Universidad de las Ciencias

Informáticas

Facultad 5

Subdivisión de Superficies mediante el uso de la Unidad de Procesamiento Gráfico.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Arasay Baryolo Morales

Yandry Fernández Font

Tutores: Ing. Tamara Martínez Labaut

Ing. Lazaro Campoalegre Vera

Ciudad de la Habana, junio de 2010

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente

A los ____ días del mes de _____ del año _____.

Arasay Baryolo Morales

_____

Firma del Autor

Yandry Fernández Font

_____

Firma del Autor

Ing. Tamara Martínez Labaut

_____

Firma del Tutor

Ing. Lazaro Campoalegre Vera

_____

Firma del Tutor

Nombre y Apellidos: Tamara Martínez Labaut.

Edad: 25 años.

Ciudadanía: cubana.

Categoría: Instructor recién graduado.

Graduada de Ingeniera en Ciencias Informáticas en el año 2008, líder del proyecto Grupo para el Desarrollo de Videojuegos (GDEVI) del Centro de Informática Industrial, perteneciente a la UCI.

E-mail: tmartinez@uci.cu.

Nombre y Apellidos: Lazaro Campoalegre Vera.

Edad: 25 años.

Ciudadanía: cubana.

Categoría: Instructor recién graduado.

Graduado de Ingeniero en Ciencias Informáticas en el año 2008, Phd Student.

E-mail: lcampoalegre@uci.cu.

Dedico esta tesis a mis padres queridos, que tanto me han apoyado, que han sabido ser buenos padres, buenos amigos y por ser mi ejemplo a seguir. Especialmente a mis abuelitos, por su amor incondicional, por demostrarme que cada día puede ser un día mejor.

En el mundo existen personas que te marcan para siempre, en mi caso es mi novio, desde que lo conocí no he sabido otra cosa que admirarlo por su gran inteligencia, comprensión, por su paciencia para enseñarme, por haberme hecho la Ingeniera que soy hoy, y una mejor persona, y por permitirme estar a su lado en todos estos años.

Arasay.

A mi madre por haberme dado más de lo que una madre debe darle a un hijo.

A mi tío y a mi abuelo, donde quiera que este último se encuentre, por haber sido las figuras paternas y ejemplo a seguir en todo momento.

Pero la realización de este Trabajo de Diploma y el título que lo acompaña van dirigidos de forma íntegra para mi abuela por alentarme a superarme cada día y ser un hombre mejor, no me alcanzará una vida entera de agradecimiento.

Yandry.

Agradecimientos

A toda mi familia por apoyarme siempre.

A mi suegra Alicia por su gran confianza y amor.

A tío Alexis por ayudarnos en todos estos años.

A abuelita Ursula por darnos el apoyo y enseñarnos de sus experiencias.

Al club de las fantásticas por siempre estar a mi lado en lo bueno y en lo malo.

A nuestra tutora por estar siempre a nuestro lado, ayudándonos, apoyándonos y aprendiendo junto a nosotros.

Al profesor Osvaldo por ayudarnos en la realización de este Trabajo de Diploma.

Arasay.

A mi novia por el amor y el cariño que me ha profesado y por tratar de hacer de mí un hombre cariñoso.

A mis suegros por todo lo que han hecho por mí y por quererme como a un hijo.

A toda mi familia por el apoyo y la constante preocupación.

A Tere por los consejos y valor en los momentos de flaqueza.

A nuestra tutora por el apoyo y la ayuda en todo momento.

A Osvaldo por los consejos y el interés mostrado para con este Trabajo de Diploma.

A mis amigos del Edificio 91 en especial a Pavel, Héctor, Alexis y Sanjony.

A mi equipo del gimnasio: Jabier, Noel y Alcides.

Al Doctor por los consejos y los debates sobre el lenguaje desconocido.

Yandry

Resumen

El desarrollo de la Realidad Virtual en la actualidad, ha abierto paso a nuevos campos investigativos con vista a aumentar las potencialidades de estos sistemas. El modelado interactivo constituye uno de los campos abiertos a partir de la necesidad de lograr escenarios más realistas y mantener la atención de los usuarios en todo momento. Sin embargo, el rendimiento es uno de los elementos que afecta directamente al deseado realismo, y es precisamente uno de los principales problemas que enfrentan hoy las aplicaciones gráficas.

En este Trabajo de Diploma, se hace un estudio comparativo de las diferentes técnicas de suavizado poligonal. Se tienen en cuenta para la implementación y la discusión de los resultados, técnicas que exploten las potencialidades de los procesadores gráficos y las operaciones sobre fragmentos. Finalmente se implementa un algoritmo que utiliza como esquema de refinamiento el Catmull-Clark, específicamente al sustituir la etapa de Fragment Shader dentro de la arquitectura de la Unidad de Procesamiento Gráfico (*Graphics Processing Unit,* GPU por sus siglas en inglés).

Los resultados obtenidos en esta investigación se muestran a través de diferentes pruebas realizadas, que dan una medida clara de las potencialidades de implementar estos tipos de algoritmos, aprovechando el paralelismo proporcionado por la arquitectura de la GPU.

Palabras Claves

Subdivisión de Superficies, GPU, Fragment Shader.

Índice de Contenido

Índice

Índice

Introducción

La primera tarjeta gráfica con aceleración tridimensional (3D) real salió al mercado en 1996 [1]. Desde entonces no han dejado de evolucionar, incorporándoseles nuevas características de acuerdo a las necesidades presentadas por los usuarios.

El uso de las tarjetas gráficas para resolver problemas de visualización, se hace común en todo el mundo por disímiles usuarios, desde artistas gráficos, jugadores, hackers, desarrolladores, etc. Producto a su alta velocidad de cálculo que puede superar en algunos casos hasta 10.000 veces a la obtenida utilizando la Unidad Central de Procesamiento (*Central Processing Unit,* CPU por sus siglas en inglés) [2].

La Subdivisión de Superficies (SSs) es una de las técnicas de modelado asistido por computadoras con optimización de rendimiento y visualización, surgen del trabajo de Edwin Catmull y Jim Clark (Catmull - Clark) [3], y de Daniel Doo y Malcom Sabin (Doo - Sabin) [4] en 1978, dando el nombre a dos de los principales esquemas de refinamiento utilizados en nuestros días. Estos esquemas parten de la subdivisión de curvas. El primer uso de esta técnica fue en 1997, en la animación de "Geri's Game" [5] un corto de la Pixar ganador de un Oscar y además se ha utilizado en otras películas animadas como: “Bichos” (Pixar/Disney) y “Toy Story 2” (Pixar/Disney).

La idea de subdividir superficies es considerar un esquema de refinamiento y aplicarlo reiteradamente sobre unos cuantos puntos de partida. Se pueden utilizar esquemas locales de refinamiento para tratar de modo diferente distintos puntos de la superficie. Existen vértices extraordinarios (esquinas) en los que el esquema de refinamiento es diferente y sólo se consigue una regularidad menor que en el resto de los puntos.

En la actualidad muchas de las aplicaciones 3D sobrecargan el proceso de visualización, pues cargan una malla detallada para colocarla en cualquier parte del entorno 3D. Se ha demostrado que en el mundo tridimensional como en el mundo real, los objetos que se encuentran más alejados de la cámara son menos perceptibles por el ojo humano, por lo que no es posible percibir todos los detalles que en realidad presenta. En los gráficos por computadora, el nivel de detalle implica mayor cantidad de triángulos, esta mayor cantidad de triángulos influye en el proceso de renderizado, provocando que se cargue demasiado la escena. Estos problemas son resueltos con una técnica llamada Multirresolución [6], que permite

colocar varios modelos de un mismo objeto con varias resoluciones, donde los modelos menos detallados son colocados a mayor distancia de la cámara y viceversa. Esta técnica acarrea otros problemas, como la necesidad de diseñar y cargar varios modelos de dicho objeto, sobrecargando así la memoria de la máquina.

Por su parte la implementación de los algoritmos de SSs en la CPU necesita de un pre-procesamiento, para no provocar efectos desagradables de parpadeo en el dispositivo de visualización. Estas implementaciones tradicionales impiden las transformaciones necesarias en tiempo real para las aplicaciones y software que realizan el suavizado de superficies y niveles de detalles de modo interactivo. La línea de Núcleo de Procesamiento Gráfico perteneciente al Centro de Informática Industrial (CEDIN), en la Facultad 5 de la Universidad de las Ciencias Informáticas no cuenta con un mecanismo computacional que contenga alguno de los esquemas de refinamiento existentes, utilizado directamente sobre la tarjeta gráfica.

Ante este grupo de inconvenientes surge la siguiente interrogante: **¿Cómo lograr un mecanismo para suavizar mallas tridimensionales mediante el procesamiento gráfico?**

Para solucionar el problema planteado se presenta como **objeto de estudio** el Modelado Geométrico Tridimensional de Superficies y como **campo de acción** la Subdivisión de mallas poligonales.

Para guiar el trabajo a realizar, se plantea como **objetivo general** elaborar una herramienta para el suavizado de modelos tridimensionales.

Las siguientes **tareas de investigación** se proponen para dar cumplimiento al objetivo de este trabajo:

- Puntualización de las características que hacen que la GPU sea actualmente una beneficiosa opción para el trabajo con gráficos por computadora.
- Discusión de la necesidad de utilizar la Subdivisión de Superficies para el modelado asistido por computadoras.
- Selección de los esquemas de refinamiento capaces de funcionar de forma paralela durante el proceso de ejecución.

- Comparación entre los distintos esquemas seleccionados de acuerdo a sus características.
- Elaboración de un mecanismo que permita el trabajo con el esquema seleccionado vía CPU.
- Implementación del esquema de refinamiento seleccionado en el hardware gráfico.

La presente investigación consistirá en unir estos dos aspectos tratados anteriormente (GPU, SSs), específicamente, al utilizar estos algoritmos a partir del uso de la GPU, aprovechando así, el inherente paralelismo que brinda el hardware antes mencionado. Actualmente este tema está siendo tratado, se han impartido conferencias y existen publicaciones que lo sustentan, existen muchísimas implementaciones y utilizaciones de estos algoritmos de subdivisión para crear entornos virtuales, juegos, modelamiento de objetos en 3D y para todo tipo de escenas, pero en la mayoría de ellas no se potencializa el uso de la GPU en su totalidad, o sea, se sigue sobrecargando la CPU y la GPU pierde su propósito.

Los **métodos científicos** que se utilizarán para darle solución a las tareas anteriormente planteadas son:

Métodos Teóricos:

Para realizar una excelente búsqueda de información, relacionada con el tema de suavizar mallas poligonales, se utilizó el método **Analítico-Sintético**, identificando varios factores que condicionan el problema, en el caso de las técnicas aparecen dos tendencias, las NURBS y la Subdivisión de Superficies (SSs) y por parte del hardware a utilizar dos posibles elecciones, la CPU y la GPU. Luego del análisis por separado de cada uno de estos factores, se arriba a la conclusión que la SSs es el factor de mayor peso en la investigación, puesto que constituye la raíz del problema.

Otro método utilizado en este trabajo fue el **Histórico-Lógico**, que permite conocer toda la trayectoria del fenómeno (SSs), específicamente definido en dos etapas fundamentales, los trabajos realizados en la CPU como primer uso del fenómeno a tratar y algunos de los logros que se tienen hoy día en la GPU, permitiendo de esta forma una ubicación a partir de lo que se ha evidenciado hasta el momento del fenómeno que se estudia, la SSs.

Métodos Empíricos:

En la investigación profunda que se realizó de la Subdivisión de Superficies a partir del estudio de documentos, bibliografías, artículos, libros reconocidos, etc., se utilizó el método **Análisis Documental.**

Capítulo 1.

Fundamentación Teórica.

Introducción

En el mundo de los gráficos 3D generados por computadora, las técnicas de suavizado poligonal permiten crear modelos con superficies continuas y realistas a partir de modelos menos detallados. Si se desea suavizar una superficie, se aplica la técnica seleccionada repetidas veces hasta lograr el resultado esperado.

En el presente capítulo se hará un análisis de las características básicas de dos de las técnicas de suavizado de mallas poligonales usadas actualmente: las NURBS y la Subdivisión de Superficies, con el objetivo de seleccionar la más óptima, enfocados fundamentalmente en las características, ventajas y desventajas que presentan.

El capítulo incluirá además una comparación entre dos propuestas de hardware, CPU y GPU, para desarrollar la técnica de suavizado una vez éste sea seleccionado. Los procesadores serán analizados de acuerdo a su rendimiento frente a aplicaciones de gráficos por computadora, obteniendo una medida de su eficiencia y velocidad.

1.1 Técnicas de modelado poligonal.

1.1.1 Técnicas NURBS.

Las Non Uniform Rational Basis Splines (NURBS) [7], como difunde su nombre en idioma español, B-splines racionales no uniformes, que son representaciones matemáticas de geometría en 3D, capaces de describir cualquier forma con precisión, desde simples líneas en 2D, círculos, arcos o curvas, hasta los más complejos sólidos o superficies orgánicas de forma libre en 3D. Gracias a su flexibilidad y precisión, se pueden utilizar modelos NURBS en cualquier proceso, desde la ilustración y animación hasta la fabricación. En la siguiente figura es representada una superficie NURBS.

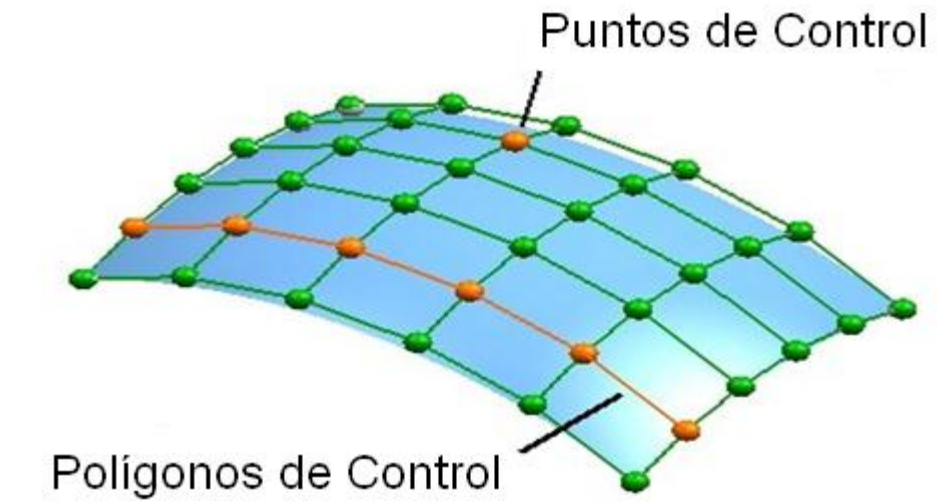


Figura 1. Representación de NURBS

1.1.1.1 Características de las NURBS.

Las NURBS pueden representar con precisión objetos geométricos estándares, tales como líneas, círculos, elipses, esferas y toroides, así como formas geométricas libres, como carrocerías de coches y cuerpos humanos.

La cantidad de información que requiere la representación de una forma geométrica en NURBS es inferior a la que necesitan por separado las aproximaciones comunes.

Una curva NURBS se define mediante cuatro elementos: grados, puntos de control, nodos y reglas de cálculo.

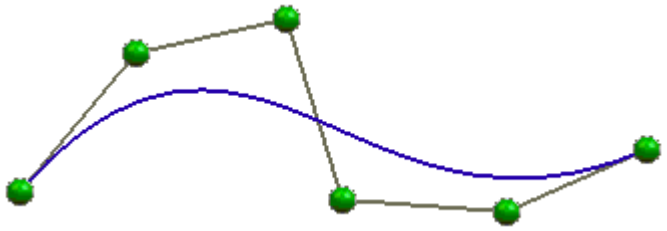


Figura 2. Representación de una curva NURBS.

