Outline

OpenGL

- OpenGL significa Open Graphics Library, un estándar para la progamación de gráfica.
- Los comandos de gráficos son implementados por el controlador de la tarjeta gráfica y, por lo tanto, son independientes del hardware de la tarjeta gráfica, del sistema operativo y del administrador de ventanas empleado.
- Los comandos de gráficos están razonablemente cercanos al hardware y son suficientes para lograr la funcionalidad principal
- Varias bibliotecas y frameworks se basan en OpenGL y permiten la programación a un mayor nivel de abstracción.

OpenGL Versions

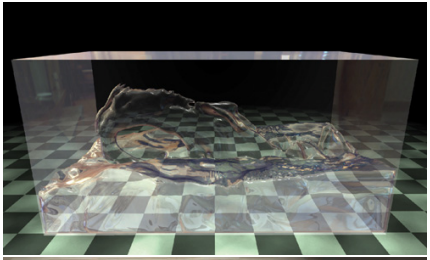
- Desde su introducción (1992) OpenGL se ha ampliado continuamente para admitir nuevas funciones de tarjetas gráficas.
- OpenGL 1.0 (1992), OpenGL 1.1 (1997), OpenGL 1.2 (1998), OpenGL 1.3 (2001), OpenGL 1.4 (2002), OpenGL 1.5 (2003)
- OpenGL 2.0 (2004), OpenGL 2.1 (2006)
- OpenGL 3.0 (2008), OpenGL 3.1 (2009), OpenGL 3.2 (2009), OpenGL 3.3 (2010)
- OpenGL 4.0 (2010), OpenGL 4.1 (2010), OpenGL 4.2 (2011), OpenGL 4.3 (2012), OpenGL 4.4 (2013), OpenGL 4.5 (2014), OpenGL 4.6 (2017)
- A partir de la versión 3.1, el Pipeline de funciones fijo ya no esta soportado, por lo cual es necesario implementar shaders, lo cual dificulta el aprendizaje.

OpenGL ES y WebGL

- OpenGL ES (Embedded System) es una versión de OpenGL con funcionalidad reducida para teléfonos móviles, televisores, tabletas, etc.
- OpenGL ES 1.0 (2003): similar a OpenGL 1.3 (Pipeline Fijo)
- OpenGL ES 1.1 (2004): similar a OpenGL 1.5 (compatible con versiones anteriores)
- OpenGL ES 2.0 (2007): similar a OpenGL 2.0 (no compatible con versiones anteriores)
- OpenGL ES 3.0 (2012): similar a OpenGL 3.3
- OpenGL ES 3.1 (2014): similar a OpenGL 4.3
- OpenGL ES 3.2 (2015): similar a OpenGL 4.3
- OpenGL ES 3.3 (2017): similar a OpenGL 4.6
- WebGL esta basado en OpenGL ES 2.0 (y WebGL 2.0 en OpenGL ES 3.0) y permite gráficos 3D en páginas web (compatible con la mayoría de navegadores)

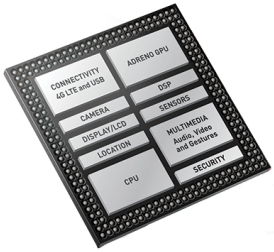
Limitaciones del Modelo de Von Neumann

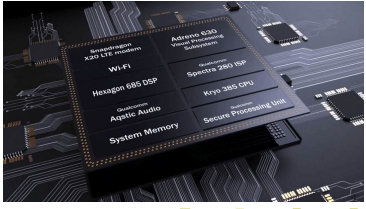
- La mayoría de las computadoras y teléfonos inteligentes tienen procesadores multicore
- Un CPU aún cuando es rápido, no esta diseñado para manejar de manera eficiente operaciones de punto flotante.
- Un CPU no tiene el suficiente poder de cómputo para generar gráficos en 3D complejos en tiempo real.
- Para procesamiento gráficos es necesario el uso de unidades de procesamiento especializadas.



SoCs para Teléfonos Inteligentes

- El término SoC significa system-on-a-chip.
- Un SoC es un sistema completo contenido en un solo circuito integrado.
- La combinación de todos sus componentes en una sola unidad de procesamiento permite un ahorro de energía significativo.
- Bloques comunes en un SoC:
  - Central Processing Unit (CPU).
  - Graphics Processing Unit (GPU).
  - Image Processing Unit (ISP).
  - Digital Signal Processor (DSP).
  - Neural Processing Unit (NPU).
  - Video encoder/decoder.
  - Modems.

