# Introducción

IA individual (autómatas, sr, bt, pathfinding) y el segundo tema es scripting.

¿Cuándo se usa, pq , etc?

La clase siguiente LUA y no se que

Miércoles no puede dar clase

Al hacer lógica o comportamientos tenemos un eje.

**Cosas frecuentes, repetitivas y rápidas /// o /// infrecuentes, diversas y lentas**.

Si hay un juego de demonios, tendremos muchos y todos se comportan igual. Y además serán rápidas en el sentido que se actualizaran en cada frame (como tiene que ser rápido debe ser en C++). 🡪 Hay pocas líneas de código.

Ejemplo 2: Witcher. ¿Cuántas posadas hay? Cientos. En cada posada hay una cinemática diferente, y pasa cada mucho tiempo (pasa infrecuentemente, es diverso) 🡪 Esto es lo que nos va a llevar al SCRIPT. De esto hay muchas líneas de código.

En gran turismo tenemos poco/nada scripting

Pero los juegos mas de misiones, que hay tantas que lo ideal es hacerlo sin que implique C++.

Maquinas Super ness, la ness, etc. ¿Cómo iba un juego de esos antiguos? Esto es anterior a la aparición de los sistemas de ficheros. Había un circuito (una ROM) e iba todo metido en la memoria.

Yo puedo meter en C un define con un array. Ese array es código o datos?

Teneiamos un chip en el que estaba ahí todo.

Juan Abadia demoscene (assemby en finlandia un concurso famoso de esos).

Sigraph feria de EEUU de graficos.

Problema: si el binario era un caos no se podía hacer un juego 2.0 de uno que ya existía.

Aparece el file system (carpetas, ficheros y extensiones).

# Scuum

**Data-driven game developent**. En el momento que separo .exe de los datos. Tengo un dato de width, height, de los efectos de sonido. El exe ahora interpreta los datos que le ponga para preparar un juego u otro. Así se saca mas dinero con menos tiempo/coste.

Por eso, cada uno tenía su engine. SCUMM (scum=basura). Pero realmente es:



Pantalla de un video juego

Descripción generada automáticamente con confianza media

Los juegos que sacaban eran así.

#### DEBERES: Jugar a “El día del tentáculo”

# DOOM / WUAKE

Un poquito más tarde, en Texas con la llegada de internet se les ocurre dos cosas:

* **Shareware**: tu jugabas un rato y al final pagas para seguir jugando. (te doy una prueba antes de pagar).
* **Publicar engine**: para la gente que compraba el juego pues le daba un editor de niveles para crear ellos sus niveles, aunque luego no se usen, pues eso vendia (son como los **mods**).

**Carmack** (el señor que hizo el DOOM), es un crack

* Quake
* Armadillo del space (compañía de vuelos).
* Fundador de Oculus Rift
* Y ahora esta con IA de texto (GPT y esas cosas)

**Publicar el Engine** = publicar el formato de fichero de lógica del juego. El formato es **WAD**. No hace un editor, solo te dice cuál es el formato. Link en <https://doom.fandom.com/wiki/WAD>

idSoftware, el engine es lo de menos. Epic Games lo contrario: pensó que las herramientas eran lo mejor.

Texto

Descripción generada automáticamente

Unreal al principio era de pago. Luego Engine gratis pero me das un %.

Baldur’s Gate se crean un Engine Aurora. De aquí luego saldrá el LUA

En un RPG lo que se necesita un árbol de diálogos.

EVOLUCIÓN

1. Codigo y datos juntos
2. Separamos código de datos
3. Mi exe es un SO pero tengo bloque de código/datos/comportamiento (mapas, triggers, diálogos…)

Esto nos permite hacer juegos más tochos. Permite meter mas trabajadores en los q cada uno se dedica a una parte.

# LUA

Lua scripting universal. Lo usan todos:

* **Sintaxis simple**.
* **Generico** (adaptable a cualquier juego)
* **Fácil de integrar**.
* **Blindado respecto a fallos** (si falla una misión no me puede petar todo el juego). Ej: punteros y liberaciones de memoria no queremos

## Propuestas:

* Propuesta Java : mala idea
* Propuesta DLLs (C++): mala idea, en C++ no sencillo, peor que la anterior
* Propuesta LUA: una lavadora lleva un SO. En SaoPaulo querían hacer esto. EMBEDDED. No tenía nada que ver con videojuegos

