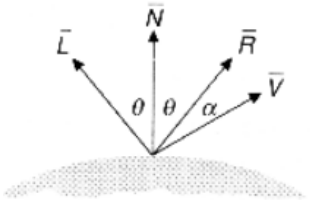
2do Coloquio

Los modelos de iluminación extraen aquellos factores que determinan el color en un determinado puno de una superficie iluminada, basándose en la interacción de la luz con el material. Para determinar a qué puntos de la escena se aplica el modelo de iluminación, se tienen modelos de sombreado. Los modelos de sombreado determinar el muestreo de la escena -tamaño de los puntos de muestra sobre la superficie a la que se le aplicará el modelo de iluminación- y un esquema de interpolación que aproximará los colores de los puntos restantes.

El modelo de Phong permite expresar I como:

$$I_\lambda = I_{a\lambda}k_{a\lambda} + f_{at}I_{p\lambda}k_{d\lambda}(\bar{L}.\bar{N}) + f_{at}I_{p\lambda}k_{s\lambda}(\bar{R}.\bar{V})^n$$



Blinn-Phong es similar a Phong, solo que el término especular se calcula como $R_sL_s = f_{at}I_pk_s(\bar{N}.\bar{H})^n$, donde el vector halfway es:

$$\bar{H} = \frac{\bar{L} + \bar{V}}{|\bar{L} + \bar{V}|}$$

Phong y Blinn-Phong dependen de la geometría local dela superficie, la ubicación y los tipos de fuente de luz, la dirección de la vista y las propiedades del material de la superficie. Tienen limitaciones dado que el color del highlight depende del tipo del material: por ejemplo en el plástico la reflexión especular es del mismo color que la luz incidente, mientras que en los metales es del color del metal.

Cook y Torrance proponen modificar la ecuación de iluminación de Phong con el coeficiente especular:

$$\rho_s = \frac{F_\lambda}{\pi}\frac{DG}{(\bar{N}.\bar{V})(\bar{N}.\bar{L})}$$

1.A. ¿Qué modela I?

I representa el color de un punto en una superficie cuando está iluminado por una luz ambiente y una determinada cantidad de luces puntuales, en este caso una. La ecuación considera tres términos: ambiente, difuso y especular. Cada término se calcula utilizando una intensidad de iluminación $\mathrm{L_i}$ recibida en un punto y una reflexión $\mathrm{R_i}$ que está relacionada con las propiedades del material de la superficie.

1.B. Diga cuáles son los elementos de la fórmula que representan las características de la superficie. Explique muy brevemente qué representa cada uno de ellos.

Los elementos que representan las características del son:

- $\mathrm{k_a}$ – coeficiente de ambiente: varía entre 0 y 1, representa la cantidad de luz ambiental reflejada. Es más visible donde no cae luz de forma directa.
- $\mathrm{k_d}$ – coeficiente de reflexión difuso: varía entre 0 y 1.
- $\mathrm{k_s}$ – coeficiente de reflexión especular: varía entre 0 y 1, grado de reflexión.
- n – exponente especular: define que tan brilloso es el material, varía entre 0 y 500.

1.C. Señale cuáles elementos de la fórmula representan la luz que llega a la superficie desde las distintas fuentes de luz. Especifique de qué tipos de fuentes de luz se trata.

Los elementos que representan la luz son:

- $\mathrm{L_a} = \mathrm{I_a}$ – luz ambiente.
- $\mathrm{L_d} = \mathrm{f_{at}} \,.\, \mathrm{I_p}$ – luz difusa.
- $\mathrm{L_s} = \mathrm{f_{at}} \,.\, \mathrm{I_p}$ – luz especular.

Tanto la luz especular como la difusa sirven para una luz puntual.

El término ambiente (o Lambertiano) considera la luz dispersa en toda la escena y produce un efecto de iluminación uniforme sobre cada punto de la superficie de la escena. Su iluminancia I_a se especifica en tres componentes RGB y la intensidad de iluminación recibida en un punto sobre una superficie es $L_a = I_a$. Por lo tanto, la cantidad de luz ambiente reflejada será $R_a L_a = k_a I_a$.

En el término difuso (la luz que ingresó a la superficie y fue parcialmente absorbida y luego dispersada) la intensidad en el punto es $L_d = f_{at}\, I_p$, siendo $\mathrm{f_{at}}$ el factor de atenuación. Así la cantidad de luz reflejada en el término difuso estará dado por $R_d L_d = f_{at} I_p k_d (\bar{L}.\bar{N})$.

En el término especular (la luz reflejada en la superficie) la intensidad en el punto es $L_s = f_{at}\, I_p$ y la intensidad reflejada será entonces $R_s L_s = f_{at} I_p k_s (\bar{R}.\bar{V})^n$. El color de la reflexión especular es el de la fuente de luz.

1.D. Dado los coeficientes ambiente, especular y difuso tanto de una fuente luminosa como de un objeto. ¿De qué color es el brillo especular de un objeto de material plástico? ¿Cómo deberían inicializarse los coeficientes ambiente, especular y difuso del objeto para que tengan esta característica?

El color del brillo especular de un objeto de plástico es del color de la luz incidente, es decir, el color de la luz especular. Para que el objeto tenga esta propiedad se debe settear el coeficiente de reflexión especular de la siguiente manera:

$$k_s = \begin{bmatrix} k_{sR} \\ k_{sG} \\ k_{sB} \end{bmatrix} \; tal\ que\ k_{sR} = k_{sG} = k_{sB} \neq 0$$

De esta forma se refleja la luz especular.

