

Índice

- Algoritmos de síntesis de imagen
 - Traza de rayos
 - Raster graphics
- Cauce gráfico
- Modelo de programación OpenGL

Cauce Gráfico en OpenGL

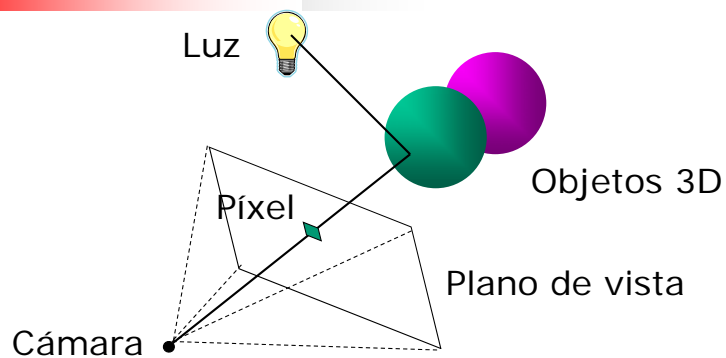
3 de Septiembre, 2010

Informática Gráfica
Ingeniería Superior Informática



2

Del mundo 3D a la imagen



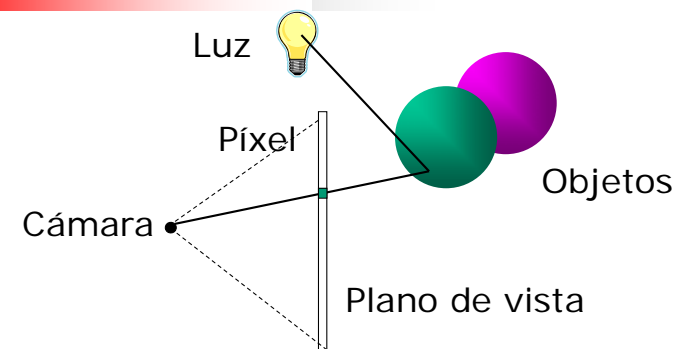
Un rayo parte de la fuente de luz, se refleja en los objetos, y llega a la cámara. El cálculo de una imagen sintética requiere la **simulación** de dicho proceso.

OpenGL sólo simula **iluminación directa** (una reflexión).



3

Ejemplo 2D



El color de un píxel viene dado por el color del rayo que pasa por el plano de imagen y llega al centro de proyección de la cámara (modelo *pinhole*).

La distancia del plano de vista a la cámara es irrelevante; no existe físicamente.



4

Traza de Rayos Vs. Raster Graphics / Rasterizado

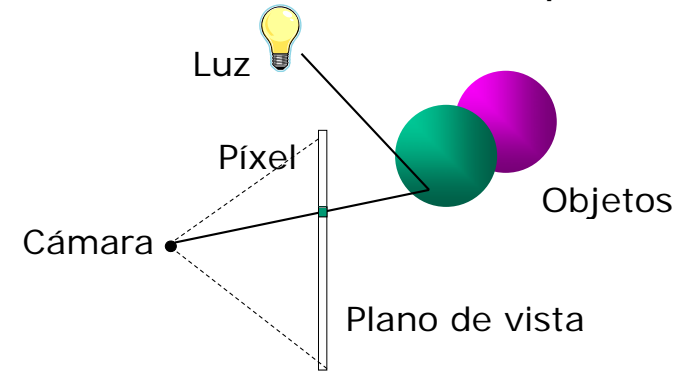
- Un rayo de luz llega tan rápido de la fuente de luz a la imagen que no nos importa en qué dirección viaja
- Traza de rayos:
 - Por cada píxel, se lanza un rayo desde la cámara, se calcula el objeto con el que intersecta, y se evalúa la iluminación
- Rasterizado (utilizado por OpenGL):
 - La superficie de cada objeto se proyecta sobre el plano de vista, y se calcula la iluminación en los píxeles ocupados.



