con soporte en cualquier renderizador, algoritmos sofisticados y soporte hardware, pero también tiene una serie de limitaciones. Su principal limitación tiene que ver con que cada triángulo, al ser una superficie plana, tiene una única normal a la superficie, y dado que el cálculo de la iluminación depende de esa dirección normal, aparecen discontinuidades de iluminación entre triángulos contiguos con diferente normal (ver figura 1.1a). Para evitar esto, se utilizan normales de *shading*, calculadas mediante interpolación entre normales asignadas a cada uno de los vértices. De esta manera se consigue una apariencia suave, más realista y ajustada a lo que la malla de triángulos estaría representando. La figura 1.1b ilustra este efecto.

(a) Esfera con normales geométricas. *(b) Esfera con normales de* shading.

Figura 1.1: Al usar normales de shading *se consigue una apariencia suave, al contrario que usando las normales de los triángulos.*

Sin embargo, por no ser esta normal de *shading* realmente la normal geométrica del objeto, pueden aparecer artefactos en ángulos rasantes (*grazing angles*) en la superficie. Esto sucede al utilizar una normal distinta a la normal geométrica para calcular la apariencia de un objeto utilizando técnicas basadas en el algoritmo de trazado de rayos (*ray tracing*), explicado en el capítulo 3, pudiendo los rayos entrar en la geometría y causar artefactos como ilustra la figura 1.2.

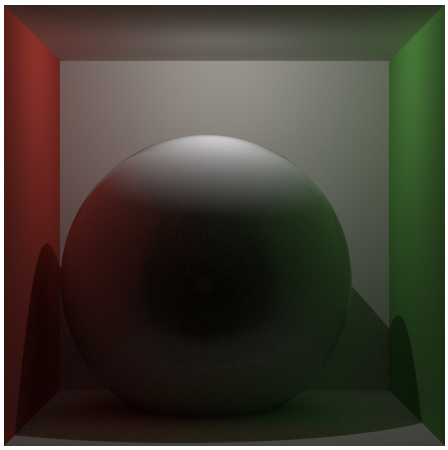


(a) Icosaedro con normales de shading. *(b) Icosaedro con normales geométricas.*

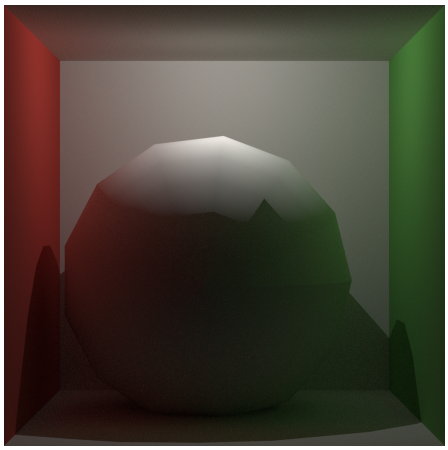
Figura 1.2: Las normales de shading *pueden causar artefactos en ángulos rasantes (izquierda) debido a la diferencia entre la normal de* shading *(que indica que el rayo reflejado sale del objeto) y la normal geométrica (que indica que el rayo reflejado entra dentro del objeto). Si ambas dos normales coinciden (derecha) esos artefactos desaparecen.*

Además, al tratarse de una discretización de la geometría real del objeto a modelar, si la geometría real es curva y suave y no se utilizan suficientes triángulos, al ser éstos planos no se consigue obtener la apariencia curva deseada, pudiendo apreciarse el contorno formado por los triángulos como se puede ver en la figura 1.3.

En este trabajo se resuelven estos dos problemas, eliminando artefactos causados por uso de normales de *shading* y consiguiendo una apariencia suave de la geometría. Para ello se propone sustituir la malla de triángulos por una malla de *triángulos cuádricos*, modelados por diversas superficies cuádricas. De esta manera se convierten geometrías ya modeladas por mallas de triángulos, consiguiendo que la normal geométrica coincida con la normal de *shading*, eliminando por tanto los artefactos en ángulos rasantes y, puesto que las superficies cuádricas son suaves, eliminando el aspecto en el contorno causado por utilizar insuficientes triángulos en la discretización de la geometría.



(a) Esfera modelada mediante ecuación implícita (geometría esférica real)



(b) Esfera modelada como malla de triángulos con insuficientes triángulos.

Figura 1.3: La esfera de la izquierda, modelada mediante una ecuación implícita, tiene una apariencia evidentemente suave. Al discretizarla mediante triángulos (derecha) aparecen artefactos evidentes tanto en su contorno como en las sombras que proyecta.

1.2. Trabajo previo

Existen varias publicaciones donde se han identificado y propuesto soluciones a los problemas expuestos en este trabajo. [BA08] expone el problema del aspecto triangular en el contorno de mallas de triángulos, proponiendo un método para suavizar los triángulos en un contexto de renderizado en tiempo real, es decir, sin iluminación global ni trazado de rayos. Sin embargo no se consigue obtener continuidad en C^1, siendo ésta una de las propiedades deseables a conservar tal y como se describe en el capítulo 4. En este trabajo se intentará resolver el mismo problema en un contexto de renderizado *offline*, para visualizarse mediante trazado de rayos con iluminación global. [RSM10] expone el problema surgido al utilizar normales de *shading* distintas a las normales geométricas, proponiendo modificar las normales de *shading* por otras que tengan en cuenta los ángulos en los que aparecen los artefactos para que nunca se entre en la geometría del objeto a renderizar. De esta manera se conserva el aspecto triangular en el contorno de los objetos. En este trabajo se proponen soluciones a ambos problemas a la vez, por medio de triángulos cuádricos.

1.3. Objetivos y tareas

El objetivo de este trabajo es la exploración de métodos para transformar una malla de triángulos a una malla de triángulos cuádricos, conservando posiciones de los vértices originales y manteniendo la normal de *shading* como

nueva normal geométrica, intentando a su vez que se pueda introducir en *pipelines* tradicionales de renderizado fácilmente. Para ello se analizarán, en paralelo, dos soluciones al problema, por una parte la obtención de ecuaciones paramétricas para modelar los triángulos cuádricos en base a ecuaciones generadoras y por otra la obtención de una ecuación implícita para modelar los triángulos cuádricos como cuádricas implícitas. Ambas soluciones poseen cualidades que serán exploradas en el capítulo 4. Las representaciones implícita y paramétrica utilizadas en geometría serán explicadas en el capítulo 2.

Las tareas a realizar son las siguientes:

- Revisión de bibliografía existente: lectura de trabajo previo relacionado, familiarización con superficies cuádricas y renderizador a usar.
- Diseño iterativo de una solución paramétrica e implícita: experimentación de distintas soluciones, validando cada una mediante los resultados obtenidos por cada solución, mejorándose hasta obtener una solución satisfactoria que resuelva los problemas propuestos.
- Experimentación numérica: exploración de posibles nuevas ideas como resultado del trabajo realizado con los modelos propuestos y la revisión de la bibliografía existente.
- Integración de las dos soluciones geométricas en un renderizador: implementación de los modelos en un renderizador para generar imágenes con iluminación global.
- Generación de imágenes con iluminación global para visualizar la resolución a los problemas expuestos y exploración de resultados, analizando las imágenes generadas con los distintos modelos y comparando tiempos de renderizado entre ellos.

1.4. Organización del documento

1. Introducción: este capítulo, motivando el trabajo realizado, haciendo una revisión del trabajo previo y detallando las tareas realizadas.
2. Geometría: introducción a las formas geométricas, conceptos relacionados con informática gráfica como normales o mallas de triángulos y una introducción a las cuádricas y triángulos cuádricos, sobre los que se fundamenta el resto del trabajo.
3. Trazado de rayos: introducción al algoritmo trazado de rayos para generar iluminación global.
4. Transformación de triángulos planos a triángulos cuádricos: explicación en detalle sobre los distintos modelos propuestos para triángulos cuádricos, definiciones matemáticas de los mismos, cómo resuelven los problemas asociados a las mallas de triángulos expuestos y cómo pueden ser integrados en un renderizador.

5. Resultados: visualización de imágenes generadas en un renderizador con iluminación global, comprobación de solución a los problemas expuestos y comprobación de restricciones descritas en el capítulo 4 y comparativa entre los distintos modelos.

6. Conclusiones: valoración sobre el trabajo realizado según los objetivos propuestos, posible trabajo futuro tomando como base el trabajo realizado y valoración personal.

Capítulo 2

Geometría

En este capítulo se hace una introducción a las figuras geométricas, cómo son usadas en informática gráfica sus formas implícita y paramétrica, conceptos relacionados con informática gráfica como normales o mallas de triángulos para poder entender las causas de los problemas que este trabajo intenta resolver y por último cuádricas y triángulos cuádricos y cómo modelarlos, ya que servirán como base de los modelos explicados en el capítulo 4.

2.1. Definición

Una figura geométrica es la forma de un objeto o su límite exterior, contorno o superficie externa, en contraposición a otras propiedades como su color, textura o tipo de material.

Existen tres representaciones fundamentales para definir una figura geométrica. Éstas son la forma implícita, forma paramétrica y forma explícita. A continuación se expandirá sobre las formas implícitas y paramétricas, ya que son las más comúnmente usadas en informática gráfica y son la base de los modelos explorados en este trabajo.

2.2. Tipos de representación

Una **ecuación implícita** es una relación de la siguiente forma:

$$f_i : \mathbb{R}^n \to \mathbb{R}, f_i(\mathbf{p}) = 0 \tag{2.1}$$

En geometría y en informática gráfica, siendo $n = 3$ para un espacio euclídeo tridimensional, se puede utilizar para representar una figura geométrica concreta,

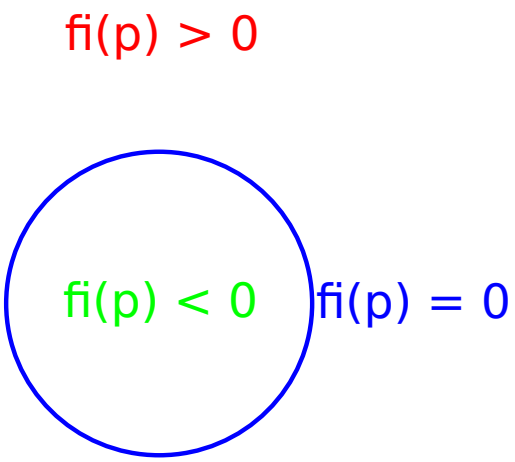


Figura 2.1: Puntos pertenecientes a la figura (verde), en la superficie (azul) o fuera de la figura (rojo).

siendo los puntos pertenecientes a ella aquellos que cumplan $f_i(\mathbf{p}) \leq 0$, los puntos pertenecientes a la superficie los que cumplan $f_i(\mathbf{p}) = 0$, los puntos no pertenecientes a la figura $f_i(\mathbf{p}) > 0$ y siendo todos los puntos del espacio $\mathbf{p}$, como se puede observar en la figura 2.1. Al estar definida para todo el espacio euclídeo $\mathbb{R}^3$, algunas superficies implícitas pueden extenderse hasta el infinito. Como ejemplo, una esfera con $r = 1$ centrada en el punto del espacio $(0, 0, 0)$ se puede expresar con la siguiente ecuación implícita:

$$f_i(x, y, z) = x^2 + y^2 + z^2 - 1 = 0 \tag{2.2}$$

Una **ecuación paramétrica** define un grupo de cantidades como funciones de una o más variables independientes llamadas parámetros:

$$f_p : \mathbb{R}^m \to \mathbb{R}^n \tag{2.3}$$

En geometría se pueden utilizar para representar la superficie de una figura geométrica, definiéndose como todos los puntos $\mathbf{p} \in \mathbb{R}^3$ generados desde un espacio $\mathbb{R}^2$ que puede estar acotado o no, como se puede ver en la figura 2.2. La esfera implícita definida en la ecuación (2.2) puede expresarse también de manera paramétrica, de la siguiente forma:

$$(0 \leq \theta \leq \pi, 0 \leq \phi < 2\pi)$$
$$f_p(\theta, \phi) = \mathbf{i}(x_0 + \sin\theta\cos\phi) + \mathbf{j}(y_0 + \sin\theta\sin\phi) + \mathbf{k}(z_o + \cos\theta) \tag{2.4}$$

Teniendo una figura geométrica $G = \mathbf{p_0}, ..., \mathbf{p_n}$ definida de cualquier manera, el problema de **implicitación** consiste en obtener una función f implícita tal que:

$$\forall \mathbf{p} \in G, f_i(\mathbf{p}) = 0 \wedge \forall \mathbf{p} \notin G, f_i(\mathbf{p}) \neq 0 \tag{2.5}$$

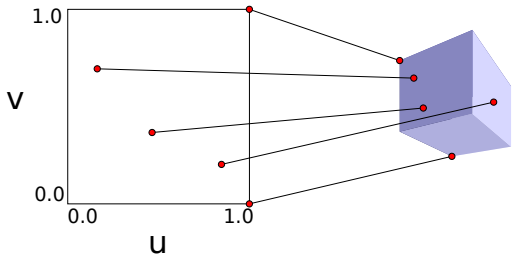


Figura 2.2: Los puntos de la figura son generados desde un espacio uv acotado según una función arbitraria f_p.

2.3. Normal a la superficie

La normal a la superficie (*surface normal*) en un punto $\mathbf{p}$ de la superficie es el vector ortogonal a la superficie (específicamente, a su plano tangente).

