UNIVERSIDAD NACIONAL DEL LITORAL
FACULTAD DE INGENIERÍA Y CIENCIAS HÍDRICAS

Tecnicatura en diseño y programación de videojuegos

UNL VIRTUAL

Manipulación de objetos en 3D

Unidad 5: Más conceptos de 3D

Docentes
Jaime Fili
Nicolás Kreiff

CONTENIDOS

1. Más conceptos de 3D (y algo de geometría)

Introducción

Se ha llegado a un punto en el que se deben incorporar nuevos conceptos del mundo del 3D que resultan necesarios para comprender los próximos temas a abordar en la materia. Es así que en este apunte se definen los conceptos que aparecerán con mayor frecuencia en adelante. También se expanden aquí algunas nociones incorporadas en la primera unidad ya que los alumnos están ahora en condiciones de comprender mejor.

Se debe tener en cuenta que esta es sólo una introducción a dichos conceptos y se pretende que puedan luego elaborar y ampliar sus conocimientos en la materia por sus propios medios.

Malla de polígonos (*Polygon Mesh*)

Una malla de polígonos o "grilla desestructurada" es un conjunto de vértices, bordes y caras que definen la forma de un poliedro en el modelado de sólidos en gráficos 3D por computadora.

Las caras usualmente consisten de triángulos, cuadriláteros u otros polígonos convexos simples ya que esto simplifica el renderizado. No obstante, se debe tener en cuenta que en ocasiones también pueden estar compuestas por polígonos cóncavos más generales o polígonos con agujeros.

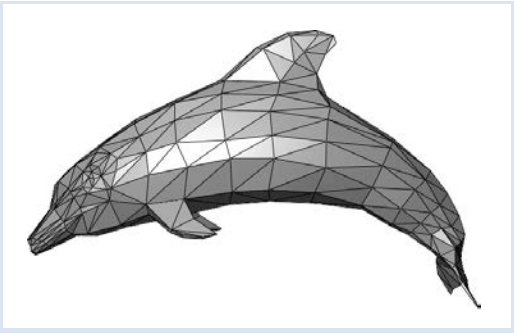


Figura 1. Ejemplo de malla de polígonos.

El estudio de las mallas de polígonos es un campo de los gráficos computarizados y la modelación geométrica.

Diferentes representaciones de mallas de polígonos se utilizan con distintos fines y aplicaciones. Además, la diversidad de operaciones que se realizan sobre mallas pueden incluir lógica booleana, suavizado, simplificación y muchas otras.

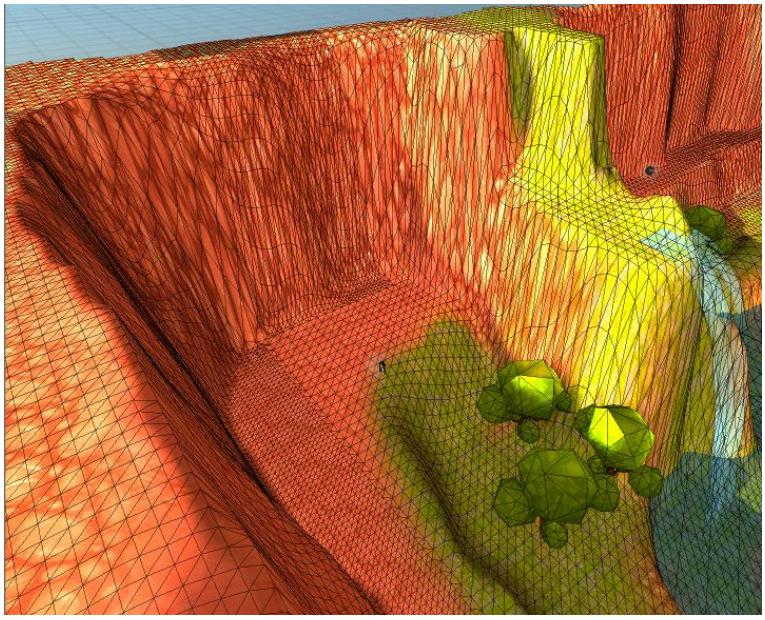
Existen representaciones de red, transmisión y mallas progresivas, que se utilizan para enviar mallas de polígonos a través de una red. Hay también mallas volumétricas, que se diferencian de las mallas de polígonos en que las primeras representan explícitamente superficie y volumen de una estructura, mientras que la malla de polígonos sólo representa explícitamente la superficie (el volumen está implícito).