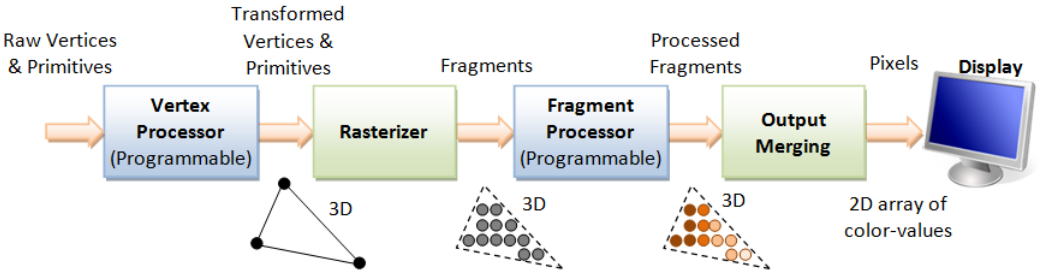




OpenGL Shading Language

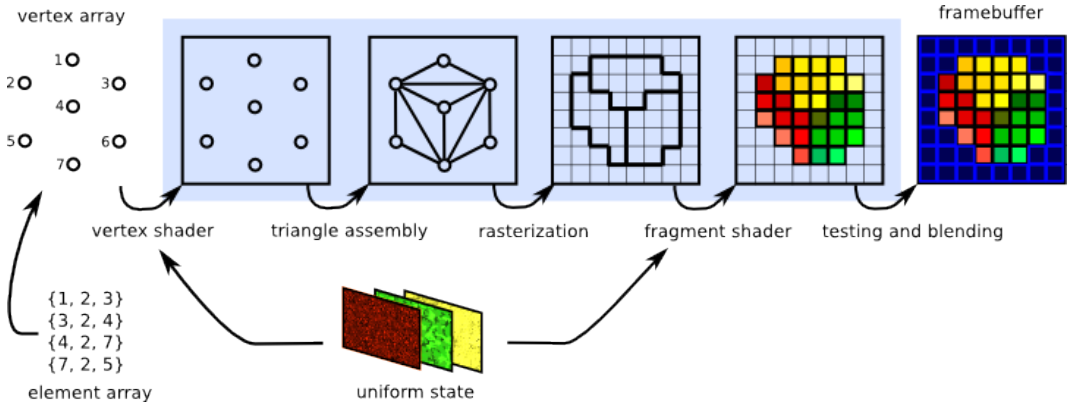
- Las aplicaciones móviles son ejecutadas principalmente en el CPU y la memoria principal
- Para procesamiento de gráficos, los programas son ejecutados en el GPU el cual tiene su propia memoria local (memoria gráfica). Existen dos tipos de procesadores
  - El vertex processor
  - El fragment processor
- Los programas del GPU son escritos en un lenguaje llamado Shading Language (Lenguaje de sombreado).
- La mayoría de los GPUs adoptaron el lenguaje de OpenGL shading Language (OGSL)

OpenGL Pipeline

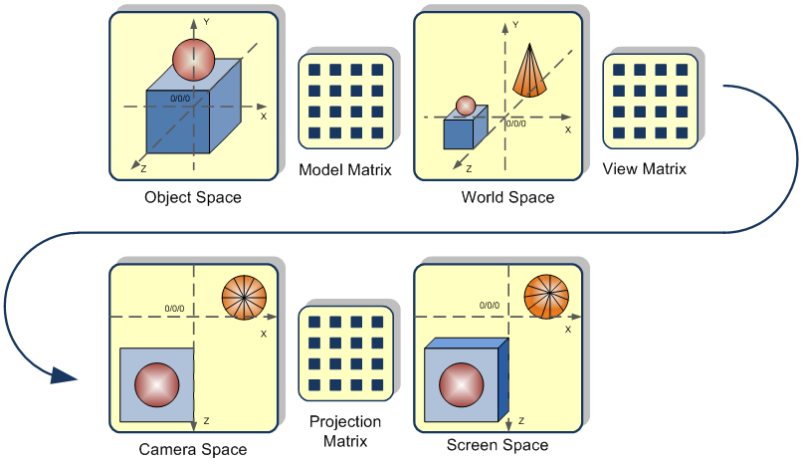


3D Graphics Rendering Pipeline: Output of one stage is fed as input of the next stage. A vertex has attributes such as (x, y, z) position, color (RGB or RGBA), vertex-normal (n_x, n_y, n_z), and texture. A primitive is made up of one or more vertices. The rasterizer raster-scans each primitive to produce a set of grid-aligned fragments, by interpolating the vertices.