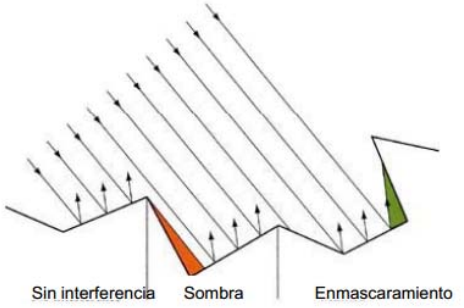
1.E. Dado el coeficiente de Cook y Torrance escriba cuál es la ecuación que permite calcular la iluminación en un determinado punto. Identifique aquellos factores que describen el material de un objeto y defina conceptualmente cada uno de ellos.

$$\mathrm{I_\lambda} = \mathrm{I_{a\lambda} k_{a\lambda}} + \mathrm{f_{at} I_{p\lambda} k_{d\lambda}} (\bar{\mathrm{L}}.\bar{\mathrm{N}}) + \mathrm{f_{at} I_{p\lambda} k_{s\lambda}} \left(\frac{\mathrm{F_\lambda}}{\pi} \frac{\mathrm{DG}}{(\bar{\mathrm{N}}.\bar{\mathrm{V}})(\bar{\mathrm{N}}.\bar{\mathrm{L}})} \right)$$

- Los k son los factores que describen el material.
- F – término de Fresnel: permite determinar cuánto proporción de la luz incidente es reflejada. Así, el color de la reflexión especular es una función de la interacción entre el material y la luz incidente, dependiendo tanto de la longitud de onda de la misma como del ángulo de incidencia.
- D – Función de Distribución de la Orientación de las Micro facetas: las normales a las micro-superficies y sus direcciones son aleatorias. Entonces tiene sentido modelarlas estadísticamente como una distribución. Para

la mayoría de las superficies la distribución de las normales es continua con un pico fuerte en la normal macroscópica de la superficie. Cook-Torrance usa la distribución de Beckmann.

- G – Factor de Atenuación Geométrica: lo "apretado" de la distribución D está determinado por la suavidad de la superficie. Una distribución más separada de las normales hace que la luz reflejada se disperse en un mayor grado de direcciones y por ende se obtenga una superficie más rugosa. Se refiere a los efectos geométricos que se producen a nivel micro-escala. Las sombras se producen por la oclusión de la fuente de luz debido a los detalles de la superficie. El enmascaramiento se produce por oclusión de la luz reflejada.



2.A. Dada la ecuación de iluminación de Phong decir para el sombreado de Phong:

a) datos de entrada/salida del programa de vértices.

b) datos de entrada/salida del programa de fragmentos.

c) que términos de la ecuación indican las propiedades del material.

d) que término indica la geometría local de la superficie.

Datos de entrada del programa de vértices:

- Uniform vec3 LightPos, EyePos;
- In vec3 Normal;
- In vec4 vPosition;
- Uniform m4 modelView, proyMatrix, normalMatrix;

Datos de salida del programa de vértices: vec4 L, N, V.

Datos de entrada del programa de fragmentos:

- In vec4 L, N, V, ka, ks, kd;
- In float coeficienteEspecular;

Datos de salida del programa de fragmentos: vec4 color del fragmento.

Los k y el coeficiente especular indican las propiedades del material.

El término que indica la geometría local de la superficie es N (el vector normal). R, el vector de reflexión, se calcula en función de N así que indica de forma indirecta la geometría local.

2.B. ¿Por qué el método de sombreado de Phong es computacionalmente más caro que el método de Gouraud?

Es computacionalmente más caro ya que en el de Phong primero deben interpolarse las normales y luego a partir de esto se calcula el color en el fragmento. Para el método de Gouraud en cambio se calcula el color en cada vértice y luego estos colores son interpolados en la etapa de renderizado.

En el método de sombreado de Gouraud la iluminación se calcula en el programa de vértices, es decir, sea hace un cálculo por vértice. En el método de sombreado de Phong se hace un cálculo de color por cada fragmento a renderizar, lo que es más costoso.

2.C. Suponga que debe aplicar el sombreado Gourard y quiere calcular la iluminación en cada punto mediante Blinnn-Phong. Indique:

a) parámetros de entrada y salida del programa de vértices, y sus tipos asociados.

b) parámetros de entrada y salida del programa de fragmentos, y sus tipos asociados.

c) ¿la iluminación la calcula el programa de vértices o el de fragmentos? Justifique.

d) ¿qué factor de la formula tiene en cuenta la geometría local de la superficie?

Datos de entrada del programa de vértices:

- In vec3 Normal;
- In vec4 vPosition;
- Uniform vec3 LightPos, EyePos;
- Uniform vec4 ka, ks, kd;
- Uniform float coeficienteEspecular;
- Uniform m4 modelView, proyMatrix, normalMatrix;

Datos de salida del programa de vértices: vec4 color.

Datos de entrada del programa de fragmentos: vec4 color.

Datos de salida del programa de fragmentos: vec4 color del fragmento.

La iluminación se calcula en el programa de vértices dado que en el sombreado de Gourard se interpola el color y no las normales.

El término que indica la geometría local de la superficie es N (el vector normal).

2.D. Suponga que debe aplicar el sombreado Gourard y quiere calcular la iluminación en cada punto mediante Cook-Torrance. Indique:

a) parámetros de entrada y salida del programa de vértices, y sus tipos asociados.

b) parámetros de entrada y salida del programa de fragmentos, y sus tipos asociados.

c) ¿la iluminación la calcula el programa de vértices o el de fragmentos? Justifique.

d) mejoras de Cook-Torrance respecto a Phong.