A continuación se presenta un ejemplo de volumen implícito:

En la siguiente imagen se observa un escenario en el que los volúmenes representados dan la sensación de ser explícitos.

Nótense las copas de los árboles, los desniveles del terreno, el agua del lago; todos estos elementos dan la impresión de ser volumétricos.

Ahora bien, la siguiente imagen muestra una vista similar pero en ella se superpone a los materiales la representación de la malla. Esto permite apreciar la representación volumétrica real de los objetos.

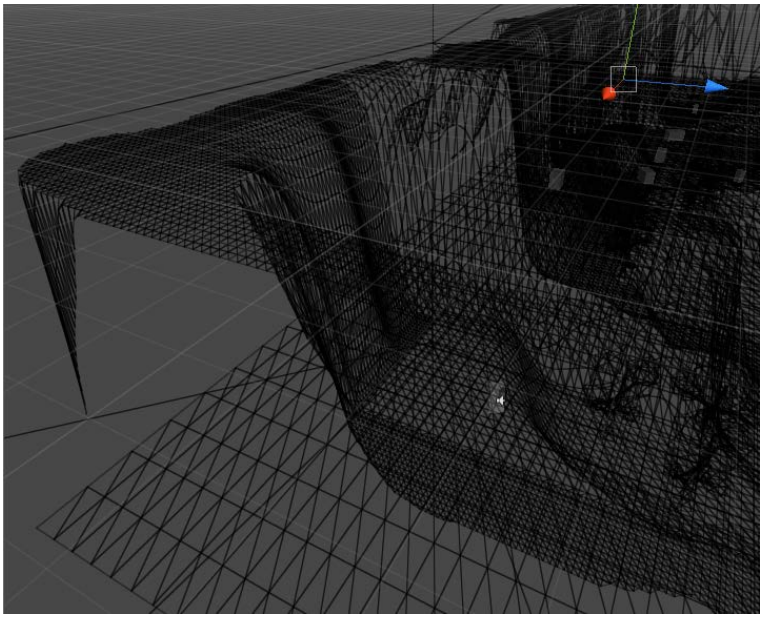


Aquí se puede apreciar la alta segmentación de la malla del terreno contra la baja densidad de polígonos de, por ejemplo, los árboles.

Es normal que posea una alta segmentación en el caso del terreno de Unity. Esto es importante para la aplicación de técnicas de optimización como el *Occlusion Culling*, que se verá más adelante.

Con respecto a los árboles, el trabajo de pintura de la textura sirve para disimular un poco la baja densidad de polígonos del objeto. En los temas que se desarrollan a continuación se ve cómo la combinación de varias técnicas permiten lograr mayor realismo simplificando la cantidad de polígonos presentes en una malla.

Finalmente, la imagen que sigue muestra la representación de malla del terreno. En la misma se puede ver que estos objetos en realidad son huecos y sin volumen real.



Elementos del modelado de mallas

Objetos creado con mallas de polígonos deben almacenar distintos tipos de elementos, incluyendo vértices, bordes, caras, polígonos y superficies.

Un ***renderer*** o procesador (pieza de hardware o software que generalmente procesa la información de la malla) puede soportar sólo caras de tres lados, por lo que los polígonos deben estar construidos por muchos de éstos. Sin embargo, existen muchos renderers que soportan polígonos de mayor cantidad de lados o que tienen la capacidad de triangular polígonos a triángulos en tiempo real, haciendo que sea innecesario el almacenar la malla ya triangulada. También en algunas aplicaciones como el modelado de cabezas, por ejemplo, es deseable poder crear tanto polígonos de tres lados como de cuatro.

El **vértice** es una posición junto con otra información como puede ser color, vector normal y coordenadas de texturas; un **borde** es una conexión entre dos vértices; la **cara** es un conjunto cerrado de bordes: una cara de triángulo tiene tres bordes y una cara de **quad** (cuadrilátero) tiene cuatro bordes. Un **polígono** es un conjunto de caras. En sistemas que soportan caras de múltiples lados, los polígonos y las caras son equivalentes. Sin embargo, como se dijo antes, la mayoría del hardware de renderizado soporta solamente caras de tres o cuatro lados, por lo que los polígonos son representados por múltiples caras.