5

Iluminación Directa

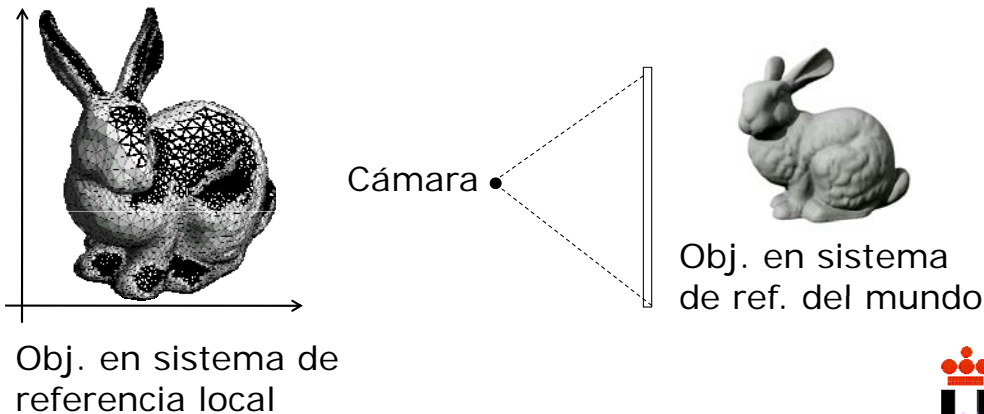
- ¿Qué datos hemos de conocer para calcular el color de un píxel?



6

1. Transformación

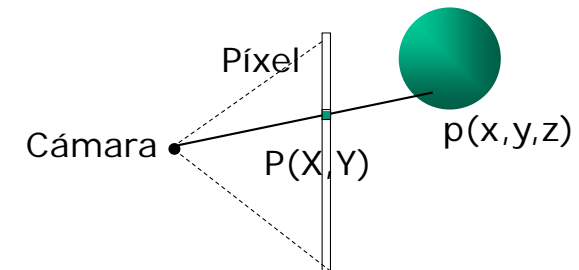
¿Cómo se representa el objeto en el mundo 3D?



7

2. Proyección

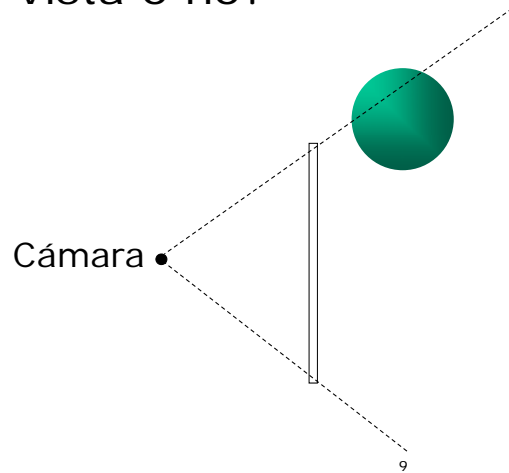
¿Qué punto del objeto es proyectado sobre el píxel?



8

3. Clipping & Culling

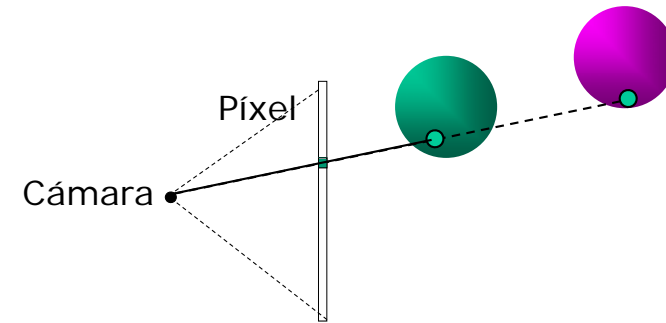
¿Este punto cae sobre el plano de vista o no?



9

4. Visibilidad

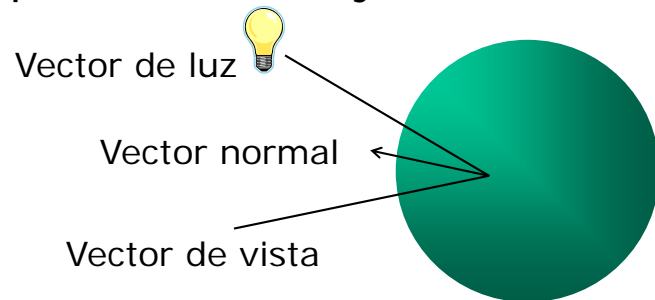
De los puntos que se proyectan sobre un píxel, ¿cuál se ve?