La normal de la ecuación implícita (2.1) en un espacio euclídeo tridimensional es la siguiente:

$$\nabla f_{i_n}(\mathbf{p}) = \frac{\nabla f_i(\mathbf{p})}{||\nabla f_i(\mathbf{p})||}$$
$$n(\mathbf{p}) = \nabla f_{i_n}(\mathbf{p}) \tag{2.6}$$

La normal de la ecuación paramétrica (2.3) en un espacio euclídeo tridimensional, con $m = 2$ y $n = 3$, es la ortogonal a sus tangentes:

$$t_u(u,v) = \frac{\partial f_p}{\partial u}(u,v)$$
$$t_v(u,v) = \frac{\partial f_p}{\partial v}(u,v)$$
$$n(u,v) = t_u(u,v) \times t_v(u,v) \tag{2.7}$$

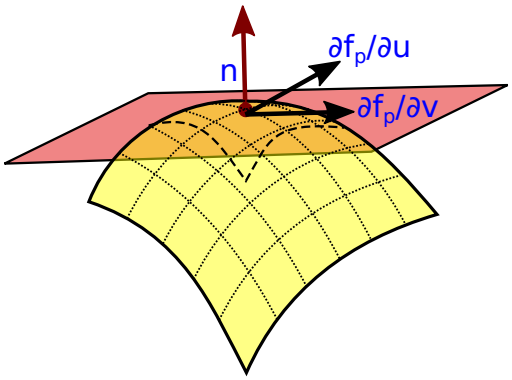


Figura 2.3: Representación gráfica de la normal a la superficie en un punto.

En la figura 2.3 se puede comprobar gráficamente a qué hacen referencia los distintos valores descritos.

2.4. Malla de triángulos

Para representar figuras geométricas complejas, es común utilizar mallas de polígonos definidas por una colección de vértices, aristas y caras. Su uso está extendido debido a que permiten representar geometrías con topologías arbitrarias fácilmente, son eficientes, cuentan con soporte en cualquier renderizador, algoritmos sofisticados y soporte hardware.

El polígono más utilizado para representar las caras en una malla de polígonos es el triángulo, ya que es el polígono más simple que puede existir en un plano euclídeo. Un triángulo está definido por las posiciones de sus tres vértices $\mathbf{v_0}, \mathbf{v_1}, \mathbf{v_2}$ en un espacio euclídeo tridimensional y las aristas que los conectan, además de un valor de normal asociado a cada vértice opcional $\mathbf{n_0}, \mathbf{n_1}, \mathbf{n_2}$, sobre el que se discutirá en la sección 2.4.1.

Al utilizar una malla de triángulos para representar figuras geométricas complejas, se está discretizando la geometría real de un objeto. Como consecuencia, si no se utiliza una discretización lo suficientemente granular es posible apreciar tanto el contorno de los triángulos (por la diferencia entre posiciones de cada punto de los triángulos y la geometría real, como se puede comprobar en la figura 1.3) como discontinuidades relacionadas con iluminación (por la diferencia entre la normal de la geometría real y la normal de cada triángulo, como se puede comprobar en la figura 1.1).

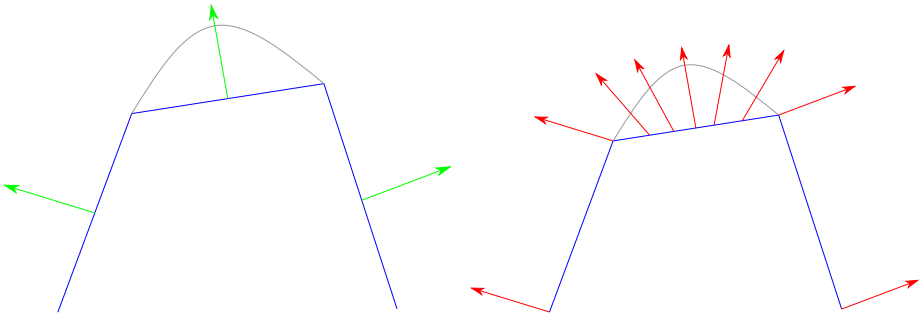
Aparte de añadir más triángulos a la malla, lo cual no es siempre posible, no existe una solución ampliamente utilizada para resolver el problema del contorno de los objetos. Para resolver el problema de diferencia entre normal real y normales de la malla se utilizan normales de *shading*.

2.4.1. Normal de *shading*

Para conseguir un aspecto suave en toda la superficie de la malla, en lugar de utilizar la normal del triángulo para cada posición del mismo se utilizan diversas técnicas de interpolación, consiguiendo así valores de normales variantes en toda la superficie.

La más sencilla de estas técnicas consiste en estimar el valor de la normal para cada vértice de una malla de triángulos mediante la media de las normales a la superficie de cada triángulo al que el vértice pertenezca. Siendo $\mathcal{N}$ todas las normales de los triángulos a los que pertenece un vértice $\mathbf{v_x}$, se define la normal de *shading* de $\mathbf{v_x}$ como:

$$n(\mathbf{v_x}) = \frac{\sum_{\mathbf{n} \in \mathcal{N}} \mathbf{n}}{|\mathcal{N}|} \tag{2.8}$$

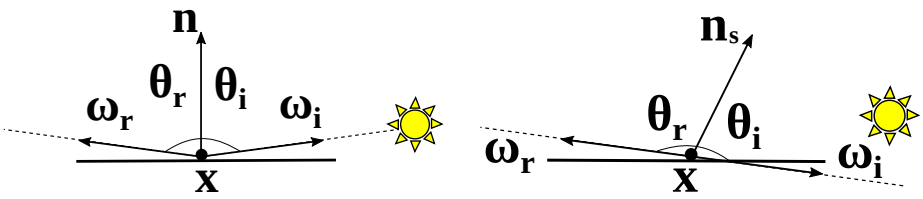


(a) Figura con normales sin interpolar. *(b) Figura con normales de shading interpoladas.*

Figura 2.4: Comparativa entre una figura sin normales interpoladas y con normales de shading interpoladas

Para cada punto perteneciente a la superficie de los polígonos se puede obtener su normal de *shading* interpolando linealmente entre las normales de los vértices del polígono. En la figura 2.4 se puede ver la diferencia entre una figura con las normales geométricas y otra utilizando normales de *shading* interpoladas.

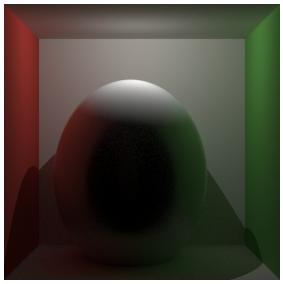
Sin embargo, utilizar normales de *shading* puede causar artefactos en determinadas circunstancias por no coincidir con la normal geométrica real. En ángulos rasantes, la diferencia entre las normales puede causar que rayos reflejados mediante las leyes de reflexión entren en la superficie de la figura en vez de apuntar afuera de la superficie, causando artefactos ya que no se puede calcular iluminación en esos puntos. En la figura 2.5 se muestra esta situación, y en la figura 1.2 se puede comprobar este artefacto con un icosaedro renderizado con trazado de rayos.



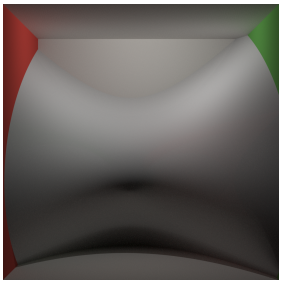
(a) Rayo reflejado con la normal geométrica. *(b) Rayo reflejado con la normal de* shading *entra en la superficie de la figura.*

Figura 2.5: Utilizar normales de shading *puede causar problemas por no ser la normal geométrica real, impidiendo un cálculo de iluminación correcto.*

En este trabajo se propone convertir los triángulos de las mallas de triángulos a una primitiva de orden superior, los triángulos cuádricos, intentando conseguir así una normal geométrica proporcional a la normal de *shading* de la malla de triángulos original, eliminando por tanto artefactos causados por la diferencia entre ambas. También se consigue eliminar el aspecto plano en los contornos de las figuras al utilizar una discretización poco granular ya que las cuádricas son de orden 2, consiguiendo como resultado un aspecto curvo. A continuación se realizará una introducción sobre las cuádricas.

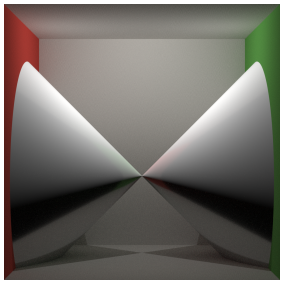


(a) Elipsoide.

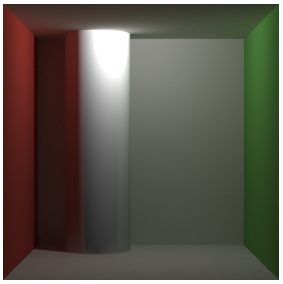


(b) Paraboloide.

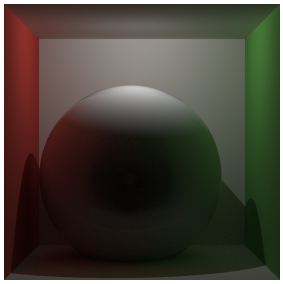
(c) Hiperparaboloide.



(d) Cono elíptico.



(e) Cilindro.



(f) Esfera.

Figura 2.6: Diferentes tipos de cuádricas.

2.5. Cuádricas

Una **superficie cuádrica** o **cuádrica** es una generalización de las secciones cónicas. Es una *hipersuperficie* D-dimensional definida como las raíces de una ecuación de segundo grado. En un espacio euclídeo tridimensional, se puede expresar implícitamente de la siguiente manera:

$$\begin{gathered} A, B, C, D, E, F, G, H, I, J \in \mathbb{R} \\ [x, y, z] \in \mathbb{R}^3 \\ f_i(x, y, z) = Ax^2 + By^2 + Cz^2 + \\ Dxy + Exz + Fyz + Gx + Hy + Iz + J = 0 \end{gathered} \tag{2.9}$$

La esfera definida implícitamente en la ecuación (2.2) es una cuádrica, con valores $A = 1$, $B = 1$, $C = 1$ y $J = -1$.

En siguientes secciones y capítulos el término cuádrica hará referencia a la superficie en un espacio tridimensional.

En la figura 2.6 se muestran las imágenes de algunas superficies cuádricas. En un espacio tridimensional existen 16 formas normalizadas.

Para calcular la normal de una cuádrica y obtener un aspecto suave en toda la superficie de la misma no es necesario utilizar normales de *shading*, únicamente hace falta utilizar las normales geométricas de las mismas, ya que al ser de orden 2 no son planos. La normal coincide con el gradiente de la ecuación (2.9) normalizado.

$$\begin{gathered} \nabla f_i(x, y, z) = \mathbf{i}(2Ax + Dy + Ez + G) + \\ \mathbf{j}(2By + Dx + Fz + H) + \\ \mathbf{k}(2Cz + Ex + Fy + I) \\ \mathbf{n} = \frac{\nabla f_i}{||\nabla f_i||} \end{gathered} \tag{2.10}$$

2.5.1. Triángulos cuádricos

En este trabajo se llamará triángulo cuádrico a la subsección de una cuádrica delimitada por 3 aristas cuádricas. Esto puede conseguirse utilizando una representación paramétrica, ya que se puede limitar los parámetros de entrada fácilmente para generar únicamente un triángulo cuádrico, o seleccionando una subsección de una cuádrica implícita. En las secciones 4.1 y 4.2 se explorarán métodos para construir el triángulo cuádrico de manera paramétrica, y en la sección 4.3 se explorará una definición implícita.

Capítulo 3

Trazado de rayos

En este capítulo se realiza una introducción breve al trazado de rayos, técnica utilizada en informática gráfica, para poder comprender a qué se deben los distintos problemas relacionados con las mallas de triángulos que se intentan resolver en este trabajo.

3.1. Introducción

La técnica de trazado de rayos o *ray tracing* [Kuc88] es utilizada por algunos de los algoritmos más populares para generar gráficos por ordenador. Consiste en simular haces de luz en el espacio y sus interacciones con distintos objetos, consiguiendo de esta manera producir imágenes fotorealistas gracias a la generación de iluminación global, pero a un coste computacional muy grande. Por ello tradicionalmente se ha utilizado el trazado de rayos para *pipelines* de renderizado offline (imágenes generadas por ordenador o efectos para películas o televisión) donde no es crítico cuánto tarda en generarse una imagen y se necesitan generar imágenes lo más realistas posible. Para *pipelines* de renderizado online (videojuegos y aplicaciones en tiempo real) donde sí existen restricciones temporales para generar imágenes tradicionalmente se han utilizado otros algoritmos que no pueden generar iluminación global, aunque en las últimas generaciones de aceleradores gráficos también se incorpora soporte hardware para el trazado de rayos [Bur20]. En la figura 3.1 se ilustra un ejemplo de imagen fotorealista generada mediante la técnica de trazado de rayos.

La idea fundamental detrás del trazado de rayos es seguir los haces de luz desde una fuente de luz, intersectando con toda la geometría presente en la escena para determinar el color de los píxeles a mostrar en imagen, simulando el funcionamiento de la luz en la realidad. Funciona trazando el camino de los haces de luz desde una cámara virtual, a través de cada píxel en una pantalla virtual, y calculando el color de los objetos visibles desde ella. Se puede dividir en tres fases: generación de rayos, intersección de rayos y estimación del color.