Matemáticamente, una malla de polígonos puede ser considerada una grilla desestructurada o un grafo sin dirección con propiedades adicionales de geometría, forma y topología.

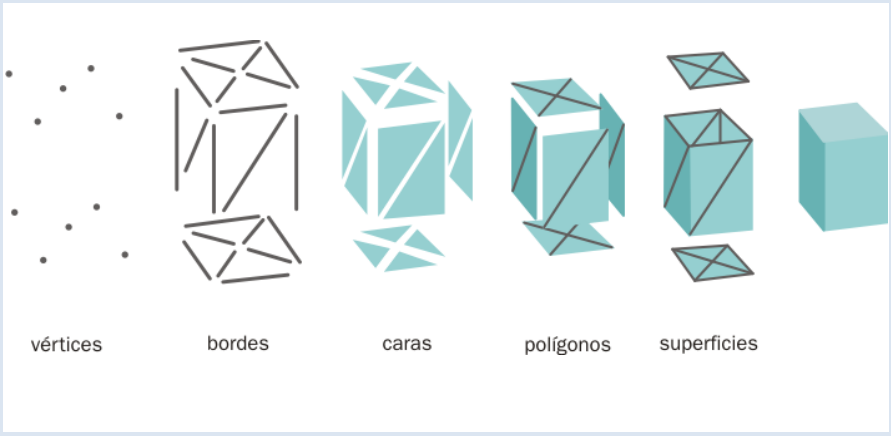
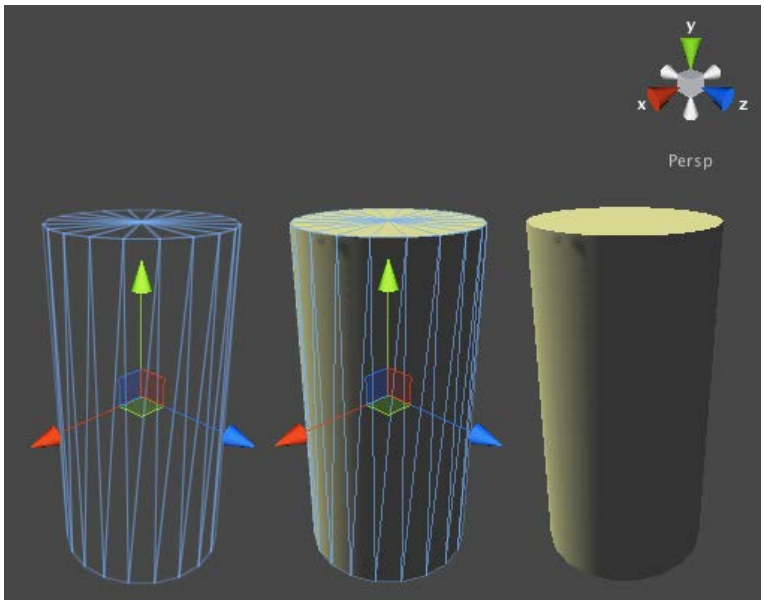


Figura 2. Conformación de un volumen.

Las **superficies** son generalmente llamadas grupos de suavizado. Considérese un cilindro con tapas, como por ejemplo una lata: para suavizar los lados, todas las *normales* (ver vectores normales: http://en.wikipedia.org/wiki/Surface_normal) de la superficie deben apuntar horizontalmente hacia fuera desde el centro, mientras que las normales de las tapas deben apuntar en la direcciones +/-(0,0,1).



En el siguiente apartado se describe el Sombreado de Phong (*Phong Shading*), que permitirá comprender más acerca del suavizado de los lados de un polígono.

Representaciones

Las mallas de polígonos pueden ser representadas de varias maneras utilizando diferentes métodos para almacenar la información de los vértices, bordes y caras. Estas representaciones incluyen:

- **Mallas *Face-Vertex***: una lista simple de vértices y un conjunto de polígonos que apuntan a los vértices que usan.

