# Análisis selección de motor gráfico para el estudio.

## Criterios de selección

// PARA EL ANALISIS DE RIESGOS MALA SELECCIÓN DE MOTOR GRÁFICO//

Para el presente proyecto una de las partes fundamentales es seleccionar un framework, motor gráfico o SDK, sobre el que realizar todos los objetivos que se pretenden alcanzar. Para seleccionar el programa se deben definir previamente unos criterios que permitan que esta elección sea lo más correcta posible ya que de ella depende en gran medida el éxito del proyecto.

Estos criterios se han pensado concretamente para cumplir los objetivos y las aplicaciones que se desean conseguir no servirían por lo tanto para otros propósitos que no sean los propios del presente proyecto.

Posteriormente a la determinación de estos criterios se le otorgará a cada uno de ellos una puntuación máxima y mínima que se podrá otorgar en la fase de análisis a cada uno de los motores, sdk o frameworks candidatos a ser seleccionados. La suma de estas puntaciones determinará cuál de estos programas se va usar para realizar el proyecto.

### Criterio 1: Facilidad de uso.

Este criterio trata de determinar la complejidad para hacer un uso básico del programa. No se habla del desarrollo en sí de proyectos y juegos, sino de elementos como la accesibilidad de la interfaz, la facilidad o no de manejar los comandos o la estabilidad del programa.

Se puede decir que es una forma de medir si el programa se adecua a los estándares que un usuario puede comprender rápidamente y moverse por el mismo de una manera rápida y efectiva consiguiendo sus propósitos funcionales rápidamente.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Pre requisitos de instalación. | 0-2 |
| Instalación. | 0-2 |
| Accesibilidad interfaz. | 0-3 |
| Visualización de elementos. | 0-3 |

### Criterio 2: Curva de Aprendizaje.

La curva de aprendizaje es un apartado importante para decidir el programa va ser objeto de las acciones de este estudio. Ya que pretende ser un proyecto que en algunas partes del mismo, como puede ser las guías de aprendizaje puedan ser realizados posteriormente por usuarios de todos los niveles.

Por esta razón se busca un programa en el que la curva de aprendizaje no sea demasiado alta y pueda ser asumida por un usuario medio. Respecto a este criterio no se trata de la curva para aprender a dominar todos y cada uno de los aspectos del programa, sino del a para poder empezar a trabajar con el programa y crear pequeños proyectos, es decir, el foco está en el comienzo de la curva de aprendizaje.

Se busca por lo tanto un programa que sea muy sencillo en los primeros momentos y de adaptación por parte de los usuarios muy rápida sin importar tanto la dificultad de dominar todos los aspectos del mismo.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Primeros pasos. | 0-2 |
| Funciones básicas. | 0-4 |
| Creación de pequeño proyecto. | 0-4 |

### Criterio 3: Documentación, tutoriales y comunidad.

Como el presente proyecto pretende ser de alcance mayoritario, sin necesidad de conocimientos previos de ingeniería o programación, es fundamental que el programa elegido cuente con una buena documentación, tutoriales y una gran comunidad de usuarios.

En este apartado se incluyen toda la documentación oficial proporcionada oficialmente por los propietarios del software y su soporte oficial, los tutoriales tanto proporcionados oficialmente como realizados por usuarios en plataformas de video online, como You Tube o Vimeo, o en blog especializados y las comunidades o foros de desarrolladores que trabajan con el programa.

Se otorgará puntuación extra a aquellos programas que con lo comentado anteriormente en castellano.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Documentación y tutoriales oficiales. | 0-4 |
| Tutoriales y guías no oficiales. | 0-3 |
| Comunidades y foros de desarrolladores. | 0-3 |
| Castellano. | 0-2 |

### Criterio 4: Conocimientos previos de programación.

La necesidad de conocimientos previos de programación de algún lenguaje especifico, como podría ser C++ o Java para poder trabajar con el software va a penalizar a los mismos, por lo tanto aquellos que no requieran este tipo de conocimientos previos su puntuación será más alta en este criterio.