Datos de entrada del programa de vértices:

- Uniform vec3 LightPos, Ka, Ks, Kd, EyePos;
- Uniform float coeficienteEspecular;
- In vec3 Normal;
- In vec4 vPosition;
- Uniform m4 modelView, proyMatrix, normalMatrix;

Datos de salida del programa de vértices: vec4 color.

Datos de entrada del programa de fragmentos: vec4 color.

Datos de salida del programa de fragmentos: vec4 color del fragmento.

Para el sombreado de Gourard la iluminación se calcula en el programa de vértices, ya que el programa de fragmentos interpola los colores calculados en los vértices de cada fragmento.

El modelo de iluminación de Phong es empírico, da buenos resultados pero no es físicamente posible. Cook-Torrance, en cambio, está basado en principios físicos: considera la conservación de energía y la longitud de onda. Sin embargo, ambos son locales y consideran únicamente las fuentes de luces a la hora de calcular la iluminación en la superficie (a diferencia de modelos globales).

El modelo de Phong siempre posiciona los high-lights en el vector reflejado R y hace que su color dependa del color de la fuente de luz, mientras que Cook-Torrance puede desplazarlo de R e incluye un término Fresnel que cambia el color del high-light según el ángulo.

La diferencia esencial entre Phong y Cook-Torrance está en el cálculo del término de Fresnel (F). Éste término introduce variaciones de color y de la energía reflejada dependientes de las propiedades de la superficie relativa al espectador. Cook-Torrance incluye también un término de distribución -D- que describe estáticamente cómo la luz se dispersa cuando choque con la superficie. El término G describe cuánta de la luz reflejada escapa la superficie.

2.E. ¿Qué tipos de material pueden modelar: Phong, Cook-Torrance y Oren-Nayar?

Phong sirve para materiales plásticos o similares, ya que los highlights son del color del brillo especular y no del color del material como sucede en los plásticos.

Cook-Torrance sirve para metales pues los highlights dependen del color del material.

Oren-Nayar sirve para superficies rugosas como el yeso y madera.

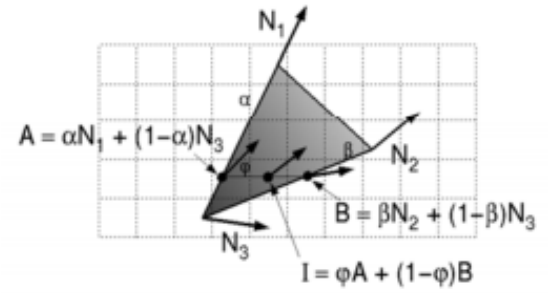
2.F. El modelo de sombreado de Phong interpola las normales de los vértices para cada primitiva a iluminar y luego calcula la iluminación en cada punto donde se tiene la normal. Especifique en que coordenadas Ci pueden interpolarse las normales y justifique.

Las coordenadas interpoladas son requeridas por el programa de fragmentos para el sombreado de Phong. Las normales son interpoladas en las coordenadas que salen del programa de vértices ya sea del Ojo o del Mundo. En nuestro caso, como la matriz Normal corresponde a las transformaciones para las normales del ModelView significa que las normales son transformadas por T0, T1 y T2 y quedan así en coordenadas del Ojo.

2.G. Se tiene un determinado triangulo y sus normales en los vértices. ¿En qué etapa del pipeline se determina en qué puntos del objeto se interpolarán las normales? Dibuje la figura y las normales que se interpolarán.

En la etapa de rasterización (T5) se determina en qué puntos del objeto se interpolan las normales.

Se calculan las normales en cada vértice del polígono. Luego se interpolan para cada punto del polígono que se proyecta en un pixel. Estas normales interpoladas se utilizan en el cálculo de intensidad en los puntos del polígono que corresponden a cada pixel proyectado.



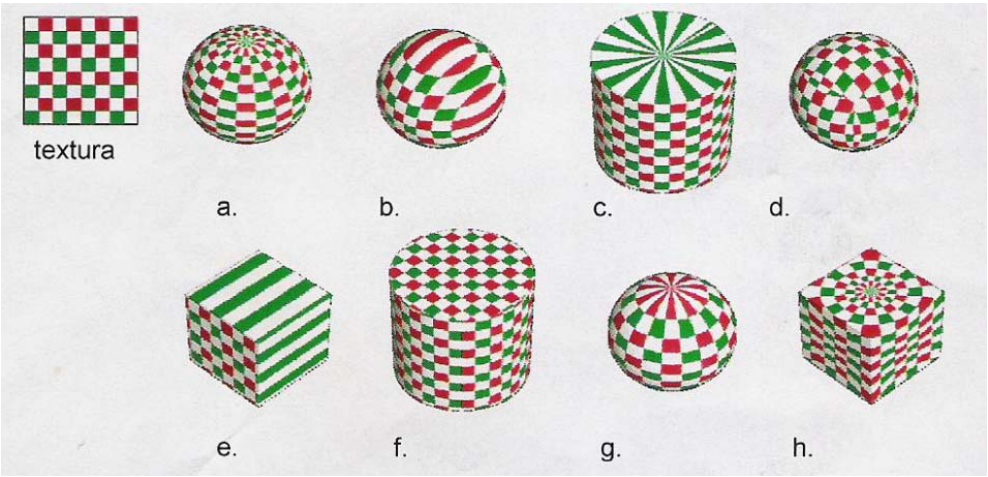
2.H. Si se desea implementar el sombreado de Phong: ¿en qué etapas se realiza la interpolación de normales y el cálculo de la iluminación? Escriba conceptualmente las tareas a realizar por el programa de vértices y/o fragmentos para calcular el sombreado de Phong.

