NPR - Implementación en 3D utilizando OpenGL

Martín Massera y Marcela Nievas

Facultad de Ciencias Exactas y Naturales – Universidad de Buenos Aires – ARGENTINA.

mmassera@dc.uba.ar (54)(011) 4783-1479 y mn6s@dc.uba.ar (54)(011) 4716-6016

Abstract

Muchas técnicas de NPR[1] *han sido aplicadas sobre imágenes 2D estáticas. El objetivo es reproducir estas técnicas sobre escenas en 3D en tiempo real.*

Para ello, se utilizó la libreria gráfica OpenGL [5], la cual permite una buena manipulación de coeficientes de texturas necesario para lograr los efectos deseados.

En el presente trabajo se muestran los resultados de la implementación de las técnicas: Phong, Technical Matte, Cartoon Flat, Cartoon Two Tones [1] y Painterly [2] con distintos Brushes en 3D en rendering en tiempo real.

Para permitir al usuario manipular los párametros de representación de las técnicas en tiempo real, se implementó una interfase gráfica.[2].

1 Introducción

Desde 1970 la tendencia preponderante en la Computación Gráfica fue la de conseguir *realismo* visual a través de la simulación más o menos *ad-hoc* de los modelos de interacción luz-materia. Sin embargo, el fotorealismo no es lo más adecuado en determinados contextos técnicos y artísticos. En efecto, en diversas situaciones del dibujo técnico, por ejemplo, el objetivo es *visualizar* la información del modelo, y no su apariencia real. Lo mismo ocurre en realidad virtual, en realidad aumentada, y en video juegos, donde la apariencia visual busca comunicar aspectos importantes de una situación sin que se desee confundir ésta con una "apariencia real".

De esa manera surge NPR, como el conjunto de técnicas adecuadas para la representación de objetos o modelos en situaciones o contextos deliberadamente artificiales.

2 Ideas

Todas las técnicas presentadas en este trabajo fueron aplicadas sobre el mismo modelo. Es decir, que todas las técnicas reciben el mismo input de la escena, más algunos parámetros necesarios para determinar su comportamiento.

2.1 Gouraud Shading

En la implementación realizada, se ha incluido el estilo Gouraud a modo de comparación con un modelo "realístico". El efecto resultante se muestra en la Figura 1. La activación de esta técnica en OpenGL, se realiza con la función: `glShadeModel(GL_SMOOTH)`.

2.2 Cartoon Flat

Este estilo es el más sencillo, ya que se logra asignando un solo color para cada triangulo. El efecto resultante se observa en la Figura 1. La función en OpenGL es `glShadeModel(GL_FLAT)`.

[1] *Non-photorealistic Rendering*

[2] http://personales.ciudad.com.ar/npr3d/

Figure 1: (Izq) Gourand Shading. (Der) Cartoon Flat.

Figure 2: (Izq) Normales y dirección de la Luz. (Der) Ángulo entre la Normal y la dirección hacia la Luz.

2.3 Cartoon Two Tones

Este método imita a los dibujos animados con efecto de sombras.

Para lograr este efecto, se coerciona el valor computado en el sombreado de Gouraud a unos pocos valores. Una manera de implementarlo es por medio de una tabla de textura de ancho 8 bits y alto 1 bit, en la cual se almacenan factores de luminosidad. Ver Figura 3.

Al momento de renderizar, se calcula el coseno entre la normal de cada vértice y la dirección hacia la luz que incide sobre él. Ver Figura 2.

Cuando menor sea el ángulo de incidencia de la luz sobre el vértice, mayor será el coseno del ángulo que los separa. Utilizando esta información, se busca el valor correspondiente en la tabla.

Por ejemplo, en la Figura 2, si el $\cos(\Phi) = 0.7$, entonces el vértice será mapeado a la coordenada de textura (0.7, 0) logrando así texturizar el valor 1 para ese vértice. Ver Figura 3.

El color final resultante, será la multiplicación entre el color del vértice y el valor obtenido de su textura.

Se obtiene así o bien el color original cuando la incidencia de la luz es directa, o un color más atenuado o sombreado (por haber sido multiplicados por valores menores a 1) cuando la incidencia de la luz es con un ángulo mayor.

Figure 3: Tabla de sombreado y su mapeo de textura.