Este criterio se refiere a conocimientos en algún lenguaje específico de programación, pero no tendrá en cuenta elementos que se presuponen que son necesarios para la creación de juegos como pueden ser la lógica básica o la estructura propia de los mismos.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| No necesidad de conocimiento específico. | 0-4 |
| No necesidad de conceptos de lógica de programación. | 0-4 |
| No necesidad de programación lenguaje propio. | 0-2 |

### Criterio 5: Editor Gráfico/Visual.

El contar con un editor visual para poder posicionar los objetos , personajes en el mundo y con el que poder crear relaciones entre ellos sin necesidad de *escribir código* solo mediante una edición gráfica es un punto esencial para seleccionar un framework u otro.

Se valorará no solo que tenga un editor gráfico sino que este sea accesible, sencillo e intuitivo para los usuarios.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Editar visual propio. | 0-2 |
| Editor visual intuitivo. | 0-4 |
| Funcionamiento general. | 0-2 |
| Idiomas del editor. | 0-2 |

### Criterio 6: Escalabilidad.

El presente proyecto en su apartado de guía de uso pretende ser una iniciación al uso de un framework de videojuegos tanto para usuarios expertos en programación como para usuarios medios.

Pero los conocimientos adquiridos durante la realización del mismo deben servir tanto para realizar pequeños proyectos como los que aquí se proponen como para futuros juegos más grandes y ambiciosos. Por lo que el programa seleccionado debe tener una gran escalabilidad que permita afrontar problemas mayores y seguir progresando en cuanto a la complejidad de los proyectos creados.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Posibilidad de grandes proyectos. | 0-4 |
| Posibilidad de crear funciones y scripts propios | 0-3 |
| Add-ons y plug-ings | 0-3 |

### Criterio 7: Resultados profesionales.

Los resultados que otros desarrolladores y/o creadores han conseguido con los diversos frameworks también es un criterio importante a la hora de seleccionar sobre el que se va realizar las acciones del presente proyecto.

Elementos como la implantación de dichos motores en el mercado de juegos profesionales y los acabados gráficos y/o estéticos que se pueden conseguir con ellos es esencial para que el motor sea adecuado para este estudio, ya que como se ha comentado pretende fomentar posteriormente un uso más profesional de dicha herramienta.

Un apartado muy importante en este apartado es la diferenciación de los juegos profesionales o la no repetición de patrones visuales. Esto hace referencia a que los proyectos puedan ser tan diferentes tanto visualmente como funcionalmente que no se parezcan entre ellos y se note la procedencia de los mismos. Es fundamental que el programa evite en la medida de lo posible que el usuario pueda detectar con un vistazo rápido con que herramienta ha sido desarrollado el juego. Se habla aquí de la primera sensación y de carácter usuario ya que un desarrollador experto puede detectar el motor gráfico, incluso en grandes proyectos de la industria o

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Juegos comerciales. | 0-4 |
| Resultados visuales. | 0-3 |
| Diferenciación de los juegos. | 0-4 |

### Criterio 8: Estado actual.

El presente trabajo se desea que tenga carácter estable y que pueda ser usado durante el mayor tiempo posible tanto por estudiante que quieran conocer este tipo de herramientas como de usuarios medios que quieran iniciarse en desarrollo de juegos por lo que es bastante importante el estado actual, es decir en el momento de la realización del estudio de la herramienta.

El estado actual se refiere principalmente a si es una tecnología que se mantiene activa, si cuenta con continuo soporte y actualizaciones por parte de sus creadores o si su uso está creciendo, se mantiene estable o ésta disminuyendo por la aparición de otra tecnología.

Por lo tanto se pretende asegurar que se selecciona un programa con persistencia y con el que los usuarios puedan seguir desarrollando en un futuro.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Tiempo de la última versión | 0-1 |
| Frecuencia de actualizaciones | 0-1 |
| Estado | 0-3 |

### Criterio 9: Requerimientos y requisitos del sistema.

El carácter mayoritario y generalista ya comentado del presente trabajo, aconseja tener en cuenta para la selección de la herramienta de desarrollo los requisitos y requerimientos del sistema para hacerla funcionar y poder trabajar con ella. Estos deben ser adecuados para llegar al mayor número posible de usuarios teniendo en cuenta tanto el hardware como el software y sistema operativo.

