## 

Título del Trabajo  
Fin de Grado

Grado en Ingeniería Multimedia



Trabajo Fin de Grado

Autor:

Nombre y Apellidos

Tutor/es:

Nombre y Apellidos

Enero 2014



Tabla de contenido

[1](#_Toc481830179)

# Justificación y Objetivos

## Objetivos

### Objetivos Generales

#### Objetivo General: Conocer los principales Frameworks de creación de videojuegos 2D para no programadores.

La principal motivación de este proyecto es analizar y conocer los Frameworks y SDK´s que permiten la creación de videojuegos sin tener conocimientos a priori de lenguaje hombre-máquina.

Dada la gran popularidad que han experimentado los videojuegos en los últimos 20 años en el mercado existen una gran cantidad de programas que permiten (o al menos lo intentan) realizar esta función, se pretende encontrar los más utilizados o los que mejor consiguen realizar esta función prestando atención a los resultados profesionales que se pueden llegar a conseguir con ellos.

Conocer accesoriamente los juegos comerciales más populares que se han desarrollado con estas herramientas así como los modelos de negocios de algunas de ellas.

#### Objetivo General: Analizar en profundidad los Frameworks y seleccionar un candidato adecuado para el estudio.

Una vez conocidos los SDK´s y Frameworks de este tipo se va a seleccionar uno de ellos en función de su importancia en el mercado, de los resultados y se va analizar todos sus aspectos y funcionalidades.

Para ellos se van a comparar los distintos programas mediante unos criterios que se determinaran para que el seleccionado se ajústate a los objetivos del presente proyecto.

/COMPLETAR /

### Objetivos Específicos.

#### Comparación de juegos en programación clásica y con un framework.

Para tener una visión global de lo que este tipo de programas que la presente memoria intenta analizar se necesita comprar los resultados, consumo de recursos, tiempos, curvas de aprendizaje o dificultad de uso entre otros aspectos en proyectos realizados con este tipo de software y la programación más clásica de videojuegos usando motores de programación pura como podría ser SFML o OGRE 3D.

Para conseguir esta comparación de ambas formas de crear juegos se va a realizar el mismo juego tanto en SFML como en él programa seleccionado y hacer una comparación de todos los elementos comentados anteriormente

# Introducción.

# Marco teórico.

En este apartado del proyecto se pretende describir los conceptos fundamentales con los que se va trabajar, así que será el punto de partida para todos los desarrollos teóricos y prácticos posteriores. También sentará las bases para lo que se realizará en los siguientes puntos.

El marco teórico se divide en tres apartados:

1. En la primera parte se abordarán los conceptos que se van a manejar durante el proyecto.
2. La parte central del marco teórico se dedicará a definir unos criterios de selección en base a los cuales se van a puntuar los **frameworks** que se analizarán en el cuerpo del trabajo.
3. Con los criterios ya definidos, se va realizar una primera selección de **frameworks**. En esta preselección se van a reducir todo lo posible el número de programas a analizar en el cuerpo del trabajo y excluir aquellos que tengan incompatibilidades con la finalidad del proyecto o impidan la consecución de los objetivos.

## Conceptos previos.

### Framework.

Un **framework** en esencia es un patrón o esqueleto para el desarrollo de una aplicación, juego o contenido multimedia. Una definición más académica o forma podría ser que un **framewok** es un conjunto estandarizado de conceptos, prácticas y criterios para desarrollar y crear aplicaciones.

Los **frameworks** suele incluir:

1. Soporte de programas.
2. Librerías.
3. Lenguaje de scripting.
4. Software accesorio para desarrollar partes especificas del programa.
5. Un lenguaje de programación o interpretado.

Su uso implica un coste de aprendizaje aunque a largo plazo facilita el desarrollo, mantenimiento de la aplicación y permiten evitar trabajar en los detalles de bajo nivel permitiendo concretar esfuerzo en las partes esenciales de programa.

Desde el punto de vista del desarrollo de software, un **framework** es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

### SDK (Kit de desarrollo).

Un **SDK o Kit de desarrollo** es un conjunto de herramientas y programas de desarrollo de software que permite al programador crear aplicaciones para un determinado paquete y estructura de software, plataforma de hardware, sistema operativo de computadora o máquina.

Un **SDK** no es más que una **API[[1]](#footnote-1)** (interfaz de aplicaciones) que permite el uso de uno o varios (aunque esta opción es menos habitual ya que suele trabajar en exclusividad con un lenguaje) lenguajes de programación.

Los **SDK** suelen llevar incluidas herramientas que ayudan al desarrollador a crear sus aplicaciones. Entre ellas destacan las siguientes:

1. Debug.
2. Soporte para le detección de errores de programación.
3. Códigos de ejmplo y documentación.
4. Entorno de programación **IDE[[2]](#footnote-2).**

### Game Engine (Motor de videojuegos).

El **Game Engine o Motor de videojuegos** proporciona un entorno de desarrollo integrado para la creación de juegos, no es más que un framework diseñado para la creación y desarrollo de juegos.