## Características del LUA

* **Embedded**: pensado para meterse en HW sencillos (lavadoras)
* **Semicompilado**:
  + **BASIC**: lenguaje interpretado. Hay un parser que coge una línea, la averigua la ejecuta. La verificación de lo que quiere hacer el programa va en tiempo de ejecución.
    - Son más lentos
    - Pero no requieren compilador
  + C / C++: lenguajs compilados. Analiza la sintaxis, detecta errores y genera un fichero compilado. El rendimiento es optimo.
  + JAVA / LUA: Semicompilado.
    - .lua o .java.
    - Tengo el código ASCII y el código maquina. Gran parte de la comodidad del basic es q no comprueba la sintaxis. Valida la sintaxis en una primera pasada. La ejecución la puedo hacer muy rápida.

El código intermedio (bytecode) y este se interpreta.

* + - **Extensible**: ojo

C qué tiene si no le pongo include? Main, for, while, =, pero el printf no. Viene con una librería con el .h pero si te vas al .cpp ya tiene la implementación. Yo extiendo el lenguaje añadiendo el include stdio.h

Empieza a haber muchas palabras reservadas y por eso usamos los namespaces.

El LUA tiene for, functions, … pero tb tiene una cosa única, lo podemos extender con cadenas ASCII

Tengo un fichero que se llama perico.cpp y ahí tengo una función void lalala(){…. }

Void main(){

Publica (“p\_lalala”, \*lalala);

}

Y tengo tamb un fichero .lua (ASCII).

Yo compilo todo y ejecuto y llamo a la llamada. La cadena asci extiende, entonces yo en mi script lua, puedo hacer una llamada p\_lalala() a C++

Tiene la capacidad de extenderse a si mismo, pudiendo llamar a funciones de otros lenguajes (C++ en nuestro caso).

Como el código está en C++ va muy rápido.

## Pasadas para sintaxis:

* Tiene palabras reservadas de por si.
* Tiene palabras publicadas por el usuario
* Si no es ninguna de esas dos, SINTAX ERROR.
* Esto lo hace en su “semicompilación”

## Arquitectura

* Implemetado en C, no C++. Para publicar miembros, atribs, clases no es tan trivial (pero se puede).
* Codigo abierto, licencia gratuita.
* Hay distintos Wrappers (uno contra C, otra contra Java, otra C++).

Nota: los compiladores tienen varias pasadas:

* Pasada léxica. Al encontrar pritf, lo pone en una tabla de símbolos. En ese momento no tiene ni idea de que es (el printf es un candidato a dar un syntax error) **(palabras)**
* Pasada sintaxis: si hay un paréntesis, si hay un punto y coma. **(relaciones entre palabras)**
* Linkado: mira el printf si está por otros includes.

El LUA tiene un mapa (ASCII, puntero). Es como que deja el linkado para el tiempo de ejecución.

### Pasos Cronologicamente

1. Publicar funciones
2. Ejecutar e interpretar

En ensablador como se hace una función a una llamada? Lua lo hace igual con una pila (push para los parámetros y desde la función pop de los parametros).

En LUA tamb se pasa luego el NUM de parámetros (para el blindaje).

Hay una maquina virtual (lua\_State). Se inicializa así:



Una función se registra así



Para ejecturar un script lua



Lua\_close(ls) para cerrar la maquina virtual

Nuestra pila es el LS. Alguien pushea los parámetros al LS. En el top de la pila está el numero de parámetros (nos dice el scope de variables locales). La cosa es que podemos acceder como si fuera un array. La pos 1 es el primer argumento

Un dibujo de un pizarrón blanco

Descripción generada automáticamente con confianza baja

Ejercicio: hacer una función en LUA que haga la suma de dos números

Vaidar que hay 2 numeros

Texto

Descripción generada automáticamente

Para devolver un resultado:

Texto

Descripción generada automáticamente

Y yo devuelvo 1 pq devuelvo un resultado solo.

RECORDAR:

Hay que publicar la función con el register su no no va

Texto

Descripción generada automáticamente

Func suma pero q reciba un num arbitrario de parámetros.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Ojo en el lua yo puedo escribir código, es un lenguaje de programación

Texto

Descripción generada automáticamente

Encapsulamos los eventos, se lanzand desde C++ y desde Lua tenemos las funciones OnGameStart, OnPlayerDead.

Captura de pantalla de un celular

Descripción generada automáticamente

Hemos visto como integrar con C, pero nos falta el C++ (objetos, atributos, miembros). Habrá que usar una librería o algo.