10

5. Iluminación

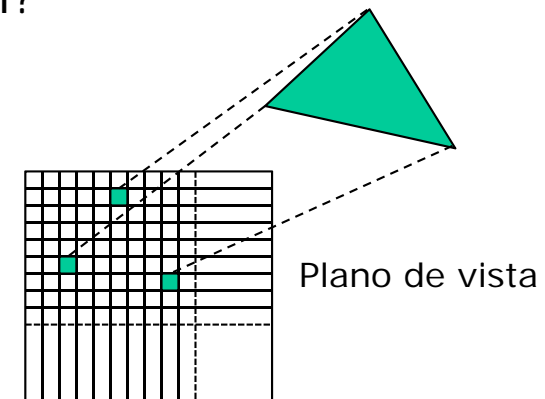
¿Cómo se calcula el color de un punto de un objeto?



11

6. Sombreado

¿Cómo se calcula el color de un píxel?



12

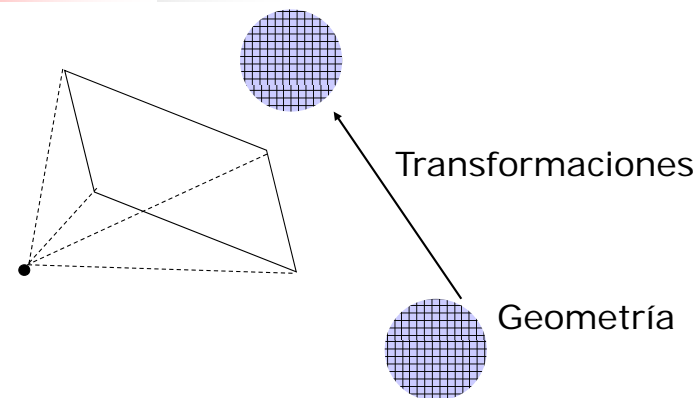
Cauce OpenGL (en orden)

1. Transformaciones
2. Proyección
3. Clipping & Culling
4. Iluminación
5. Visibilidad
6. Rasterizado
7. Sombreado



13

1: Transformaciones

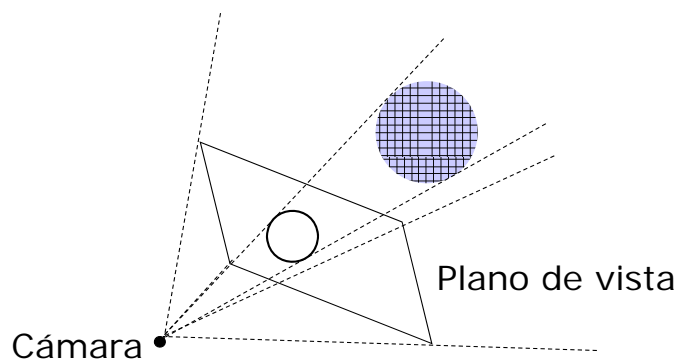


Los vértices de un objeto vienen descritos en su sistema de referencia local. Se han de transformar al sistema de referencia global, con los ejes en la cámara.



14

2. Proyección



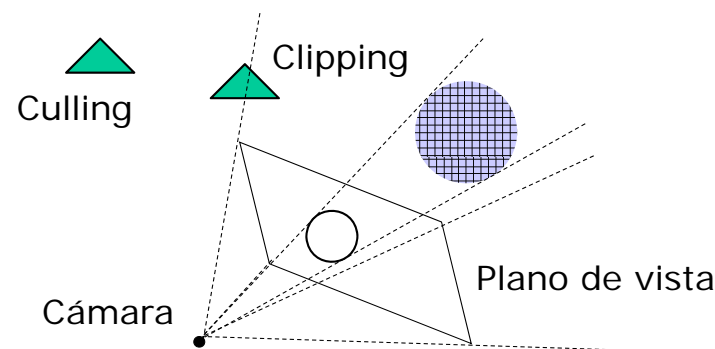
Las matrices de transformación también se utilizan para proyectar los vértices, es decir para calcular las coordenadas de píxel 2D que corresponden a cada vértice 3D.

En OpenGL, la proyección se define mediante parámetros de la cámara y del plano de vista.