1.1.1.2 Elementos que conforman las NURBS.

Grado:

Un grado es un número entero positivo.

Este número normalmente es 1, 2, 3 ó 5 pero puede ser cualquier número entero positivo. Las líneas y polilíneas NURBS son grado 1, los círculos son grado 2 y la mayoría de las formas libres son grado 3 ó 5. A veces se utilizan los siguientes términos: lineal, cuadrático, cúbico y quíntico. Lineal significa de grado 1, cuadrático significa de grado 2, cúbico significa de grado 3 y quíntico significa de grado 5.

El orden de una curva NURBS es un número entero positivo igual a (grado+1). En consecuencia, el grado es igual a orden-1.

Existe la posibilidad de incrementar los grados de una curva NURBS sin cambiar su forma. Generalmente, no es posible reducir el grado de una curva NURBS y no cambiar su forma.

Puntos de Control:

Los puntos de control son una lista de puntos de grado+1 como mínimo.

Una de las maneras más sencillas de cambiar la forma de una curva NURBS es mover los puntos de control.

Los puntos de control tienen un número asociado denominado peso. Con algunas excepciones, los pesos son números positivos. Cuando todos los puntos de control de una curva tienen el mismo peso (normalmente 1), la curva se denomina no racional; de lo contrario, se trataría de una curva racional. En NURBS, la R significa racional e indica que una curva NURBS tiene la posibilidad de ser racional. A la práctica, la mayoría de las curvas NURBS son no-racionales. Algunas curvas, círculos y elipses NURBS, ejemplos significativos, son siempre racionales.

Nodos:

Los nodos son una lista de números de grado+N-1, donde N es el número de puntos de control. A veces esta lista de números se denomina vector nodal. En este contexto, la palabra vector no significa una dirección 3D.

Esta lista de nodos debe cumplir varias condiciones técnicas. El modo estándar para asegurar que las condiciones técnicas se cumplan, es requerir que el número se mantenga igual o aumente a medida que vaya bajando en la lista y limitar el número de valores duplicados a que no sea superior al grado. Por ejemplo, para una curva NURBS de grado 3 con 15 puntos de control, la lista de números 0,0,0,1,2,2,2,3,7,7,9,9,9 es una lista de nodos satisfactoria. La lista 0,0,0,1,2,2,2,2,7,7,9,9,9 no es aceptable porque hay cuatro 2, y cuatro es un número mayor que el grado.

El número de veces que un valor nodal se duplica se denomina multiplicidad nodal. En el ejemplo anterior de lista satisfactoria de nodos, los valores nodales 0, 1, 2, 3, 7 y 9 poseen multiplicidad tres, uno, tres, uno, dos y tres respectivamente. Se dice que un valor nodal es un nodo de multiplicidad total si se multiplica por su grado varias veces. En el ejemplo, los valores de nodo 0, 2, y 9 tienen multiplicidad total. El valor de un nodo que aparece una sola vez se denomina nodo simple. En el ejemplo, los valores del nodo 1 y 3 son nodos simples.

Si una lista de nodos se inicia con un nodo de multiplicidad completa, la siguen nodos simples, termina con un nodo de multiplicidad completa y los valores se espacian uniformemente, entonces los nodos son uniformes. Por ejemplo, si una curva NURBS de grado 3 con 7 puntos de control tiene nodos 0, 0, 0, 1, 2, 3, 4, 4,4, la curva tendrá nodos uniformes. Los nodos 0, 0, 0, 1, 2, 5, 6, 6,6 no son uniformes. Los nodos

que no son uniformes se denominan no uniformes. Las letras N y U de la palabra NURBS significan no uniforme e indican que los nodos de una NURBS pueden ser no uniformes.

Los valores duplicados del nodo en la mitad de la lista del mismo, hacen que una curva de NURBS sea menos suave. En caso extremo, un nodo de completa multiplicidad en la mitad de la lista significa que hay un lugar en la curva NURBS que se puede doblar en un punto de torsión. Por esta razón, a algunos diseñadores les gusta agregar y quitar nodos y luego ajustar los puntos de control para hacer curvas más suaves o figuras torsionadas. Debido a que el número de nodos es igual a (N+grado-1), donde N es el número de puntos de control, si se agregan nodos también se agregan puntos de control, y si se quitan nodos se quitan puntos de control. Los nodos se pueden añadir sin cambiar la forma de la curva de NURBS. En general, quitar nodos cambiará la forma de una curva.

Nodos y Puntos de Control:

Un error frecuente se produce cuando cada nodo se empareja con un punto de control, y ocurre sólo en las NURBS de grado 1 (polilíneas). Para curvas NURBS de grados más altos, existen grupos de nodos de 2 x grado que corresponden a grupos de puntos de control de grado+1. Por ejemplo, suponga que tiene curvas NURBS de grado 3 con 7 puntos de control y nodos 0, 0, 0, 1, 2, 5, 8, 8,8. Los primeros cuatro puntos de control están agrupados con los primeros seis nodos. Del segundo hasta el quinto punto de control están agrupados con los nodos 0, 0, 1, 2, 5,8. Del tercer al sexto punto de control están agrupados con los nodos 0, 1, 2, 5, 8,8. Los últimos cuatro puntos de control están agrupados con los últimos seis nodos.

Regla de Cálculo:

La regla de cálculo de una curva utiliza una fórmula matemática que dado un número le asigna un punto. La regla de cálculo NURBS es una fórmula que comprende el grado, los puntos de control y los nodos. En la fórmula hay lo que se llama funciones básicas de B-spline. Las letras B y S de la palabra NURBS significan “basis spline.”El número de cálculo con que empieza la regla de cálculo se denomina parámetro. Puede idealizarse la regla de cálculo como una caja negra que se recibe un parámetro y produce un punto. El grado, los nodos y los puntos de control determinan el funcionamiento de la caja negra [8].

1.1.2 Técnicas de Subdivisión de Superficies.

En el mundo de los gráficos 3D generados por computadora, una subdivisión de superficies, es un método para representar una superficie suave mediante la especificación de una malla poligonal menos detallada. La superficie subdividida puede calcularse a partir de una malla más burda, iterando el proceso de dividir cada cara poligonal en caras más pequeñas que se aproximan mejor a la superficie suavizada. Esta técnica parte del modelado poligonal clásico (Hard Surface Modeling [9]), pues utiliza las mismas herramientas, aunque de un modo diferente, y optimiza además, a diferencia de la técnica poligonal, el espacio en memoria [10]. La SSs permite construir estructuras jerárquicas y el suavizado puede ser más local en determinadas zonas de la superficie límite.

1.1.2.1 Qué es la Subdivisión de Superficies.

La técnica de Subdivisión de Superficies [11] permite crear modelos de superficies suavizadas, continuas y realistas a partir de modelos menos detallados. Permite trabajar con modelos a baja resolución, es decir, con pocos polígonos, definiendo su geometría de un modo que se podría denominar como tosco, poco depurado. En esta técnica, aplicando una modificación final al modelo, sus polígonos se multiplican, suavizando las esquinas entre polígonos, y creando un modelo definitivo de superficies suavizadas que puede seguir siendo modelado y mejorado, actuando sobre los vértices del modelo a baja resolución. Ejemplo en la **Figura 3**.

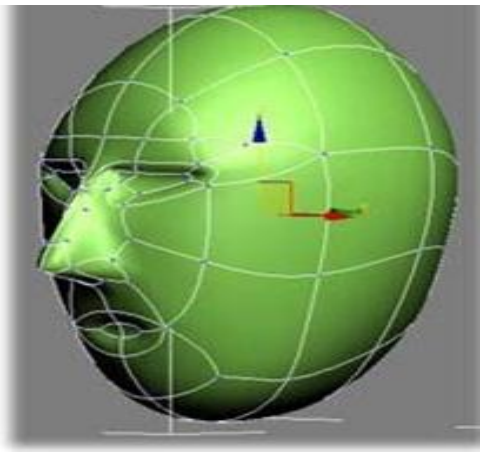


Figura 3. Representación de Subdivisión de Superficies.

1.1.2.2 Cómo Funciona.

La técnica de Subdivisión de Superficies añade polígonos de detalle para depurar una malla base del modelo, suavizando las esquinas de los polígonos. Usando una malla de una única superficie, la técnica de SSs emplea operaciones algorítmicas de manera automática para generar un efecto de suavizado [11]. Ver **Figura 4**.

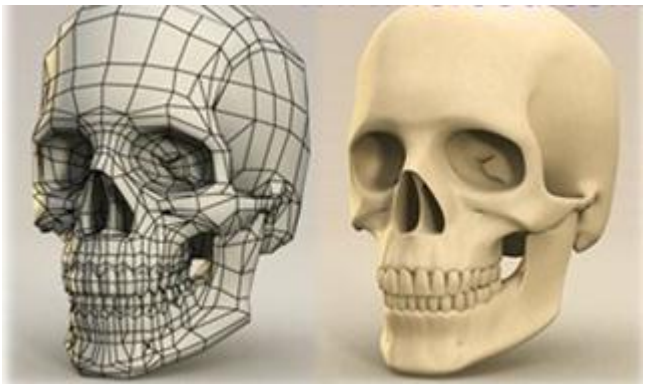


Figura 4. Representación de la superficie límite.

Aunque esta técnica se utiliza generalmente para suavizar las superficies de un modelo que ha sido realizada a un bajo nivel de detalle, en ocasiones también se aplica para añadir algunos polígonos más al modelo de forma homogénea y poder manipularlo a mano con más detalle posteriormente (en lugar de usar Quickslices para hacer divisiones).

Su aplicación se conjuga frecuentemente con el box modeling, es decir, a partir de una primitiva box, ir deformándola hasta obtener una aproximación al modelo deseado cuando se aplica la SSs para suavizar las superficies. Por lo tanto, es habitual también ayudarse de imágenes de fondo en los visores, que representan el alzado, planta o perfil del objeto que se quiere hacer, y en casos en los que el modelo sea simétrico, se suele trabajar sobre una mitad del objeto para luego hacer un espejo del mismo y soldarlo para completar el modelo.

1.1.2.3 Propiedades de la Subdivisión de Superficies.

La SSs crea una superficie suavizada que se gestiona a partir de una celosía con puntos de control. **Figura 5**.

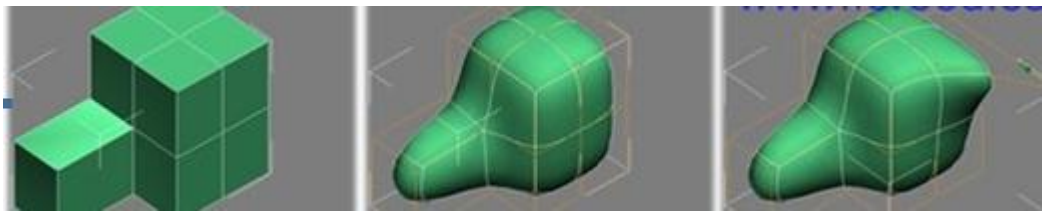


Figura 5. Esta celosía se crea a partir de un modelo poligonal a baja resolución.

La geometría del modelo puede ser controlada también a partir de los vértices del modelo poligonal original a baja resolución [11]. **Figura 6**.

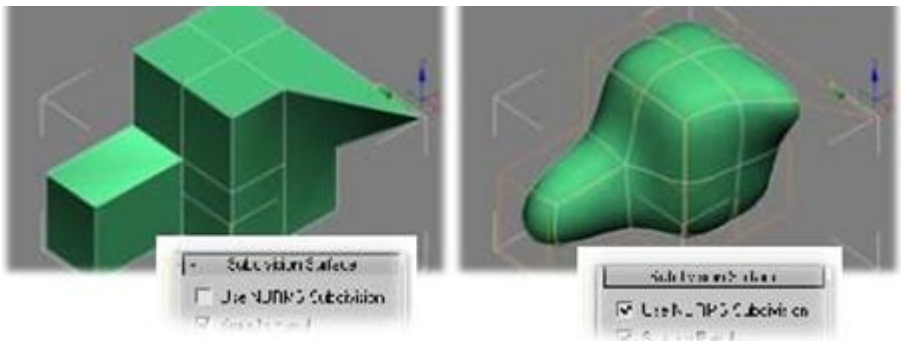


Figura 6. Subdivisión con vértices transformados.

Otra propiedad fundamental de la SSs es que los vértices de control, que son aquellos que definen la malla a baja resolución, pueden tener una propiedad llamada Peso (Weight), que permite atraer hacia ellos más o menos la superficie que éste gestiona, en función de su valor: **Figura 7**.

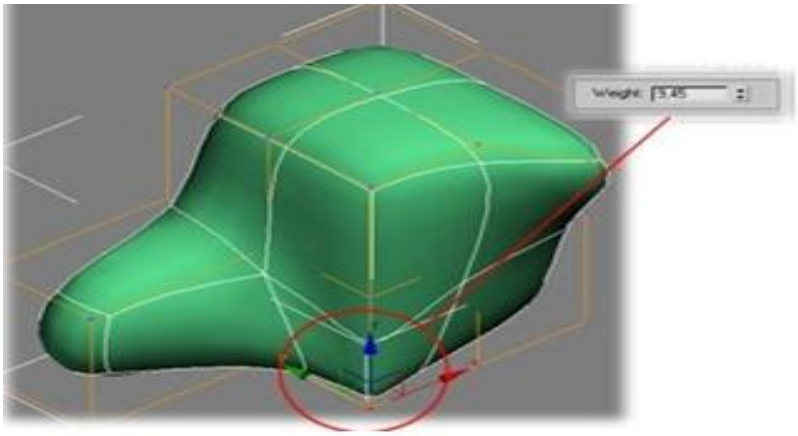


Figura 7. Superficie límite con función peso incluida.

1.1.2.4 Ventajas que tiene la Técnica de Subdivisión de Superficies.

- Se ven los resultados inmediatamente: se empieza con un modelo áspero, donde se puede definir fácilmente las dimensiones correctas.
- Se pueden añadir más detalles fácilmente mientras se avanza: no se tiene que modelar todo de una vez, sino que se puede por partes e ir avanzando, extruyendo, creando nuevos polígonos.
- Se consigue una malla de multirresolución: Importante para la rapidez de animación en los viewports. Trabajando a baja resolución no se sobrecarga la escena y se evita que el ordenador vaya demasiado lento.
- Se consigue una superficie lisa – Superficies continuas, y suavizadas.
- Facilita el modelado de figuras complejas: Con la práctica se comprobará lo fácil que puede ser modelar personajes complejos [11].

1.2 Esquemas de Refinamiento.

Los esquemas de refinamiento establecen mecanismos sencillos y eficientes para construir una superficie límite suave a partir de un poliedro de control inicial burdo, cuya conectividad y geometría son progresivamente refinadas. Su utilidad puede ser encontrada al facilitar la manipulación de superficies a distintos niveles de detalle (LODs), mencionados por primera vez en 1976 por Jim Clark [12].

La Tabla 1 contiene una clasificación resumida de los principales esquemas clásicos de subdivisión. En cuanto a su taxonomía son clasificados de acuerdo a cuatro criterios:

1. Polígonos que forman la malla de control inicial (triángulos o cuadriláteros).
2. Elementos subdivididos (facetas en los esquemas primales o vértices en los duales).
3. Relación entre la malla de control y la superficie límite.
4. Grado de suavidad.