Las normales de cada vértice son interpoladas en la salida del programa de vértices, previo a la rasterización y a la ejecución del programa de fragmentos. Éstas se utilizan en el programa de fragmentos para calcular la iluminación en T5.

El programa de vértices recibe las posiciones del vértice y lo transforma a coordenadas de clipping; y recibe la normal del vértice y la transforma a coordenadas del ojo y las pasa para ser interpolada. Además, calcula el vector ojo y el vector luz en coordenadas del ojo y las pasa en el pipe para ser usadas en el programa de fragmentos.

El programa de fragmentos recibe las normales interpoladas, el vector ojo y luz (en coordenadas del ojo) y los parámetros del material para calcular la intensidad del fragmento usando el modelo de Phong.

3. Hay determinados mapeos de textura que se realizan en dos etapas, la de parametrización y la de renderizado. 3.A. En la etapa de parametrización se utilizan distintas superficies intermedias, diga en cada caso, cuál es la superficie intermedia.

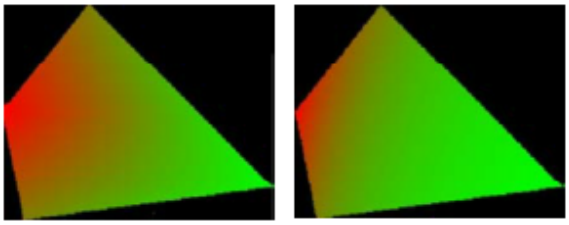


Superficie Intermedia	Figuras
Plano	B, E
Cubo	D, F
Cilindro	C, G
Esfera	A, H

3.B. En la etapa de renderizado, OpenGL utiliza interpolación bilineal en el espacio de la pantalla para encontrar un valor de textura. Para proyecciones ortogonales el mapeo bilineal es correcto. Explique por qué. Esto no es correcto para la proyección perspectiva. Explique por qué.

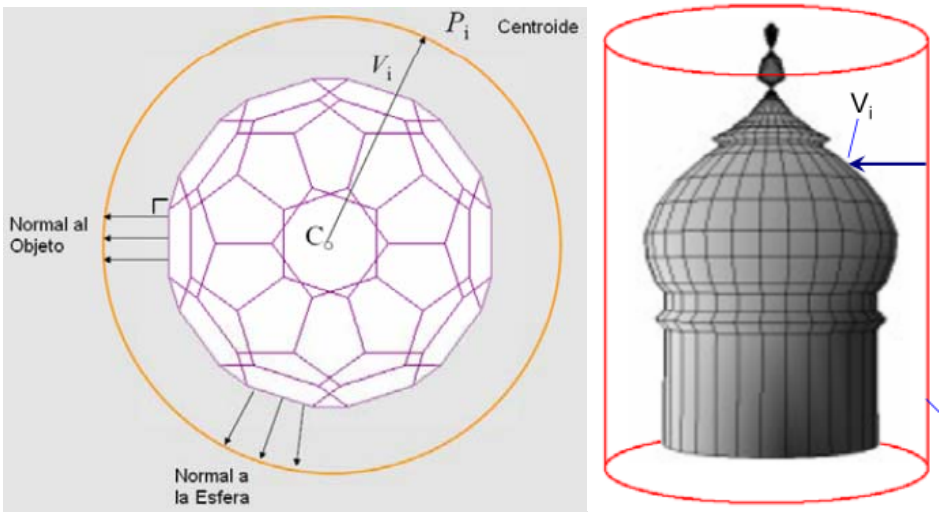
Para proyecciones ortogonales el mapeo bilineal es correcto ya que en la proyección ortogonal los planos que delimitan la visión de la cámara son perpendiculares (a diferencia de la proyección perspectiva donde son oblicuos), delimitando la escena a un rectángulo formado por los mismos. Esto produce que en la imagen generada los objetos se ubiquen a la misma distancia de los planos que delimitan la visión sin importar la profundidad en la que se encuentren. Por esta razón es correcto utilizar interpolación bilineal, ya que no hay problemas de distorsión por la perspectiva. La proyección ortogoal se da por rayos paralelos, por lo que las tangencias se preservan y líneas paralelas se proyectan a líneas paralelas.

No es correcto para la proyección perspectiva, debido a que surge una distorsión por la perspectiva, anomalías que ocurren debido a que la interpolación es llevada a cabo luego de la transformación en perspectiva en el sistema de coordenadas 3D de la pantalla: la interpolación lineal hace que la información de sombreado sea incrementada en forma constante de una línea de scan a la siguiente.



Sombreado de Gouraud Sombreado correcto

3.C. ¿Cuál considera que es la superficie intermedia más apropiada en este caso? ¿Por qué? Puede dibujar en la figura lo que considere necesario.



Para la primer imagen utilizaría el método de proyección esférica, dado que puedo utilizar la normal de la esfera y el centroide del objeto de forma indistinta para el indexado, o sino recurrir a las normales del objeto. Para la segunda imagen utilizaría el método de proyección cilíndrica dada que se trata de un objeto alongado. Para el indexado utilizaría las normales del objeto o el rayo reflejado sobre la superficie intermedia.

La superficie intermedia elegida es la más adecuada ya que es la más similar al objeto en sí por lo que el indexado de la superficie intermedia al objeto producirá así una menor deformación del mapa.

Métodos de proyección: El mapeo se realiza en dos etapas un Mapeo S, donde se mapea la textura 2D a una superficie intermedia 3D simple, y una mapeo O donde se mapea la superficie intermedia –con la textura- a la superficie destino del objeto que está siendo renderizado.