OpenGL Pipeline (2)



Transformaciones

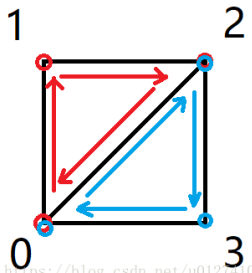


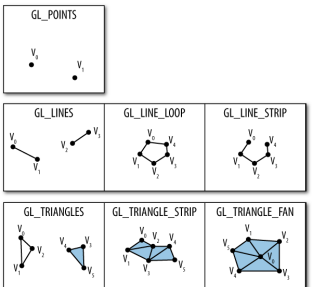
Dibujo de Primitivas

- Para dibujar cualquier superficie en OpenGL, se debe triangular los vértices adyacentes para formar polígonos (generalmente triangulos).
- Se requieren de dos triangulos para dibujar un cuadrado.

Demo #1

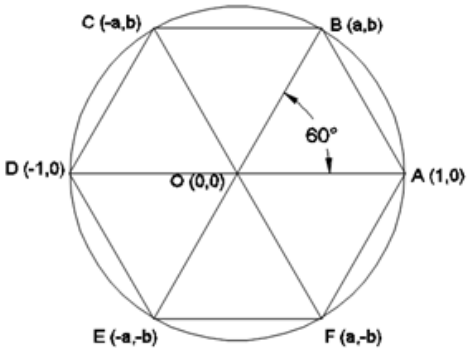
Dibujar un Triangulo en 2D





Experimentado con Primitivas

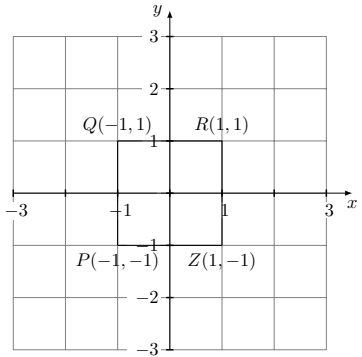
- Es posible dibujar objetos complejos si se tiene un entendimiento correcto de las primitivas
- Por ejemplo: Dibujar un Hexagono Relleno/Alambrico.

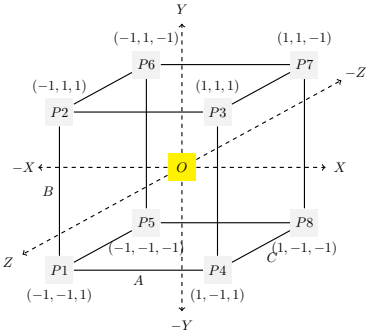


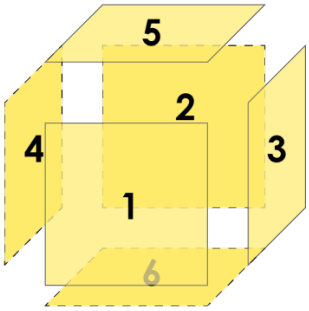
Cubo 3D

Demo #2

Cubo en OpenGL







Objetos Complejos en OpenGL

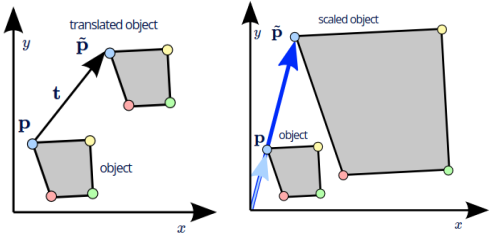
- Una vez que se entiende como generar las coordenadas de poligonos y asignarles un color, es posible construir objetos 3D complejos.

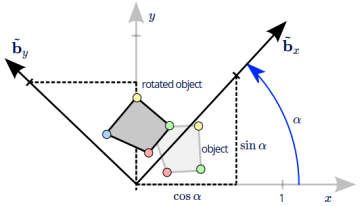
Demo #3

Objetos Complejos en OpenGL (Letra A)

Transformaciones

- **Translación.** Es un cambio en la posición de los objetos en cualquiera de los ejes.
- **Escalamiento.** Es un cambio en el tamaño de un objeto con respecto a sus dimensiones en cualquiera de los ejes.
- **Rotación.** La orientación de un objeto puede ser cambiada mediante un ángulo de rotación.