- **Mallas *Winged-Edge***: en ellas cada borde apunta a dos vértices, dos caras, y los cuatro vértices que la tocan (en sentido horario y anti horario).

- **Mallas *Half-Edge***: similares a las anteriores pero en estas sólo se usa la mitad de la información *traversal* (recorrido de la información del árbol de datos) de la superficie.

- **Mallas *Quad-Edge***: almacenan bordes, semi-bordes y vértices sin ninguna referencia a polígonos. Estos últimos están implícitos en la representación y pueden ser encontrados traversando la estructura. Los requerimientos de memoria son similares a los de las mallas *half-edge*.

- ***Corner-Tables***: almacenan vértices en una tabla predefinida de manera que, al traversar la misma, implícitamente defina los polígonos. Es en esencia el "ventilador de triángulos"(del inglés triangle-fan); un conjunto de triángulos interconectados por un vértice central que se utiliza en el renderizado de gráficos por hardware. La representación es más compacta y resulta más eficiente recuperar polígonos pero las operaciones para modificar polígonos son lentas. Incluso, *corner-tables* no representan completamente las mallas; es necesario el uso de múltiples corner-tables (ventiladores de triángulos) para representar a la mayoría de ellas.

- **Mallas *Vertex-Vertex***: una malla "VV" representa solo vértices que apuntan a otros vértices. Tanto la información del borde como de la cara está implícita en la representación. La simplicidad de la representación permite que muchas operaciones se realicen más eficientemente sobre las mallas.

Cada una de las representaciones descriptas tiene ventajas y desventajas particulares. La elección de una estructura de datos está determinada por la aplicación, la performance requerida, el tamaño de la información y las operaciones que se realizarán.

Para ciertas operaciones es necesario tener un acceso rápido a la información topológica, como bordes y caras adyacentes, lo que requiere estructuras más complejas, como la representación *winged-edge*. Para el renderizado por hardware, en cambio, se necesitan estructuras compactas y simples por lo que la representación *corner-table* (ventilador de triángulos) es comúnmente incorporada en APIs de renderizado de bajo nivel, como DirectX y OpenGL.

2. Shaders

Un ***shader*** es un programa que corre en la unidad de procesamiento gráfico (GPU) y se utiliza para producir los niveles apropiados de luz y sombra en una imagen. Actualmente se usa también para generar efectos especiales y realizar postprocesado de imagen mediante la programación del canal de renderizado programable (*programmable rendering pipeline*) de la GPU. La posición, tono, saturación, brillo y contraste de todos los píxeles, vértices o texturas utilizadas para construir una imagen final se pueden modificar sobre la marcha usando algoritmos definidos en el *shader* y, a su vez, éste puede ser modificado por variables externas o texturas introducidas por el programa que llama al mismo *shader*.

Se utilizan ampliamente para producir infinidad de efectos en el postprocesado de cine, así como en imágenes generadas por computadora y en videojuegos. Algunos usos más complejos incluyen alterar el matiz, la saturación, el brillo y/o el contraste de una imagen; producir desenfoque, sombreado de celda, distorsión, detección de bordes y una gran variedad de otros efectos.

Descripción de la tecnología

Los *shaders* son programas sencillos que describen las características ya sea de un vértice o un píxel. Los *Vertex Shaders* describen las características (posición, coordenadas de texturas, colores, entre otras) de un vértice, mientras que los *Pixel Shaders* describen las características (color, profundidad *z* y el valor *alfa*) de un pixel. Un *vertex shader* se llama para cada vértice de una primitiva (posiblemente después de teselación), por lo que entra un vértice y sale uno actualizado. Cada vértice se representa como una serie de píxeles sobre una superficie (bloque de memoria) que finalmente se envía a la pantalla.

Tipos de shaders

Comúnmente se utilizan tres tipos de *shaders*:

Vertex shaders

Se ejecutan una vez por cada vértice que se envía a la GPU. El propósito es transformar la posición 3D en espacio virtual de cada vértice a las coordenadas en 2D en la que aparecen en la pantalla (así como el valor de la profundidad para el *z-buffer*).