El método de proyección de Wrapping permite que las texturas sean proyectadas sobre una superficie 3D de manera directa, pero también deformada ajustando los cuatro lados del mapa sobre la superficie. Este tipo de proyección es útil para proyectar texturas sobre objetos que pueden requerir distintos estiramientos sobre el mapa para adecuarlo y para aplicar texturas a pequeñas porciones de objetos 3D.

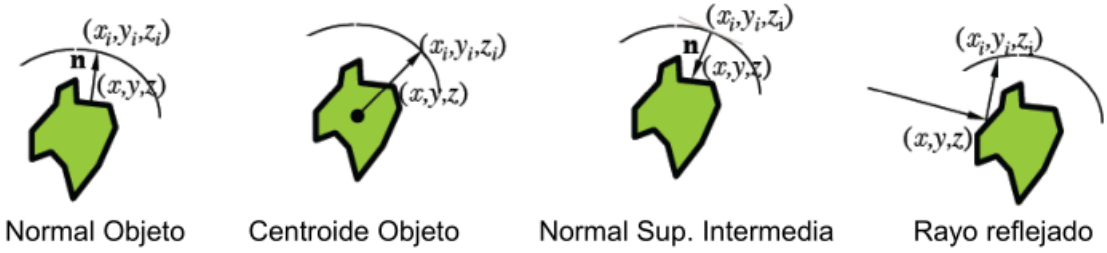
El mapeo de superficies intermedias se puede dar sobre un plano, un cubo, cilindro o una esfera:

- El método de proyección plano es ideal para aplicar mapas de imágenes sobre superficies planas porque los resultados son totalmente predecibles y la distorsión es mínima en tanto las superficies tridimensionales sean paralelas a los planos de proyección. También puede aplicarse sobre objetos curvos para simular distintos efectos.
- El método de proyección cúbica es una variación del método plano que contiene un mapa en cada una de las caras del cubo. Este método es particularmente efectivo con cubos, en tanto uno de los lados del cubo sea paralelo al plano de proyección.
- El método de proyección cilíndrica aplica el mapa sobre la superficie juntando los lados del mapa de modo de envolver el objeto. Es útil aplicarlo sobre objeto alongados tales como una zanahoria o una botella de vino. Puede aplicarse de modo tal que la parte de arriba y/o abajo no se cubran. Las coordenadas del objeto se pasan a coordenadas cilíndricas *(radio, ϑ, altura)* y así las coordenadas de texturas resultan ser *s=ϑ* y *t=altura*.
- El método de proyección esférica rodea al objeto con el mapa y luego junta los extremos inferior y superior hasta cubrir completamente el objeto. Esta técnica es útil para proyectar mapas sobre objetos redondos. Las coordenadas del objeto pasan a coordenadas esféricas *(radio, ϑ, φ)* siendo *s=φ* y *t=ϑ*.

3.D. Describa gráfica y conceptualmente las cuatro maneras para indexar el mapa.

Hay cuatro formas de indexar el mapa:

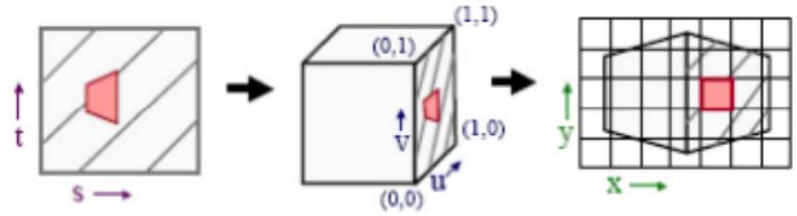
- Usando la normal del objeto: obtengo las coordenadas del mapa de la superficie intermedia mediante la superficie perpendicular a las normales del fragmento.
- Usando el centroide del objeto: similar al anterior, sólo que utilizo los vectores que salen del centro del objeto y atraviesan los vértices del fragmento en vez de las normales.
- Usando la normal de la superficie intermedia.
- Usando el rayo reflejado a través de la superficie intermedia.



3.E. ¿Cuáles son las tareas realizadas en la etapa de renderizado de la textura? En esta etapa, cuando se establece la correspondencia entre pixeles y texels pueden presentarse 3 casos, ¿cuáles? Para cada caso, la elección del texel para cada pixel puede realizarse eligiendo el texel más cercano o mediante interpolación. Ejemplifique gráficamente la elección en ambos casos.

Tareas realizadas en la etapa de renderizado de la textura:

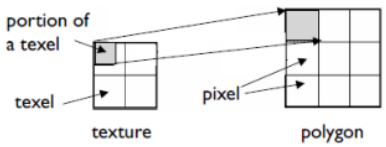
1. Se buscan los valores de textura durante la conversión scan (se mapea desde la coordenada x,y de la pantalla al sistema de coordenadas de modelado y de ahí a la imagen de textura).
2. Interpolacion hiperbolica de las coordeanadas de textura.
3. Filtrado de Magnificación -suaviza la transición entre pixels- y Desmagnificacion.



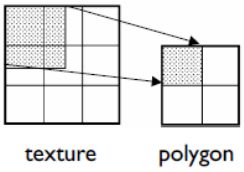
Espacio de la textura Espacio del objeto Espacio de la pantalla

Una textura consiste de primitivas o elementos denominados texels; éstos se asimilan a un conjunto contiguo de elementos (pixels en 2D) con alguna propiedad tonal o regional. Los tres casos que pueden presentarse son:

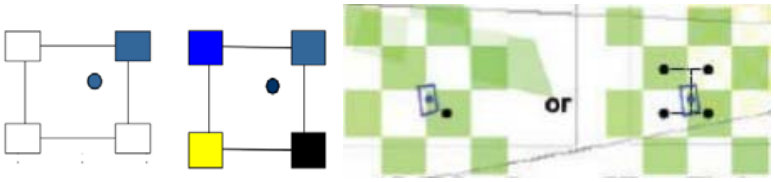
- Un pixel corresponde a un texel.
- Un pixel corresponde a un fragmento de un texel: resulta en muchos pixels con el mismo texel, sin filtrado se produce aliasing. Magnificación. La solución son los filtros de magnificación.