Tabla 1 Comparación entre los diferentes esquemas de refinamiento.

	Tris./Cuads.	Primal/Dual	Aprox./Interp.	Suavidad
Catmull-Clark	C	P	A	C^2
Doo-Sabin	C	D	A	C^1
Loop	T	P	A	C^2
Mariposa	T	P	I	C^1

- Catmull-Clark: es una generalización de la subdivisión en B-splines bi-cúbicos (cúbicos en las dos direcciones u y v) uniformes.
- Doo-Sabin: es una generalización de la subdivisión en B-splines bi-cuadráticos (cuadráticos en las dos direcciones u y v) uniformes.
- Loop [13]: generalización de la subdivisión de splines cuadráticos sobre parches triangulares.
- Mariposa [14] (Butterfly).

1.3 Comparación entre las Técnicas NURBS y Subdivisión de Superficies.

Las NURBS son definidas mediante un conjunto de curvas B-splines parametrizadas en dos direcciones, muy flexibles y que poseen un conjunto de puntos de control utilizados para cambiar su forma. Aunque parece una técnica bastante sencilla poseen un conjunto de desventajas con respecto a la SSs, pues tienen una ampliación muy limitada, es decir, será extremadamente difícil agregarle algún tipo de transformación al modelo una vez haya sido terminado, esta técnica se obtiene a través de una parametrización plana, modelan láminas, cilindros, etc., una representación más compleja de la técnica constituyen las Trimmed NURBS que a su vez son susceptibles a errores numéricos y es difícil mantener su regularidad en animaciones [5]. En cambio la técnica de Subdivisión de Superficies se plantea como el medio para resolver los problemas existentes hasta el momento y facilitar el problema del suavizado poligonal, pues no es necesario modelar todo de una vez como las NURBS, ella puede por partes ir avanzando, extruyendo y creando nuevos polígonos. Parte de poliedros arbitrarios y son obtenidas superficies de topología compleja, donde las caras del poliedro se subdividen de modo sucesivo dando lugar a una superficie suave incluso en animaciones, facilitando de esta forma el modelado de figuras complejas. La técnica de Subdivisión de Superficies es más eficiente que la técnica NURBS en cuanto a que: los nuevos puntos de una superficie se calculan mediante pequeños números de operaciones de punto flotante de forma local [15], es decir, los nuevos puntos dependen solo de una región pequeña (región que se encuentra a su alrededor). Otras de las ventajas de la técnica, que le aporta un mayor grado de adaptabilidad, es que si el conjunto de puntos es transformado, trasladado o rotado, la superficie límite será subdividida a partir de la transformación realizada, brindando la posibilidad de interactuar con el modelo sobre la marcha. Otra de las ventajas es, que a diferencia de las NURBS, no es necesario efectuar un cosido de parches para eliminar errores de unión, como la ocurrencia de agujeros en la superficie límite. Debido a su carácter local es posible ejecutar los algoritmos de SSs de forma paralela pues las nuevas posiciones de los vértices del próximo nivel de subdivisión, solo dependen de las posiciones de los vértices del nivel actual.

1.5 Utilización de la GPU.

La unidad de procesamiento gráfico se convierte hoy en día en una ventajosa opción a la hora de trabajar los gráficos por computadora e incluso al desarrollar algoritmos de propósito general [16], debido a su velocidad de procesamiento con comas flotantes, y la posibilidad que nos brinda de paralelismo con la CPU, además del alto ancho de banda de memoria en la GPU, donde ésta, actúa como una memoria caché de nivel 2, que proporciona altísimas velocidades de transferencia de datos entre la GPU y el resto del sistema para mejorar el rendimiento en actividades de renderizado, sombreado, texturizado y operaciones de cálculo en general, proporcionando el menor coste de la GPU y la rápida evolución del hardware gráfico.

A continuación serán mostrados los resultados hechos por el grupo de *Tom's Hardware* [17], para eliminar cualquier vestigio de duda entre la problemática existente, entre Intel y NVIDIA, evidenciando el poder de las tarjetas gráficas sobre los microprocesadores de propósito general, Tabla 2 y 3.

En las Tablas 2 y 3 el porciento (%), es el resultado de un análisis de los (frame per secound) fps, donde el menor valor de la tabla simboliza el 100 %.

Tabla 2 Resultados obtenidos utilizando distintos tipos de GPUs.

Resultados para las GPUs	fps	%
GeForce 9800 GTX (512 MB)	15263.6	561.2
GeForce 8800 GTS OC (512 MB)	15257.4	561.0
GeForce 8800 GT OC (512 MB)	14609.2	537.1
GeForce 9600 GT OC (1024 MB)	13148.6	483.4
GeForce 7950 GT (512 MB)	6500.9	239.0

GeForce 6800 GT (256 MB)	2719.8	100.0

Tabla 3 Resultados obtenidos utilizando distintos tipos de CPUs.

Resultados para CPUs	fps	%
Q6600@3.2	12284.2	135.9
X6800EE@2.94	12097.5	133.9
E6750@2.67	11985.9	132.6
Q6600@2.4	11583.3	128.2
E2160@2.41	10512.4	116.3
E2160@1.8	9036.2	100.0

Por su parte y aunque la SSs es un mecanismo que se plantea sea aplicado a un modelo infinitas veces, sobre un número finito de caras, la implementación en la CPU se encuentra limitada, a modelos con un pequeño número de caras y el esquema de refinamiento solo puede ser aplicado también un número pequeño de veces, debido al crecimiento en complejidad de la malla en cada una de las iteraciones del esquema de refinamiento. Por esta razón y la posibilidad que brinda la GPU de trabajar con una gran cantidad de vértices se hace necesario implementar estos esquemas de refinamiento en el hardware antes mencionado.

Los lenguajes de Programación en la GPU antes del año 2006, permitían trabajar en dos etapas programables en la GPU: Vertex Shader [18] y Fragment Shader [19]. A partir de ese año sale a la luz otra etapa conocida como Geometry Shader [20] fundamentalmente para trabajar con primitivas, si bien es

cierto que tiene varias limitantes actualmente, como la cantidad de vértices que son posibles recibir para emitir una geometría específica. Puede ser considerada a la hora de tratar algoritmos que trabajen con geometría en futuros trabajos, pues para esto fue diseñada.

1.5.1 Vertex Shader.

La etapa de sombreado de vértice es capaz de trabajar con la estructura de vértices de los modelos tridimensionales y con ello realizar operaciones matemáticas modificando estas variables y así definiendo colores, texturas e incidencia de la luz. Esto da libertad a los programadores para realizar diferentes efectos desde la deformación de un objeto hasta la recreación de las olas del mar.

En caso de representaciones gráficas de pelo se basaría en los vértices de la malla dando un efecto más realista al resultado. Lo que conlleva a una rápida ejecución de la imagen puesto que se utiliza el hardware específico, en este caso el de las tarjetas gráficas.

Lo que en realidad pretende esta herramienta es adicionar a una malla de polígonos elementos que se alojan en los vértices de dichos polígonos o simplemente modificarlos.

Incluido en Direct3D y OpenGL, el Vertex Shader puede reproducir diferentes efectos realistas. El mismo ha evolucionado con el tiempo encontrándose en la actualidad en la versión 4.0.

Esta etapa permite trabajar con los vértices de forma individual. Se tiene como entrada un vértice con su respectiva información. Poder modificar la información o parámetros de los vértices tales como la posición, normal, coordenadas de textura, etc. Esta etapa brinda siempre como salida un objeto del mismo tipo que la entrada, o sea, un vértice. Una de sus limitantes es que no se tiene información de otros vértices de la malla salvo del que se está modificando en ese momento.

1.5.2 Geometry Shader.

Esta etapa de sombreado de geometría salió a la luz, por parte de NVIDIA, el 13 de noviembre del 2006, con entrada de primitivas simples (point, line, triangle).

La misma tiene como entrada los vértices obtenidos del procesador de vértices y su relación geométrica con el resto. En este caso la entrada no son vértices sino entidades geométricas (triángulos, líneas, conjuntos de triángulos, etc.). En ella pueden ser modificadas las posiciones y atributos de los vértices, teniendo en cuenta su relación geométrica. Un aspecto muy importante que dota de mayor potencia a esta etapa es la capacidad de generar nueva geometría o eliminar la que ya existe. Es la única etapa en la que la entrada no siempre genera una salida de la misma naturaleza. Las operaciones implicadas pueden duplicar geometría (escenarios de mallas idénticas), seleccionar geometría con respecto a criterios geométricos (nivel de detalle, composición, etc.) y también modificar características individuales de los vértices.

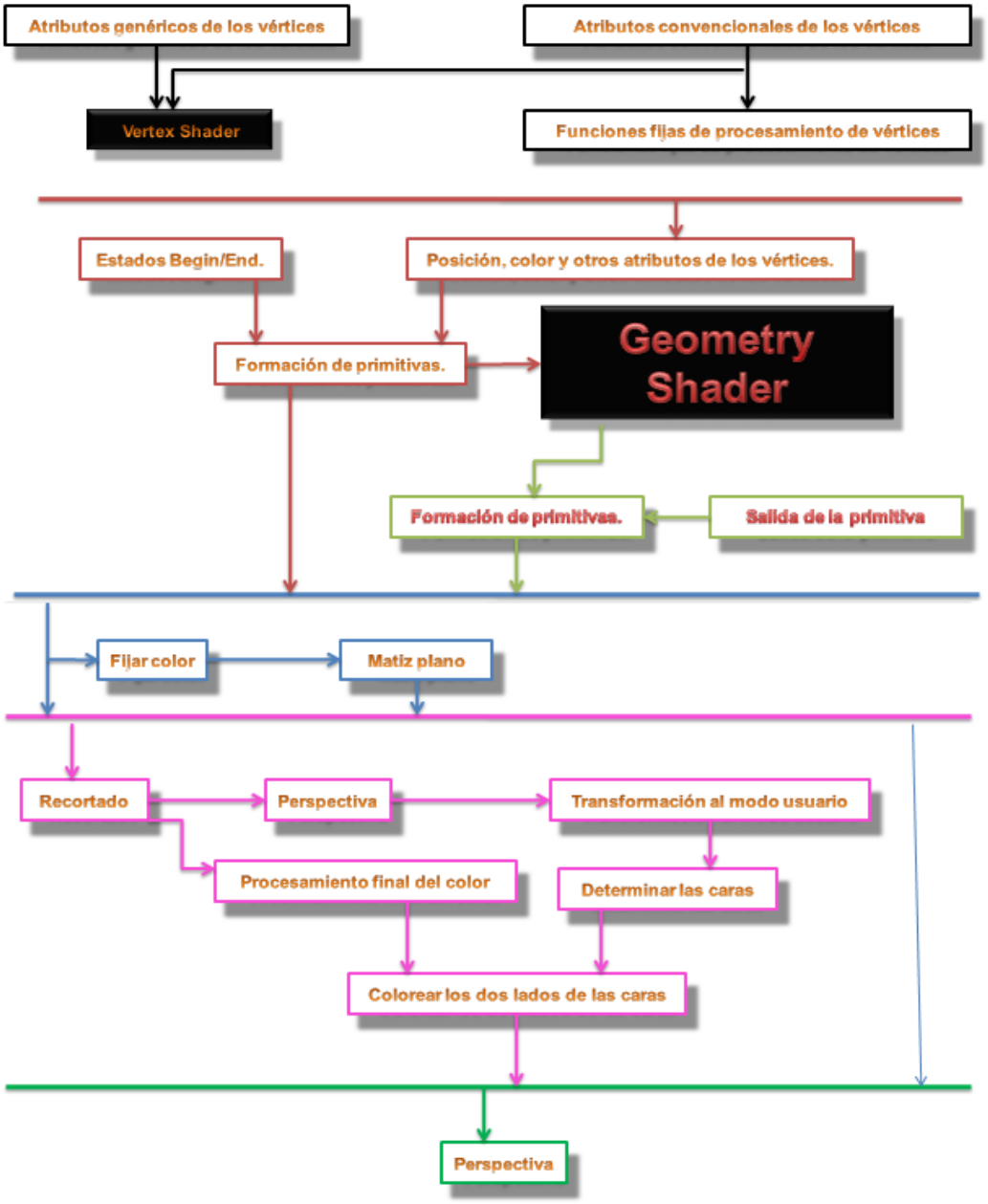


Figura 8. Vertex y Geometry Shader, dentro del pipeline de Opengl.

1.5.3 Fragment Shader.

Actualmente identificada como la última etapa programable dentro del pipeline gráfico, pues es la encargada del trabajo con las texturas y la generación de los píxeles que serán visualizados en pantalla.

El procesador de fragmentos ejecuta programas de fragmentos. En una GPU existen más de una unidad de procesamiento de fragmentos, cada una capaz de procesar un fragmento a la vez. Las entradas del programa de fragmentos son los atributos que surgen de hacer una interpolación lineal de los atributos de los vértices que le dieron origen al fragmento. Además de las instrucciones típicas disponibles en los programas de vértices, los de fragmentos agregan instrucciones para acceder a una textura mediante un par de coordenadas reales. Este mecanismo utiliza lo que se denominan “sampler objects” [18], que toman una muestra de la textura en base a la interpolación de los valores de los píxeles cercanos.

Este procesador cuenta también con un conjunto de registros de lectura/escritura para almacenar valores intermedios. Como salida, el procesador de fragmentos genera una serie de valores que representan atributos de un píxel de la imagen final como el color, la transparencia, etc. El Fragment Shader para calcular una iluminación por píxel, necesita la orientación del triángulo, la orientación del vector de luz, y en algunos casos la orientación del vector vista.

La GPU como bien es anunciado por su nombre es creada como acelerador gráfico y de esta forma librar a la CPU de estos menesteres. Pero no es esta la única utilización que se le brinda actualmente, pues su gran velocidad y paralelismo ha despertado la curiosidad de muchos y están siendo utilizadas para resolver problemas de propósito general [21, 22], y es aquí donde juega un papel fundamental la etapa de sombreado de fragmentos y específicamente las texturas que son utilizadas en esta etapa, tanto es así que se ha tratado durante la evolución del hardware de facilitar el trabajo con las texturas y por ende del mecanismo en general haciéndolo aún más programable.

Conclusiones del capítulo

Luego de analizado el estado del arte de la modelación de superficies 3D, se concluye después de haber investigado las características, funcionalidades, hasta incluso las fórmulas que se utilizan para lograr el suavizado, que la técnica que se utilizará en el presente Trabajo de Diploma será la Subdivisión de Superficies. Una vez elegida la SSs se enfatiza en la necesidad de usar la GPU, para obtener un mejor resultado. Finalmente se muestra una panorámica de los esquemas de refinamiento de mayor relevancia en la comunidad científica y las diferentes etapas programables de la GPU, de la cual se escogerá el Fragment Shader como etapa a sustituir dentro de la arquitectura de la GPU.

Capítulo 2.

Esquemas de Refinamiento.

Introducción

En el capítulo que se presenta a continuación se hará alusión a los principales esquemas de refinamiento que dan soporte a la técnica de subdivisión de superficies, se explicarán cada uno de ellos, en qué consisten y cómo facilitan el trabajo a la hora de definir y suavizar una superficie regular o irregular.

El análisis profundo de cada uno de los esquemas de refinamiento que existen permitirá establecer una comparación a partir de una serie de características y de esta forma elegir el esquema que sea más factible de acuerdo al problema que se compete.