Figure 4: (Izq) Two Tones con sombra suave. (Der) Sombra más fuerte. Al pie, la tabla de sombreado utilizada.

En OpenGL, el array de coeficientes se asocia al *rendering pipeline* con la función `glTexImage2D` y se le indica la forma de calcular el color final con la función `glTexEnvi`

Los parámetros permitidos al usuario de modificar son la cantidad de bit que corresponden a cada zona de sombra y el factor de luminosidad asociado. El efecto resultante se observa en las Figura 4.

2.4 Technical Matte

Este estilo, tiene por objetivo destacar las zonas cálidas de las zonas frías, dependiendo de la incidencia de la luz recibida.

Se define un color que representa el calor, por ejemplo amarillo, y otro que representa el frío, por ejemplo azul.

Al momento de renderizar se calcula el color cálido y el color frío a partir del color puro del vértice, utilizando el factor de incidencia. El cálculo del color frío puro será `colorfriopuro=colorvertice*(1-factorfrio) + color_frio*(factorfrio)`

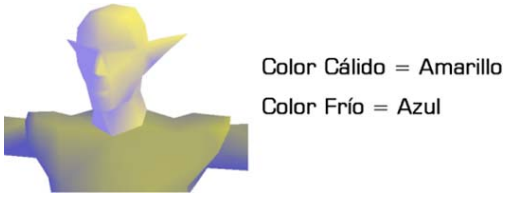


Figure 5: Technical Matte

Por ejemplo, si el color puro es verde, entonces su color cálido será un verde amarillento y su color frío será un verde azulado.

Luego, y al igual que para cartoons, se calcula el coseno entre la normal del vértice y la dirección hacia la luz que incide sobre él.

Entonces, se interpola el color puro cálido y el color puro frío en función del coseno obtenido, logrando así el efecto deseado. Si a es el coseno antes mencionado, entonces el color final del vértice es `colorfinal=colorcalidopuro*a+colorfriopuro*(1-a)`

El usuario puede definir el color cálido, el color frío y el factor de incidencia de estos colores sobre el color puro del vértice. El efecto resultante se observa en la Figura 5.

2.5 Silueta

Para lograr el efecto de silueta, se vuelven a redibujar todos los vértices, exagerándolos y alejándolos de la cámara con un color negro. Esto se logra, sumando a cada coordenadas de cada vértice su normal multiplicando por un factor de exageración. Por ejemplo, para la coordenada x el cálculo será: `x=x+(normal*factorexageracion)`

En el caso de las coordenada x e y, el efecto que produce es una ampliación, mientras que para la coordenada z, el efecto es un alejamiento de la cámara.

El usuario puede elegir el factor de exageración.

2.6 Painterly

Para lograr el efecto denominado *painterly*, se recurre primero a la construcción de un sistema de partículas a partir de las caras de la figura.

Cada partícula tiene su coordenada en el espacio, su alpha blending, tamaño y rotación. En la construcción del modelo, se recorre cada cara y se distribuyen uniformemente sobre ella, la cantidad de partículas dadas por los parámetros elegidos.

El tamaño, orientación y alpha blending, son obtenidos también a partir de los parámetros elegidos.

En el momento de renderizar las partículas, se le aplica primero la transformación (traslación y/o rotación) de la escena a cada una de las partículas, obteniendo así su coordenada en el espacio (al igual que se haría con los vértices de las caras) y se las ordena según el criterio elegido por el usuario.

Luego, para cada partícula, se crea un *billboard* mirando hacia la cámara sobre un plano z fijo entre la cámara y la partícula más cercana a ésta.

A cada billboard, se lo texturiza con el *brush* elegido, y se lo rota sobre el plano z fijo según la orientación informada por la partícula.

La acción final, es enviar cada billboard de cada partícula a la escena en el orden preestablecido.

Orden de la partículas

En una escena tradicional (sin clipping), las caras más alejadas a la cámara deben ser dibujadas en primera instancia, para que luego las caras más cercanas las obstruyan. Con painterly, además se debe pintar primero los billboard más grandes y luego los más pequeños, al igual que lo haría un pintor que esta trabajando sobre un lienzo. Los criterios de orden y render para elegir son:

SIZE: Solo por el tamaño del billboard, de más grande a más pequeño.