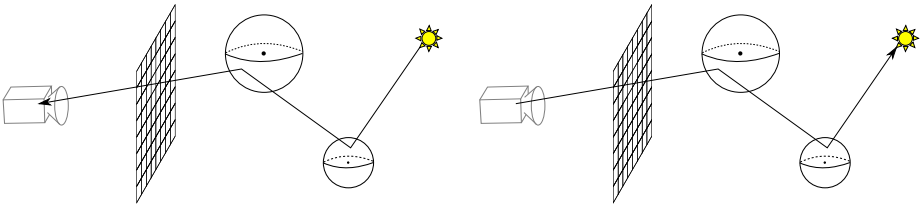
Figura 3.1: Imagen fotorealista renderizada con trazado de rayos.

Fuente: Gilles Tran, dominio público, via Wikimedia Commons

La ecuación del rayo sobre la que se fundamenta el algoritmo es la siguiente:

$$\mathbf{p} = \mathbf{o} + t\mathbf{d} \tag{3.1}$$

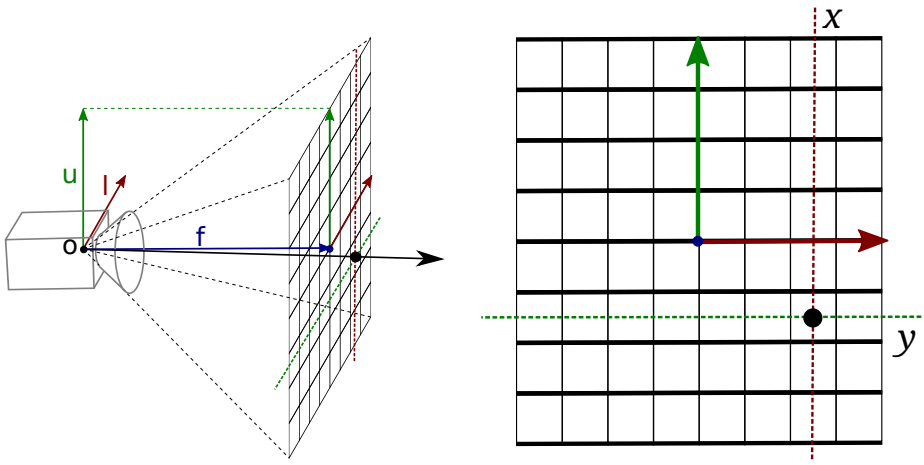
Éste rayo describe una trayectoria infinita desde un origen $\mathbf{o}$ en dirección $\mathbf{d}$ (como vector unitario), siendo $\mathbf{p}$ cualquier punto a lo largo del rayo y t la distancia entre $\mathbf{p}$ y $\mathbf{o}$. Para visualizar las figuras de una escena será necesario realizar una intersección entre rayo y figura geométrica, determinando así la visibilidad de los mismos o el color de cada píxel en la imagen resultante, como se explicará a continuación.



(a) En la realidad los rayos de luz son generados por fuentes de luz.

(b) En el trazado de rayos los rayos son generados desde la cámara.

Figura 3.2: Generación de rayos desde fuentes de luz o desde la cámara.



(a) Se generan los rayos desde un punto en la escena. Este punto es el $(0, 0, 0)$ *en espacio de la cámara.*

(b) Se generan uno o varios rayos por cada píxel de la pantalla.

Figura 3.3: Generación de rayos desde una cámara estenopeica.

3.2. Generación de rayos

En vez de trazar rayos desde las distintas fuentes de luz, como sucede en la realidad y se ilustra en la figura 3.2a, se trazan los rayos desde la cámara por razones de eficiencia, como se ilustra en la figura 3.2b. La mayor parte de los rayos generados desde una fuente de luz no llegan a la cámara, por lo que la mayoría de algoritmos no los utiliza.

Se generan uno o varios rayos desde la cámara por cada píxel, trazándolos en la escena para buscar posibles intersecciones entre el rayo y cualquier objeto en la escena. En la figura 3.3 se puede ver cómo se generan los rayos por cada píxel de la pantalla desde la cámara. A estos rayos se les llama rayos primarios (o rayos de cámara), ya que son los primeros rayos trazados en la escena (los rayos secundarios son usados para calcular sombras, reflexiones, refracciones, etc). Se utiliza un modelo de cámara estenopeica o *pinhole camera*, de modo que se generan los rayos desde un único punto en el espacio de la escena, como se puede visualizar en la figura 3.3a.

En espacio de la cámara cada rayo se genera con $\mathbf{o} = (0, 0, 0)$ y dirección $\mathbf{d} = (x, y, 1)$ normalizada, donde x e y son las posiciones de cada píxel de la pantalla.

3.3. Intersección de rayos

Para comprobar si un rayo intersecta con una figura o no hay que resolver un sistema de ecuaciones. Utilizando la forma implícita de la ecuación (2.1), se obtienen 3 grados de libertad por la ecuación del rayo (3.1) y 1 por la forma implícita.

$$\begin{aligned} \mathbf{p} &= \mathbf{o} + t\mathbf{d} \\ f_i(\mathbf{p}) &= 0 \end{aligned} \tag{3.2}$$

La intersección de un rayo con un plano, se resolvería de la siguiente manera. Tomando la ecuación del rayo (3.1) y $f_i(\mathbf{p}) = \mathbf{p} \cdot \mathbf{n} + \mathbf{c} = 0$ como ecuación del plano, por sustitución:

$$\begin{aligned} (\mathbf{o} + t\mathbf{d}) \cdot \mathbf{n} + \mathbf{c} &= 0 \\ \mathbf{o} \cdot \mathbf{n} + t\mathbf{d} \cdot \mathbf{n} &= -\mathbf{c} \\ t &= -\frac{\mathbf{c} + \mathbf{o} \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}} \end{aligned} \tag{3.3}$$

Si t es mayor que cero se ha encontrado el punto del rayo que intersecta con el plano. Si t es negativo el punto no pertenece al rayo. Si t es cero, todos los puntos del rayo se encuentran en el plano.

Si se utiliza la forma paramétrica definida en la ecuación (2.3) es necesario comprobar si el punto pertenece o no a la figura, bien utilizando $(u, v) = f_p^{-1}(\mathbf{p})$ y comprobando si los valores u, v obtenidos pertenecen a la figura o definiendo la forma paramétrica como combinación de formas implícitas (como en las mallas de polígonos).

La intersección de un rayo con un triángulo comienza como la intersección con un plano. A continuación se deben comprobar los límites del triángulo, ya que la ecuación implícita del plano no está limitada. Para ello se pueden tomar planos por cada arista del triángulo o utilizar algoritmos más eficientes y complejos.

Si hay varias figuras en la escena, un rayo puede intersectar con varias de ellas. Solo se utilizará la intersección con una figura que como resultado tenga la distancia t positiva mínima entre todas las intersecciones, como se ilustra en la figura 3.4.

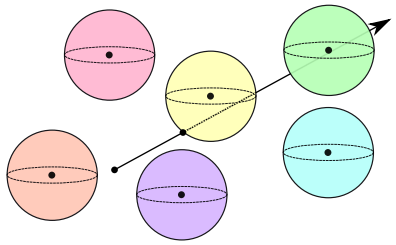


Figura 3.4: Solo se utilizará la intersección con la esfera amarilla.

3.3.1. Intersección de rayo con cuádricas

Para poder visualizar cuádricas genéricas se ha implementado la siguiente intersección de un rayo con una cuádrica en el renderizador Mitsuba [Jak10].

Tomando la ecuación del rayo (3.1) y la ecuación de una cuádrica genérica (2.9), por sustitución:

$$
\begin{gathered}
at^2 + bt + c = 0 \\
a = Ad_x^2 + Bd_y^2 + Cd_z^2 + Dd_xd_y + Ed_xd_z + Fd_yd_z \\
b = 2Ao_xd_x + 2Bo_yd_y + 2Co_zd_z \\
+D(o_xd_y + o_yd_x) + E(o_xd_z + o_zd_x) + F(o_yd_z + d_yo_z) \\
+Gd_x + Hd_y + Id_z \\
c = Ao_xo_x + Bo_yo_y + Co_zo_z \\
Do_xo_y + Eo_xo_z + Fo_yo_z + Go_x + Ho_y + Io_z + J
\end{gathered}
\tag{3.4}
$$

Se obtienen dos soluciones. Los dos valores de t encontrados al resolver la ecuación cuadrática anterior son los dos valores tal que $\mathbf{o} + t\mathbf{d}$ sean los puntos donde el rayo intersecta con la cuádrica.

Cualquier valor que sea negativo no pertenece al rayo. Por tanto, si el discriminante de la raíz es negativo, entonces el rayo no intersecta a la cuádrica.

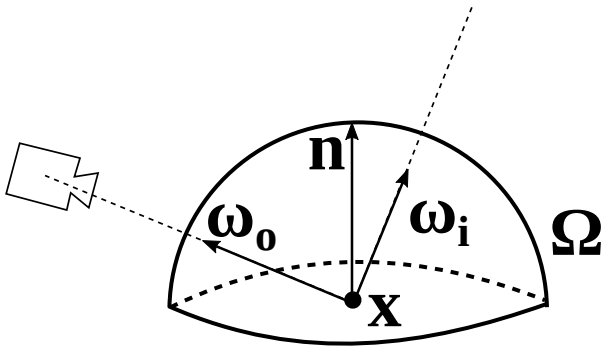


Figura 3.5: Representación gráfica de los distintos valores de ecuación de renderizado.

3.4. Estimación del color

Para calcular el color de cada píxel de la imagen final, es necesario resolver la ecuación de renderizado:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega L_i(\mathbf{x}, \omega_i) f_r(\mathbf{x}, \omega_i, \omega_o) |\mathbf{n} \cdot \omega_i| d_{\omega_i}$$

Donde

- $\mathbf{x}$ es una posición en el espacio
- ω_o es la dirección de la luz saliente
- $L_o(\mathbf{x}, \omega_o)$ es la radiancia espectral total de salida en dirección ω_o desde una posición $\mathbf{x}$
- $L_e(\mathbf{x}, \omega_o)$ es la radiancia espectral emitida desde $\mathbf{x}$
- $\mathbf{n}$ es la normal de superficie en $\mathbf{x}$
- Ω es la unidad hemisferio centrada en $\mathbf{n}$ conteniendo todos los posibles valores para ω_i
- $\int_\Omega \dots d_{\omega_i}$ es una integral sobre Ω, representando toda la radiancia espectral entrante a $\mathbf{x}$ desde cualquier dirección
- $L_i(\mathbf{x}, \omega_i)$ es la radiancia espectral entrante a $\mathbf{x}$ con dirección ω_i
- ω_i es la dirección negativa de la luz entrante a $\mathbf{x}$
- $f_r(\mathbf{x}, \omega_i, \omega_o)$ es la función de distribución de reflectancia bidireccional (o *BRDF* por sus siglas en inglés)
- $|\mathbf{n} \cdot \omega_i|$ es el factor de debilitamiento de la radiancia saliente debido al ángulo de incidencia, ya que el flujo de luz cubre una superficie con área mayor al área proyectada perpendicular al rayo

En la figura 3.5 se pueden visualizar los distintos parámetros de la ecuación de renderizado.

Para calcular la luz directa incidente en cada posición $\mathbf{x}$ en el espacio tras haber intersectado con los rayos desde la cámara, es necesario lanzar rayos de sombra (*shadow rays*) desde ese punto a las distintas fuentes de luz para saber si esa luz contribuye en el color del píxel o no. Tanto en las sombras como en la intersección del rayo principal se pueden apreciar los problemas mencionados en la sección 2.4 con el uso de mallas de triángulos. En la figura 3.6 se ilustran los rayos de sombra con dos fuentes de luz.