2.1 Consideraciones iniciales.

Todos y cada uno de los esquemas de refinamiento antes mencionados y que serán el punto de partida para la explicación de este capítulo tienen peculiaridades que hacen que se utilice uno u otro en dependencia del objetivo a lograr, por ejemplo, existen esquemas interpolantes y aproximantes, los esquemas interpolantes como el Mariposa brindan superficies límites de continuidad C^1 en las proximidades regulares de la malla, mientras que otros como el Catmull-Clark y el Loop proporcionan continuidad C^2, o sea, se obtendrá una superficie de mayor suavidad con un menor número de iteraciones con un algoritmo aproximante. Por lo que serán obviados los esquemas interpolantes, pues la necesidad es suavizar una superficie en la menor cantidad de iteraciones posibles.

No todos los esquemas aproximantes, como fue visto en el capítulo anterior, tienen continuidad C^2, y es el caso del Doo-Sabin que posee continuidad C^1. Por esta razón dicho esquema no será analizado dentro de los esquemas aproximantes.

2.2 Máscaras.

Una máscara es la aplicación del esquema de forma local, o sea, particularmente en una región específica delimitada según el esquema de refinamiento que se esté utilizando. Las máscaras están formadas por un grupo de coeficientes que acompañan a los vértices e indican cómo se construyen los nuevos puntos a partir de los anteriores. En la **Figura 9** se muestran algunas máscaras, luego durante el desarrollo del presente trabajo, nuevas máscaras serán incorporadas.

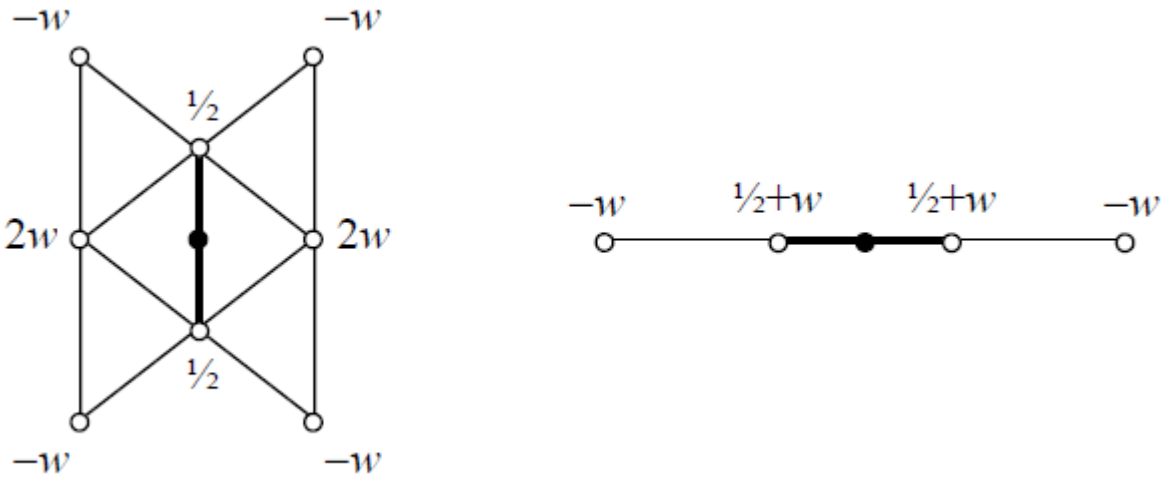
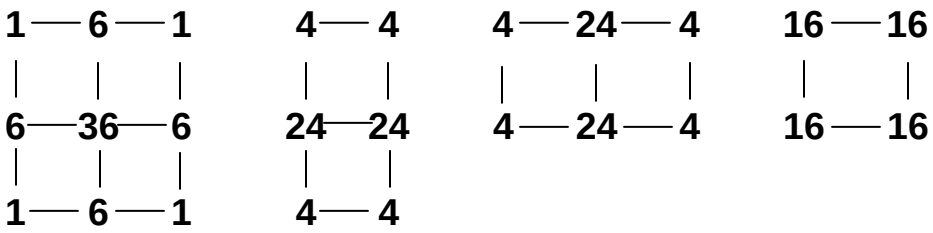


Figura 9. Máscaras del esquema interpolante de la Mariposa.
(Izquierda: para aristas interiores; derecha: para aristas fronterizas o vivas).

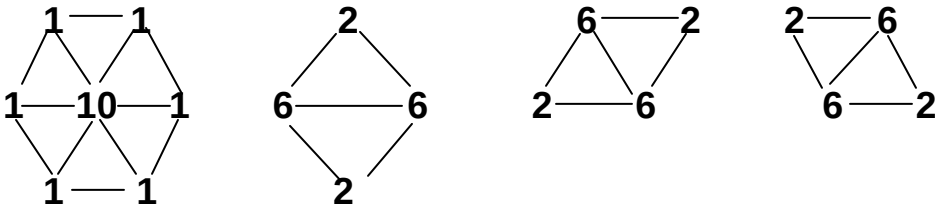
Los esquemas que han prevalecido hasta el momento en el estudio que se realiza son: el esquema elaborado por Catmull y Clark y el de Loop, de los cuales se hará un análisis de las máscaras que presentan.

2.2.1 Máscaras Regulares.

El esquema de Catmull-Clark, se aplica sobre cuadriláteros, la malla debe estar formada por cuadriláteros, aunque también son aceptados triángulos, siendo mucho más general, las máscaras regulares pertenecientes al mismo son:



En el caso del Loop que es un tipo de esquema aplicado a mallas triangulares, se tienen las máscaras siguientes:



En la práctica, estas máscaras no son utilizadas en esta forma, sino que son obtenidas a través de la incorporación de una variable de peso en las máscaras genéricas, dichas variables dan un nivel de continuidad y hacen que la superficie resultante tenga resultados diferentes para distintos pesos, la suma de los coeficientes debe ser la unidad para que el esquema se mantenga invariable ante traslaciones y rotaciones.

En las Figuras 10 y 11 son mostradas las máscaras utilizadas para los esquemas de refinamiento, Catmull-Clark [23] y Loop [24] respectivamente.

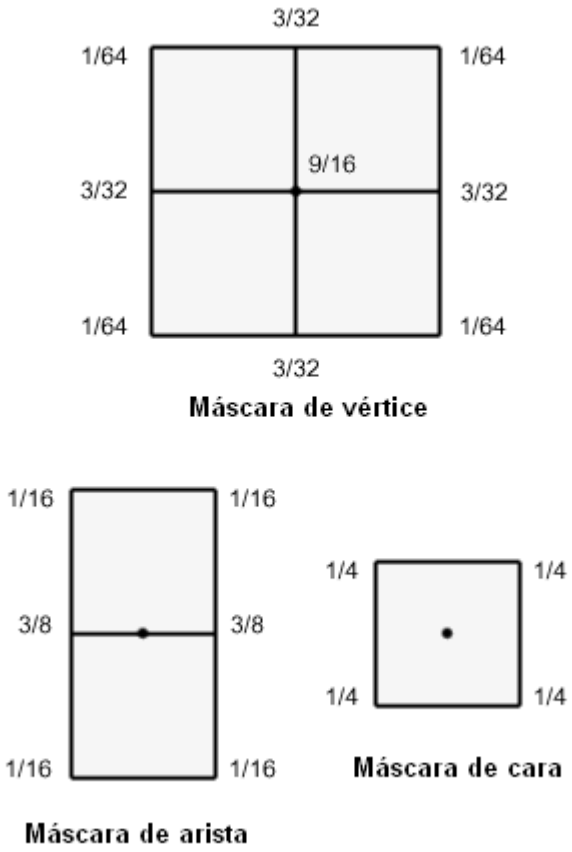


Figura 10. Máscaras del esquema de Catmull-Clark para casos de regularidad con una variable de peso de 1/64.

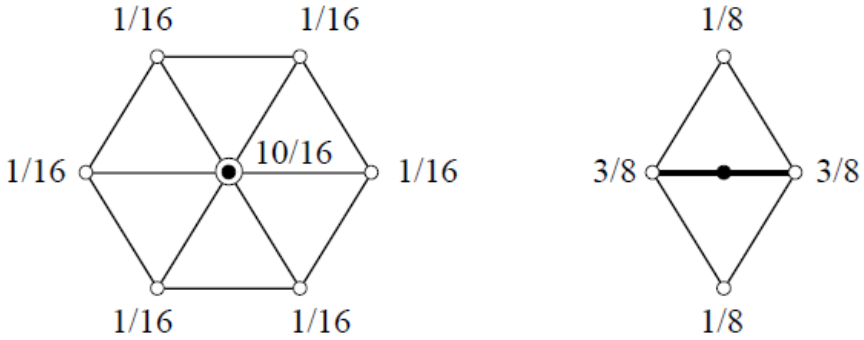


Figura 11. Máscaras del esquema aproximante de Loop, con variable de peso 1/16.
(Izquierda: para recolocar vértices antiguos; derecha: para aristas interiores)

2.2.2 Máscaras Irregulares.

Hasta el momento se ha hablado de subdivisión, esquemas de refinamiento y máscaras de estos esquemas pero sólo en partes regulares de la malla, o sea, en lugares donde el esquema encaja perfectamente y la continuidad es la esperada. Luego se demostró que existían lugares donde los esquemas no actuaban correctamente, por lo cual, se realizaron nuevas máscaras para estos casos atípicos y se fueron agrupando para dar cierta generalidad, y es el caso de la **Figura 13**, donde se muestra la máscara para una arista fronteriza perteneciente a una malla abierta. No solo fueron encontrados casos degenerados en las mallas abiertas, sino también en los poliedros, donde existe un tipo de irregularidad que en el modelado 3D generalmente se encuentra presente, y es la irregularidad en algunos vértices interiores, estos vértices serían las esquinas del objeto y son llamados vértices extraordinarios, en la **Figura 12** se muestra un caso de este tipo de vértice para el esquema de Catmull-Clark.

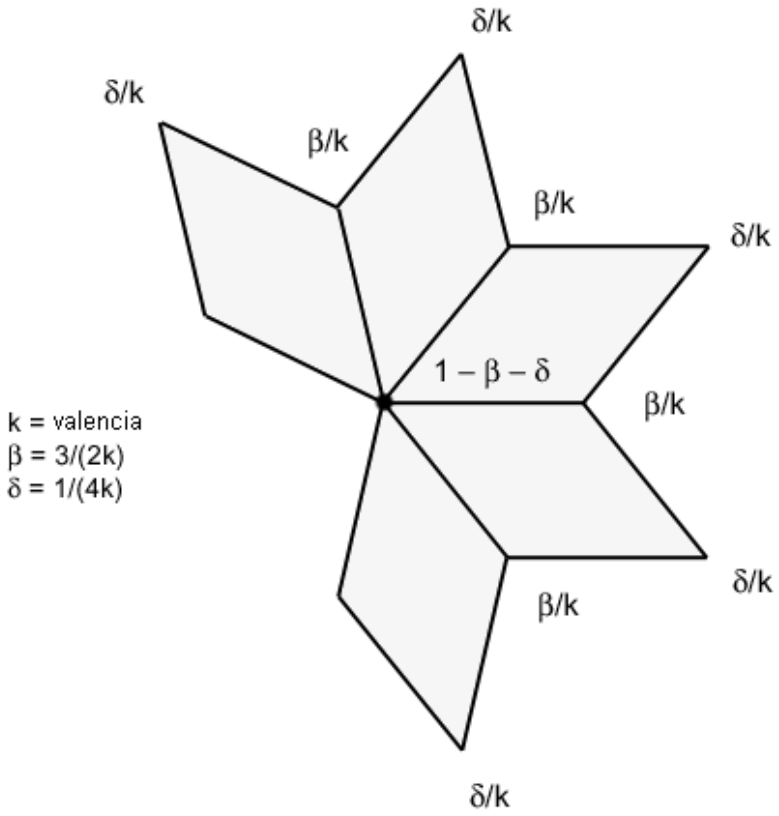


Figura 12. Máscara para vértices extraordinarios del esquema Catmull-Clark.

En esta formación de mallas, el problema que existe es que no todas poseen sus vértices, aristas y caras regulares, o sea, hasta el momento, en el caso de Catmull-Clark, todos los vértices poseían 4 aristas que los comunicaban con 4 vértices (existían 4 caras de las que formaban parte), esto se conoce como la valencia del vértice y muchas veces se denota por *K*, en el caso de regularidad, *K* = 4. Las caras regulares están definidas por 4 vértices y existen casos donde no es posible definir la cara de esta forma y es necesario usar triángulos formados por sólo 3 vértices, apareciendo otro caso de irregularidad.

En el caso de Loop la valencia de los vértices regulares es de 6 (*K* = 6) y los vértices irregulares serían todos los que no cumplan esta condición. Para las mallas abiertas las máscaras utilizadas son las mismas que para el esquema de Catmull-Clark, **Figura 13.**

El trabajo con la SSs asegura que una vez que es incluido un nuevo vértice en la malla este tiene la propiedad de regularidad. Por lo que los vértices irregulares serán encontrados en la primera iteración del algoritmo posibilitando la identificación inmediata de los mismos.

La irregularidad fronteriza mostrada en la **Figura 13** es utilizada cuando las mallas son abiertas, en este caso se trabajará con poliedros, pero si es necesario mencionarlas para futuros trabajos sobre el tema.

Las máscaras irregulares que se les proporciona a continuación contienen la variable de peso incluida en su notación.

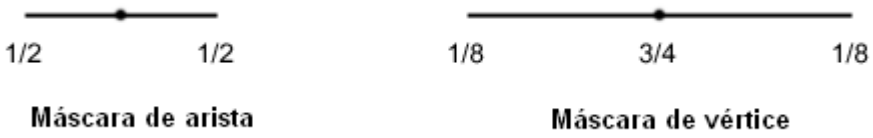


Figura 13. Irregularidad fronteriza (Catmull-Clark y Loop).

Analizando el caso de irregularidad en el esquema de Loop, al igual que en el de Catmull-Clark, es necesario utilizar una máscara diferente, es decir, para recolocar un vértice extraordinario de valencia $K \neq 6$, se ha de tomar, por mor de simetría, β para cada uno de sus vecinos y $1-K$ β para él mismo, existiendo varias propuestas para β [13, 25] **Figura 14.**

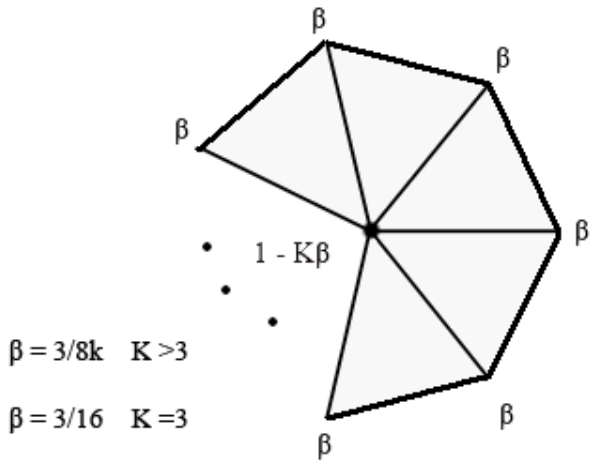


Figura 14. Máscaras Irregulares para vértices extraordinarios. (Loop).

2.3 Esténciles.

En la terminología, aparece usualmente relacionado el concepto de máscara con el de esténcil [26], sin embargo, en formación topológica son exactamente iguales. Los esténciles son utilizados para almacenar los identificadores de los vértices vecinos de la malla que contribuyen en la formación del nuevo vértice y las máscaras, como se explicó anteriormente, están formadas por los coeficientes que acompañan a cada uno de los vértices vecinos.