Vertex Shaders pueden manipular propiedades tales como la posición, color y coordenadas de texturas pero no pueden crear nuevos vértices. La salida de un *vertex shader* va a la próxima fase en el *pipeline*, que es un *shader* de geometría (*geometry shader*), si se encuentra presente el ***rasterizador*** (el cual convierte la imagen en pixels o puntos para salida por pantalla o para ser guardada en un archivo de imagen).

Geometry shaders

Los *shaders* de geometría son un tipo relativamente nuevo y fueron introducidos en Direct3D 10 y OpenGL 3.2; anteriormente estaban disponibles en OpenGL 2.0+ con el uso de extensiones. Este tipo de *shader* puede generar nuevas primitivas gráficas tales como puntos, líneas y triángulos a partir de aquellas primitivas que fueran enviadas al principio del *pipeline* gráfico.

Los programas de *shaders* de geometría se ejecutan después de los *shaders* de vértice. Toman como entrada una primitiva entera, posiblemente con información de adyacencia. Por ejemplo, cuando se trabaja con triángulos, los tres vértices son la entrada del *geometry shader*. Luego el *shader* podrá emitir cero o más primitivas, que serán rasterizadas y sus fragmentos pasados finalmente a un pixel *shader*.

Típicamente, el uso de *shaders* de geometría incluye la generación de *point sprites*, teselación[1] de geometría, extrusión de sombra de volúmenes y renderizado de una pasada a un *cube map*[2].

Uno de los beneficios de este tipo de *shaders* es la modificación automática de complejidad de malla. Una serie de tiras de línea que representan los puntos de control de una curva se pasan al *shader* de geometría y, en función de la complejidad requerida, éste puede generar automáticamente líneas adicionales, cada una de las cuales proporciona una mejor aproximación de la curva.

[1] Teselación es el proceso de crear un plano bi-dimensional usando la repetición de una figura geométrica, sin superposición ni saltos.

[2] *Cube mapping* es una técnica de mapeado de entornos que usa un cubo de seis lados como la forma del mapa.

Pixel shaders

Son también conocidos como *fragment shaders* y calculan el color y otros atributos de cada pixel. Pueden variar desde cambiar la salida del color, aplicar un valor de iluminación, sombras, reflejos especulares, translucidez y otros efectos. También se puede modificar la profundidad del pixel para *z-buffering*, o más de una salida de color si están activos múltiples *render targets*.

Un pixel *shader* por sí mismo no puede producir efectos muy complejos, ya que únicamente funciona en un solo píxel, sin el conocimiento de la geometría de una escena.

Programando shaders

El lenguaje con el que los *shaders* se programan depende del entorno de destino.

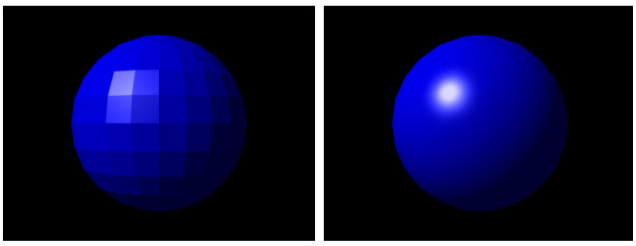
El lenguaje oficial de OpenGL y OpenGL ES (la versión de OpenGL para sistemas embebidos como los teléfonos celulares) es OpenGL Shading Language, también conocido como GLSL. Por su parte, el lenguaje oficial para Direct3D es High Level Shader Language también llamado HLSL. Sin embargo, CG es un lenguaje desarrollado por la empresa Nvidia que genera *shaders* tanto para OpenGL y Direct3D.

3. Sombreado de phong

El **sombreado de phong** (*Phong Shading*) se refiere a una técnica de interpolación que permite obtener el sombreado (intensidad de color) de las superficies en gráficos 3D por computadora. En concreto, se calculan las normales a cada vértice, luego se interpolan en cada pixel de los polígonos rasterizados para finalmente calcular el color del pixel, basándose en la normal interpolada y el método de iluminación.

El sombreado de phong también puede referirse a la combinación específica de interpolación de phong y el modelo de reflexión de phong.