SIZE, Z: Primero por tamaño, de más grande a más pequeño. Y ante igual tamaño, se los ordena

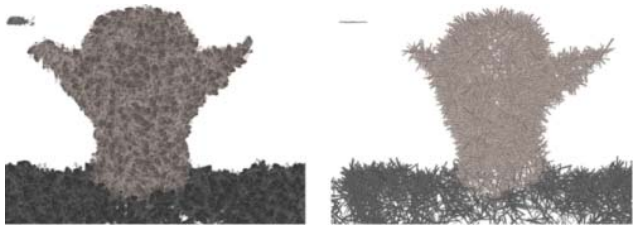


Figure 6: Painterly con distintos brushes. En el margen superior izquierdo, se muestra el brush utilizado.

por su cercanía a la cámara, de más alejado a más cercano.

Z, SIZE: Primero por su cercanía a la cámara, de más alejado a más cercano. Y ante igual distancia, se los ordena por tamaño, de más grande a más pequeño.

El usuario puede elegir el "brush"[3], el rango de cantidad de partículas a distribuir por cada cara, el rango de tamaño de las partículas, la orientación de las partículas, el alpha blending, y el orden de las partículas. Los efectos resultantes se observan en las Figura 6.

3 Conclusiones

La performance obtenida en tiempo real, resulto ser muy satisfactoria tanto para Cartoon Two Tones como para Technical Matte. Esto se debió a que los efectos se lograron, en un caso, cambiando la forma de componer la textura sobre cada cara, y en el otro, cambiando el color de cada vértice al momento de renderizar.

En cambio, para Painterly, la performance bajó considerablemente, debido a la cantidad de partículas necesarias para completar cada escena.

La interfase implementada para permitir al usuario realizar cambios en tiempo real de los parámetros de cada técnica, fue muy útil a la hora de interpretar visualmente los efectos logrados.

[3]Imagen de la pincelada

4 Trabajos Futuros

Las técnicas utilizadas, contemplan solo una luz direccional en la escena, quedando pendiente agregar más luces direccionales.

Cuando se exagera el borde de la silueta, se obtiene un efecto visual desagradable. Queda pendiente, lograr la coherencia entre el borde de la figura y ésta, sin importar el grosor de la silueta.

Dado que el incremento de partículas degrada la performance, es vital seguir analizando la forma de eliminar aquellas partículas que no tendrán gran incidencia sobre el resultado visual final.

En Painterly, las partículas creadas y distribuidas a partir del mesh de entrada, no evolucionan. Queda pendiente el nacimiento, crecimiento y muerte de las mismas.

Agradecimiento

Queremos agradecer a nuestro profesor Claudio Delrieux[4] por el apoyo recibido para llevar adelante este trabajo.