Cuando es usado algún esquema de refinamiento para conseguir los resultados esperados en la superficie límite, es necesario "combinar" estos dos conceptos para computar las nuevas posiciones de los vértices en la geometría, pero constituye un error "mezclarlos" en una misma definición.

2.3.2 Esténciles Regulares.

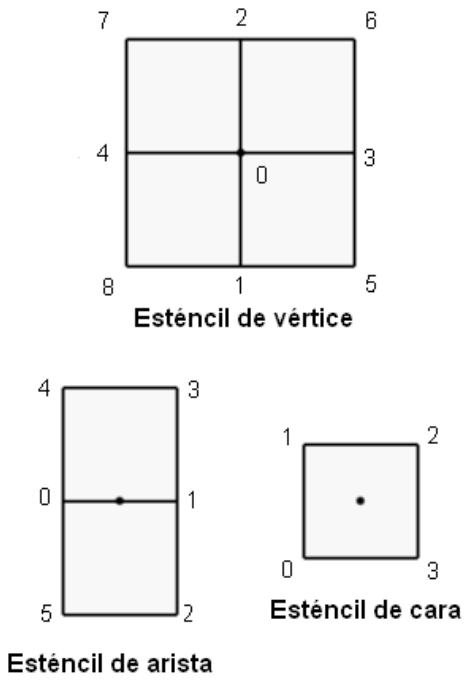


Figura 15. Esténciles del esquema de Catmull-Clark para casos de regularidad.

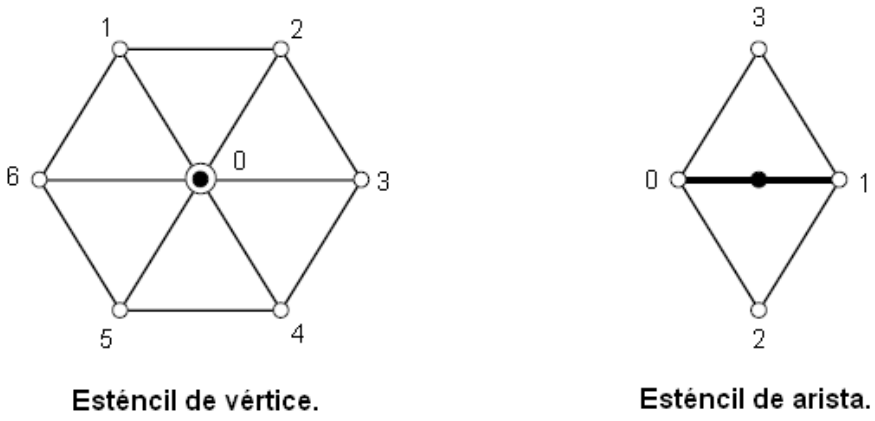


Figura 16. Esténciles regulares para el esquema de Loop.

2.3.2 Esténciles Irregulares.

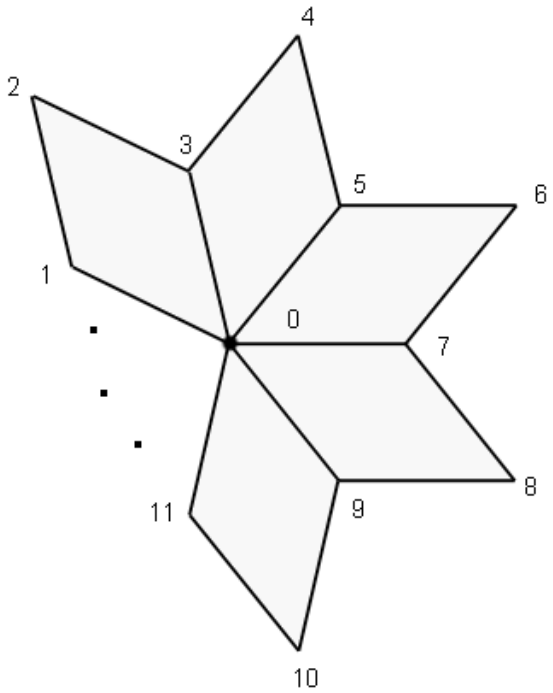


Figura 17. Esténcil de vértice extraordinario para el esquema de Catmull-Clark

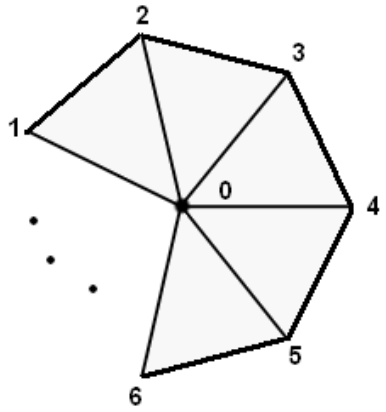


Figura 18. Esténcil de vértice extraordinario para el esquema de Loop.

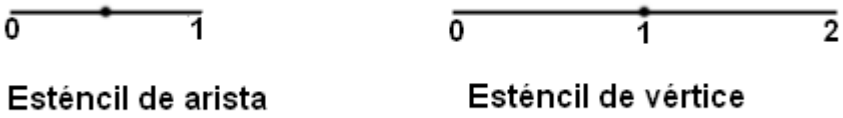


Figura 19. Esténcil para mallas abiertas.
(Catmull-Clark y Loop).

2.4 Aplicaciones.

CPU

Los trabajos más significantes hechos hasta el momento fueron realizados en la CPU, y aunque no se hayan aprovechado al máximo las potencialidades de los algoritmos de subdivisión, si es necesario destacar la importancia de su ocurrencia ya que se ha logrado obtener una mayor experiencia en su utilización, y se han descubierto nuevos casos degenerados, como las irregularidades fronterizas, donde los algoritmos no logran en la superficie límite la continuidad esperada. Por esta razón es que hoy se cuentan con nuevas máscaras para trabajar todos estos casos extraordinarios. La diferencia entre las primeras máscaras y las que existen hoy, son numéricamente significativas.

Si bien es cierto que las referencias que se tienen de los trabajos hechos hasta el momento están en idioma inglés, vale destacar que han sido los más trascendentales y que han marcado etapas significativas dentro de la evolución de los algoritmos de subdivisión.

El primero de estos trabajos en el caso de Catmull-Clark fue el hecho por estos mismos autores en el año 1978 [3], el cual introduce la técnica y entre sus aplicaciones anuncian la posibilidad de generar superficies suavizadas e incluso la incorporan al espacio tridimensional. Aunque los ejemplos mostrados como resultados del trabajo eran bastante sencillos, si hay que destacar que fueron el punto de partida para los descubrimientos posteriores, no olvidar al esquema Doo-Sabin [4], como otro prócer de esta naciente técnica de modelado. Los siguientes trabajos que fueron apareciendo, trataron de hacer aún más sencillo y general el trabajo realizado por los antes mencionados pioneros de esta técnica. Entre ellos se destaca el elaborado por Kari Pulli y Mark Segal, en 1996 [27] el cual muestra una implementación del algoritmo Loop con el fin de potenciar el renderizado de la geometría, en esta solución de Pulli se habla por primera vez de llevar la topología de la malla a un arreglo de 2D para resolver un problema de subdivisión y aunque solo tratan los casos de regularidad es planteada la de ampliar la generación a mallas de topología irregular. Seguidamente aparece el estadounidense Jos Stam, el cual introduce las funciones básicas, las cuales permitían trabajar con las máscaras del esquema de Catmull-Clark de forma directa, pudiendo calcular el nivel de subdivisión deseado directamente, conociendo de antemano cuanto influye cada vértice de la malla inicial en la formación de los nuevos vértices de la superficie límite [28], en ese mismo año el autor publica otro artículo con el mismo algoritmo pero utilizando el esquema de refinamiento introducido en 1987 por Loop [29]. Las funciones básicas eran cargadas en una matriz y multiplicadas por las posiciones de los vértices de la malla, haciendo coincidir cada vértice con su determinado coeficiente de la máscara, la ventaja era que las funciones básicas eran reutilizables, gracias al carácter local del algoritmo, podían cargarse desde un fichero para ser utilizadas en cualquier aplicación.

Los trabajos venideros estuvieron enfocados a lograr mayores velocidades de ejecución y poder incorporar así estos algoritmos en aplicaciones afectando en la menor medida posible su rendimiento, evidenciado en trabajos como el de Hong Qin y Baba C. Vemuri [30], en 1998, muestran una implementación del esquema de refinamiento Catmull-Clark, logrando un nuevo modelo dinámico de superficie, aprovechando la funcionalidad brindada por la SSs de realizar transformaciones en los vértices

de la malla sin que esto afecte la superficie límite obtenida ni la topología de la malla, otro de los trabajos que muestran el uso de estos algoritmos es el propuesto por Jeffrey Bolz y Peter Schröder [23], en el año 2002, el cual introduce una nueva forma de trabajo con las funciones básicas, mencionadas anteriormente, utilizando el Catmull-Clark como esquema base de refinamiento. Lo trascendental de la arquitectura de subdivisión mostrada en este artículo es la ejecución en paralelo del esquema de subdivisión, abriendo las puertas a una nueva etapa en la historia, donde se comienza a pensar en la utilización de la GPU como una necesaria variante si de rapidez se trata.

GPU.

Los logros descritos hasta el momento, logrados en la CPU, tenían una desventaja en común: el no aprovechamiento del paralelismo brindado por el carácter local de los algoritmos de subdivisión. Es entonces que se hace necesario para lograr subdividir con una mayor velocidad de cómputo, encontrar alguna variante que permitiese el deseado paralelismo. Fueron entonces centro de atención las GPU, cuya arquitectura estaba diseñada para trabajar en paralelo, a través de sus múltiples núcleos de procesamiento. Es por esta razón que los trabajos que se realizan hoy en día, se hacen diseñados sobre el antes mencionado hardware.

Entre los principales logros, y que han servido de punto de partida para el desarrollo de este trabajo de diploma, aparece nuevamente un artículo de Jeffrey Bolz y Peter Schröder [31], pero esta vez publicado en el 2007. El artículo propone una implementación en el Fragment Shader utilizando las funciones básicas, introducidas en su trabajo del 2002, las cuales eran cargadas inicialmente en texturas y sólo era necesario el envío de los vértices o puntos de control. Con la única diferencia que además de proveer una mayor velocidad de cómputo, garantizaban una impermeable superficie límite (no existían agujeros en la superficie subdividida), propiedad que no presentaban muchos de los trabajos realizados anteriormente, incluyendo la publicación del 2002. Es importante destacar en esta técnica de implementación la introducción al análisis de la simetría encontrada en algunos lugares de la malla, permitiendo no subdividir toda la malla, sino primeramente identificar simetrías para luego hacer la subdivisión en una ocurrencia por simetría encontrada.

Algo muy difícil de lograr cuando se subdivide en la GPU en profundidad, o sea, de forma iterativa, es lograr llevar una geometría, dígase la topología de la malla, a una matriz 2D, pues es necesario enviar los

datos a través de texturas. Existe publicaciones sobre el tema como la de Xianfeng Gu, Steven J. Gortler y Hugues Hoppe [32], del 2002, en la cual se lleva a un espacio 2D cualquier geometría 3D, haciendo un corte de la malla a través de alguna de sus aristas y todo esto es parametrizado a un cuadrilátero, de igual forma son almacenadas las normales y coordenadas de textura. La técnica anteriormente expuesta permite el trabajo con mallas irregulares de topología arbitraria, creando una estructura regular completa, a diferencia de las elaboradas hasta ese momento que sólo obtenían estructuras semi-regulares.

En 2007, vuelve a aparecer Charles Loop [33] con un algoritmo basado en el esquema de refinamiento de Catmull-Clark, el cual no subdividía la superficie de forma completa sino que identificaba los parches que debían ser sometidos al proceso de subdivisión. Esta implementación podía ser migrada a la GPU, con la condición de obtener futuros dispositivos gráficos, éstos últimos necesariamente debían cumplir con una arquitectura que es brindada por el autor al final del artículo.

Juraj Konečný [15], compara la subdivisión en la CPU contra la realizada en la GPU, basando su investigación específicamente en el trabajo sólo con vértices regulares, obviando así los demás tipos de irregularidad, pero no obstante brinda la posibilidad de constatar algo que se pensaba desde hace algún tiempo y era que: *la subdivisión en la GPU es mucho más rápida que la subdivisión en la CPU*.

Conclusiones del capítulo

Con el capítulo presentado quedan fundamentados los cimientos que servirán para darle cumplimiento al objetivo trazado para este trabajo. Una vez hecho alusión a los principales esquemas de refinamiento que dan soporte a la Subdivisión de Superficies, estudiada sus peculiaridades, y de ser realizada una comparación entre ellos basándose en sus características, se concluye que el esquema de refinamiento más factible para resolver el problema que compete es el de Catmull-Clark, incluyendo las máscaras correspondientes a este esquema. El discriminante utilizado en este caso, para decidir entre el esquema de refinamiento de Loop y el de Catmull-Clark, no fue precisamente el nivel de suavidad presentado por la superficie resultante, sino la topología de la malla inicial, puesto que aunque el esquema de Catmull-Clark está diseñado para el trabajo con cuadriláteros, acepta de igual forma triángulos en la malla inicial, haciéndolo más general y proporcionando una variante estándar para resolver cualquier tipo de problemas, si de subdivisión se trata.

Capítulo 3.

Solución Propuesta.

Introducción

En el siguiente capítulo se pondrá en consideración el resultado de este Trabajo de Diploma para darle cumplimiento al objetivo trazado inicialmente. Se detallará minuciosamente el conjunto de acciones necesarias para utilizar la herramienta propuesta y se explicará brevemente el trabajo con las bibliotecas utilizadas como soporte para la obtención de la herramienta.

3.1 Objetivo de Automatización.

Con la realización de esta investigación se pretende automatizar el modelado de objetos en 3D, permitiendo a los grafistas y desarrolladores el trabajo con mallas burdas de baja resolución, optimizando así la carga de los ficheros contenedores de escenas 3D. Posibilitando además la generación de objetos de varias resoluciones en tiempo real.

3.2 Información que se maneja.

La información manejada será referente al conjunto de valores y propiedades que poseen los modelos 3D, dígase posición de los vértices, información de las diferentes caras, así como puede incluirse también en dependencia del tipo de fichero, la información de las normales y coordenadas de textura.

3.3 Propuesta del Sistema.

Se propone la implementación en la GPU de un algoritmo de SSs que utiliza como esquema de refinamiento base el de Catmull-Clark, permitiendo al usuario lograr la subdivisión de objetos 3D y poder incorporar la solución a la aplicación de gráfico por computadora que esté desarrollando.

Las principales propiedades de los objetos que podrán ser modificadas serán las posiciones de los vértices de la malla inicial.

Como lenguaje de programación en la GPU se utiliza *(C for graphic,* Cg por sus siglas en inglés*)*, pues es capaz de insertarse en los pipelines de dos de las APIs gráficas más usadas por estos días Direct3D y OpenGL, posibilitando un amplio uso de la tecnología y la adición de futuros trabajos sobre el tema de la SSs. El hardware utilizado para darle solución al objetivo de la investigación es una NVIDIA GeForce 9800 GT y *cg* constituye el lenguaje de programación que NVIDIA elaboró para el trabajo con su hardware [34].

La herramienta va acompañada de una propuesta de utilización, mostrando el enlace y flujo de datos entre la CPU y la GPU. Se señala además que esta no es la única forma de utilizar la herramienta sólo es una propuesta base, que puede ser enriquecida en dependencia del trabajo y el tipo de aplicación gráfica que se quiera desarrollar.