- Un pixel se corresponde con varios texels, común en el acortamiento de la perspectiva. Desmagnificación.



Una de las formas para seleccionar el texel que corresponde al pixel es el más cercano (nearest point sample), es simple y rápido pero de baja calidad. Otra opción consiste en hacer una interpolación lineal de los texels más cercanos.

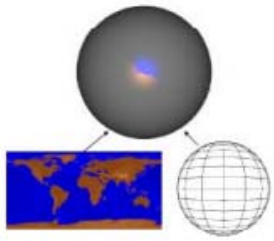


3.F. Dado un objeto 3D y una textura que se le aplica al mismo, explique en qué etapa/s del pipeline 3D se adiciona la textura al objeto. Tenga en cuenta las dos etapas de aplicación de las texturas.

Inicialmente se hace la parametrización, el mapeo de coordenadas de textura al objeto. Esto se hace en coordenadas del objeto.

En el programa de fragmentos se determinan las coordenadas de textura de cada fragmento y se accede a la textura para procesar los fragmentos en coordenadas Normalizadas del Dispositivo (3D de pantalla).

4.A. Detalle la ecuación de iluminación I utilizando el método de Phong y el mapeo de textura adecuado para obtener la imagen.



El color de un objeto es el color de su componente de luz difusa, se varía así el coeficiente de reflexión difusa:

$$I_{phong} = I_a k_a + f_{at} I_p (GRIS)(\bar{L}.\bar{N}) + f_{at} I_p T(s,t)(\bar{R}.\bar{V})^n$$

El mapeo de textura es obviamente a través de una superficie intermedia esférica mediante un indexado a través del centroide o normales del objeto/superficie intermedia (que es lo mismo en este caso).

4.B. Dada la ecuación anterior escriba los programas de vértices y fragmentos.

Programa de vértices:

```
in vec4 vPosition, Normal;
in vec2 textCoord; //corresponde a ks
uniform vec4 ka;
uniform float coeficienteEspecular;
uniform vec4 LuzPos;
uniform mat4 matrizProy, matrizMV, matrizMVNormal;
uniform textureSample sample;
const vec4 kd=(0.5f, 0.5f, 0.5f, 1f);
out vec4 color;
    void main(){
        gl_Position = matrizProy * matrizMV * vPosition;
        vec4 Norm = matrizMVNormal * Normal;
        vec3 N = normalize(Norm.xyz);
        vec3 L = normalize(LuzPos.xyz - (matrizMV  * vPosition).xyz);
        vec3 E = normalize((matrizMV * vPosition).xyz – vec3(0,0,0));
        vec3 R=relfect(L, N);

        float auxSpec = pow( max( dot(R,E), 0.0), coeficienteEspecular);
```

```
        float ks= texture(sample, textCoord);

        vec4 ambient, diffuse, specular;
        ambient = ka;
        diffuse= kd * max(dot(L,N),0.0);
        specular = ks * max(auxSpec,0.0);
        color = vec4((ambient +diffuse + specular).xyz,1.0);
        }
```

Programa de fragmentos:

```
In vec4 color;
Out vec4 colorSalida;
void main(){
        colorSalida = color;
        }
```

4.C. Hace un shader que tenga 50% de color calculado con sombreado de Phong, y 50% con color de textura cúbica.

Programa de vértices:

```
in vec4 norm, posV;
in vec2 textV;
uniform mat4 MV, MN, Proy, MVProy;
out vec3 vNE, vLE, vVE; // vector normal, luz y de vista (al ojo)
out vec2 TexCoord;
void main()
{       TexCoord = textV;
        gl_Position = MVProy * posV;
        vec4 vE =  vec4(MV * posV);
        vVE = -normalize(vE);
        // Calc. vector luz en Esp. ojo
        vLE = vec4(MV*(posL - posV));
        vLE = normalize(vLE);
        // transformar normal del Eo al Ee
        vNE = normalize(MN * norm);
}
```

Programa de fragmentos:

```
in vec2 TexCoord;
in vec3 vNE, vLE, vVE;
uniform samplerCube myTexture;//sampler2D
uniform vec4 ka, kd, ks;
uniform float CoefEsp;
out vec4 colorSalida;
```

```
void main()
{       vec4 texColor = texture( myTexture, TexCoord );

        vec3 N = normalize(vNE);
        vec3 L = normalize(vLE);
        vec3 R = reflect(-L, N);
        vec3 V = normalize(vVE);
        vec3 H = normalize(L+V);

        // Calc térm difuso+espec de Blinn-Phong
        float difuso = max(dot(L,N), 0.0f);
        float specBlinnPhong = pow(max(dot(N,H), 0.0f), CoefEsp);
        if (dot(N, L) < 0.0)
                specBlinnPhong = 0.0f;

        vec4 colorPhong = ka + kd *difuso + ks*specBlinnPhong;
        colorSalida = mix(colorPhong, texColor, 0.5f);
}
```

4.D. Dado el pipeline visto en clase: ¿qué operaciones debe proveer el programa de vértices? ¿Y el de fragmentos? ¿Cuáles son los datos de entrada y salida que DEBE tener un programa de vértices? ¿Y uno de fragmentos?