Según el tipo del material de la figura, ésta puede refractar la luz por completo (espejo) o parcialmente (al cambiar de un medio a otro, como de aire a agua). Para tener en cuenta este fenómeno, se generan rayos secundarios desde el punto $\mathbf{x}$ de intersección del rayo primario, refractados con un ángulo concreto

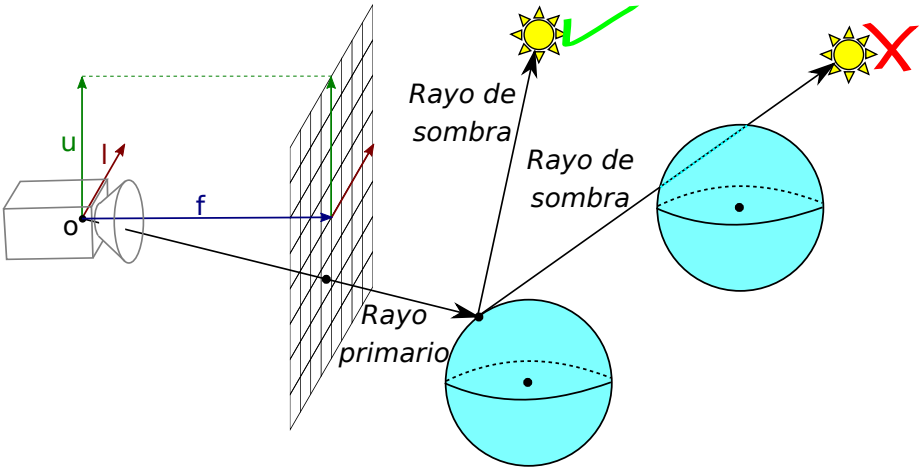
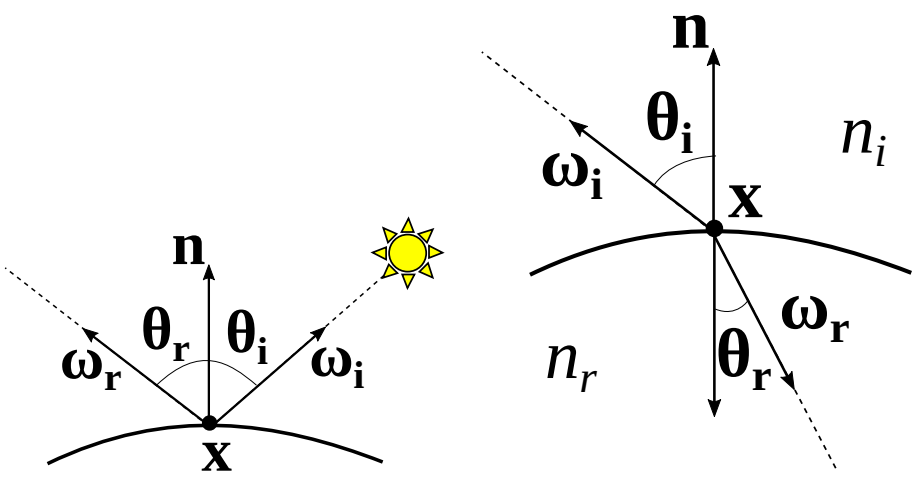


Figura 3.6: El rayo de sombra debe llegar a la fuente de luz o esa fuente de luz no se tiene en cuenta para el cálculo del color del píxel.

según el tipo del material. En la figura 3.7 se puede visualizar cómo se calculan los ángulos utilizados para lanzar los rayos secundarios. En este paso del algoritmo es donde aparecen los artefactos comentados en la sección 2.4.1. Los rayos secundarios actúan igual que un rayo primario, con cada intersección con otras figuras generarán rayos de sombra recursivamente.

Además de la luz directa y la luz recibida por espejos, la ecuación de renderizado tiene en cuenta toda la luz recibida en un punto $\mathbf{x}$ del espacio, esto es, luz indirecta que otras figuras refractan en todas direcciones, ya que los materiales no absorben el 100 % de la radiancia y emiten un porcentaje de la luz recibida en diversas direcciones según imperfecciones del material. Para calcular la luz indirecta sería necesario lanzar rayos secundarios en múltiples direcciones por cada intersección de un rayo primario/secundario recursivamente. Para conseguir terminar con la computación se emplean estimaciones y técnicas más avanzadas para calcular la ecuación de renderizado completa, las cuales quedan fuera del ámbito de este trabajo. En la figura 3.8 se ilustra este efecto, sería necesario generar rayos secundarios infinitos para resolver por completo la ecuación de renderizado.



(a) La luz es refractada con el mismo ángulo de incidencia en el punto de la superficie.

(b) La luz es refractada siguiendo la ley de Snell (cambio de aire a agua).

Figura 3.7: Distintos tipos de materiales y fenómenos requieren generación de rayos adicionales.

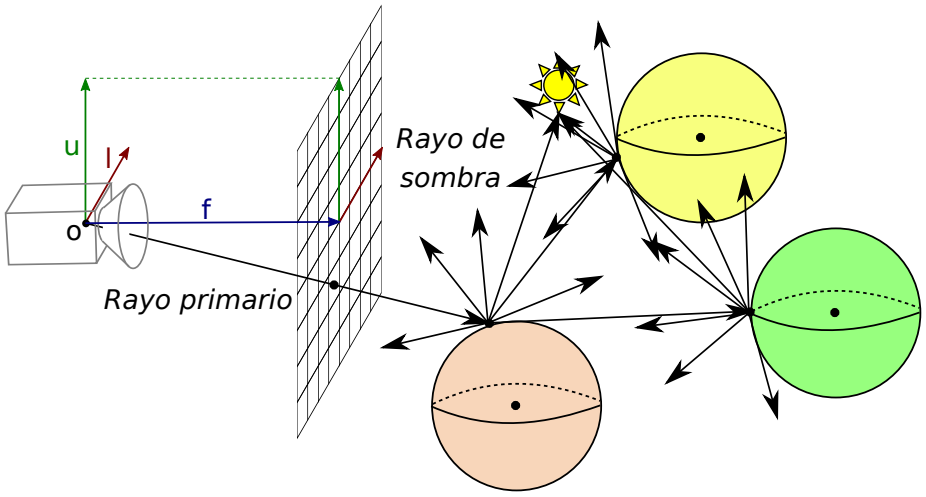


Figura 3.8: Para conseguir resolver la ecuación de renderizado en su totalidad sería necesario generar infinitos rayos secundarios.

Capítulo 4

Transformación de triángulos planos a triángulos cuádricos

En este capítulo se describen tres métodos como solución a los problemas asociados al uso de mallas de triángulos explicados en los capítulos 2 y 3: el contorno es visible al discretizar con poca granularidad y artefactos por utilizar normales de *shading* que no corresponden con las normales geométricas de la figura a visualizar.

Para ello, se propone crear un *triángulo cuádrico* para cada triángulo de la malla original, generándolo como superficie paramétrica y como superficie implícita.

Los triángulos cuádricos generados deben cumplir con las siguientes restricciones:

(1) Las coordenadas de los puntos correspondientes a los vértices deben formar parte del nuevo triángulo cuádrico.

(2) Las normales de los puntos correspondientes a los vértices del nuevo triángulo cuádrico deben corresponderse con las normales de *shading* del triángulo original.

(3) Debe asegurarse continuidad en C^0 entre triángulos, no puede haber discontinuidades entre distintos triángulos cuádricos.

(4) Debe asegurarse continuidad en C^1 entre triángulos, las normales entre los triángulos (en las aristas) deben ser suaves (las aristas entre triángulos deben compartir una dirección tangente).

Con estas restricciones se garantiza que la estructura de la malla se conserve gracias a (1) y a (3), ya que al conservar las posiciones para cada vértice de cada

triángulo de la malla localmente se consigue conservar la estructura globalmente, conservando así la apariencia de la malla original, y consiguiendo además que donde antes se intentaba simular una apariencia suave en toda la superficie de la figura ahora realmente se consiga gracias a (2) y (4).

Las restricciones (1) y (2) se pueden asegurar al resolver el sistema, asegurándose de que los puntos de los vértices pertenezcan al nuevo triángulo y asegurándose de que las normales en los mismos sean proporcionales a las normales de *shading* del triángulo original.

Las restricciones (3) y (4) no son tan sencillas de garantizar, se ha intentado encontrar experimentalmente modelos que cumplan con ellas, exponiendo a continuación los tres más prometedores.

Para resolver los problemas cumpliendo con estas restricciones se ha planteado encontrar las ecuaciones de las superficies mediante algoritmos eficientes, utilizando sistemas de ecuaciones lineales ya que resuelven problemas muy rápidamente y permiten modelar el problema de manera sencilla, e intentando resolver el problema de manera local frente a una solución global (resolviendo triángulo a triángulo frente a resolver para toda la figura) ya que así se consigue resolver mucho más rápidamente.

4.1. Triángulo paramétrico de aristas cuádricas

Esta representación se basa en modelar las aristas de los triángulos originales como aristas cuádricas, haciendo coincidir una misma arista para 2 triángulos adyacentes, intentando así cumplir con la restricción (3) por construcción del modelo, como se puede comprobar en la figura 4.1.

Para cumplir con la restricción (1) se fuerza a que las aristas comiencen en un vértice y terminen en otro, haciendo coincidir también inicio y fin de arista entre pares de aristas.

Para cumplir con la restricción (2) se fuerza a que la tangente en cada comienzo y fin de arista (en cada vértice del triángulo original) sea ortogonal a las normales en ambos vértices, intentando así que la normal geométrica coincida en todo el triángulo con la de *shading*.

La última restricción (4) es la más compleja de cumplir, en las siguientes subsecciones se explica cómo se genera el nuevo triángulo intentando cumplir con todas las restricciones y como se intersecta en un renderizador basado en trazado de rayos.

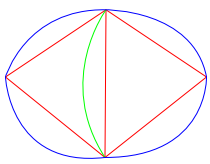


Figura 4.1: Las aristas que pertenecen a varios triángulos se comparten también en el nuevo triángulo (arista cuádrica verde).

4.1.1. Generación

La arista cuádrica está formulada de la siguiente manera:

$$\begin{gathered}\mathbf{c_{u2}}, \mathbf{c_{u1}}, \mathbf{c_{u0}} \in \mathbb{R}^3 \\ u \in [0,1] \subset \mathbb{R} \\ f_u(u) = u^2\mathbf{c_{u2}} + u\mathbf{c_{u1}} + \mathbf{c_{u0}}\end{gathered} \tag{4.1}$$

Una vez definidas las aristas, es necesario interpolar entre ellas para poder definir completamente el nuevo triángulo en toda su superficie. Esto se consigue con una interpolación transfinita (*transfinite interpolation*) [GH73], definida de la siguiente manera. Dadas tres funciones (4.1) para cada arista del triángulo, $f_u(u)$, $f_v(v)$ y $f_w(w)$:

$$\begin{gathered}w = 1 - v - u \\ u + v \leq 1 \\ f_p(u,v) = u(f_u(v) + f_w(1-w) - f_u(0)) \\ +v(f_v(w) + f_u(1-u) - f_v(0)) \\ +w(f_w(u) + f_v(1-v) - f_w(0))\end{gathered} \tag{4.2}$$

Para obtener una ecuación (4.2) que cumpla con las restricciones (1), (2), (3) y (4) se propone resolver un sistema de ecuaciones lineal, siendo éste un método sencillo, flexible y rápido para realizar pruebas. Se deben obtener por tanto 27 constantes que definan las 3 aristas. En la figura 4.2 se ilustran los valores de las distintas variables gráficamente.

Para cumplir con la restricción (1) se fuerza a que las aristas comiencen en un vértice y terminen en otro, obteniendo así 18 ecuaciones y quedando 9 grados de libertad.

$$\begin{array}{lll} f_u(0) = \mathbf{v_0} & f_v(0) = \mathbf{v_2} & f_w(0) = \mathbf{v_1} \\ f_u(1) = \mathbf{v_1} & f_v(1) = \mathbf{v_0} & f_w(1) = \mathbf{v_2} \end{array} \tag{4.3}$$

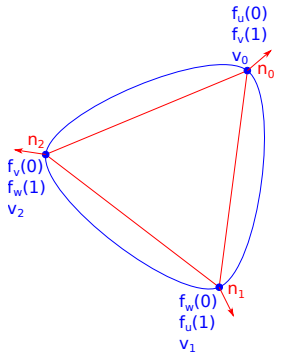


Figura 4.2: Figura representando las variables definidas en el sistema de ecuaciones a resolver.

Para cumplir con la restricción (2) se fuerza a que la tangente en cada comienzo y fin de arista (en cada vértice del triángulo original) sea ortogonal a las normales de *shading* en ambos vértices, intentando así que la normal geométrica coincida en todo el triángulo con la de *shading*, obteniendo 6 ecuaciones adicionales.

$$\begin{array}{lll} \frac{\partial f_u}{\partial u}(0) \cdot \mathbf{n_0} = 0 & \frac{\partial f_v}{\partial v}(0) \cdot \mathbf{n_2} = 0 & \frac{\partial f_w}{\partial w}(0) \cdot \mathbf{n_1} = 0 \\ \frac{\partial f_u}{\partial u}(1) \cdot \mathbf{n_1} = 0 & \frac{\partial f_v}{\partial v}(1) \cdot \mathbf{n_0} = 0 & \frac{\partial f_w}{\partial w}(1) \cdot \mathbf{n_2} = 0 \end{array} \tag{4.4}$$

Para cumplir con la restricción (3) e intentar cumplir con (4) se fuerza a que el punto intermedio de cada arista se genere en un plano dado por los puntos de inicio y fin de la arista y un tercero generado como el punto intermedio entre ellos desplazado según una normal interpolada entre ambos puntos. De esta manera se consigue que una arista compartida entre dos triángulos sea siempre la misma (independientemente de si el comienzo o el fin de la misma es un punto u otro), cumpliendo con (3), e intentando que la normal sea suave entre triángulos adyacentes para cumplir con (4). Dados dos vértices $\mathbf{v_i}, \mathbf{v_j}$ por cada arista del triángulo, y sus normales $\mathbf{n_i}, \mathbf{n_j}$:

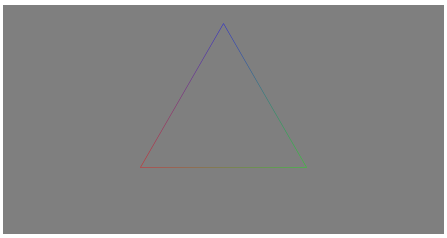
$$\begin{aligned} \mathbf{v_{medio}} &= \frac{\mathbf{v_i} + \mathbf{v_j}}{2} \\ \mathbf{n_{medio}} &= \frac{\mathbf{n_i} + \mathbf{n_j}}{2} \\ \mathbf{v_{desplazado}} &= \mathbf{v_{medio}} + \mathbf{n_{medio}} \\ \mathbf{n_{plano}} &= (\mathbf{v_j} - \mathbf{v_i}) \times (\mathbf{v_{desplazado}} - \mathbf{v_i}) \\ f_{u|v|w}(\frac{1}{2}) \cdot \mathbf{n_{plano}} &= \mathbf{v_i} \cdot \mathbf{n_{plano}} \end{aligned} \tag{4.5}$$

Siendo $f_{u|v|w}(\frac{1}{2})$ el punto intermedio de la arista generada.

