Texto

Descripción generada automáticamente

Esto es un engine que usa librerías.

CALL3D, JSON, PHYSICS y alguno más. Esto genera un engine.exe y hay una carpeta data con datos (texturas, mallas, las dlls, json).

Al lado de la carpeta engine hay carpetas adicionales.

Texto

Descripción generada automáticamente

El programa abre un ficheor que se llama boot.

Vamos a usar esto el primer mes para ir aprendiendo cositas. Código de C++

* Carpeta ai para la IA
* CAL3D: para animación esqueletal, vieja pero funciona, hoy en día no lo usuaria es muy limitada.
* Components: son extras (modelo entidad componente). Antes había POO y en los juegos teníamos enemigos, y un player y teníamos una jerarquía de clases, pero llego un momento que no encajaba bien. Hacíamos un update de cada clase y se repetía código. ERA ANTI CACHE FRIENDLY. Entonces una mejor manera es con componentes. Ej de componentes:
  + **Nombre**: entidad ahora tiene un nombre

Al ponerle un nombre, tendremos una estructura que tenemos en memoria todos los componentes nombre. La entidad

Imagen que contiene pizarrón, texto, firmar, computer

Descripción generada automáticamente LA entidad por tanto no tiene un update, la entidad no piensa

* + **Transform**: contiene posición, orientación y escala
  + **Controller First Person**: coge datos de otro componente para saber donde está. Y le también el de input
  + **Input**.

No es lo mismo acceder con registros que con cache. Este es el principal cuello de botella.

Los controladores de los enemigos. Mas ventajas de esto. Si tengo una maquina con 8 CPUs, puedo hacer que cada procesador haga 100, en paralelo.

Unity no lo hace: las entidades tienen componentes, pero son un monobehaviour. La entidad ¿????

Una entidad es un array de componentes.

* **Base**: no tiene virtuales, ocupa 0 bytes

El transform ocupa en memoria algo. (quaternion, posición y escala).ç

Un pizarrón blanco

Descripción generada automáticamente

El siguiente transform

SSL, MMX, vx. Es un conjunto de avanzados de instrucciones. Los registros son de 64 bits en la maquinas de 64 bits. MMX, registros de 128 bits. Extended instructions. Y ahí encajan bien vectores, matrices…

Pizarrón blanco con letras negras

Descripción generada automáticamente con confianza media

Esto está alineado.

Yo hago ahora una clase y le metodo un método virtual update, para que los hijos lo sobreescriban.

El compilador mete una dirección para que sepa que que tiene que ejecutar en memoria.

La transform con un método virtual es un puntero (64 bits) a la tabla virtual + tus datos. Esto sirve para qu cuando el juego necesita ejecutar un método virtual pues va a la dirección de memoria esa. Entonces todo esto para que no tenga virtuales el componente base.

* Transform tiene un método load y el componente base tiene otro que se llama igual

Texto

Descripción generada automáticamente

* Fsm: finite state machine
* Geometry: usaremos las cosas de matemáticas

Cuando borro una entidad se me puede quedar un ptr a null. Entonces no me guardo un puntero. Un dibujo de un pizarrón blanco

Descripción generada automáticamente con confianza baja

Me guardo un ID (handle). ERntonces nosotros pedimos un handle.

Texto

Descripción generada automáticamente

Cast pq si tengo un handle

Devuelve un handle.

Captura de pantalla de un celular

Descripción generada automáticamente

* Imgui: estándar de videojuego. Se puede poner como un gizmo para mover el objeto.
* Input:
* Json
* Cmp physics: hace de puente cogiendo la transform y pasándosela al physics y la physics me dice cual es la nueva posición.
* Modulo: parte del código q se encarga de una feature concreta.
  + De manager: para gestionar los managers
  + Multithreading
  + De physics
* Particles
* Profiling: para saber donde estamos perdiendo tiempo
* Render:
  + Hay un componente render que le permite a la entidad decirle al render manager que quiere pintar x cosas.

Texto

Descripción generada automáticamente

* Resources: es una cosa estática que se aprovecha por mucha gente. Una textura por ejemplo. Aparece en la memoria solo una vez. Hay un propietario de esos recursos.
  + CResourceType me dice el tipo de recurso.
* Skeleton:
* Murmur: hash, tu me das un byte array y yo te devuelvo un id asociado. A partir de algo yo siempre genero lo mismo. Y es muy muy muy difícil que yo te devuelva el mismo. No va a pasar nunca. Es muy rápido. Sacamos un ID a partir de un texto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para ponerlo UTF8 todo.

Includes

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Si vemos que tarda mucho en compilar, el compilador parsea todos los include de las std. Precompile headers. El primer h es un h que tiene precompilado y lo tiene ya para todos.

El precompilado es mcv\_plaform.h y tiene que tener todo los h que necesitemos sabiendo que si tocamos este .h se precompilara tooodo el proyecto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Hay un fichero que en vez de Use tendrá un Crear. Es el cpp que tendrá el

Cualquier fichero .cpp la primera línea será #include “mcv\_platform”

Entonces Unreal, hace coge los cpps, y genera un único fichero unity.cpp q incluye otros cpps.

Ficheros importantes

* Componentes.json: tiene el max de objetos, y el orden de construcción de las entidades.

Google Chrome tiene chrome://tracing/ que si le meto el archivo

Tabla

Descripción generada automáticamente

Tenemos para un profiler