PROGRAMACIÓN AVANZADA EN UNITY3D

- Game Programming Patterns -

Enrique Sainz para Level-Up

Qué es Game Programming Patterns

- ***Game Programming Patterns*** es un libro escrito por Robert Nystrom
- Está basado (parcialmente) en el ***Design Patterns*** de **Erich Gamma**, **Richard Helm**, **Ralph Johnson** y **John Vlissides**
- En ***Game Programming Patterns***, **Nystrom** propone y adapta diferentes patrones de diseño para el desarrollo de videojuegos

- Contexto -

Contexto: de Ingeniero a Jefe de Equipo

- **Robert Nystrom** se inició en el desarrollo de videojuegos en **Electronic Arts**, donde trabajó desde el **2001 al 2010**

- Durante este tiempo, trabajó como **desarrollador** en juegos como **Madden PC**, pero muy pronto giró hacia el desarrollo de **herramientas y tecnología**

- En el **2011**, cuando ya había entrado a trabajar en **Google**, publica su ***Game Programming Patterns***
    - Robert Nystrom estuvo trabajando en su libro durante varios años, motivado por la cantidad de veces que sus compañeros “reinventaban la rueda”

Contexto: Programar Juegos sin Unity

- En los años en los que **Robert Nystrom** trabajo en **EA**, no existía **Unity**, y hacía relativamente poco que habían aparecido motores gráficos como **Irrlicht**, **Ogre** o **Torque 3D**
  - **Unreal Engine** ya existía (en 1998 se publicó Unreal), pero pasaron varios años hasta que se popularizó su uso como herramienta Third Party y alcanzó la madurez actual
  - Para entonces ya existían otras herramientas de terceros como **Renderware**, que se utilizó en el **GTA III**

- Muchas de las soluciones se tenían que **implementar a mano**, en **C++**, específicamente para cada juego
  - Y pocos compañeros de **Robert Nystrom** conocían el libro de ***Design Patterns***

- Contenido -

Contenido (I)

- **Patrones de Diseño - Revisitado**
  - Command
  - Flyweight
  - Observer
  - Prototype
  - Singleton
  - State

- **Patrones de Secuenciado**
  - Double Buffer
  - Game Loop
  - Update Method

Contenido (II)

- **Patrones de Comportamiento**
  - Bytecode
  - Subclass Sandbox
  - Type Object
- **Patrones de Desacoplamiento**
  - Component
  - Event Queue
  - Service Locator
- **Patrones de Optimización**
  - Data Locality
  - Dirty Flag
  - Object Pool
  - Spatial Partition

- Formato -

Formato del Libro : Énfasis

- El libro de **Robert Nystrom no es específico** para videojuegos, pero sí que **resuelve problemas muy usuales** en éste tipo de software

- Por eso hace énfasis en:
  - **Tiempo y Secuencia:** las cosas tienen que ocurrir en el orden y momento correctos
  - **Ciclos de Desarrollo Comprimidos:** tiene que poderse iterar rápidamente sin que eso los desarrolladores se estén pisando entre ellos
  - **Escalabilidad a través del Desacoplamiento:** cada elemento nuevo tiene que poder interactuar con todos los anteriores, pero sin que esto implique una intrincada red de llamadas entre elementos
  - **Rendimiento:** cuanto más le sacamos a la plataforma de destino, mejor es nuestro juego... ¿no? (respuesta: ***SÍ, POR SUPUESTO***)

Formato del Libro : Estructura

- El libro describe cada patrón en varias secciones:
  - **Intención** - para qué sirve el patrón, en términos de los problemas que permite resolver
  - **Motivación** - ejemplo de problema que se resuelve con este patrón
  - **El patrón** - describe el patrón, y cómo implementarlo
  - **Cuando usarlo** - indicaciones sobre cuándo es mejor usar o evitar el patrón
    - Tiene una subsección **A tener en cuenta** sobre los riesgos y consecuencias de utilizar el patrón
  - **Ejemplo** - muestra un ejemplo de implementación paso a paso
  - **Decisiones de Diseño** - comenta las diferentes decisiones que se pueden tener que tomar al implementar el patrón para diferentes casos
  - **Ver También** - otros patrones de diseño que pueden estar relacionados.

- Objetivo -

Nuestro Objetivo

- Durante este curso, vamos a dedicarnos a estudiar los diferentes **patrones de diseño propuestos por Nystrom**, e intentar adaptarlos a nuestro entorno (**Unity3D**)
  - Sabemos que existen **diferencias** importantes entre C++ y C#, o entre **programar** un juego y **“montarlo/scriptarlo”** en **Unity3D**, pero nos adaptaremos

- El objetivo final, como siempre: conseguir un código/proyecto mantenible, escalable, elegante y **SOLID**o