En la CPU el lenguaje de programación C++ es seleccionado debido a su eficiencia y comodidad al trabajar los gráficos por computadora, es un lenguaje versátil, potente, robusto, libre e imperativo. Soporta paradigmas como el de programación orientada a objetos, programación estructurada y programación genérica. Presenta un grupo de cualidades como el trabajo con plantillas (templates), sobrecarga de operadores e identificación de tipos en tiempo de ejecución (RTTI). Muy usado por la mayoría de los desarrolladores de software gráfico en el mundo y por los proyectos de aplicaciones gráficas de la Universidad de las Ciencias Informáticas, específicamente en la Facultad 5.

Computational Geometry Algorithms Library (CGAL, por sus siglas en inglés) [26] es una biblioteca robusta y eficiente de estructuras de datos y algoritmos geométricos implementada en C++. La librería es multiplataforma y tiene licencia de libre distribución “Open Source” desde su release 3.0 (2003). Es utilizada específicamente para la carga y trabajo inicial con el fichero que posee la información del objeto 3D. La biblioteca brinda la posibilidad de trabajar con una de las estructuras de malla más eficiente conocida como *HalfEdges* y de amplia utilización en otros trabajos de SSs.

Objet File Format (OFF, por sus siglas en inglés) [26, 35] es el tipo de fichero utilizado para obtener la información del objeto 3D, el formato almacena la información de tres elementos: vértices, caras y aristas, sólo serán utilizados los dos primeros. Este tipo de fichero se caracteriza por su simple estructura, idónea para el objetivo que se persigue, un ejemplo es mostrado en la **Figura 20**.

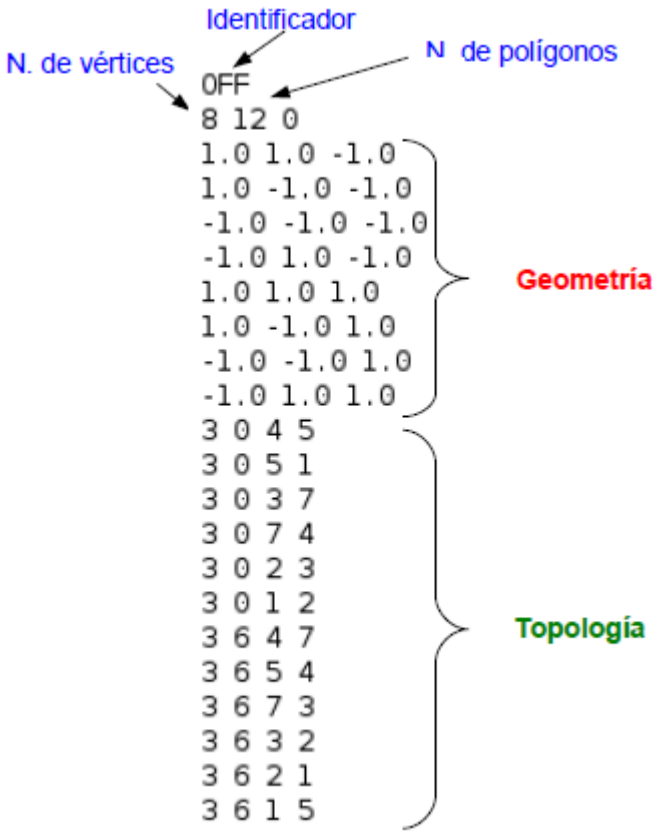


Figura 20. Formato de archivos OFF.

Como Entorno de Desarrollo Integrado (*Integrated Development Environment,* IDE por sus siglas en inglés), Microsoft Visual Studio 2008 es utilizado para sistemas operativos Windows. Permite a los desarrolladores crear rápidamente aplicaciones de alta calidad y riqueza, pues cuenta con un conjunto de herramientas de desarrollo profesional, las cuales conforman un sistema altamente productivo. Proporciona un poderoso compilador para el lenguaje seleccionado y permite además la adición de *plugins* ampliando el trabajo e incorporando otras fuentes para el desarrollo.

OpenGL constituye la API seleccionada, a diferencia de Direct3D es multiplataforma, esto permitirá la fácil migración del trabajo realizado sobre Windows para cualquier sistema UNIX, y redefine tamaño de punto y ancho de línea.

Se brinda a continuación el conjunto de pasos necesarios con el objetivo de crear las precondiciones de utilización del algoritmo propuesto, ilustrados en la **Figura 21.**

1. Subdividir una vez en la CPU, por lo que cada vértice irregular estará rodeado por un anillo de vértices regulares.

2. Construir un arreglo unidimensional con la información de la malla subdividida, consintiendo en los vértices originales re-posicionados y dos anillos que lo rodean (*Fragment Mesh*), donde cada *Fragment Mesh,* formará una de las filas que compondrán la textura 2D que será enviada al procesador de fragmentos en la GPU.

3. En el procesador de fragmentos se accede a las posiciones de texturas correspondientes a cada uno de los nuevos vértices que serán obtenidos en ese nivel de subdivisión. Para conocer los vecinos necesarios y obtener los nuevos vértices, la información de los índices de dichos vecinos deberá ser almacenada en tablas *Look-Up*.

4. El tercer paso es repetido tantas veces como niveles de subdivisión se deseen, teniendo presente que una vez que el modelo llega a la GPU poseen un nivel 1 de subdivisión, correspondiente a la subdivisión en la CPU. Los resultados obtenidos en cada nivel constituyen la entrada del siguiente.

5. Por último se descarta el anillo externo de cada *Fragment Mesh*, obteniendo así una malla impermeable sin agujeros en su estructura.

3.3.1 Fragment Mesh.

Para cumplir el objetivo propuesto es utilizada una estructura de almacenamiento de malla, conocida como *Fragment Mesh*, ideada por los señores Le-Jeng Shiue, Ian Jones y Jörg Peters [36] en el 2007. Dicha estructura permite tratar al objeto 3D como una "*Geometry Images*" [32], "Imágenes Geométricas" en español, pero de una forma más compleja.

La principal ventaja de esta estructura es que fue ideada para trabajar la SSs y permite:

- Reutilizar los esténciles de un nivel de subdivisión anterior en el actual nivel, siendo sólo necesario incorporar los nuevos esténciles al final de la lista, que a su vez serán reutilizados en el próximo nivel.
- La forma en que se almacena cada *fragmento,* alberga la conectividad de la malla en sí. No siendo necesario tomar medida alguna de mantenimiento, las cuales son muy costosas a la hora de trabajar los gráficos por computadora.
- Permite llevar la malla a un espacio imagen 2D de una forma completa.

Tabla 4 Comparación entre las diferentes implementaciones de SSs.

	Flexibilidad			Eficiencia		
	n-gon	Dual/√3	Adaptabilidad	Tiempo	Espacio	Localidad
HalfEdge	sí	sí	sí	lento	peor	no
QuadTree	no	no	sí	lento	peor	no
Patch - based	no	no	Por parche	rápido	mejor	sí
Frag - mesh	sí	sí	Por fragmento	rápido	mejor	sí

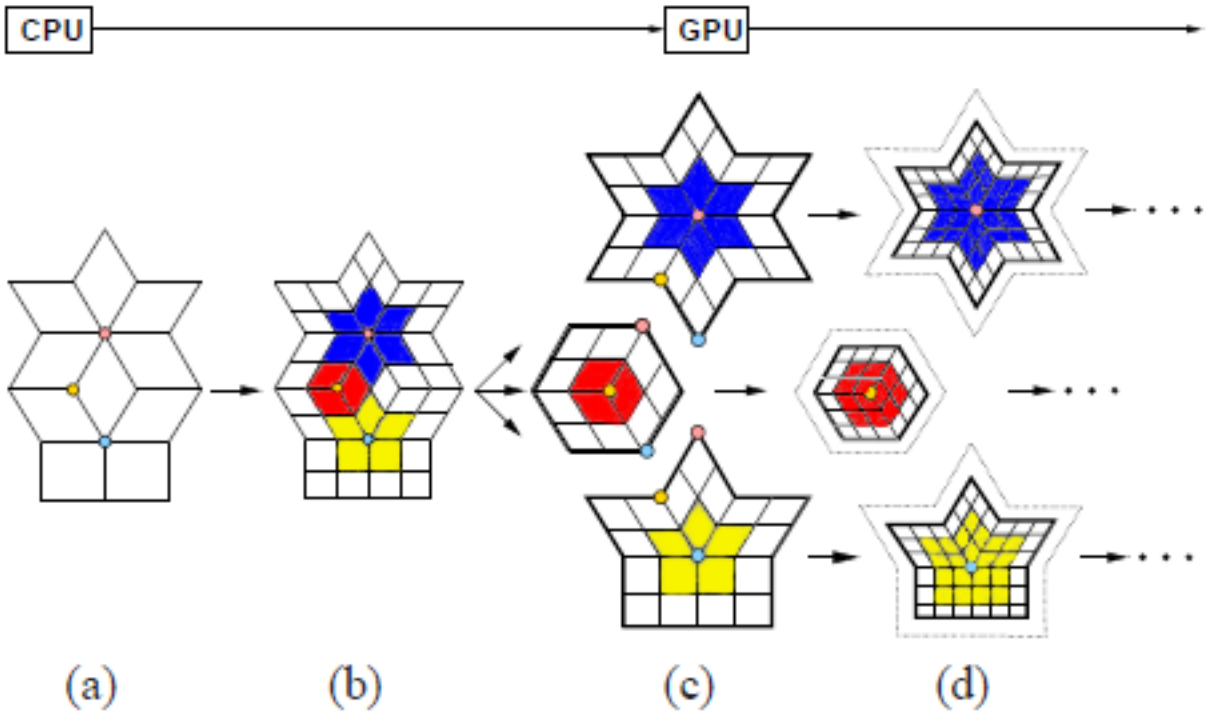


Figura 21. Flujo de ejecución de la implementación de utilización del algoritmo.

En la **Figura 21** se muestra en que consiste físicamente un *Fragment Mesh* donde cada vértice centro del fragmento, se encuentra rodeado por dos anillos de vértices vecinos. La SSs se realizará a nivel de fragmento y debido a que un fragmento solapa a sus fragmentos vecinos, esto evita la ocurrencia de agujeros en la superficie límite ya subdividida.

La forma de almacenar los vértices de un fragmento, se hace comenzando por el vértice que constituye el centro del mismo, luego se continúa con los restantes vértices pertenecientes a cada uno de los anillos.

Los esténciles son almacenados de forma similar, teniendo en cuenta que durante su formación deberán almacenarse en el mismo orden que tendrán los vértices nuevos en el fragmento. Por ejemplo, el primer vértice del fragmento siempre es su centro, pues entonces, el primer esténcil en cada una de las pasadas de render siempre va a ser el correspondiente a la formación del vértice centro. Por otro lado, no será necesario re-calcular ninguno de los esténciles ya obtenidos en anteriores niveles de subdivisión, pues los

coeficientes de los vértices del fragmento que forman los esténciles “serán los mismos”. Otra información de necesario almacenamiento es la *valencia*, que coincide con la valencia del vértice centro.

Como se puede apreciar cada fragmento alberga en sí su conectividad; la información, por ejemplo, del primer vértice del anillo a, sería: sumar la cantidad de vértices de todos los anillos hasta a, más el vértice centro, donde la cantidad de vértices del anillo a se calcula mediante la fórmula: $valencia * a * 2$.

Es válido destacar que la estructura representada no sólo soporta el esquema de refinamiento Catmull-Clark sino que puede ser utilizada en cualquiera de los esquemas de refinamiento existentes.

3.3.2 Técnica de Ping Pong.

Cuando se trabaja en el procesador de fragmentos para enviar datos y recibir datos de forma recursiva utilizando texturas, se puede implementar la técnica Ping-Pong [37].

La técnica permite utilizar como textura de entrada de la iteración *i* la textura resultado de la iteración *i-1*, sólo basta hacer un cambio entre las texturas de lectura y escritura. Esta técnica puede ser implementada de tres formas distintas:

1. Utilizar FBOs distintos con asociaciones a cada una de las texturas a las que se renderiza, y cambiarlos con glBindFramebufferEXT().
2. Utilizar un FBO, reasociar la textura del render target en cada pasada utilizando glFramebufferTexture2DEXT ().
3. Utilizar un FBO con múltiples puntos de asociación, e intercambiarlos con glDrawBuffer().

Las tarjetas actuales, como la utilizada en la solución, soportan hasta ocho puntos de asociación por cada FBO y las más antiguas hasta cuatro, no existiendo ningún inconveniente para la utilización de la tercera opción, que constituye la más rápida y eficiente [38].

En el problema que compete a este trabajo de diploma, la técnica antes mencionada, se utiliza almacenando en las texturas los vértices actuales que sirven como entrada y constituyen la textura de la cual se va a leer. Los nuevos vértices, resultado del nivel de subdivisión actual, quedarán almacenados en

la textura en la que se escribió durante la pasada de render. La textura de escritura será la de lectura en la próxima pasada o se tomará como resultado final si ya se ha logrado el nivel de subdivisión deseado.

A continuación se muestran las características del hardware donde se elaboró la implementación del esquema de refinamiento Catmull-Clark.

- Tarjeta de Video NVIDIA GeForce 9800 GT, 512 MB de memoria y contando con drivers versión 190.38.
- Procesador de la generación Pentium 4 de Intel a 3.0 GHz y 1 Gb de memoria RAM.
- Sistema Operativo Windows 7.

3.4 Resultados.

En las pruebas realizadas, la solución propuesta fue comparada con una de las implementaciones más rápidas con las que se cuenta actualmente en la CPU, que puede ser encontrada en la biblioteca CGAL, descrita anteriormente. En todos los casos de prueba realizados, los modelos fueron subdivididos hasta obtener una superficie límite de nivel 5. Obteniéndose los resultados mostrados en la Tabla 5, donde $\mathbf{T_{CPU}^{1}}$ es el tiempo que se demora en alcanzar el primer nivel de subdivisión en la CPU, $\mathbf{T_{GPU}}$ se refiere al tiempo obtenido al subdividir los restantes niveles en la GPU y finalmente $\mathbf{T_{CPU}^{2}}$ es el tiempo logrado en la CPU para alcanzar el nivel deseado.

Tabla 5 Resultados comparativos obtenidos (nivel 5 de subdivisión).

Fragmentos	Caras	Vértices	T_{CPU}^{1}	T_{GPU}	T_{CPU}^{2}
8	6144	6146	0.015	0.031	2.594
8	6144	6146	0.015	0.032	2.593
16	14336	14338	0.015	0.094	6.844
16	14336	14338	0.015	0.078	6.094
20	18432	18434	0.016	0.047	7.828
65	69632	69628	0.097	0.078	30.719
581	503808	504482	0.594	0.703	248.281

Conclusiones del capítulo

En el capítulo que concluye se presentó como resultado obtenido de la investigación realizada un algoritmo de SSs, utilizando como esquema de refinamiento base el Catmull-Clark. El esquema seleccionado fue implementado en la GPU. Se sustituyó, específicamente dentro de la arquitectura del hardware antes mencionado, la etapa de sombreado de fragmentos. La implementación propuesta incluye además una estructura, *Fragment Mesh*, elaborada y potencializada para el trabajo con la SSs en la GPU.

Conclusiones

Luego de una ardua investigación de la bibliografía que relaciona el tema tratado, mayoritariamente en idioma Inglés y de alto contenido teórico, fue seleccionada, para resolver el problema planteado inicialmente, la SSs utilizando la GPU. Posterior al análisis de los esquemas de refinamiento existentes y las diferentes etapas programables dentro de la GPU, se decidió utilizar el esquema de Catmull-Clark y sustituir la etapa de sombreado de fragmentos pues es la única etapa que actualmente permite la recolección de los datos de salida.