En dicho criterio se mide los requerimientos de la herramienta en sí, no de los juegos y proyectos realizados con las mismas.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Requisitos | 0-2 |
| Sistemas operativos soportados | 0-3 |

### Criterio 10: Licencia.

El precio del software necesario y el tipo de licencia de cada programa a analizar también es un punto importante a la hora de seleccionar el mismo.

Al pretender ser un proyecto abierto y generalista tendrán mejor puntuación aquellas herramientas con un precio menor o aquellas que sus versiones gratuitas cuenten con unas menores limitaciones de uso.

Pese a ello, este criterio tiene menos importancia que otros ya que no se debe desvirtuar el objeto del estudio, ni que por seleccionar un software de menor precio o licencia con menos limitaciones no se llegue a la consecución de determinados objetivos o no se completen de manera satisfactoria.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Precio de Licencia | 0-3 |
| Modo prueba o licencia free | 0-2 |
| Precio módulos o add-ons | 0-2 |

## Resumen puntuación

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Facilidad de uso. | 0-10 |
| Curva de Aprendizaje. | 0-10 |
| Documentación, tutoriales y comunidad. | 0-12 |
| Conocimientos previos de programación. | 0-10 |
| Editor Gráfico/Visual. | 0-10 |
| Escalabilidad. | 0-10 |
| Resultados profesionales. | 0-10 |
| Estado actual. | 0-5 |
| Requerimientos y requisitos del sistema. | 0-5 |
| Requerimientos de los juegos. | 0-5 |
| Licencia. | 0-5 |

## Selección de motor gráfico

## Primera selección

Antes de proceder a un análisis de los posibles candidatos, se ha realizado una primera lista de posibles motores, que pueden cumplir los criterios referidos en él punto anterior y ser a ser el medio con el que se van a cumplir los objetivos del presente trabajo.

Esta primera lista parte de la búsqueda en general de motores gráficos y frameworks para desarrollar juegos 2D, sin entrar a valorarlos en los criterios anteriormente citados. De ellos se descartaran automáticamente aquellos que posean características incompatibles con los objetivos planteados, aquellos que se vea sin necesidad de estudio completo que no cumplen los requisitos buscados o aquellos que por otros motivos justificados no puede ser una opción.

|  |  |
| --- | --- |
| Click Team Fusion 2.5 | Torque 2D |
| RPG-Maker | Gdevelop |
| Unreal 4 2D | Game Maker |
| Cocos2D | GODOT |
| Construct 2 | Marmalade |
| Eclipse Origins | Unity 2D |
| Adventure Game Studio | Stencyl |
| Angel 2D |  |

#### Motores descartados y justificación

##### Torque 2D

Torque 2D o Torque Game Builder es la adaptación a 2D del motor gráfico Torque 3D creado por la compañía GarageGames para su juego Tribes 2 en 2001. Esta versión 2D del motor fue lanzada en 2013 bajo licencia MIT\* lo que le confiere un punto a favor al tratarse de software libre. Pese a ser a contar con una licencia libre Torque 2D posee la posibilidad de comprar una interfaz de desarrollo Torque Studio.

Después de una primera aproximación se ha podido comprobar que la instalación y configuración es demasiado compleja para el usuario medio, la creación de proyectos es excesivamente tediosa y la indispensabilidad de contar con conocimientos de programación para poder crear contenido se ha descartado.

##### Unreal 4 2D

Unreal 4 es la evolución del motor desarrollado en 1998 para el juego del mismo nombre y que bajo él y sus constantes evoluciones se han desarrollado juegos tan populares como Deus Ex, Bioshock, Batman: Arkham Asylum o Mass Effect.

Como pasará en su día que evoluciono desde un motor exclusivo de juegos en primera persona o shooters a abarcar también juegos tercera persona ahora en sus últimas versiones intenta también dar soporte a juegos 2D.

Aunque su potencia es muy grande y se pueden realizar grandes proyectos con él toda su interfaz es pensada para elementos 3D con lo que en algunos aspectos resulta bastante complejo el manejo de las 2D dimensiones, además es un motor bastante exigente respecto a los requerimientos del sistema tanto para trabajar con el cómo de los proyectos resultantes.