15

3. Culling & Clipping



En la imagen sólo se han de pintar los triángulos que se encuentran dentro de la ventana de la imagen. La zona visible del espacio es la **pirámide de vista** (view frustum).

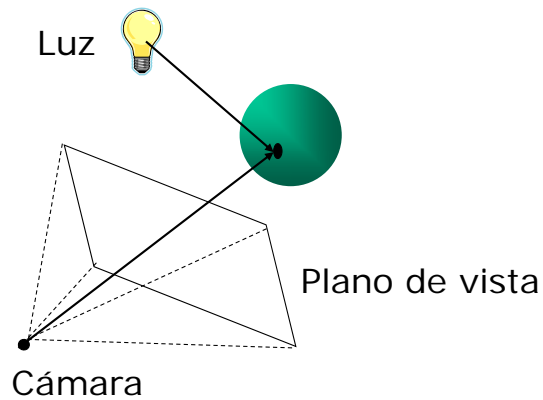
Culling: eliminar triángulos que caen fuera de la pirámide.

Clipping: recortar triángulos que caen parcialmente fuera.



16

4. Iluminación



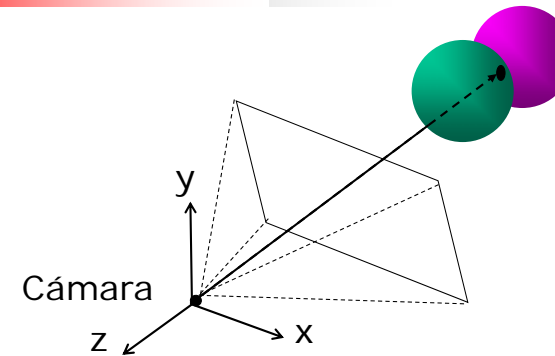
En los vértices proyectados, se evalúa el modelo de iluminación de Phong.

La visibilidad se evalúa después de la iluminación!



17

5. Visibilidad

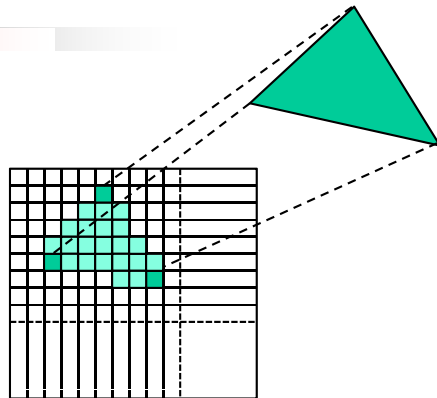


El problema de visibilidad consiste en no pintar las superficies ocultas, sólo las visibles. En OpenGL se resuelve mediante el algoritmo de **Z-Buffer**: Los valores de profundidad (z) de los píxeles pintados se almacenan en un buffer. Cuando se pinta un nuevo píxel, si su valor de z es menor que el almacenado, se desecha.



18

6. Rasterizado

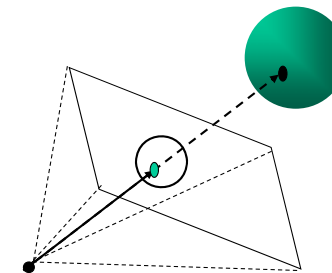


Dados los píxeles en los que se proyectan los vértices, la etapa de rasterizado se encarga de determinar los píxeles cubiertos por un triángulo y en los que hay que aplicar sombreado.



19

7. Sombreado



Finalmente, se ha de asignar color a cada píxel, mediante un algoritmo de sombreado: plano, Gouraud o Phong (no implementado en OpenGL).

La aplicación de texturas se hace a medias entre las etapas de iluminación y sombreado.