El programa de vértices debe proveer como mínimo transformaciones de vértices del espacio del objeto al de clipping (T0, T1, T2, T3). Después debería además incorporar transformación de normales, generación y transformación de coordenadas de textura, cálculo de iluminación por vértices y aplicación del color del material.

El programa de fragmentos debe proveer como mínimo la asignación de color al fragmento. Luego puede además proveer operaciones sobre valores interpolados, iluminación por fragmento, acceso a texturas, aplicación de textura, niebla, suma de colores, Alpha test. Corresponde a T5 en el pipe visto en clase.

Programa de…	Datos de Entrada	Datos de Salida
Vértices	Vértice en coordenadas del Objeto	Vértice en coordenada de Clipping
Fragmentos	Fragmento	Intensidad/Color del fragmento

4.E. Escriba uno o más shaders que permitan simular el efecto de luz parcialmente reflejada y parcialmente refractada. Para modelar la reflexión debe usarse el modelo de sombreado de Phong.

Programa de vértices:

```
const float eta = 0.66;
const float fresnelPower = 5.0;
const float F = [(1.0-eta)*(1.0-eta)] / [(1.0 + eta) * (1.0 + eta)];
in vec4 norm, posV;
uniform mat4 MV, MN, Proy, MVProy;
out vec3 refractColor, reflectColor;
```

```
out float uMix;
void main( ){
        vec3 E = vec3( MV * posV );
        vec3 EE = normalize( E ) - vec3(0,0,0);
        vec3 N = vec3(normalize( MN * norm));
        refractColor = refract(EE, N, eta);
        reflectColor = reflect(EE, N );
        uMix = F + (1.0 - F) * pow{ [1.0 – dot(-EE, N)], fresnelPower};
        gl_Position = MVProy * posV;
}
```

Programa de fragmentos:

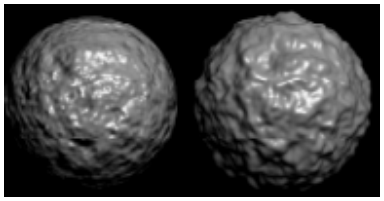
```
        in vec3 reflectColor, refractColor;
        uniform float uMix;
        uniform samplerCube myTexture;
void main( ){
        vec4 refract = texture( myTexture, refractColor );
        vec4 reflect = texture( myTexture, reflectColor );
        gl_FragColor = mix( refract, reflect, uMix );
}
```

4.F. ¿Cuáles son los tipos de parámetros que pueden utilizarse en los shaders? Especifique en que caso utiliza cada uno de ellos.

Los tipos de parámetros de lo shaders son:

- Uniform: permite el pasaje de parámetros que no van a cambiar durante las múltiples ejecuciones del programa de vértices/fragmentos durante la renderización de una primitiva.
- In (varying): sirve para variables de entrada de los shaders. Estas variables poseen valores establecidos por las etapas previas y no son constantes, es decir, no pueden ser modificadas por código del usuario. Las variables IN del programa de fragmentos pueden tener calificadores de interpolación (flat, smooth, etc).
- Out: sirve para variables de salida de los shaders, Estas variables son pasadas a la siguiente etapa del pipeline.
- Const: para variables que deben ser declaradas y su valor no será cambiado una vez inicializado, la declaración debe inicializar el valor.

5. A. ¿Qué tipo de textura se aplica para cada uno de los siguientes casos? Para cada caso describa qué debería almacenar el mapa de textura que aplica.



La primera utiliza bump-mapping o normal-mapping y la segunda es un displacement-mapping.

El bump-mapping almacena información para modificar (o reemplazar) la normal de cada punto antes del cálculo de iluminación. En bump o normal mapping el mapa de textura es un mapa que tiene la alteración que debe sufrir (o reemplazar) a la normal.

La segunda almacena un desplazamiento respecto de la posición original para así deformar la topología de la red poligonal. En casos de los terrenos, estos se pueden construir con mapas de desplazamientos basados en imágenes fotográficas de vistas aéreas en las que las elevaciones están codificadas con distintos tonos de grises. Los terrenos tridimensionales también pueden crearse generando imágenes 2D con técnicas fractales y usando estas como mapas de desplazamiento. El mapa en sí contiene las alteraciones que deben sufrir los vértices.

5.B. Describa qué es un mapa de transparencia y qué es un mapa de luces. ¿Para qué se usa cada uno de estos?

Mapa de transparencias: una estrategia para simular transparencia consiste en aplicar mapas de transparencia a la superficie de un objeto 3D. Un mapa de transparencia consiste en una imagen bidimensional que se aplica sobre la superficie de un objeto 3D con el propósito de hacer todo o parte del objeto transparente. Algunos programas usan el valor de negro para indicar zonas totalmente transparentes, otros usan el blanco. Los tonos de grises indican objetos translucidos. Se modifica así el canal Alpha y se generan hoyos en superficies sólidas sin modificar la geometría de la superficie.

Mapa de Luces: consisten en añadir una segunda textura a todas y cada una de las caras existentes en una escena 3D. Esta textura contendría la luz que recibe cada cara. Los mapas de luces son pre-computados y generalmente se usan en objetos estáticos. Son comúnmente usados en ambientes urbanos e interiores con grandes superficies planas. Los mapas de luces son mapas de texturas coloreadas. Generalmente son planas, con información acerca de la dirección de la luz. Otros motores utilizan múltiples mapas de luces para proveer información direccional aproximada para combinar con mapas de normales.

Las ventajas de su uso son muchas, desde su espectacularidad y bajo coste computacional, hasta su relativa sencillez de implementación. Tiene algunas desventajas, como su bajo nivel de detalle de las sombras, o que no son recomendables en escenarios exteriores.