Por consiguiente se obtiene un shader para la SSs, que permite obtener poliedros más detallados a través de un objeto inicial burdo, lográndose resultados de subdivisión en tiempo real gracias al inherente paralelismo proporcionado por la GPU y a que técnicamente se obtiene el próximo nivel de subdivisión en solo una pasada de render, esto último, logrado mediante la incorporación en solo un shader, del cálculo de los vértices extraordinarios y de los vértices regulares. De igual forma la solución identifica como centro de un fragmento no sólo a los vértices irregulares, sino también a los vértices de la malla inicial que posean valencia cuatro ($K = 4$), aumentando de esta forma la velocidad de cómputo, pues apoya la posibilidad de resolver un nivel de subdivisión en una pasada de render.

Recomendaciones

- Utilizar el algoritmo propuesto en una aplicación de gráfico por computadora, que aproveche las potencialidades descritas, beneficiándose de la velocidad de cómputo lograda.
- Incluir al algoritmo el tratamiento de mallas abiertas.
- Rediseñar el algoritmo propuesto para ser utilizado en la etapa Geometry Shader, una vez sea liberada una salida directa similar a la de Fragment Shader.

Bibliografía Referenciada.

[1]. **ATI.** http://ati.amd.com/companyinfo/press/1996/mar.html. [Online] marzo 31, 1996.

[2]. **Palazzesl, Arie.** *GPU vs. cifrado WiFi: ¿Adivina quién gana?***.** 2008.

[3]. **Catmull, Edwin and Clark, Jim.** *Recursively generated B-Splines surfaces on arbitrary topological meshes.* Old Westbury, New York : s.n., 1978, pp. 350---354.

[4]. **DOO, D and SABIN, M.** *Behaviour of recursive division surfaces near extraordinary points.* Septiembre 10, 1978, Computer Aided Design, pp. 356–360.

[5]. **DeRose, Tony, Kass, Michael and Truong, Tien.** *Subdivision Surfaces in Character Animation.* s.l. : SIGGRAPH '98 Conference Proceedings, 1998, pp. 85–94.

[6]. **BIERMANN, H, et al., et al.** *Cut-and-paste editing of multiresolution surfaces.* s.l. : SIGGRAPH '02 Conference Proceedings, ., 2002, pp. 312–321.

[7]. Qué significa NURBS? *Rhinoceros NURBS modeling for windows.* [Online] http://www.es.rhino3d.com/nurbs.htm.

[8]. **Altmann, Markus.** *About Nonuniform Rational B-Splines - NURBS.*

[9]. Introducción al modelado por subdivisión. *Etérea.* [Online] http://www.etereaestudios.com/training_img/subd_tips/introduccion.htm.

[10]. **Gutiérrez, Diego.** *Superficies de subdivisión.* Grupo de Informática Gráfica Avanzada Universidad de Zaragoza : GIGA.

[11]. **Cobian, Alvaro Blanco.** *Clase5 Apuntes sobre Superficies de subdivisión.* s.l. : Alvaro Blanco Cobian portafolios.

[12]. **Clark, J H.** *Hierarchical Geometric Models for Visible Surface Algorithms.* s.l. : Communications of the ACM, Octubre 1976, pp. 19-10, 547-554.

[13]. **Loop, Charles.** *Smooth Subdivision Surfaces Based on Triangles.* s.l. : University of Utah, 1987. Master thesis.

[14]. **Dyn, Nira, Levin, David and Gregory, John A.** *A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control.* s.l. : ACM Transactions on Graphics, abril 1990, pp. 9-2, 160-169.

[15]. **Konečný, Juraj.** *Catmull–Clark Subdivision Surfaces on GPU.* Slovakia : Faculty of Mathematics, Physics and Informatics Comenius University.

[16]. **Göddeke, Dominik.** *GPGPU::Reduction Tutorial.* Germany : University of Dortmund.

[17]. **Kreiss, Tino.** *GPU vs. CPU Upgrade: Extensive Tests.* mayo 15, 2008.

[18]. **NVIDIA** *. The CG Tutorial.* 2005.

[19]. **Marino, Federico Jorge.** *Visualización de escenas 3D fotorrealistas mediante hardware gráfico programable (GPU).* s.l. : Facultad de Ingeniería, Universidad de Buenos Aires . Tesis de Grado en Ingeniería Informática.

[20]. **Yongming, Xie.** *Geometry Shader Tutorials.* Noviembre 15, 2006.

[21]. **Göddeke, Dominik.** *GPGPU::Fast Transfers Tutorial.* Germany : University of Dortmund.

[22]. **Göddeke, Dominik.** *GPGPU::Reduction Tutorial.* Germany : University of Dortmund.

[23]. **BOLZ, J and Schr¨oder, Peter.** *Rapid evaluation of Catmull-Clark subdivision surfaces.* s.l. : Proceedings of the Web3D Symposium, 2002, pp. 11–18.

[24]. **Lozano, Oscar Mateo.** *REAL-TIME SUBDIVISION SURFACES ON THE GPU.* s.l. : Universidad Autónoma de Madrid.

[25]. **Warren, Joe.** *Subdivision Methods for Geometric Design.* s.l. : Rice University, Noviembre 1995.

[26]. **CGAL.** *CGAL Tutorial.* s.l. : CGAL.

[27]. **Pulli, Kari and Segal, Mark.** *Fast rendering of subdivision surfaces.* s.l. : Proceedings of the EUROGRAPHICS Workshop on Rendering Techniques, 1996, pp. 61–70.

[28]. **Stam, Jos.** *Exact Evaluation Of Catmull-Clark Subdivision Surfaces At Arbitrary Parameter. Values*Seattle, U.S.A. : s.n.

[29]. **Stam, Jos.** *Evaluation of Loop Subdivision Surfaces.* Seattle, U.S.A. : s.n.

[30]. **Qin, Hong and Vemuri, Baba C.** *Dynamic Catmull-Clark Subdivision Surfaces.* 3, s.l. : IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, Julio-Septiembre 1998, Vol. 4.

[31]. **Bolz, Jeff and Schr¨oder, Peter.** *Evaluation of Subdivision Surfaces on Programmable Graphics Hardware.* 2007.

[32]. **Gu, Xianfeng, Gortler, Steven J. and Hoppe, Hugues.** *Geometry Images.* s.l. : SIGGRAPH ’02 Conference Proceedings, 2002, pp. 335–361.

[33]. **LOOP, CHARLES and SCHAEFER, SCOTT.** *Approximating Catmull-Clark Subdivision Surfaces with Bicubic Patches.* 2007.

[34]. **Corporation, NVIDIA.** *Cg Toolkit.* s.l. : NVIDIA, Septiembre 2005.

[35]. **geomview.** [Online] geomview. http://www.geom.uiuc.edu/software/geomview/oogltour.html.

[36]. **Shiue, Le-Jeng, Jones, Ian and Peters, J¨org.** *A Realtime GPU Subdivision Kernel.* s.l. : SIGGRAPH 2007 Conference Proceedings, 2007.

[37]. **Göddeke, Dominik.** *Playing Ping Pong with Render-To-Texture.* Germany : University of Dortmund.

[38]. **NVIDIA.** *NVIDIA Corporation.* [Online] http://download.nvidia.com/developer/presentations/2005/GDC/OpenGL_Day/OpenGL_FrameBuffer_Object.pdf.

Bibliografía Consultada

[1]. **Peters, J. and Wu, X.** *The Distance of a Subdivision Surface to its Control Polyhedron.* Septiembre 15, 2008.

[2]. **Newball, Andrés Adolfo Navarro, Wyvill, Geoff and McCane, Brendan.** *Efficient Mesh Generation Using Subdivision Surfaces.* s.l. : SISTEMAS & TELEMÁTICA, 2008.

[3]. **Stam, Jos and Loop, Charles.** *Quad/Triangle Subdivision.* z, Vol. xx (200y), pp. 1–5.

[4]. **Brönnimann, Hervé.** *Designing and implementing a general purpose halfedge data structure.* Brooklyn, U.S.A : Polytechnic University.

[5]. **Shiue, Le-Jeng and Peters, J¨org.** *A Pattern-based Data Structure for Manipulating Meshes with Regular Regions.* Florida : Computer and Information Science and Engineering University of Florida.

[6]. **Kazakov, Maxim.** *Catmull-Clark Subdivision for Geometry Shaders.* s.l. : Digital Media Professionals, Inc.

[7]. **Kenneth, I. Joy.** *BICUBIC UNIFORM B-SPLINE SURFACE REFINEMENT.* California : Visualization and Graphics Research Group, Department of Computer Science, University of California.

[8]. **Johnson, Travis.** *Jacobi and Gauss-Seidel Methods and Implementation.* abril 23, 2009.

[9]. **Karˇciauskas, K. and Peters, J.** *Adjustable Speed Surface Subdivision.* Florida : Department of Mathematics and Informatics, Vilnius University, University of Florida.

[10]. **Göddeke, Dominik.** *GPGPU Performance Tuning – An illustrated example.* Germany : University of Dortmund.

Glosario de Términos.

Vertex Shader: sombreado de vértice, en idioma español.

Geometry Shader: sombreado de geometría, en idioma español.

Fragment Shader: sombreado de fragmentos, en idioma español.

Pipeline: Conjunto de elementos procesadores de datos, conectados en serie.

Modelo 3D: Es una representación matemática de un objeto tridimensional en un computador mediante vértices, aristas y caras.

Píxel: Es el acrónimo inglés de “picture element”, es la menor unidad homogénea en color que forma parte de una imagen digital.

Plug-in: es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Topología: rama de las matemáticas que estudia las propiedades de las figuras con independencia de su tamaño y forma.

Geometría: parte de las matemáticas que estudia el espacio y las figuras que se pueden formar en él a partir de puntos, líneas, planos y volúmenes.

Toroide: es la superficie de revolución generada por una curva plana cerrada que gira alrededor de una recta exterior coplanaria (el eje de rotación situado en su mismo plano). Su forma se corresponde con la superficie de los objetos que en lenguaje cotidiano se denominan *argollas*, *anillos* o *aros*. La palabra toroide también se usa para referirse a un poliedro toroidal, la superficie de revolución generada por un polígono que gira alrededor de un eje. Según Weisstein, Eric W. *Toroid* From MathWorld (ac. 01-04-09).

Por mor de: indica que una cosa se hace o no se hace a causa de otra o en consideración a alguien: no me obligues a mentir por mor de librarte de la culpa.
Diccionario Manual de la Lengua Española Vox. © 2007 Larousse Editorial, S.L.

Índice de Tablas

Índice de Figuras

Índice de Figuras

Resultados Obtenidos:

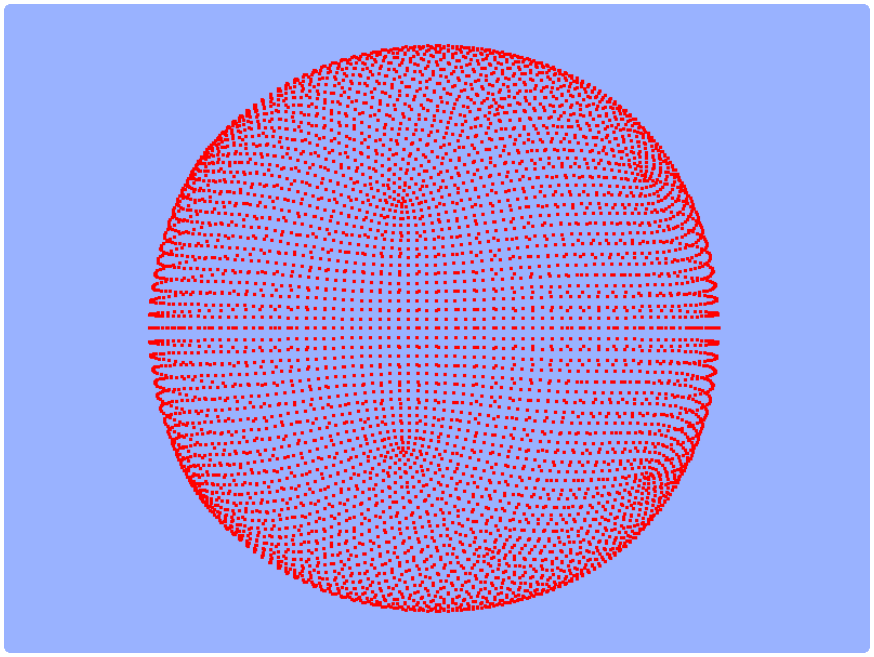


Figura 22. Nivel 5 de Subdivisión para 8 fragmentos (0.031 seg).

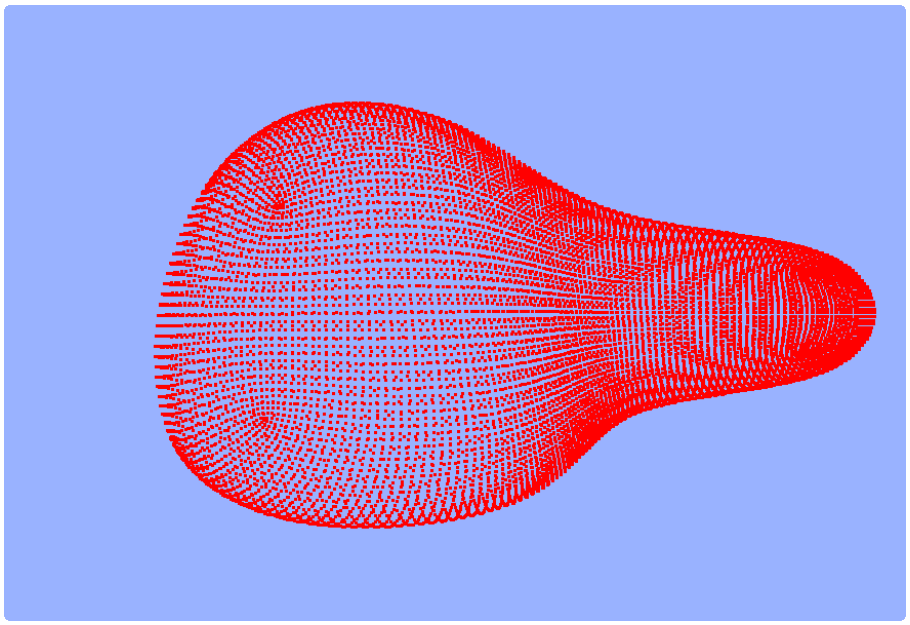


Figura 23. Nivel 5 de Subdivisión para 16 fragmentos (0.094 seg).

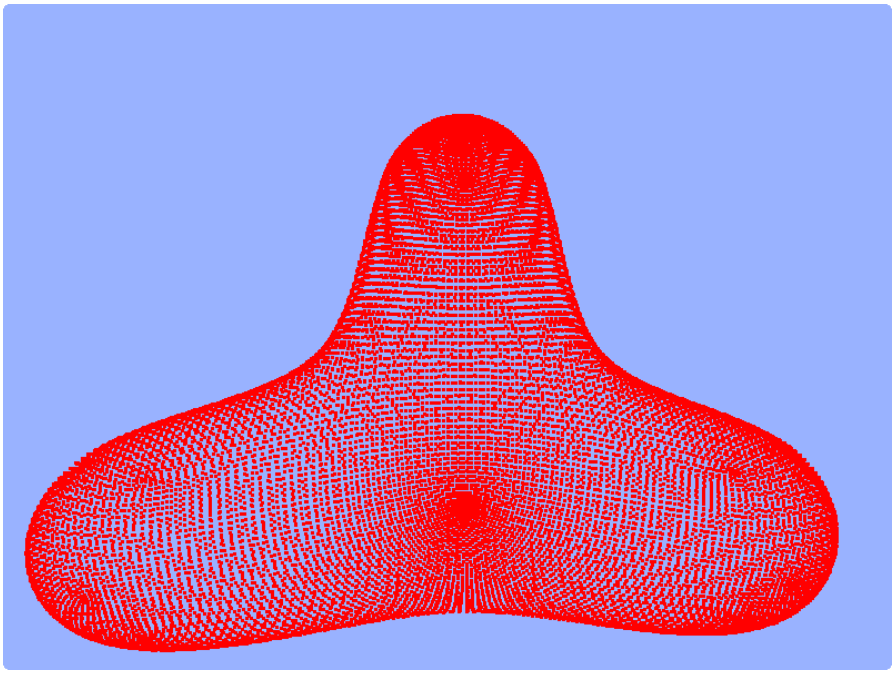


Figura 24. Nivel 5 de Subdivisión para 20 fragmentos (0.047 seg).