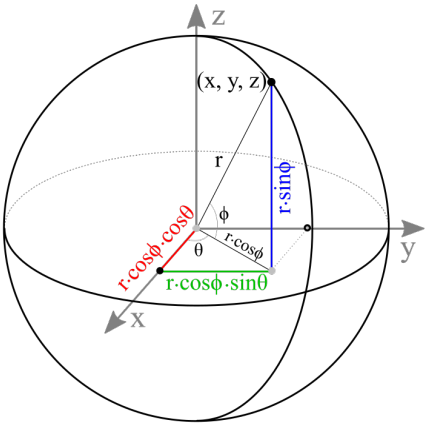


Demo #4

Transformaciones y Animación en OpenGL

Dibujar una esfera (1)

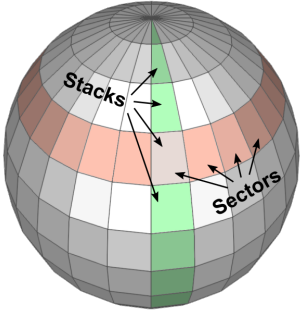
- La definición de esfera es una superficie cerrada en 3D donde cada punto de la esfera está a la misma distancia (radio) de un punto dado.
- Dado que no podemos dibujar todos los puntos en una esfera, solo tomamos muestras de una cantidad limitada de puntos dividiendo la esfera por sectores (longitud) y pilas (latitud).

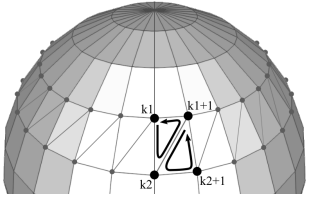


* http://www.songho.ca/opengl/gl_sphere.html

Dibujar una esfera (2)

- Para dibujar la superficie de una esfera en OpenGL, debe triangular los vértices adyacentes para formar polígonos. Es posible usar una sola tira triangular para renderizar toda la esfera.
- Cada sector de una pila requiere 2 triángulos.



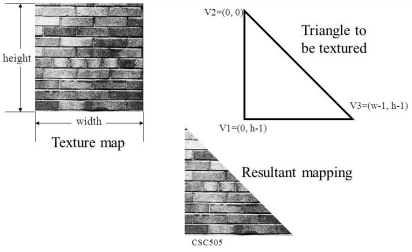


Demo #4

Esfera en OpenGL ES

Mapeo de Texturas

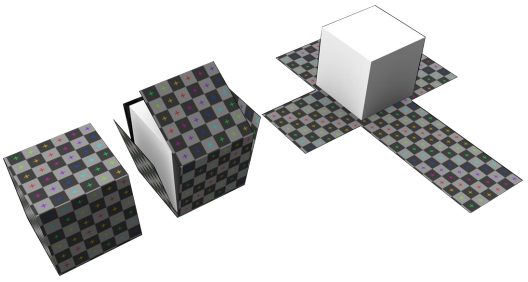
- El mapeo de texturas en gráficos por computadora se refiere a la aplicación de un tipo de superficie a una imagen 3D.
- El método común es crear una imagen de mapa de bits 2D de la textura, llamada *mapa de textura*, que luego se *envuelve* alrededor del objeto 3D.
- Para texturas mas precisas se emplean funciones matemáticas en lugar de mapas de bits.

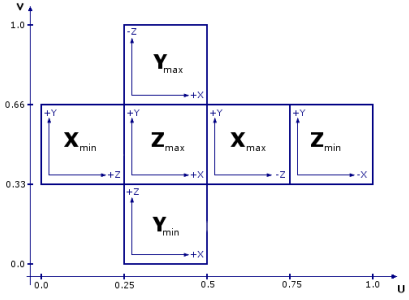


Demo #5

Texturas en OpenGL

Multiples Texturas en OpenGL



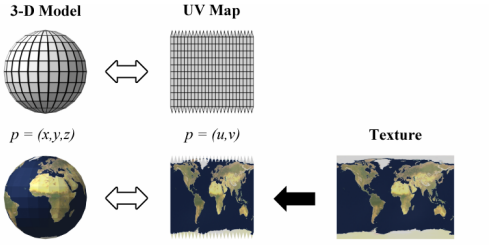


Demo #6

Multiples Texturas en OpenGL

UV mapping

- El mapeo UV es el proceso de modelado 3D de proyectar una imagen 2D en la superficie de un modelo 3D para el mapeo de texturas. Las letras U y V denotan los ejes de la textura 2D porque X, Y y Z ya se utilizan para denotar los ejes del objeto 3D en el espacio modelo,