En la siguiente imagen se muestra un ejemplo de Phong Shading:



FLAT SHADING PHONG SHADING

El sombreado de phong mejora al sombreado de Gouraud[3] y proporciona una mayor aproximación a la sombra de una superficie suave. La interpolación de Phong asume una variación suave del vector normal a la superficie y es un método que funciona mejor que el sombreado Gouraud cuando se aplica a un modelo de reflexión que tenga una pequeña cantidad de reflejo especular, tal como el modelo de reflexión de phong.

[3] Técnica usada en gráficos 3D que simula efectos de luz y color sobre superficies de objetos. Publicada por Henri Gouraud en 1971, esta técnica de sombreado permite suavizar superficies con una carga computacional menor que con otros métodos basados en el cálculo píxel a píxel.

El problema más grave con el sombreado de Gouraud se produce cuando los reflejos especulares se encuentran en el centro de un polígono de gran tamaño. Dado que los reflejos especulares están ausentes en los vértices del polígono y que el sombreado Gouraud interpola en base a los colores en los vértices, el reflejo especular se pierde en el interior del polígono. Este problema se resuelve con el sombreado de phong.

En efecto, a diferencia del sombreado de Gouraud, que interpola los colores en la superficie de los polígonos en base a los colores de cada vértice, en el sombreado phong el vector normal es interpolado linealmente en la superficie del polígono y en base a las normales de cada vértice del polígono. La normal de la superficie se interpola y se normaliza en cada píxel, y luego se utiliza en un modelo de reflexión - por ejemplo, el modelo de reflexión de phong- para obtener el color final del pixel. El sombreado de phong es, por tanto, más costoso computacionalmente que el sombreado Gouraud ya que el modelo de reflexión debe calcularse en cada píxel en lugar de en cada vértice.

4. Mapas de coordenadas UV (*UV Mapping*)

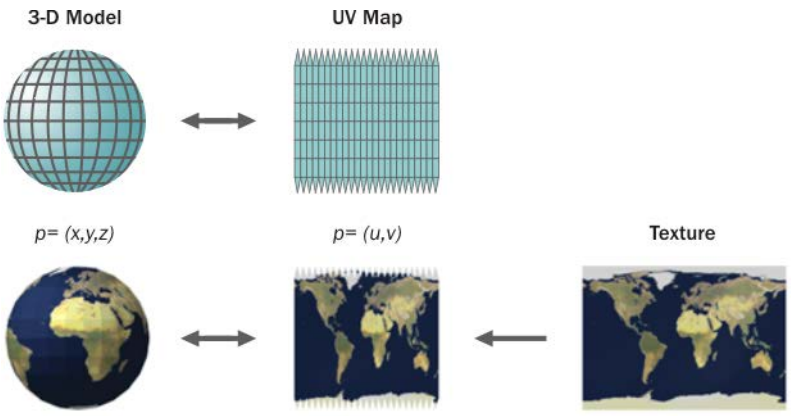
El mapeado UV es el proceso de modelado 3D de crear una representación 2D de un modelo 3D.

Este proceso proyecta un mapa de texturas en un objeto tridimensional. En él, las letras “U” y “V” denotan los ejes de la textura 2D ya que “X”, “Y” y “Z” se utilizan para expresar los ejes del objeto 3D en el espacio modelo.

El texturado UV permite que los polígonos que generan un objeto 3D sean pintados con color de una imagen. Esta imagen se denomina mapa de texturas UV y no es cualquier imagen. El proceso de mapeado UV implica la asignación de pixels en la imagen al mapeado en el polígono, usualmente se programa copiando una pieza en forma de triángulo del mapa de imagen en un triángulo dentro del objeto.

La denominación UV es entonces la alternativa a XY, que sólo mapea en un espacio de textura en lugar del espacio geométrico del objeto.

El cálculo de representación utiliza así la textura de coordenadas UV para determinar cómo pintar la superficie tridimensional.



Por ejemplo, en la imagen se ve una esfera con una figura cuadriculada, primero sin y luego con un mapeado UV. Sin un mapa UV, el cuadriculado en espacio X,Y,Z aparecería como tallado fuera de la esfera. Con un mapa UV, el cuadriculado en el espacio UV y los puntos en la esfera mapean a este espacio respecto a su latitud y longitud.