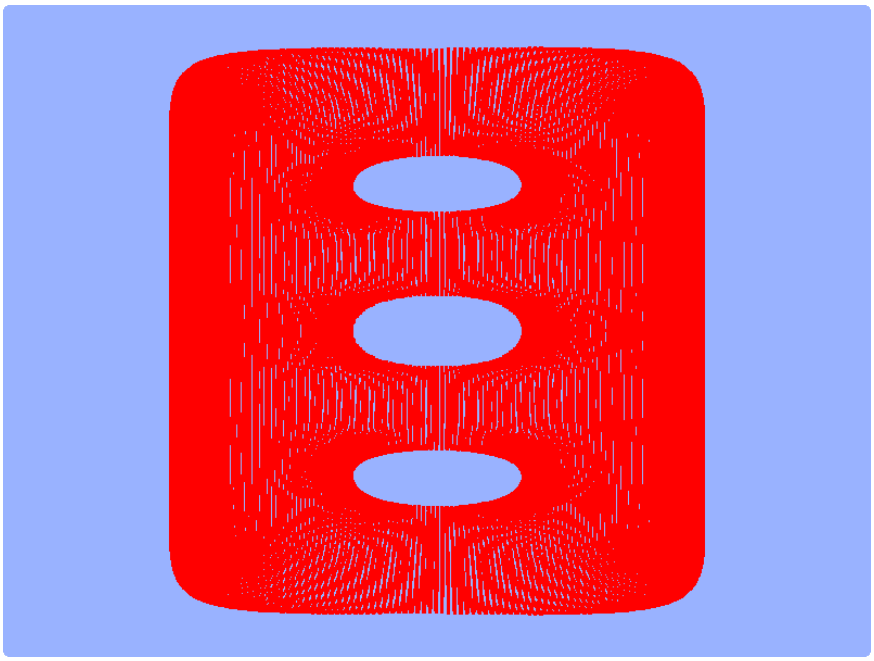


Figura 25. Nivel 5 de Subdivisión para 65 fragmentos (0.078 seg).

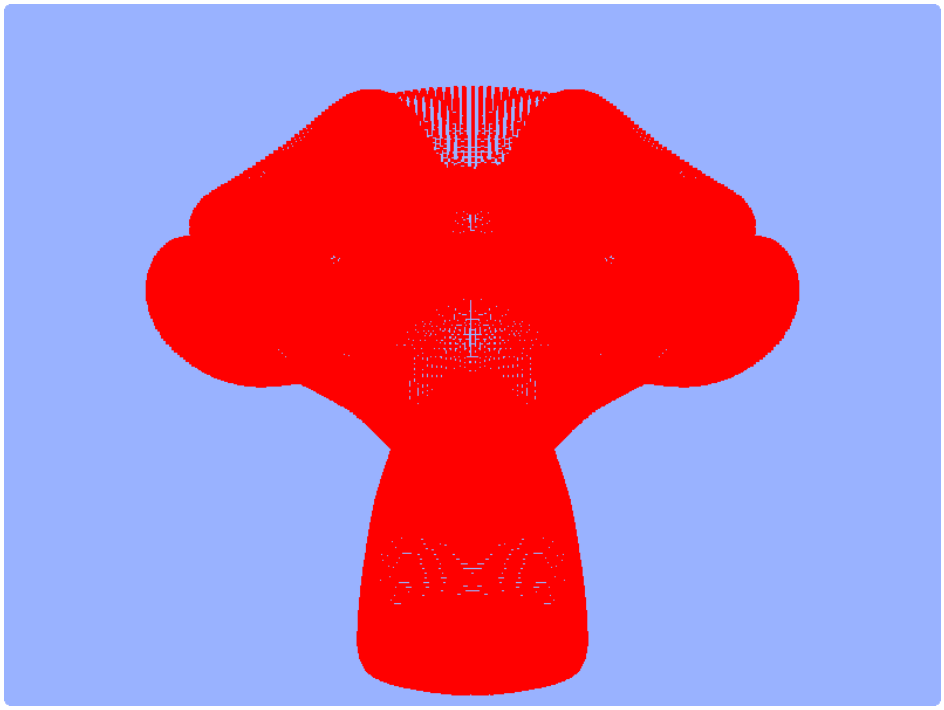


Figura 26. Nivel 5 de Subdivisión para 581 fragmentos (0.703 seg).

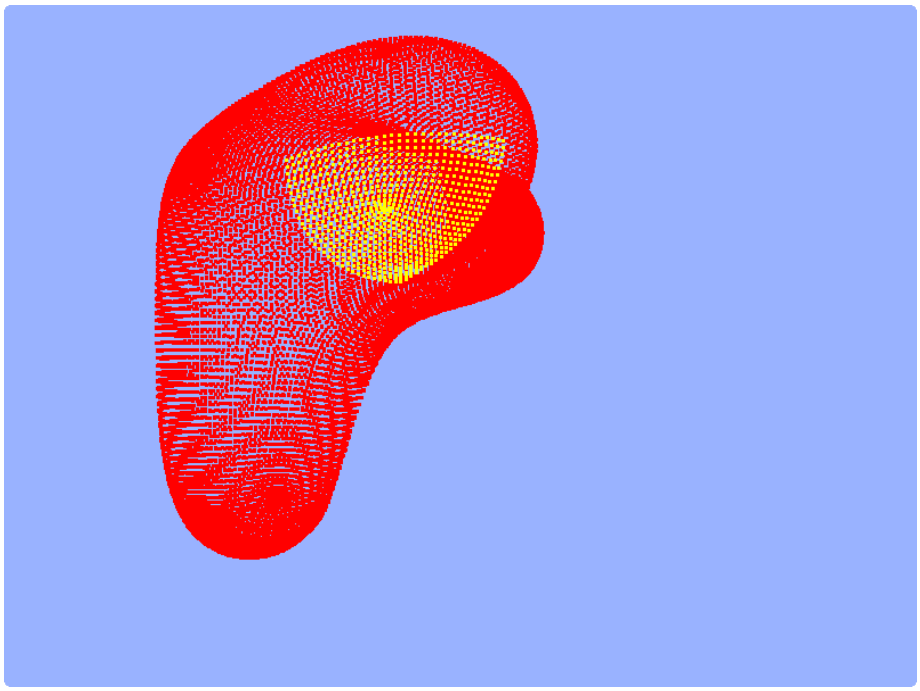


Figura 27. Nivel 5 de Subdivisión, resaltado fragmentos de valencia 3.

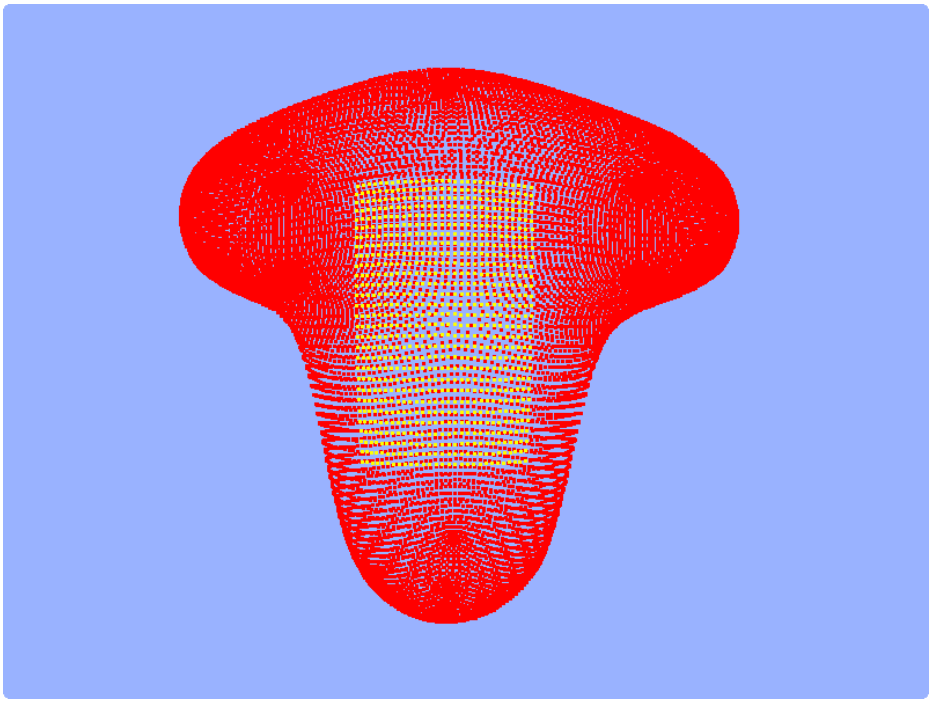


Figura 28. Nivel 5 de Subdivisión, resaltado fragmentos de valencia 4.

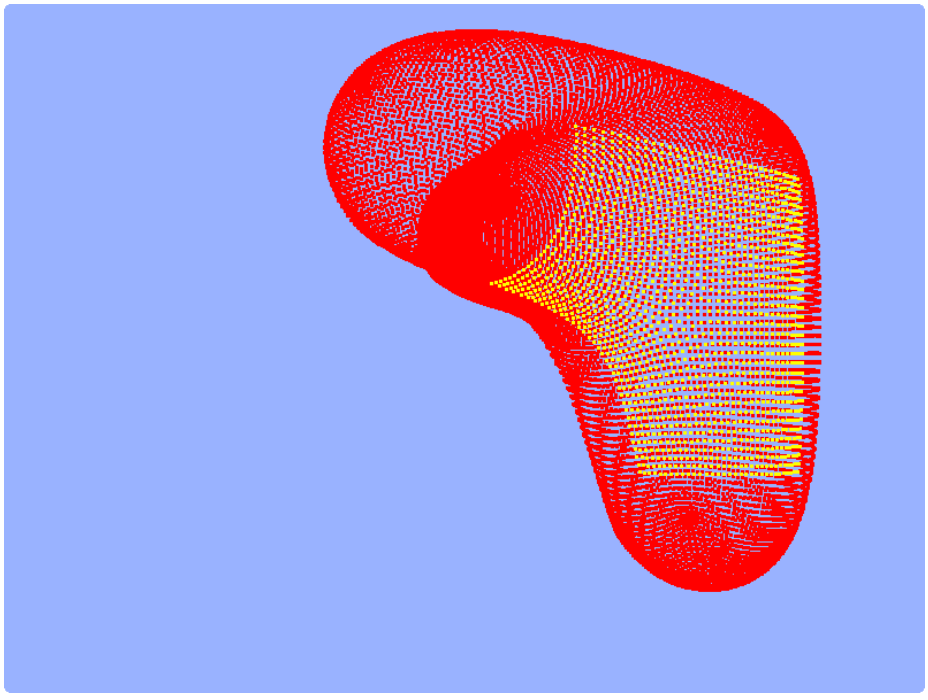


Figura 29. Nivel 5 de Subdivisión, resaltado fragmentos de valencia 5.

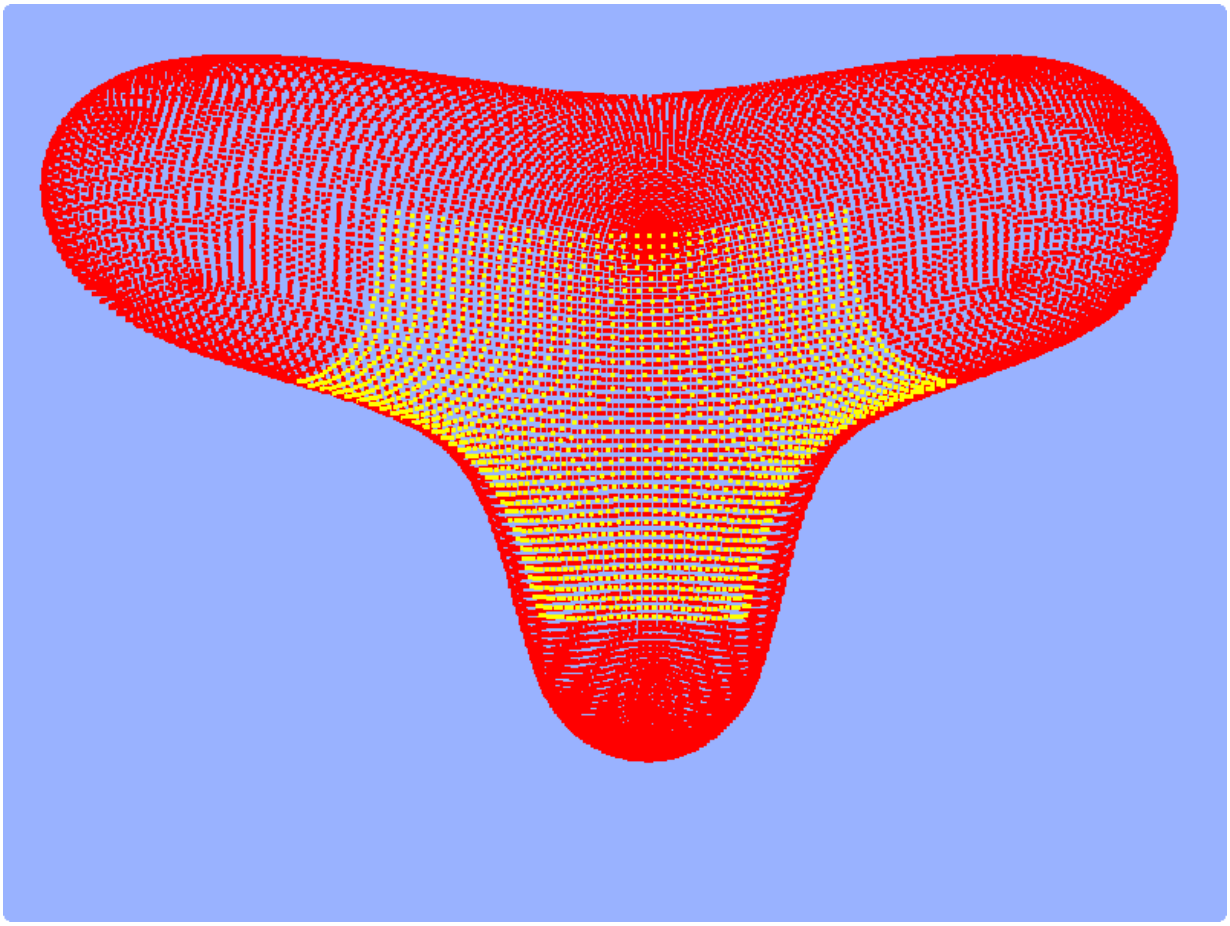


Figura 30. Nivel 5 de Subdivisión, resaltado fragmentos de valencia 6.