##### Cocos2D.

Cocos2D es un framework para diseñar y crear juegos 2D inicialmente basado en Phyton\* aunque diferentes versiones lo han adaptado a diferentes lenguajes de programación como por ejemplo la versión X o Js que dan soporte a C++ y Javascript respectivamente.

Cocos2D se descarta en esta primera fase ya que aunque posee una creación rápida de proyectos y un manejo fácil y rápido de Sprites y sus acciones es imprescindible programar ya sea en Phyton con la versión original o en los lenguajes propios de otras versiones.

Cuenta además con la contra de no contar con una interfaz visual de manejo, creación y posicionamiento de los elementos, su funcionamiento es similar a motores como SFML, en los cuales tienes que desde código llamar a las funciones propias del mismo se realizan diversas acciones, pero añadiéndole su propio entorno de programación.

##### GODOT Engine.

A GODOT Engine es un motor de videojuegos multiplataforma y de código abierto. Su funcionamiento basado en nodos y escenas lo diferencia en gran medida de los motores y framworks analizados en el presente proyecto.

Uno de los objetivos de este proyecto es que usuarios no programadores se acerquen al desarrollo de videojuegos y cree sus primeros proyectos, para posteriormente con la base adquirida poder aumentar su conocimiento, esto incluye poder utilizar cualquier otro motor o framework que no sea el seleccionado, por lo que uno que trabaje de forma tan distinta y diferencia del resto no es un buen candidato.

A parte del su funcionamiento diferenciado nos encontramos también en un caso como el de Cocos 2D, donde necesitamos nociones de programación para poder realizar proyectos en GODOT.

##### Marmalade.

Marmalade es un SDK y motor gráfico basado en C++. Es uno de los SDK´s de desarrollado más utilizados por los grandes estudios para crear juegos 2D y 3D , juegos como Lara Croft and the Guardian of Light, Plants vs Zombies , Cut the Rope o el modo Zombies 2D de Call of Duty: World at War. Ha sido premiado en los TIGA\* Games Industry Awards 2015 en la categoría Engine and Middleware, Tools and Tech.

Pese al gran potencial de Marmalade y las cotas de calidad que pueden alcanzar sus juegos, su entorno basado en el lenguaje C++ que se debe conocer y dominar para crear contenido y su alta curva de aprendizaje hacen que sea descartado.

// tiga.org Asociacion de desarrolladores, productores publishers y vendedores de videjuegos.

##### Eclipse Game Engine.

Eclipse Originigs es un motor gráfico y una interfaz o suite para la creación de MMORPG\* 2D. Su uso es sencillo y su interfaz visual hace bastante sencillas las tareas de creación de escenarios y colocación de Sprites.

Su limitación a un solo género, la escasa documentación y comunidades activas unido a que es motor que prácticamente ha quedado en desuso hacen que no sea un buen candidato.

##### Unity 2D

Unity es uno de los motores de videojuegos más utilizados hoy en día tanto por grandes equipos como por pequeños estudios independientes. Su gran escalabilidad, gran interfaz visual, carácter multiplataforma y el potente sistema de scripts lo convierten un motor apto para cualquier nivel usuarios.

Pero el problema principal que hace que sea descartado es que Unity esta pensado principalmente para desarrollo de juegos 3D, aunque sus ultimas versiones incorporan la posibilidad de desarrollar juegos 2D todavía no tiene toda la potencia que si tiene para juegos 3D. Aunque no se duda de que en futuras versiones sigan mejorando en ese sentido.

Al igual que con Unreal 2D, al ser un motor 3D adaptado a 2D, es pesado y consume excesivos recursos.

|  |  |
| --- | --- |
| MOTIVO | CANDIDATO EXCLUIDO |
| Complejidad. | Torque2D, Unreal 2D, Marmalade. |
| Alto consumo de recursos | Unreal 2D, Unity2D |
| Necesidad de conocimientos en programación | Cocos2D, GODOT Engine. Marmalade |
| Diferencia excesiva con el resto | GODOT Engine. |
| Limitaciones | Eclipse Game Engine. |
| Escasa documentación | Marmalade. |
| Programa en desuso | Eclipse Game Engine. |