Cuando un modelo es creado como una malla de polígonos utilizando una herramienta de modelado 3D, las coordenadas UV pueden ser generadas para cada vértice de la malla. Una forma es que el modelador 3D despliegue la malla de triángulos en los bordes o "costuras", mostrando automáticamente los triángulos en una página plana. Si la malla es una esfera UV, por ejemplo, el modelador podría transformarla en una proyección cilíndrica equidistante.

Una vez que el modelo está abierto, el artista puede pintar una textura en cada triángulo individualmente usando la malla abierta como una plantilla. Cuando la escena es renderizada, cada triángulo mapeará a la textura correspondiente de la plantilla.

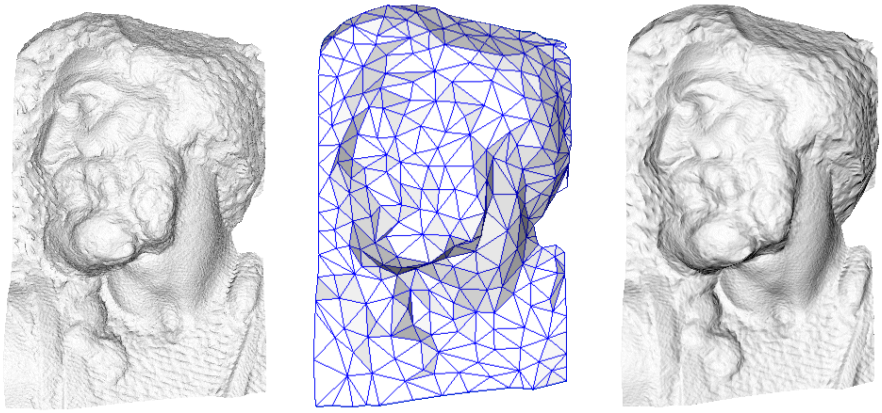
Un mapa UV puede ser generado ya sea automáticamente por un software, manualmente por un artista o por una combinación de ambos. Generalmente un mapa UV será generado y luego el artista deberá realizar ajustes y optimizarlo para minimizar costuras y superposiciones. Si el modelo es simétrico, el artista puede superponer triángulos opuestos para pintar en dos lados simultáneamente.

Las coordenadas UV se aplican por cara, no por vértice. Esto significa que un vértice compartido puede tener diferentes coordenadas UV en cada uno de sus triángulos, por lo que triángulos adyacentes pueden ser cortados y posicionados en diferentes áreas del mapa de texturas.

El proceso de mapeado UV en su forma más simple requiere tres pasos: abrir la malla, crear la textura y aplicar la textura.

5. Normal Maps

El mapeo de normales (***normal mapping***) es una técnica utilizada para simular la iluminación de "bultos y abolladuras y se utiliza para agregar detalles sin usar más polígonos. Un uso común de esta técnica es el de mejorar considerablemente la apariencia y los detalles de un modelo de baja cantidad de polígonos generalizando un mapa de normales de un modelo de alta densidad de polígonos.



original mesh
4M triangles

simplified mesh
500 triangles

simplified mesh
and normal mapping
500 triangles

En la primera imagen se puede observar una malla con 4 millones de triángulos. En la segunda vemos una malla simplificada con mucho menos detalle y con 500 triángulos. Aplicando un mapa de normales a la malla simplificada se puede lograr la semejanza que se observa con respecto a la primera imagen.

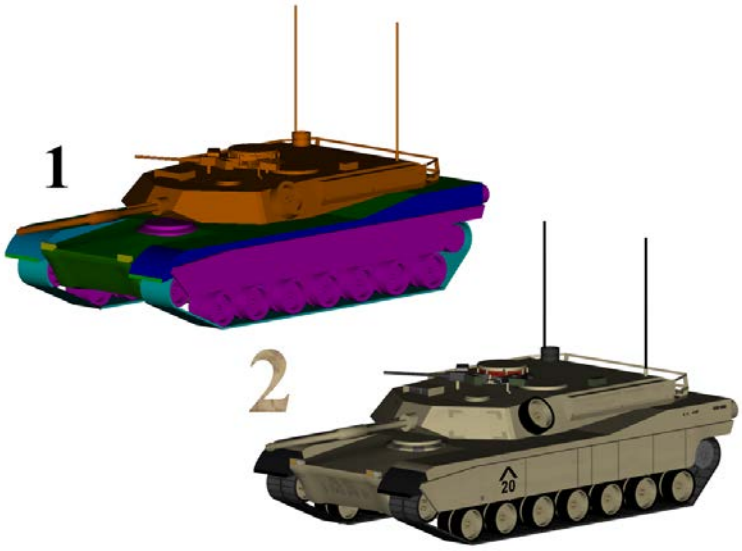
Los mapas de normales generalmente se almacenan como imágenes RGB donde las componentes RGB corresponden a las coordenadas X, Y, y Z, respectivamente, de la normal de la superficie.