Un motor de juego suele incluir diferentes componentes que se corresponden con los elementos de un videojuegos. Los elementos que suele incluir entre otros son los siguientes:

1. **Programa del juego principal.**
2. **Motor de render.**
3. **Motor de audio.**
4. **Motor de físicas.**
5. **Motor de IA.**
6. **Motor de Red.**

Antes de los Game Engine se escribían como entidades singulares y no reutilizables. El término motor de juego apareció a mediados de los 90 ligado al crecimiento de los juegos 3D especialmente shooters en primera persona, ya que permitía en juegos muy complejos técnicamente no comenzar desde cero en cada nuevo desarrollo.



Figura 1 : Wof3d Pimer motor de juegos 3D

## 3.2 Criterios de selección

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: Requisitos e instalación.

El primer criterio que hay tener en cuenta a la hora de analizar los diferentes motores a analizar es el requisito de funcionamiento e instalación.

En cuanto los requisitos de instalación se valorará con mayor puntuación, dado el carácter generalistas del presente proyecto, a aquellos programas que necesiten menos recursos del sistema para funcionar y que además funcionen en más sistemas operativos.

Es importante que el proceso de instalación sea tenido en cuenta, para comprobar la sencillez del mismo, la posible necesidad de instalación de otros programas o librerías accesorias y una buena guía al usuario durante todo el proceso. En este apartado será también tenido en el tiempo que dura el proceso.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Requisitos de instalación. | 0-4 |
| Instalación. | 0-2 |

Tabla 1: Criterio: Instalación y requisitos

### Criterio 2: 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 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,. en blog especializados y las comunidades o foros de desarrolladores que trabajan con el programa.

El pilar fundamental de este criterio es el análisis de los tutoriales oficiales. Estos tutoriales son la primera aproximación de los usuarios al programa y por lo tanto deben contar con todos los elementos para que se puedan llevar a cabo de manera correcta, rápida y entendible. Se necesita un programa con unos tutoriales que aporten conocimiento e inciten al a seguir aprendiendo y desarrollando habilidades, ya que se pretende que el presente proyecto sea un punto de partida a un aprendizaje en profundidad del programa y por lo tanto es necesario contar con los recursos necesarios.

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

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Documentación, foros y tutoriales oficiales | 0-4 |
| Análisis de los tutoriales oficiales | 0-8 |
| Comunidades y foros de desarrolladores. | 0-4 |

Tabla 2: Criterio : Documentación , tutoriales y comunidad

### Criterio 3: Funcionamiento y dificultad de uso.

El funcionamiento general del programa trata de detectar fallos técnicos tanto en la fase de instalación o posteriormente durante el uso del programa tales como errores que impiden la ejecución, el manejo o la realización de las distintas acciones y 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 y usar de una manera rápida y efectiva consiguiendo sus propósitos funcionales.

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.

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.

El apartado de un lenguaje propio de programación no penaliza a aquellos programas que disponen de un sistema propio de programación, sino la necesidad de su aprendizaje o su manejo para la creación de las funcionalidades más básicas de un juego y para un uso básico del mismo, ya que a priori el contar con un lenguaje propio de programación para desarrollar nuevas funcionalidades y expandir las posibilidades no previstas por el proyecto es una ventaja que valoraremos en el criterio escalabilidad, es la obligatoriedad o no lo que se ésta valorando aquí.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Funcionamiento General | 0-4 |
| Primeros Pasos | 0-4 |
| Funciones básicas | 0-12 |
| Lógica de programación necesaria | 0-4 |

Tabla 3: Criterio: Funcionamineto general y curva de aprendizaje

### Criterio 4: 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. Se tendrá en cuenta también el funcionamiento general , es decir que no tenga bugs ni errores que impidan su funcionamiento o que dificulten el proceso de creación y desarrollo de los proyectos.

La usabilidad[[3]](#footnote-3) en el funcionamiento de interfaz y de sus elementos también será un criterio a tener en cuenta.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Editor/Gráfico visual | 0-8 |

Tabla 4: Criterio: Editor gráfico/visual

### Criterio 5: 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 crear funciones y scripts propios y/o lenguaje de programación propio. | 0-8 |
| Add-ons y plug-ins | 0-6 |

Tabla 5: Criterio: Escalabilidad

### Criterio 6: 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 |
| Principales juegos desarrollados | 0-8 |
| Diferenciación | 0-4 |

Tabla 6: Criterio: Resultados profesionales

### Criterio 7: Estado actual.

A la hora de seleccionar un framwork, se tiene que tener en cuenta el estado actual del mismo. Se trata de encontrar un framework que permita un uso prolongado en el tiempo y para ello debe estar activo, es decir, no abandonado por sus desarrolladores. Se valorará también la frecuencia de actualizaciones ya que permite detectar que los bugs y problemas que surgen en todo programa con su uso se van solucionando, además que estas nuevas versiones permiten incluir nuevas funcionalidades.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Estado actual | 0-2 |
| Frecuencia de actualizaciones. | 0-1 |
| Fecha desde la última actualización. | 0-1 |

Tabla 7: Criterio: Estado actual

### Criterio 8: Sistemas operativos soportados y requerimientos.

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 hacer funcionar los juegos. Estos deben ser adecuados para llegar al mayor número posible de usuarios teniendo en cuenta tanto el hardware como el software y sistema operativo.