Al tener 27 ecuaciones y 27 incógnitas se obtiene una única solución a las 9 constantes para cada arista definida en (4.1) del triángulo definido en (4.2). El triángulo final no es *cuádrico* per se ya que aunque las aristas sean cuádricas, al interpolar se pasa a una ecuación de orden 3.

Una vez obtenida la representación paramétrica del triángulo es necesario intersectarlo con rayos para obtener una imagen con iluminación global, explicado a continuación.

(a) El triángulo resultante se genera con tres coordenadas desde el espacio uv correspondientes a los vértices del triángulo original.



(b) Triángulo resultante al generar $f_p(u, v)$ para las coordenadas u, v en la figura de la izquierda. El color corresponde al valor de la normal en cada punto de la superficie.

Figura 4.3: Cuando se triangula utilizando únicamente los puntos desde uv correspondientes a los límites de las aristas se obtiene el triángulo original.

4.1.2. Intersección

Para intersectar la figura paramétrica obtenida existen dos alternativas para poder visualizarse mediante trazado de rayos:

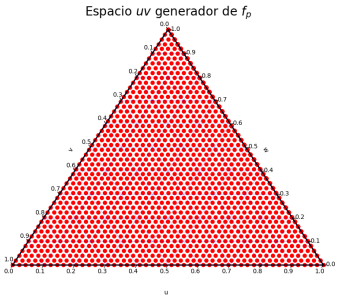
(1) Obtener la ecuación inversa de la ecuación (4.2), para comprobar si dado un punto $\mathbf{p}$ los valores uv pertenecen a la figura o no.

(2) Generar puntos pertenecientes a la superficie y triangular entre ellos, obteniendo una malla de triángulos más granular que la original.

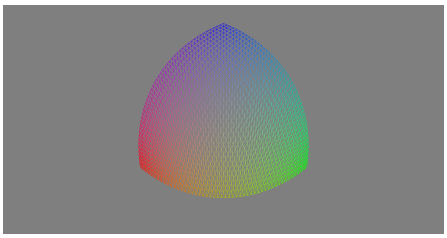
Se comenzó intentando intersectar utilizando la primera opción, intentando utilizar sistemas de optimización para obtener f_p^{-1}, tratando de minimizar $\|f_p(u, v) - \mathbf{p}\|^2$, pero los resultados obtenidos contenían muchos artefactos y no se apreciaba la figura obtenida.

Por tanto se decidió triangular utilizando la ecuación paramétrica obtenida, generando N puntos separados linealmente en un espacio uv bidimensional, usándose para generar puntos pertenecientes al nuevo triángulo y triangulando entre ellos para obtener una malla de triángulos tradicional que modele el nuevo triángulo representado por la ecuación paramétrica (4.2). Si se genera el triángulo resultante con únicamente 3 puntos correspondientes a los límites del espacio uv bidimensional, se puede comprobar gráficamente que las restricciones (1) y (2) han sido cumplidas, ya que se genera el triángulo original, como se ilustra en la figura 4.3.

Tras obtener la malla de triángulos que modela al nuevo triángulo (generada con 2500 puntos en el espacio uv, como se ilustra en la figura 4.4) se puede



(a) El triángulo resultante se genera con 2500 coordenadas desde el espacio uv.



(b) Triángulo resultante al generar $f_p(u, v)$ para las coordenadas u, v en la figura de la izquierda. El color corresponde al valor de la normal en cada punto de la superficie.

Figura 4.4: Con suficientes puntos al triangular la malla de triángulos resultante se aproxima al nuevo triángulo representado por la ecuación paramétrica.

utilizar directamente en cualquier renderizador ya que es la figura estándar utilizada en informática gráfica. En la figura 4.5 se muestran el triángulo original con normales de *shading* y el nuevo triángulo.

4.2. Triángulo cuádrico paramétrico

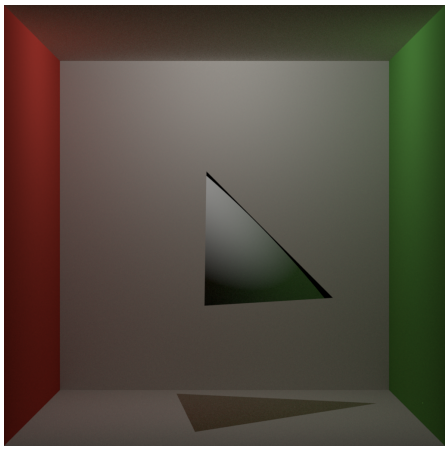
Tras generar un primer triángulo paramétrico mediante aristas cuádricas se procedió a experimentar con un modelo que fuese cuádrico en toda la superficie, pudiendo utilizarse por tanto como base para realizar una implicitación (explicada en la sección 2.1).

4.2.1. Generación

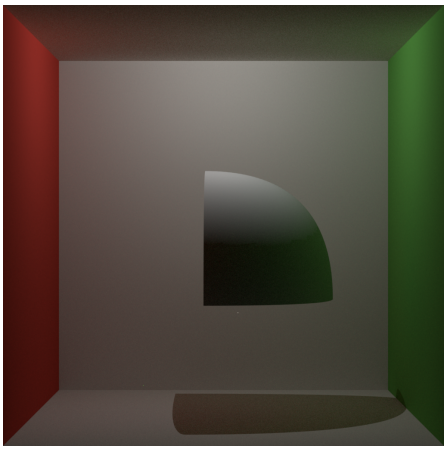
El triángulo cuádrico paramétrico está formulado de la siguiente manera:

$$\begin{gathered} u + v \leq 1 \\ u, v \in [0, 1] \subset \mathbb{R} \\ \mathbf{c_5}, \mathbf{c_4}, \mathbf{c_3}, \mathbf{c_2}, \mathbf{c_1}, \mathbf{c_0} \in \mathbb{R}^3 \\ f_p(u, v) = u^2\mathbf{c_5} + v^2\mathbf{c_4} + uv\mathbf{c_3} + u\mathbf{c_2} + v\mathbf{c_1} + \mathbf{c_0} \end{gathered} \tag{4.6}$$

Para obtener una ecuación (4.6) que cumpla con todas las restricciones se va a resolver de nuevo un sistema de ecuaciones lineal. Esta vez se deben obtener 18 constantes que definan el triángulo cuádrico paramétrico entero. En la figura 4.6 se ilustran los valores de las distintas variables gráficamente.



(a) Triángulo normal con normales de shading.



(b) Malla de triángulos modelando el nuevo triángulo. Al igualar normales de shading y de geométricas se eliminan artefactos.

Figura 4.5: Comparativa entre un triángulo normal y un triángulo nuevo. Se aprecia un contorno suave (en las sombras) y un aspecto suave.

Para cumplir con la restricción (1) se fuerza a que los límites del triángulo cuádrico paramétrico coincidan con los vértices del triángulo original, obteniendo así 9 ecuaciones y quedando 9 grados de libertad.

$$f_p(0,0) = \mathbf{v_0} \qquad f_p(1,0) = \mathbf{v_1} \qquad f_p(0,1) = \mathbf{v_2} \tag{4.7}$$

Para cumplir con la restricción (2) se fuerza a que la tangente en los límites del triángulo (en cada vértice del triángulo original) sea ortogonal a sus normales de *shading*, intentando así que la normal geométrica coincida en todo el triángulo con la de *shading*, obteniendo 6 ecuaciones adicionales.

$$\begin{array}{lll} \dfrac{\partial f_p}{\partial u}(0,0) \cdot \mathbf{n_0} = 0 & \dfrac{\partial f_p}{\partial u}(1,0) \cdot \mathbf{n_1} = 0 & \dfrac{\partial f_p}{\partial u}(0,1) \cdot \mathbf{n_2} = 0 \\ \dfrac{\partial f_p}{\partial v}(0,0) \cdot \mathbf{n_0} = 0 & \dfrac{\partial f_p}{\partial v}(1,0) \cdot \mathbf{n_1} = 0 & \dfrac{\partial f_p}{\partial v}(0,1) \cdot \mathbf{n_2} = 0 \end{array} \tag{4.8}$$

Para cumplir con la restricción (3) e intentar cumplir con (4) se genera un plano por arista de la misma manera que en la sección 4.1.1, dados $\mathbf{n_{plano0}}, \mathbf{n_{plano1}}, \mathbf{n_{plano2}}$ de la ecuación (4.9) calculados con $i, j = [(0,1),(0,2),(1,2)]$ respectivamente:

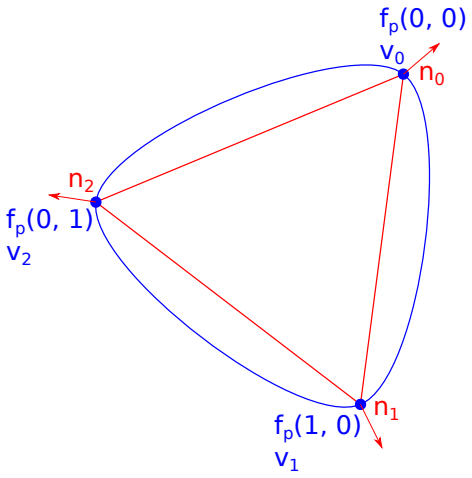


Figura 4.6: Figura representando las variables definidas en el sistema de ecuaciones a resolver.

$$
\begin{aligned}
f_p(\frac{1}{2}, 0) \cdot \mathbf{n_{plano0}} &= \mathbf{v_i} \cdot \mathbf{n_{plano0}} \\
f_p(0, \frac{1}{2}) \cdot \mathbf{n_{plano1}} &= \mathbf{v_i} \cdot \mathbf{n_{plano1}} \\
f_p(\frac{1}{2}, \frac{1}{2}) \cdot \mathbf{n_{plano2}} &= \mathbf{v_i} \cdot \mathbf{n_{plano2}}
\end{aligned} \tag{4.9}
$$

Al tener 18 ecuaciones y 18 incógnitas se obtiene una única ecuación a las 18 constantes que definen el triángulo cuádrico paramétrico en la ecuación (4.6).

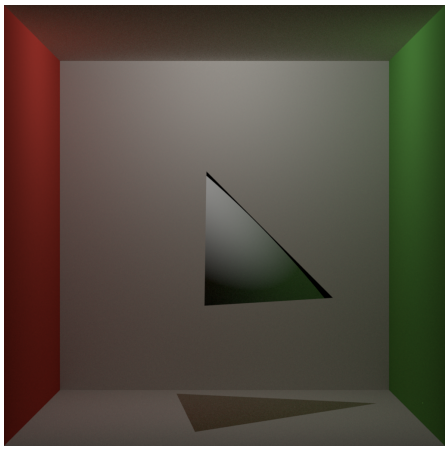
Una vez obtenida la representación paramétrica del triángulo cuádrico es necesario intersectarlo con rayos para obtener una imagen con iluminación global.

4.2.2. Intersección

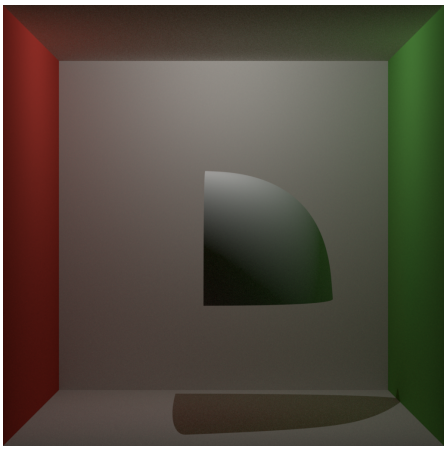
Para intersectar un triángulo (4.6) se sigue el mismo método que en la sección 4.1.2, generando puntos en un espacio *uv* y triangulando entre ellos para obtener una malla de triańgulos. En la figura 4.7 se muestran el triángulo original con normales de *shading* y el nuevo triángulo cuádrico paramétrico.

4.3. Triángulo cuádrico implícito

En los dos modelos paramétricos obtenidos, a la hora de generar una malla de triángulos para poder renderizarse se generan 2500 puntos de la superficie y se triangula entre ellos, dando como resultado 4802 triángulos por cada triángulo original, para poder aproximarse lo suficiente a la superficie paramétrica real para solucionar los problemas asociados al contorno y el uso de normales de



(a) Triángulo normal con normales de shading.



(b) Malla de triángulos modelando el nuevo triángulo cuádrico paramétrico. Al igualar normales de shading y de geométricas se eliminan artefactos.

Figura 4.7: Comparativa entre un triángulo normal y un triángulo cuádrico paramétrico. Se aprecia un contorno suave (en las sombras) y un aspecto suave.

shading. A continuación se propone un modelo implícito, para poder intersectar con un único objeto (la superficie implícita) en vez de los 4802 triángulos de los modelos anteriores, obteniendo resultados similares.

4.3.1. Generación

Para obtener una ecuación (2.9) que defina una cuádrica de la que se obtendrá el triángulo cuádrico se va a tratar de implicitar (ver sección 2.1) la forma paramétrica del triángulo cuádrico. No se puede implicitar el triángulo paramétrico generado a partir de aristas cuádricas porque no es de orden 2, como se ha explicado en la sección 4.1.1.