6. Mapeado de texturas (*Texture Mapping*)

El mapeado de texturas es un método para agregar detalles, textura de superficie o color a un modelo generado por computadora.

Un mapa de textura es aplicado (mapeado) a la superficie de una figura o polígono. En este proceso cada vértice en un polígono es asignado a coordenadas de textura, ya sea a través de una asignación explicita o por definición procedural.

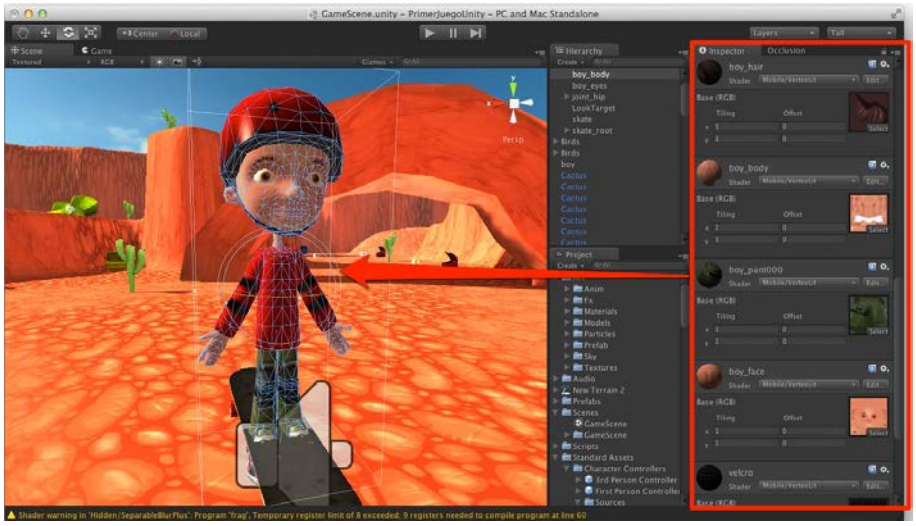
El muestreo de ubicación de imagen es luego interpolado a través de la cara del polígono para producir un resultado visual que aparenta tener más detalle del que se podría lograr con un número limitado de polígonos.



El **multitexturado** es el uso simultáneo de más de una textura sobre un polígono. De hecho, una textura de iluminación (o *lightmap*) puede ser utilizado para iluminar una superficie como alternativa a recalcular la iluminación cada vez que la superficie es renderizada.

Otra técnica de multitexturado es el *bump mapping*, que permite a una textura controlar directamente la cara visible de una superficie con el propósito de calcular su iluminación, lo que puede dar una muy buena apariencia de una superficie compleja como la corteza de un árbol, por ejemplo.

El siguiente es un ejemplo de multitexturado en un modelo. Como se puede observar en el inspector de Unity, las diferentes partes del cuerpo del personaje (las distintas mallas que lo componen) utilizan distintos materiales.



Referencias

Polygon Mesh: http://en.wikipedia.org/wiki/Polygon_mesh

Vectores Normales: http://en.wikipedia.org/wiki/Surface_normal

Sombreado de Phong: http://es.wikipedia.org/wiki/Sombreado_de_Phong

Texture Mapping: http://en.wikipedia.org/wiki/Texture_mapping

UV Mapping: http://en.wikipedia.org/wiki/UV_mapping

Normal Mapping: http://en.wikipedia.org/wiki/Normal_mapping

OpenGL Shading Language, Third Edition (por Randi J. Rost - Bill Licea-Kane) Addison – Wesley 2011

3D Rendering: http://en.wikipedia.org/wiki/3D_rendering