5.C. Suponga que quiere texturizar una esfera mediante bump mapping. Para esto debe modificar las normales antes de calcular la iluminación. Entonces, a partir de una esfera modelada con polígonos, la normal y el mapa de textura para modificar las normales, especifique los pasos a realizar para aplicar dicha textura en cada punto de la esfera.

Inicialmente mapeo la textura a la esfera (con o sin superficie intermedia).

En el programa de vértices:

- Calcular los vectores L y V en coordenadas del ojo
- Obtener los vectores normal, tangente y bitangente en coordenadas del ojo.
- Formo la matriz.
- Calcular las direcciones de vista y de la luz en coordenadas locales de la superficie como variables de salida.

En el programa de fragmentos:

- Buscar la perturbación en la textura.
- Construir el vector normal perturbado en coordenadas locales de la superficie.

- Calcular el color del fragmento usando la normal perturbada y las direcciones de la luz y de vista.

5.D. ¿Es el algoritmo de Back Face Culling un algoritmo de cara oculta? ¿Por qué? Justifique.

No es un algoritmo de Cara Oculta porque no hace comparaciones para verificar que va adelante o atrás, solo calcula el ángulo entre la normal de la cara y la vista para decidir si esta se encuentra dentro del ángulo de visión y pueda ser entonces dibujada.

Pero esto acarrea un problema, imaginemos el caso de un toroide al cual observamos lateralmente, solo veríamos la cara exterior del mismo que esta frente a nosotros, pero en el anillo interior también una parte estará frente a nosotros por lo tanto según el algoritmo BFC deberá ser dibujada, cuando en realidad no es visible ya que está siendo tapada por la cara exterior.

5.E. El algoritmo de Z-buffer ¿es un algoritmo en el espacio de la imagen o del objeto? ¿Por qué? Justifique.

Es un algoritmo de precisión de la imagen, ya que resuelve la visibilidad en puntos discretos de la imagen. Para cada polígono verifica por cada pixel del mismo si su coordenada z está más cerca que el valor almacenado en el z-buffer, de ser así, actualiza con su valor al z-buffer y setea el pixel correspondiente de la pantalla con su color.

Cara Oculta

Dado un conjunto de objetos 3D y una especificación del sistema de vista se quieren determinar qué líneas o superficies de los objetos son visibles. Una superficie puede estar ocluida por otros objetos o por sí misma (auto oclusión).

Para la determinación de superficies visibles hay 3 tipos de algoritmos:

- Algoritmos simples que **testean visibilidad** (culling): determinan las caras que no pueden verse desde el observador. Por ejemplo back-face culling, clipping del volumen de vista canónico.
- **Precisión de la Imagen**: resuelven la visibilidad en puntos discretos de la imagen. Muestrean el modelo y luego resuelven la visibilidad. Por ejemplo ray-tracing o Z-buffer y los scan-line con buffers de profundidad (ambos en hardware).
- **Precisión del Objeto**: resuelven para todas las posibles direcciones desde un determinado punto de vista. Resuelven la visibilidad exactamente y luego muestrean los resultados. Son independientes de la dirección de vista o de la densidad de muestreo. Por ejemplo ordenamiento 3-D en profundidad, árboles BSP.

Sombreado

Sombreado Plano: Se calcula la iluminación una sola vez por polígono y se determina así un solo valor de intensidad que se aplicada a todo el polígono. Trabaja bien para todos los objetos hechos con todas las caras realmente planas. Si el modelo no tiene caras planas la apariencia depende de la cantidad de polígonos usados para simular caras curvas. Es válido cuando la fuente de luz está infinitamente lejana (paralela) o el observador está infinitamente lejano (R·V es constante).

Métodos Interpolantes

Para no evaluar la ecuación de iluminación en cada punto que corresponda del polígono, se propuso el sombreado interpolante. La interpolación se realiza a lo largo de las líneas de scan y, para aumentar la eficiencia se puede utilizar, para los parámetros que corresponda, un cálculo incremental.

Sombreado de Gourard: Se calcula la intensidad de la luz en los vértices del polígono y se interpolan estas intensidades para encontrar los valores en los pixels en los que proyecta el polígono en pantalla. La intensidad en cada vértice se calcula usando un modelo de reflexión local, por ejemplo, el método de Phong. La interpolación se realiza a lo largo de las líneas de scan de manera incremental. Interpolar linealmente el color en toda la cara. Tiene la ventaja que los cálculos incrementales son rápidos en el proceso de rasterizado.

Sombreado de Phong: Se calculan las normales en cada vértice del polígono. Luego se interpolan para cada punto del polígono que se proyecta en un pixel. Estas normales interpoladas se utilizan en el cálculo de intensidad en los puntos del polígono que corresponden a cada pixel proyectado. El sombreado de Phong da una imagen de mejor calidad que el de Gouraud. El costo computacional es más alto, sin embargo, da mejor calidad y especularidades más exactas.

Problemas del sombreado interpolantes:

- Distorsión de perspectiva
- No hay invariancia rotacional: Los resultados pueden cambiar con la orientación del polígono.
- Siluetas poligonales: No importa cuán bueno sea el modelo de sombreado interpolante usado en una superficie curva: la silueta es aun claramente poligonal. Esto se mejora subdividiendo la superficie en mayor número de polígonos a expensas de costo computacional.
- Vértices compartidos puede generar discontinuidades en el sombreado
- Normales no representativas en los vértices: Las normales calculadas en los vértices pueden no representar adecuadamente la geometría de la superficie.