Para obtener los coeficientes que definan f_i de la ecuación (2.9), se plantea resolver un sistema de ecuaciones lineales, esta vez sobredeterminado ya que será necesario utilizar más grados de libertad de los 10 que se pueden obtener con la ecuación (2.9), resolviéndose por mínimos cuadrados.

Dada una ecuación (4.6) f_p:

Para cumplir con la restricción (1) se fuerza a que los límites del triángulo cuádrico pertenezcan a la cuádrica, obteniendo así 3 ecuaciones y quedando 7 grados de libertad.

$$f_i(\mathbf{v_0}) = 0 \qquad f_i(\mathbf{v_1}) = 0 \qquad f_i(\mathbf{v_2}) = 0 \tag{4.10}$$

Para cumplir con la restricción (2) se fuerza a que las tangentes en los límites del triángulo cuádrico paramétrico sean ortogonales a las normales de esos

puntos en la cuádrica implícita, intentando así que la normal de la cuádrica sea proporcional con la del triángulo cuádrico paramétrico, obteniendo 6 ecuaciones adicionales.

$$\begin{aligned}
&\nabla(f_i \circ f_p)(0,0) \cdot \frac{\partial f_p}{\partial u}(0,0) = 0 \qquad \nabla(f_i \circ f_p)(0,0) \cdot \frac{\partial f_p}{\partial v}(0,0) = 0 \\
&\nabla(f_i \circ f_p)(1,0) \cdot \frac{\partial f_p}{\partial u}(1,0) = 0 \qquad \nabla(f_i \circ f_p)(1,0) \cdot \frac{\partial f_p}{\partial v}(1,0) = 0 \\
&\nabla(f_i \circ f_p)(0,1) \cdot \frac{\partial f_p}{\partial u}(0,1) = 0 \qquad \nabla(f_i \circ f_p)(0,1) \cdot \frac{\partial f_p}{\partial v}(0,1) = 0
\end{aligned} \tag{4.11}$$

Para cumplir con las restricciones (3) y (4), ya gestionadas en el triángulo cuádrico paramétrico, se añaden las siguientes restricciones al sistema sobre el punto baricentro del triángulo cuádrico paramétrico.

$$\begin{aligned}
(f_i \circ f_p)(\frac{1}{3}, \frac{1}{3}) &= 0 \\
\nabla(f_i \circ f_p)(\frac{1}{3}, \frac{1}{3}) \cdot \frac{\partial f_p}{\partial u}(\frac{1}{3}, \frac{1}{3}) &= 0 \\
\nabla(f_i \circ f_p)(\frac{1}{3}, \frac{1}{3}) \cdot \frac{\partial f_p}{\partial v}(\frac{1}{3}, \frac{1}{3}) &= 0
\end{aligned} \tag{4.12}$$

Obtenido un sistema sobredeterminado con 11 ecuaciones y 10 incógnitas, se añade el siguiente término de regularización para ayudar a encontrar una única solución:

$$A + B + C + D + E + F + G + H + I + J = 1 \tag{4.13}$$

Terminando con un sistema con 12 ecuaciones y 10 incógnitas, resuelto por mínimos cuadrados.

Una vez obtenida la cuádrica es necesario delimitarla de alguna manera para visualizar únicamente la parte de la superficie que corresponde al triángulo cuádrico implícito, obteniendo finalmente un triángulo cuádrico. Para ello se generan 4 planos, definidos por los siguientes puntos:

(1) $f_p(0,0), f_p(1,0), f_p(0,1)$

(2) $f_p(0,0), f_p(0{,}5,0), f_p(1,0)$

(3) $f_p(0,0), f_p(0,0{,}5), f_p(0,1)$

(4) $f_p(1,0), f_p(0{,}5,0{,}5), f_p(1,1)$

La intersección viene explicada en la siguiente sección.

4.3.2. Intersección

Para poder visualizar el triángulo cuádrico implícito definido por una cuádrica y 4 planos es necesario intersectar con la cuádrica tal y como se explica en la sección 3.3. Una vez obtenido un punto $\mathbf{p}$ perteneciente a la cuádrica, se debe comprobar si pertenece o no al triángulo cuádrico, comprobando que el punto

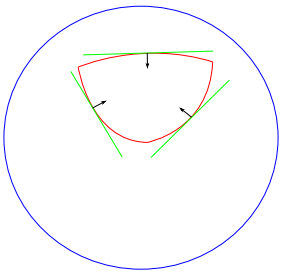
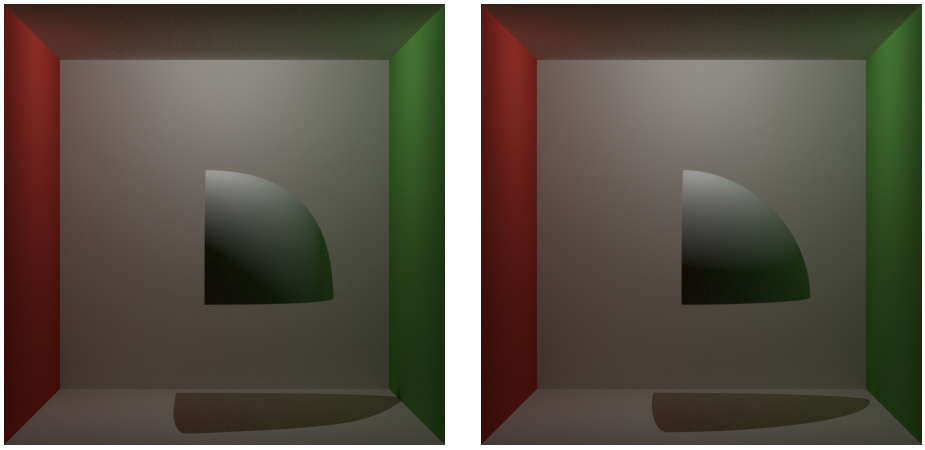


Figura 4.8: En azul la cuádrica generada, en verde los planos (líneas en 2d) utilizados para cortar la cuádrica para generar (en rojo) el triángulo cuádrico.

esté dentro de los 4 planos definidos en la sección anterior, como se puede ver en la figura 4.8.

Para ello se genera la normal de cada plano definido por $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$ como $\mathbf{n_p} = (\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_3} - \mathbf{p_1})$, invirtiéndola ($\mathbf{n_p} = -\mathbf{n_p}$) si no apunta al punto baricentro del triángulo cuádrico paramétrico $f_p(\frac{1}{3}, \frac{1}{3})$.

Para comprobar que el punto está dentro del triángulo cuádrico (las 4 normales de los planos apuntan al punto $\mathbf{p}$ de la cuádrica obtenido) se comprueba que el ángulo entre las normales y los vectores dirección desde un punto de cada plano al punto de la cuádrica sean mayores o igual a $\frac{\pi}{2} rad$, es decir, $\mathbf{n_p} \cdot (\mathbf{p_1} - \mathbf{p}) \geq 0$.



(a) Malla de triángulos modelando un triángulo cuádrico paramétrico.

(b) Triángulo cuádrico implícito.

Figura 4.9: Comparativa entre un triángulo cuádrico paramétrico y un triángulo cuádrico implícito. Se puede ver la diferencia entre posiciones y normal entre ambos triángulos.

En la figura 4.9 se muestra una comparativa entre un triángulo cuádrico paramétrico y un triángulo cuádrico implícito.

Capítulo 5

Resultados

En este capítulo se presentan los resultados de aplicar las técnicas de transformación de triángulos con normales interpoladas de *shading* en las diferentes versiones de triángulos cuádricos presentadas en el capítulo 4. Primero se comienza con una comparativa visual (sección 5.1) en la que se comprueba que el hecho de utilizar las diferentes formas de triángulos cuádricos, dado que no existe diferencia entre normal interpolada de *shading* y normal geométrica, efectivamente eliminan los artefactos típicos de las diferencias entre ambas normales. Después (sección 5.2) se comprueba el cumplimiento de las restricciones planteadas en la introducción al capítulo 4. Por último (sección 5.3) se explorarán las diferencias entre las diferentes representaciones de geometrías cuádricas.

5.1. Comparativa visual

En esta sección se comparan las tres diferentes representaciones cuádricas con su equivalente mediante triángulos con normales interpoladas. Se tratarán una serie de geometrías razonablemente sencillas (triángulo, tetraedro e icosaedro) que aproximan en realidad superficies suaves. En esta comparativa se explora principalmente que los artefactos más comunes producidos por la interpolación de las normales desaparezcan gracias a que ambas dos normales coinciden en las correspondientes formas cuádricas.

Las imágenes de la figura 5.1 han sido generadas con un material difuso y una luz de punto para comprobar que el problema a los contornos visibles (en pico) de mallas de triángulos con insuficientes triángulos como se comenta en la sección 2.4 ha sido resuelto. Las imágenes de la figura 5.2 han sido generadas con un material dieléctrico (diferentes tipos de cristal) y una luz de área para comprobar si se corrigen artefactos causados por la normal de *shading* en ángulos rasantes producidos por la reflexión y/o refracción del material, como se ha explicado en la sección 2.4.1.

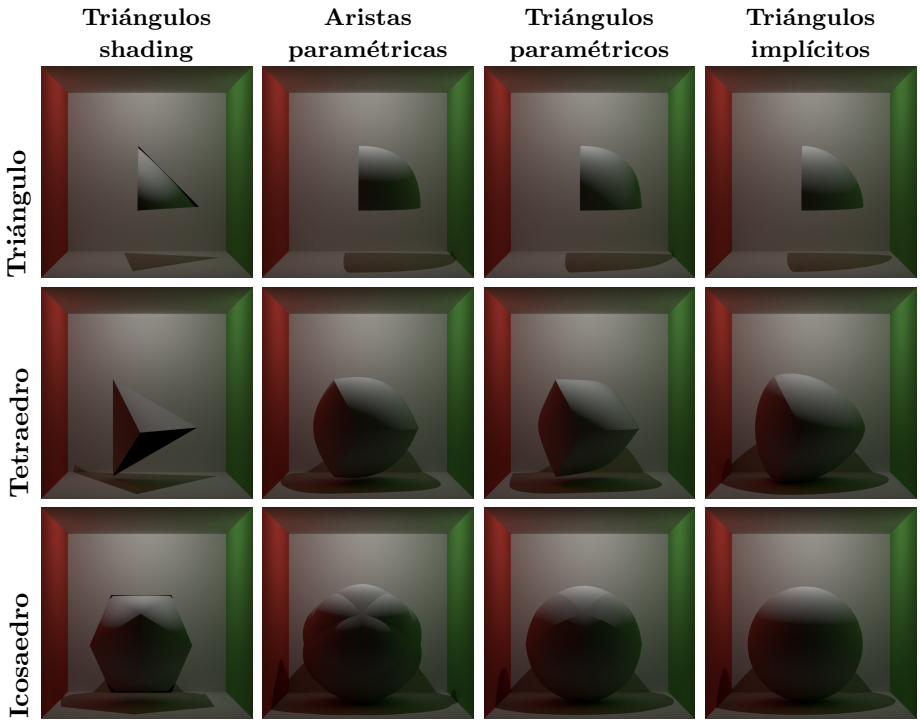


Figura 5.1: Figuras generadas con una fuente de luz puntual y un material difuso. En el contorno de las diferentes figuras generadas mediante triángulos con normales de shading *(columna izquierda) se presentan contornos que no concuerdan con la iluminación. También se pueden ver las aristas claramente en las sombras. Las diferentes formas cuádricas (segunda, tercera y cuarta columnas) renderizadas, sin embarago, presentan una geometría (contorno y sombras) más acorde con la iluminación, siendo clara la mejoría en el caso del icosaedro formado por triángulos implícitos.*

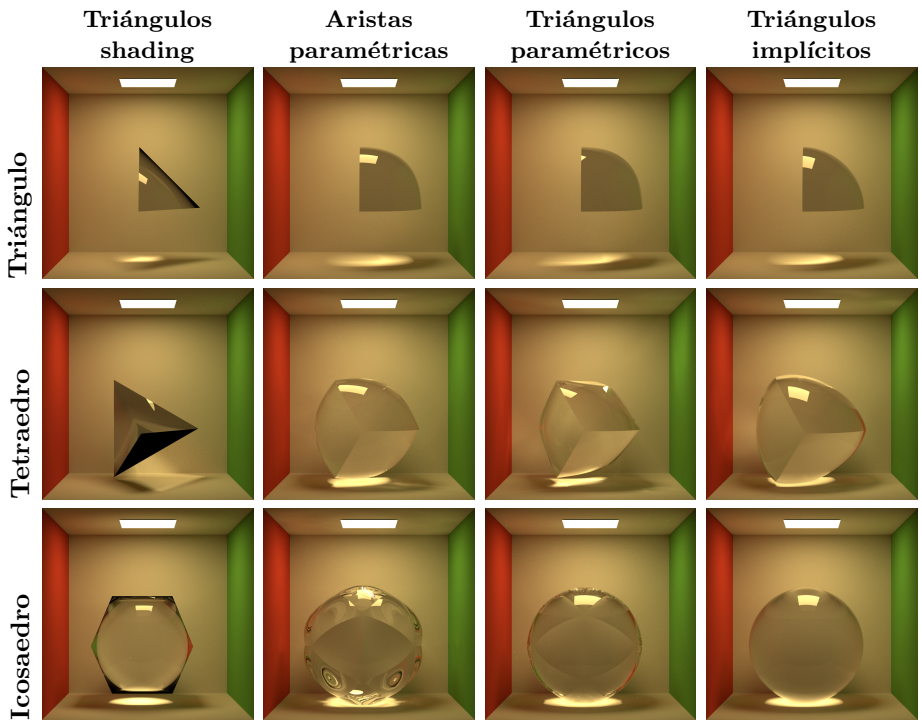


Figura 5.2: Figuras generadas con una fuente de luz de área y un material dieléctrico (diferentes tipos de cristal). En la apariencia de las diferentes figuras generadas mediante triángulos con normales de shading (columna izquierda) aparecen artefactos por discordancia entre normales geométrica y de shading. Las diferentes formas cuádricas (segunda, tercera y cuarta columnas) renderizadas, sin embarago, presentan una apariencia libre de artefactos, ya que las normales sí concuerdan, siendo más evidente en el caso del tetraedro, consiguiendo arreglar el artefacto en una cara entera con todos los modelos propuestos.