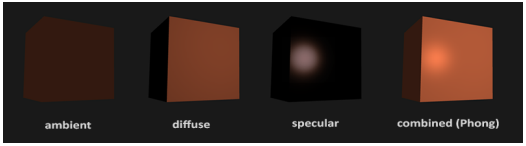
Demo #7

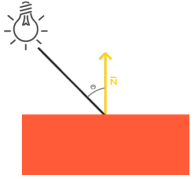
Textura sobre una Esfera

Outline

Iluminación Ambiente y Difusa

- Iluminación ambiental: lo que los objetos nunca están completamente oscuros.
- Iluminación difusa: simula el impacto direccional que tiene un objeto ligero sobre un objeto.
- Iluminación especular: simula el punto brillante (del color de la fuente) de una luz que aparece sobre objetos brillantes.





Demo #8

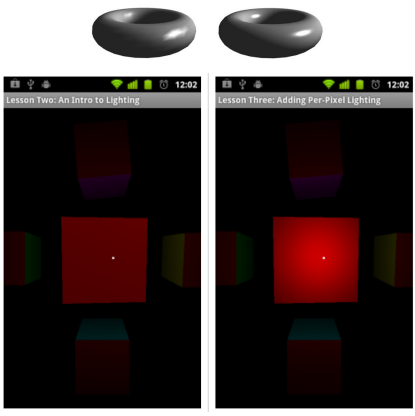
Iluminacion Ambiente y Difusa

Iluminación por Vertices

- En la iluminación por vértice, la cara frontal del cubo muestra un sombreada plana sin evidencia de indicencia de luz. Los cuatro puntos de la cara frontal son equidistantes de la luz, y intensidad en cada puntos es interpolada mediante los triángulos que forman la cara frontal.
- En la iluminación por fragmento hay un efecto mucho más notorio en la misma cara del cubo.

Demo #9

Iluminación por Fragmentos



Textura e Iluminación

- El arte del mapeo de texturas (junto con la iluminación) es relevabnte para construir ambientes 3D de apariencia realista. Sin esto, el sombreado suavemente parecerá artificial, como un viejo juego de consola de los años 90.
- Para lograr esto, es necesario enviar al vertex shader una matriz con información de coordenadas de textura.

Demo #10

Textura e Iluminación

Textura e Iluminación Sobre una Esfera

- Para agregar realismo a las escenas, es necesario combinar texturas más iluminación.
- El código en el *Fragment Shader* es más complejo para lograr dicha tarea.

Demo #11

Textura e Iluminación sobre una Esfera

Simulación de Particulas

- Una partícula es un diminuto objeto 2D que siempre está frente a la cámara y que generalmente contiene una textura con partes transparentes.
- Una partícula en sí misma es efectivamente solo un sprite (mapa de bits), que al combinarse con otros crear efectos asombrosos.
- Existe un objeto llamado emisor o generador de partículas que, desde su ubicación, genera continuamente nuevas partículas que se descomponen con el tiempo.
- Si un emisor de partículas de este tipo generara, por ejemplo, partículas diminutas con una textura similar al humo, las coloreara menos brillantes cuanto mayor sea la distancia del emisor y les diera una apariencia brillante, obtendría un efecto similar al del fuego.

Demo #15

Particulas

Skybox

- Imagina que hay una montaña en la distancia dentro de un juego. Una montaña a la que el jugador nunca podrá viajar, porque está muy lejos.
- Modelar la geometría real de esa montaña y luego renderizarla en un rango de ángulos realmente estrecho todo el tiempo requeriría recursos de la GPU.
- Es mucho más eficiente y bastante efectivo tener solo una imagen renderizada previamente de esa montaña para mostrar.
- Un Skybox es caja grande con textura alrededor del mundo para mostrar objetos muy distantes que no se pueden alcanzar.

Demo #16

SkyBox