# PPT

## Motor de videojuegos:

Es un software que facilita el proceso de creación de juegos o aplicaciones graficas. (Unreal, Cryengine, Renderware, Unity, Godot, Cocos2d, Marmalade, libGDX)

Su objetivo es proveer y encapsular tecnologías necesarias para:

* Renderizado.
* Física.
* Sonido.
* Animación.
* AI.
* Networking.

Analiza el proceso de desarrollo, provee una interfaz de fácil uso para desarrolladores, no reinventa la rueda.

## Componentes:

Renderimg, animación, Physics, AI, HID, Audio, game assets(recursos), core sytems, 3rd party SDK’s, drivers/OS y hardaware (cross plat)

## Recursos:

Texturas, Imágenes, Audio, Modelos y animaciones 3D, customizados.

## Sistemas:

Nos permiten almacenar y procesar la información de los recursos para ser usada dentro del motor.

## MiddleWare:

La mayoría de los motores utilizan software de terceros para implementar sistemas de gran calidad y así ahorrar tiempo.

Juegos:

Se ejecutan en tiempo real, se ejecutan constantemente.

Update:

Process input 🡪 update game 🡪 render 🡪 process imput ….

Tipos:

Acoplado: Los sistemas se actualizan en orden especifico,

Pros: proceso ordenado, todos los procesos tienen igual importancia.

Cons: No hay control sobre que tan frecuente cada rutina se actualiza.

En paralelo

Pros: Cada sistema se actualiza de manera independiente

Cons: Problemas de sincronización, hay dispositivos que no poseen buen manejo de threads.

Vertex Buffer Object:

Provee métodos para subir Vertex data (posición, normal, vector, color, etc.) al dispositivo de video para un rendering no inmediato.

Ofrecen mejoras sustanciales de rendimiento en comparación con el modo inmediato, principalmente porque los datos residen en el dispositivo de video en lugar de la memoria del sistema, por lo tanto, pueden ser procesados directamente por este.

Vertex Array Object:

es un objeto OpenGL que almacena todo el estado necesario para suministrar datos de vértice. Almacena el formato de los datos de vértice, así como los objetos de búfer.

Shaders:

**Vertex shade**r: Maneja el procesamiento de vértices individuales.

**Fragment shader**: Los sombreadores de fragmentos toman un solo fragmento como entrada y producen un solo fragmento como salida.

La salida de un fragment shader es un valor de profundidad, un posible valor de plantilla (no modificado por el fragment shader), y cero o más valores de color que potencialmente se escribirán en los buffers de los actuales marcos de imágenes.