En la figura 5.1 se puede comprobar cómo se reduce significativamente el problema de los contornos por uso de insuficientes triángulos, ya que el contorno de las figuras renderizadas con los nuevos modelos es suave, no apreciándose los triángulos utilizados en los modelos originales más, siendo más aparente en las sombras de las mismas. En las aristas de los modelos paramétricos para el tetraedro y el icosaedro todavía se puede apreciar la división entre los distintos triángulos cuádricos, así como en el tetraedro modelado con triángulos implícitos, debido a que no se consigue cumplir con la restricción sobre C^1, como se explicará en la sección 5.2. En el icosaedro implícito sí se eliminan todos los problemas de contorno, por conseguir transformar a una esfera el icosaedro.

En la figura 5.2, especialmente en el tetraedro, se puede comprobar cómo desaparecen los artefactos causados por la normal de *shading* en determinados ángulos rasantes en los nuevos modelos, ya que la normal geométrica coincide con ésta. Al hacer coincidir ambas normales, y tras haber modificado la geometría del objeto original, la normal no apunta hacia dentro del objeto como se explica en la sección 2.4.1, sino que realmente es ortogonal al plano tangente a cada punto de los nuevos modelos, eliminando por tanto las zonas negras.

A continuación se explicará cómo se conservan o no las restricciones planteadas en la introducción al capítulo 4 en los distintos modelos.

5.2. Cumplimiento de restricciones

En esta sección se comprobará cómo los distintos modelos cumplen con las restricciones propuestas en la introducción al capítulo 4. Se realizará una comprobación sobre los dos modelos paramétricos propuestos, dejando el modelo implícito para la sección 5.3 ya que no se puede comprobar gráficamente de la misma manera que los modelos paramétricos por no ser un modelo generador. Se modelarán un tetraedro y un icosaedro, mostrándose coloreados con un valor proporcional a la normal en cada punto, calculado como el valor RGB resultado de normalizar cada normal en el rango $[0, 256)$ según los valores mínimo y máximo de las normales.

Las restricciones (1) y (2) son cumplidas por construcción de los modelos paramétricos, ya que se fuerza a ello en el sistema de ecuaciones y el sistema es determinado. Se puede comprobar además en las figuras 5.1 y 5.2 cómo la estructura de las figuras originales permanece intacta. En la figura 5.3 se genera una malla de triángulos para cada modelo paramétrico utilizando únicamente los puntos uv de los límites tal y como se explica en la sección 4.1.2, mostrando únicamente las aristas de cada vértice, pudiéndose ver cómo el resultado es la malla original.

Tanto en la figura 5.1 como en 5.2 se puede comprobar cómo se cumple con la restricción (3), ya que no hay discontinuidades entre distintos triángulos, puesto que las aristas son compartidas entre triángulos adyacentes. Esto es más aparente en la figura 5.4, en la cual se han generado visualizaciones con los 2500 puntos mencionados en la sección 4.1.2 mostrando únicamente las aristas de cada triángulo normal generado como resultado de la triangulación.

(a) Tetraedro generado con triángulo de aristas paramétricas cuádricas. *(b) Tetraedro generado con triángulo cuádrico paramétrico.* *(c) Icosaedro generado con triángulo de aristas paramétricas cuádricas.* *(d) Icosaedro generado con triángulo cuádrico paramétrico.*

Figura 5.3: Imágenes generadas únicamente con puntos uv pertenecientes a los límites de cada triángulo, mostrando las aristas de cada triángulo, generando el modelo original.

(a) Tetraedro generado con triángulo de aristas paramétricas cuádricas. *(b) Tetraedro generado con triángulo cuádrico paramétrico.* *(c) Icosaedro generado con triángulo de aristas paramétricas cuádricas.* *(d) Icosaedro generado con triángulo cuádrico paramétrico.*

Figura 5.4: Imágenes generadas con 2500 puntos desde espacio uv. Se puede comprobar C^1 en los vértices, C^0 en toda la superficie de los objetos, siendo las aristas compartidas para triángulos adyacentes.

La restricción (4) puede visualizarse mejor en la figura 5.4. Se puede apreciar cómo la normal varía suavemente al acercarse a cada vértice, pero no en las aristas compartidas entre dos triángulos, siendo más evidente en el tetraedro, no consiguiendo por tanto cumplir con la restricción (4) en toda la superficie pero sí en los vértices. En la figura 5.2 también se puede comprobar como las aristas son apreciables en prácticamente todas las figuras, excepto en el icosaedro implícito, ya que consigue obtener una esfera perfecta.

El modelo implícito, al ser resuelto por mínimos cuadrados, puede no cumplir con las restricciones (1) y (2) aunque se tengan en cuenta en la construcción del modelo, ya que al implicitar el modelo del triángulo cuádrico paramétrico (que sí las cumple), el modelo implícito también debería cumplirlas. Esto será explorado en la siguiente sección.

5.3. Comparativa entre representaciones

En esta sección se comparará los tiempos de renderizado para las figuras 5.1 y 5.2 y se comparará el modelo del triángulo cuádrico paramétrico con el triángulo cuádrico implícito.

Los siguientes resultados han sido generados utilizando el renderizador Mitsuba [Jak10], en un computador con una CPU i5 4690k con 8 GiB de memoria DDR3, utilizando un integrador *Path Tracer* con un *sampler* independiente. En la tabla 5.1 se presentan el número de primitivas (triángulos en una malla de triángulos normales o triángulos cuádricos implícitos en el nuevo modelo implícito propuesto) para cada modelo.

Los resultados de la tabla 5.2 pertenecen a la generación de las imágenes de la figura 5.1, generadas con 64 SSP (*samples per pixel*) y teniendo en la escena una luz de punto en la parte superior de la sala, estando cada figura renderizada con un material difuso.

Los resultados de la tabla 5.3 pertenecen a la generación de las imágenes de la figura 5.2, generadas con 500 SSP (*samples per pixel*) y con la opción *adaptive integration* activada, teniendo en la escena una luz de área en la parte superior de la sala y estando cada figura renderizada con un material dieléctrico (diferentes tipos de cristal).

Modelo	# primitivas		
	Triángulo	Tetraedro	Icosaedro
Triángulos shading	1	4	20
Aristas paramétricas	4802	19208	96040
Triángulos paramétricos	4802	19208	96040
Triángulos implícitos	1	4	20

Tabla 5.1: Número de primitivas presentes en cada modelo.

Modelo	Tiempo (s)		
	Triángulo	Tetraedro	Icosaedro
Triángulos shading	40.1680	43.9700	52.1100
Aristas paramétricas	67.4640	93.8160	100.2960
Triángulos paramétricos	72.5040	90.6240	112.1400
Triángulos implícitos	50.2120	71.6220	165.3240

Tabla 5.2: Tiempos de renderizado para distintas combinaciones de modelos y figuras con material difuso y punto de luz.

Modelo	Tiempo (h)		
	Triángulo	Tetraedro	Icosaedro
Triángulos shading	0.2882	0.6134	2.0992
Aristas paramétricas	1.6140	5.3269	7.7267
Triángulos paramétricos	1.7650	3.5300	6.6993
Triángulos implícitos	1.4336	3.4404	10.6203

Tabla 5.3: Tiempos de renderizado para distintas combinaciones de modelos y figuras con material dieléctrico y luz de área.

El modelo implícito es más rápido al renderizar el triángulo y el tetraedro, siendo más lento al renderizar el icosaedro. Esto se debe a que Mitsuba utiliza algoritmos que subdividen el espacio para minimizar el número de intersecciones necesarias a comprobar entre un rayo y las primitivas en la escena, no realizando intersecciones de figuras que están detrás de otras, necesitando para ello tener los límites de cada primitiva para poder generar las estructuras necesarias para subdividir el espacio. Sin embargo la implementación de cuádricas en Mitsuba realizada como parte de este trabajo no ha sido optimizada para todos los casos de uso de Mitsuba, ya que el renderizador incluye multitud de técnicas y algoritmos sofisticados y no era parte del objetivo del trabajo implementar todas.

En la figura 5.5 se han trazado líneas correspondientes a la relación de tiempos de renderizado entre el modelo del triángulo paramétrico cuádrico y el triángulo paramétrico implícito de los resultados mostrados en las tablas 5.2 y 5.3 conforme aumentan el número de primitivas a renderizar en el modelo implícito según la tabla 5.1. Se puede observar cómo el renderizado de la triangulación del modelo paramétrico es más lento (1,4440 veces más lento) con una única primitiva implícita y cómo la diferencia entre ambos modelos disminuye hasta llegar el modelo implícito a ser más lento (0,6783 veces más lento) al renderizar con 20 primitivas.

Asumiendo que la pérdida de eficiencia conforme aumenta el número de primitivas se debe a una implementación no óptima de cuádricas implícitas en Mitsuba, por tener que comprobar intersecciones entre figuras tapadas por otras, y asumiendo que la relación de tiempos de renderizado entre el modelo del triángulo cuádrico paramétrico y el triángulo cuádrico implícito fuese constante con respecto al número de primitivas si la implementación fuese óptima (si tuviese en cuenta las optimizaciones consistentes en subdividir el espacio usadas por Mitsuba), se puede afirmar que el modelo del triángulo cuádrico implícito renderiza en 0,6925 veces menos tiempo que el del triángulo cuádrico paramétrico.

Idealmente se podría convertir entre un modelo paramétrico y un modelo implícito sin pérdida, pudiéndose modelar el triángulo como un modelo paramétrico, por ser más sencillo expresar ecuaciones que cumplan con las restricciones del capítulo 4, y renderizar como modelo implícito. Sin embargo, aunque las tres representaciones son similares, no son intercambiables entre sí, como se puede comprobar en las diferencias entre ellos de las imágenes en las figuras 5.1 y 5.2. El modelo de triángulo paramétrico de aristas cuádricas no es cuádrico en toda la superficie del triángulo, sólo en las aristas, como se ha explicado en la sección 4.1.1. A continuación se explica por qué no se pueden intercambiar los modelos triángulo cuádrico paramétrico y triángulo cuádrico implícito.

En la figura 5.6 se pueden visualizar en mapas de calor las diferencias entre el modelo paramétrico y el implícito. En la primera imagen se visualiza $(f_i \circ f_p)(u, v)$, para comprobar que la figura implícita contiene todos los puntos de la figura paramétrica, dando idealmente 0 o próximo a 0 en toda la superficie, significando que las dos figuras contienen las mismas posiciones. En la segunda imagen se visualiza $\|\nabla f_p(u, v) \times \nabla(f_i \circ f_p)(u, v)\|$, para comprobar que la normal en ambos modelos es proporcional en toda la superficie. Valores próximos a 0 indican un ángulo entre las dos normales pequeño. Se puede comprobar cómo en los vértices tanto los puntos como las normales sí que coinciden en ambos modelos, cumpliendo así con las restricciones (1) y (2) el modelo implícito. En la imagen implícita del icosaedro de la figura 5.2 se puede apreciar cómo aunque el modelo del triángulo cuádrico paramétrico no cumpla con (4) en toda su superficie, el modelo implícito sí lo consigue, ya que la figura aproximada es una esfera, que es una figura cuádrica. También se puede ver cómo todas las figuras implícitas cumplen con (3), ya que no se aprecian discontinuidades entre triángulos.