References

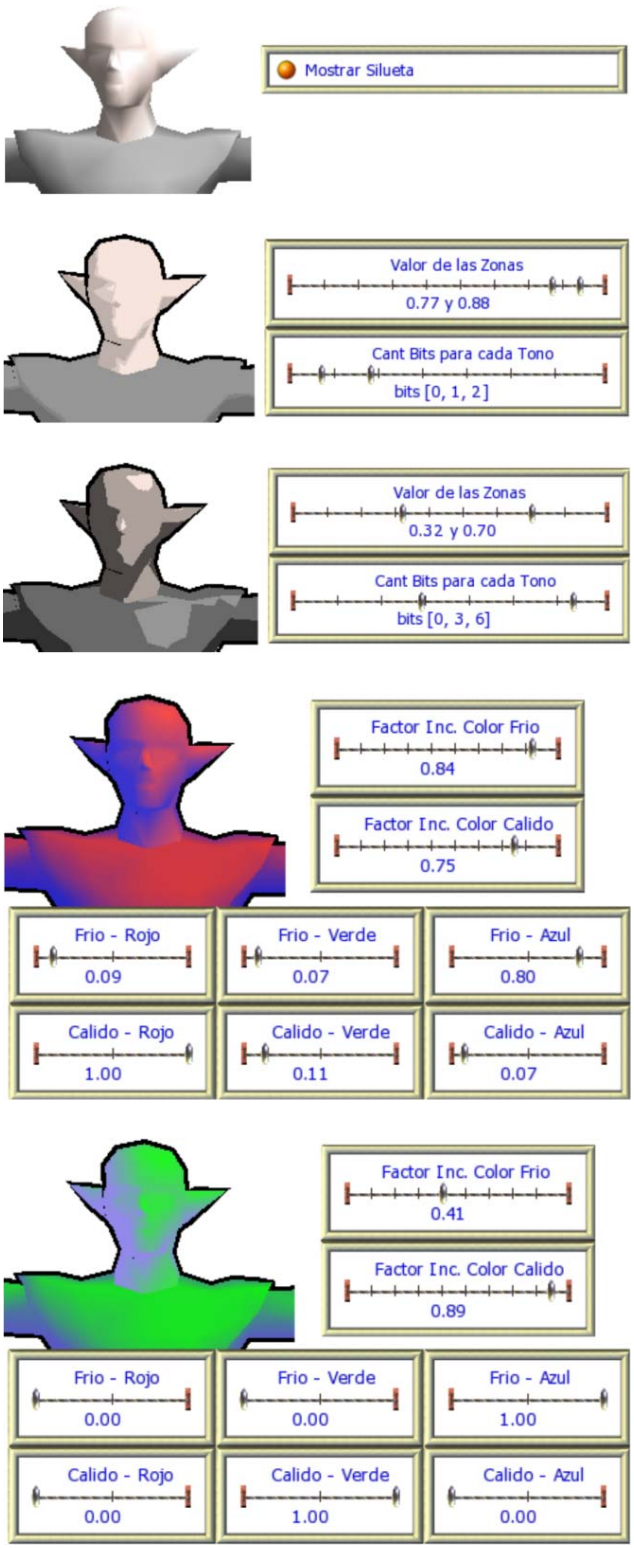
[1] Interactive Non-Photorealistic Technical Illustration. Amy Gooch. December 1998.

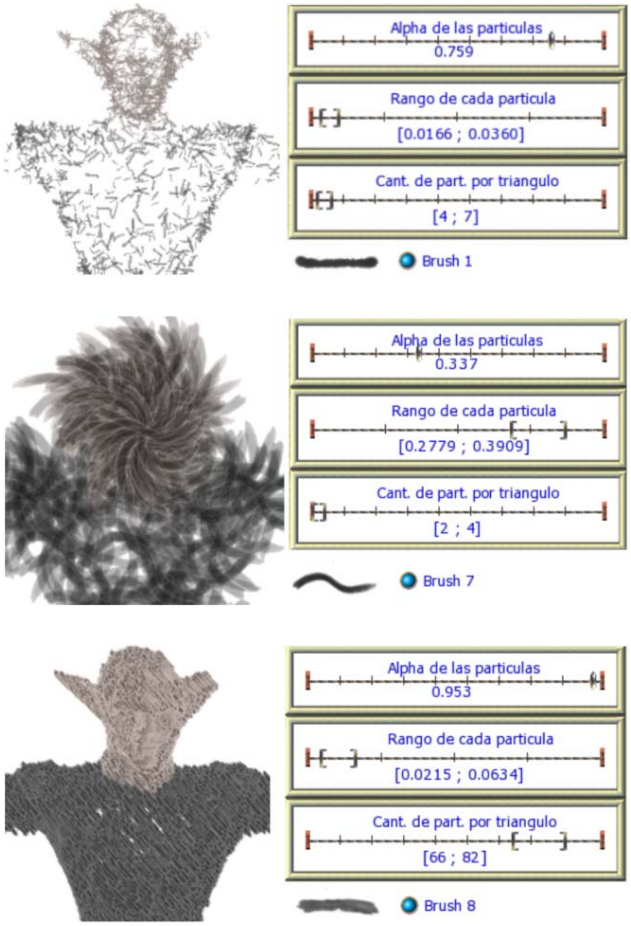
[2] Paint By Numbers: Abstract Image Representations. Paul Haeberli. Computer Graphics, Volumen 24, Number 4, August 1990.

[3] Painterly Rendering for Animation. Barbara J. Meier. SIGGRAPH 96 Conference Proceedings, pp. 477–484. August 1996.

[4] Particle System – A Technique for Modeling a Class of Fuzzy Objects. William T. Reeves. ACM Transactions on Graphics, Vol. 2, No. 2, April 1983, Pages 91-108.

[4]claudio@acm.org





[5] http://www.opengl.org/