20

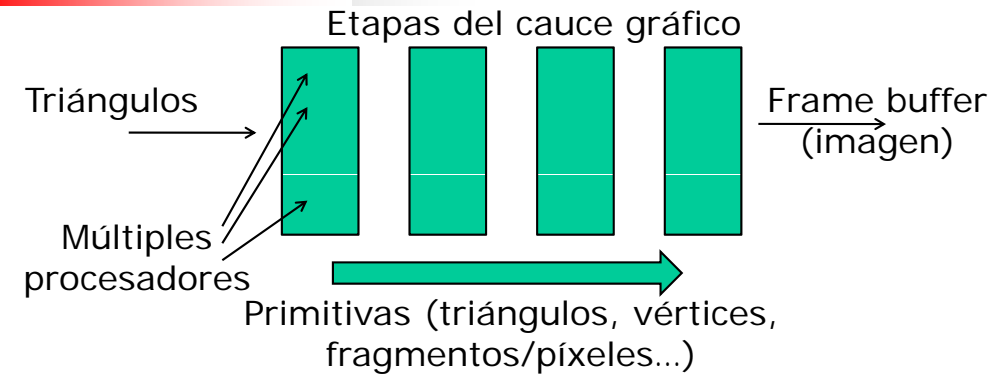
Modelo de Programación OpenGL

- En una CPU tradicional, un programa se ejecuta con una serie de instrucciones secuenciales.
 - For "todos los triángulos" do:
 - Calcular transformación
 - ...
- En una GPU, hay múltiples procesadores dedicados a cada etapa del cauce, como una arquitectura segmentada, y los datos se van pasando de una etapa a otra.
 - Stream processing
 - SIMD: Single instruction, multiple data

21



Modelo de Programación OpenGL



Los procesadores trabajan sobre las primitivas en función del estado definido en ese momento.

La programación de los algoritmos propiamente dichos, la sincronización, y el ensamblado de resultados son tareas transparentes para el programador. Esa es la virtud de OpenGL.

22



Modelo de Programación OpenGL

- En el modelo OpenGL, los procesadores de las distintas etapas ejecutan siempre la misma función mientras no se les indique lo contrario.
- OpenGL se ejecuta de acuerdo a un **estado**. Programar en OpenGL supone modificar el estado antes de enviar nuevos triángulos.
- Ejemplos de variables del estado OpenGL:
 - Transformación: la matriz a utilizar...
 - Proyección: ángulo de la pirámide de visión...
 - Culling & clipping: dimensiones de la pirámide...
 - Iluminación: luces, coeficientes del material...
 - Sombreado: plano o Gouraud...

23



Ejemplo 1

- Una luz, un objeto rojo y otro azul.
 - Inicialización:
 - Activar luz
 - Definir otros valores de estado constantes
 - En cada fotograma:
 - Definir color rojo.
 - Definir transformación del objeto rojo.
 - Enviar triángulos del objeto rojo.
 - Definir color azul.
 - Definir transformación del objeto azul.
 - Enviar triángulos del objeto azul.

24



Ejemplo 2

- Dos luces, pero sólo una afecta al objeto azul.

Inicialización:

Activar luz 1

Definir otros valores de estado constantes

En cada fotograma:

Activar luz 2

Definir color rojo.

Definir transformación del objeto rojo.

Enviar triángulos del objeto rojo.

Desactivar luz 2

Definir color azul.

Definir transformación del objeto azul.

Enviar triángulos del objeto azul.



25

Ejemplo 3

- Objeto con color distinto en cada vértice.

Inicialización:

Activar luz

Definir otros valores de estado constantes

En cada fotograma:

//Triángulo 1

Definir color vértice 1.

Enviar vértice 1.

Definir color vértice 2.

Enviar vértice 2.

Definir color vértice 3.

Enviar vértice 3.

//Triángulo 2

etc.



26

Arquitectura OpenGL

- La librería OpenGL ofrece funciones de lenguaje C para cambiar el estado del cauce gráfico o enviar y leer datos. OpenGL tiene sus propios tipos de datos.
- La gestión de ventanas, periféricos de entrada (ratón, teclado) y el pintado se realizan con librerías adicionales montadas sobre OpenGL como una capa externa. Un ejemplo muy habitual es la librería GLUT. Generalmente, todas estas funciones (p.ej. de GLUT) se programan como callbacks.



27

Ejemplo de Código

```
main ()
{
    AbrirVentana ();
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho (-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
    glBegin (GL_POLYGON);
        glVertex2f (-0.5, -0.5);
        glVertex2f (-0.5, 0.5);
        glVertex2f (0.5, 0.5);
        glVertex2f (0.5, -0.5);
    glEnd ();
    glFlush ();
}
```



28