Para poder intercambiar sin pérdida entre ambos modelos, el modelo pa-

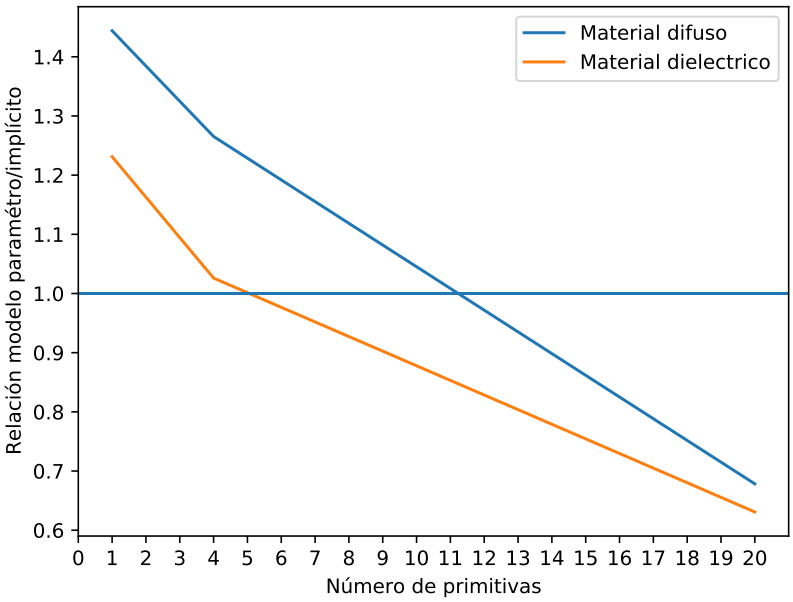
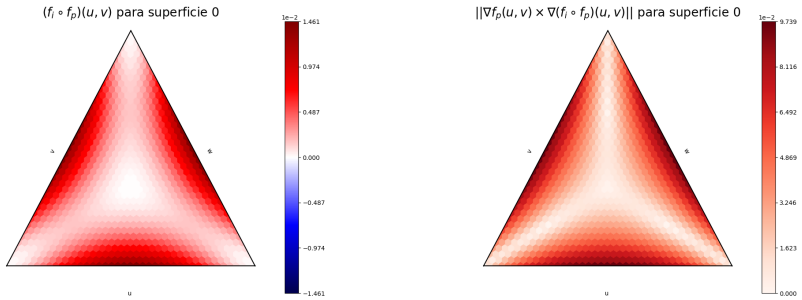


Figura 5.5: Gráfica mostrando relación entre tiempos de renderizado de la triangulación del modelo triángulo cuádrico paramétrico y el triángulo cuádrico implícito.



(a) Mapa de calor de $(f_i \circ f_p)(u,v)$ para el icosaedro.

(b) Mapa de calor de $\|\nabla f_p(u,v) \times \nabla(f_i \circ f_p)(u,v)\|$ para el icosaedro.

Figura 5.6: Mapas de calor para comprobar las diferencias entre los modelos triángulo cuádrico paramétrico y triángulo cuádrico implícito.

ramétrico debería estar fundamentado sobre un sistema de ecuaciones racionales, como se explicará en el siguiente capítulo.

Capítulo 6

Conclusiones

El objetivo de este trabajo era resolver dos problemas asociados al uso de mallas de triángulos:

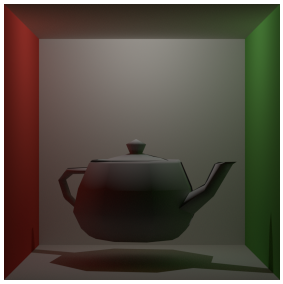
- Artefactos en determinados ángulos rasantes causados por uso de normales de *shading*.
- Contorno de los triángulos apreciable si no se utilizan suficientes triángulos.

Estos problemas han sido resueltos, como se ha mostrado en el capítulo 5, gracias a la introducción del triángulo cuádrico como nueva primitiva de renderizado siendo obtenido como transformación de un triángulo normal con normales de *shading*, asegurándose que el nuevo triángulo cuádrico conserve las posiciones de los vértices del triángulo original y también las normales a dichos vértices. Además, al utilizar sistemas de ecuaciones lineales se consigue obtener el nuevo triángulo cuádrico muy rápidamente.

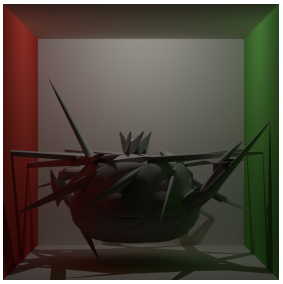
Para geometrías simples y sus discretizaciones los resultados son óptimos, consiguiendo transformar un icosaedro en una esfera perfecta, eliminando todos los artefactos por uso de normales de *shading* y consiguiendo una apariencia suave, sin picos en el contorno de la figura y sin artefactos en ángulos rasantes, como se explica en la sección 5.1.

Para geometrías complejas, pese a contener artefactos causados por normales extremas (explicado a continuación), se consiguen resolver también los problemas planteados en este trabajo. En la imagen de la tetera original de la figura 6.1 se pueden ver artefactos (zonas negras) por la discordancia entre normales, pudiéndose apreciar también el contorno de los triángulos. En los modelos paramétricos los artefactos por diferencia de normales no están presentes y el contorno es suave, consiguiendo el modelo de triángulo cuádrico paramétrico una apariencia suave en la base de la tetera.

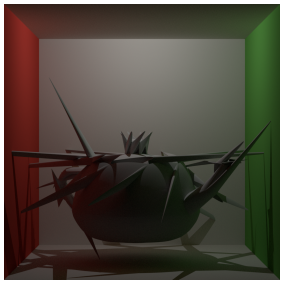
Con el modelo implícito no se consigue renderizar nada de la figura, ya que al estar el triángulo cuádrico limitado por cuatro planos, cuando el triángulo paramétrico a partir del cual se obtienen esos planos está muy distorsionado



(a) Figura compleja original, con normales de shading.



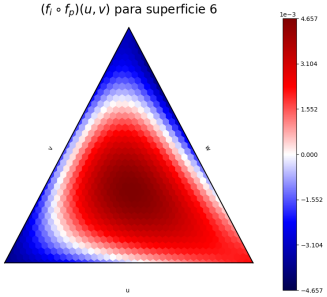
(b) Figura compleja con modelo basado en aristas cuádricas.



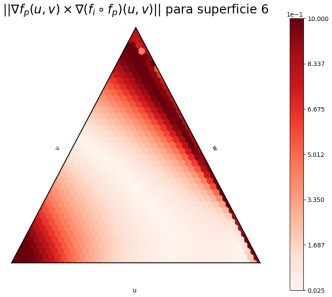
(c) Figura compleja con modelo cuádrico paramétrico.

Figura 6.1: Diferencia de resultados entre una geometría cuádrica y una geometría no cuádrica con respecto al modelo original. Además de los artefactos, en el modelo de aristas se puede observar el contorno de las aristas en la parte inferior de la tetera, mientras que en el modelo del triángulo cuádrico no.

Fuente: Tetera de Utah, Martin Newell

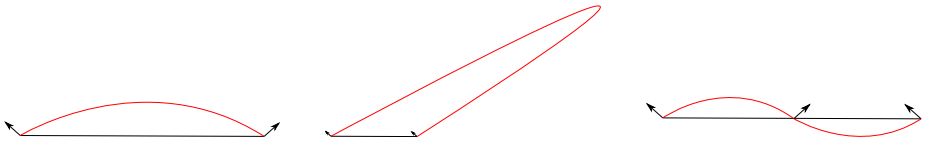


(a) Mapa de calor de $(f_i \circ f_p)(u, v)$ para una figura compleja.



(b) Mapa de calor de $\|\nabla f_p(u, v) \times \nabla(f_i \circ f_p)(u, v)\|$ para una figura compleja.

Figura 6.2: Mapas de calor para comprobar las diferencias entre los modelos triángulo cuádrico paramétrico e implícito.



(a) Cuando las normales de shading *en los vértices coinciden con normales para una cuádrica los resultados son óptimos.*

(b) Cuando las normales de shading *no pueden ser modeladas con una cuádrica los resultados no son satisfactorios.*

(c) Al contar con más grados de libertad (dividiendo el triángulo original en más subtriángulos) situaciones con normales extremas pueden modelarse con cuádricas.

Figura 6.3: Diferencia de resultados entre una geometría cuádrica y una geometría no cuádrica, y contando con más grados de libertad.

esos planos no son correctos. En la figura 6.2 se pueden observar las grandes diferencias entre ambos modelos para la figura compleja.

Esto es debido a que las normales en geometrías complejas resueltas por los modelos cuádricos propuestos pueden generar geometrías cuádricas muy forzadas. En la figura 6.3 se puede observar el problema, en la imagen de la izquierda las normales pueden ser resueltas con una cuádrica (las normales son perpendiculares a la cuádrica, ya que las normales se han obtenido de una figura cuádrica). En la imagen central, las normales no han sido obtenidas desde una cuádrica, siendo necesario un grado adicional para poder modelarse satisfactoriamente. El modelo cuádrico intenta obtener una cuádrica que sea perpendicular a ambas normales y para ello genera resultados que pueden considerarse como artefactos, por modificar en exceso la geometría original. En la última imagen de la derecha se puede ver una posible solución, pudiéndose resolver por medio de 2 cuádricas gracias a añadir grados de libertad al sistema, en vez de tener que modelarse la superficie con una ecuación ecuación de orden superior como una cúbica.

Para poder obtener un triángulo cuádrico para geometrías arbitrarias es necesario contar con más grados de libertad [Guo93]. Se ha realizado un pequeño experimento subdividiendo el triángulo original en 6, resuelto por un sistema lineal compuesto por 6 subtriángulos como triángulos cuádricos paramétricos. En la figura 6.4 se muestra el resultado, consiguiendo eliminar los artefactos producidos por normales incompatibles con un modelo cuádrico pero introduciendo otros artefactos en la continuidad de las normales. Esto es debido a que el sistema resuelto contiene 108 grados de libertad, más grados de los necesarios para poder ser resuelto con un sistema de ecuaciones lineal, quedando por tanto fuera del objetivo del trabajo.

Como trabajo futuro se puede explorar esta opción, al utilizar sistemas de ecuaciones racionales se pueden modelar correctamente los triángulos cuádricos de forma paramétrica para hacer uso de todos los grados de libertad proporcionados por los 6 subtriángulos [Vla+01]. Además, al utilizar modelos racionales los modelos paramétrico e implícito serían equivalentes, pudiendo por tanto

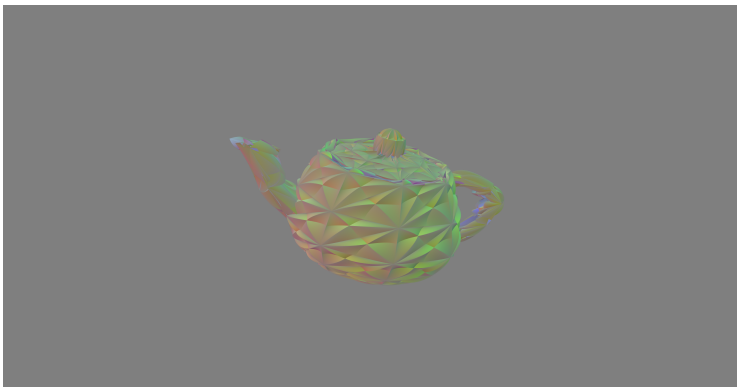


Figura 6.4: Al introducir grados de libertad adicionales es posible eliminar los artefactos obtenidos en los modelos paramétricos anteriores, pero para forzar una superficie suave es necesario realizar optimizaciones más sofisticadas no pudiéndose representar mediante ecuaciones lineales.

transformar entre uno u otro modelo sin pérdida de precisión [CSD07].

El trabajo ha sido gratificante, he aprendido mucho sobre geometría e informática gráfica, y espero que las exploraciones realizadas en este trabajo puedan servir como base para futuras publicaciones.

Bibliografía

[BA08] Tamy Boubekeur y Marc Alexa. "Phong Tessellation". En: *ACM Trans. Graph.* 27.5 (dic. de 2008). ISSN: 0730-0301. DOI: 10.1145/1409060.1409094. URL: https://doi.org/10.1145/1409060.1409094.

[Bur20] J. Burgess. "RTX on—The NVIDIA Turing GPU". En: *IEEE Micro* 40.2 (2020), págs. 36-44. DOI: 10.1109/MM.2020.2971677.

[CSD07] F. Chen, L. Shen y J. Deng. "Implicitization and parametrization of quadratic and cubic surfaces by μ-bases". En: *Computing* 79.2 (abr. de 2007), págs. 131-142. ISSN: 1436-5057. DOI: 10.1007/s00607-006-0192-0. URL: https://doi.org/10.1007/s00607-006-0192-0.

[GH73] William J. Gordon y Charles A. Hall. "Construction of curvilinear co-ordinate systems and applications to mesh generation". En: *International Journal for Numerical Methods in Engineering* 7.4 (1973), págs. 461-477. DOI: 10.1002/nme.1620070405. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620070405. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620070405.

[Guo93] Baining Guo. "Representation of arbitrary shapes using implicit quadrics". En: *The Visual Computer* 9.5 (mayo de 1993), págs. 267-277. ISSN: 1432-2315. DOI: 10.1007/BF01908449. URL: https://doi.org/10.1007/BF01908449.

[Jak10] Wenzel Jakob. *Mitsuba renderer*. http://www.mitsuba-renderer.org. 2010.

[Kuc88] Roinan Kuchkuda. "An Introduction to Ray Tracing". En: *F40, Theoretical Foundations of Computer Graphics and CAD, edited by R.A. Earnshaw, Springer-Verlag*. Springer, 1988, págs. 1039-1060.

[RSM10] Alexander Reshetov, Alexei Soupikov y William Mark. "Consistent Normal Interpolation". En: *ACM Trans. Graph.* 29 (dic. de 2010), pág. 142. DOI: 10.1145/1866158.1866168.

[Vla+01] Alex Vlachos y col. “Curved PN Triangles”. En: *Proceedings of the 2001 Symposium on Interactive 3D Graphics*. I3D ’01. New York, NY, USA: Association for Computing Machinery, 2001, págs. 159-166. ISBN: 1581132921. DOI: 10.1145/364338.364387. URL: https://doi.org/10.1145/364338.364387.