Teniendo una gran influencia positiva en su puntuación aquellos programas que posibiliten la exportación de sus juegos a un mayor número de dispositivos posibles. En este apartado no se valorara si esta exportación a los diversos dispositivos y sistemas operativos es una función propia del programa base y viene incluida en su licencia principal o necesita módulos especiales de pago o gratuitos para llevarla a cabo, ya que esto ya será teniendo en cuenta en el criterio licencia.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MAX-MIN |
| Posibilidades de exportación: Sistemas y plataformas | 0-8 |

Tabla 8: Criterio: Sistemas operativos soportados

# Criterio 9: 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 Licencias | 0-4 |
| Modo de prueba o free | 0-2 |
| Precio módulos de exportación | 0-2 |

Tabla 9: Criterio: Licencias

### Puntuación total.

|  |  |
| --- | --- |
| CRITERIO | PUNTUACIÓN MIN-MAX |
| Requisitos e instalación | 0-6 |
| Documentación, tutoriales y comunidad | 0-16 |
| Funcionamiento y dificultad de uso | 0-24 |
| Editor Gráfico/Visual | 0-8 |
| Escalabilidad | 0-14 |
| Resultados profesionales | 0-12 |
| Estado actual | 0-4 |
| Sistemas operativos soportados y requerimientos. | 0-8 |
| Licencias | 0-8 |
| TOTAL | 0-100 |

Ilustración 1: Puntuación Total

### 

## 3.2 Primera selección de framework.

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 |  |

Tabla 10: Primera selección de motores

### Motores descartados y justificación

#### Resultado de imagen de torque 2d logoTorque 2D

Figura 2: Torque 2D

Tabla 11: Logo 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[[4]](#footnote-4) lo que le confiere un punto a favor al tratarse de software libre.

Pese a ser a contar con una licencia libre Torque 2D pose 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 es prácticamente indispensable de contar con conocimientos de programación para poder crear contenido por lo que se 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.

Tabla 12: Unreal 4

Figura 3: Unreal Engine

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 la versión Js que dan soporte a C++ y Javascript respectivamente.

Figura 4: Cocos2D

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

Ilustración 2: Cocos 2D

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.

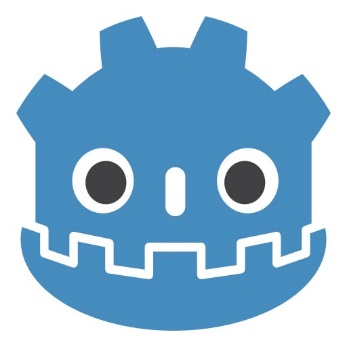
A GODOT Engine es un motor de videojuegos multiplataforma y de código abierto. Su funcionamiento basado en nodos y escenas lo diferencia en gran medida de los motores y framworks analizados en el presente proyecto.

Figura 5: GODOT Engine

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 Games Industry Awards 2015 en la categoría Engine and Middleware, Tools and Tech.

Figura 6: Marmalade

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.

#### Eclipse Game Engine.

Eclipse Origins 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.

Figura 7: Unity

Pero el problema principal que hace que sea descartado es que Unity ésta pensado principalmente para desarrollo de juegos 3D, aunque sus últimas 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.

#### Stencyl

Stencyl es un programa de creación de videojuegos 2D para computadoras y dispositivos móviles. Es un software gratuito con la opción de comprar módulos de pago.

Stencyl funciona dividiendo cada ciclo de ejecución en tres pasos creación , dibujado y actualización de datos, este ciclo de ejecución. Para las físicas hace uso del motor Box2D.

El principal inconveniente de Stencyl es que usa Flash para la exportación de sus juegos y las vulnerabilidades de Flash ha hecho que la mayoría de navegadores hayan dejado de dar soporte a dicha tecnología, aunque es posible seguir usando las aplicaciones en Flash incluyendo Flash Player en el sitio Web es conveniente tratar de usar otros programas que hagan uso de otras tecnologías.

|  |  |
| --- | --- |
| 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. |
| Motivos de Seguridad | Stencyl |
| Programa en desuso | Eclipse Game Engine, Stencyl |

Tabla 13: Exclusiones de Frameworks

### Motores preseleccionados.

#### Gdevelop

Gdevelop es un software para la creación e juegos 2D y 3D, si necesidad de utilizar ningún lenguaje de programación. Gdeveloper es un programa de código abierto y totalmente gratuito.

Figura 8: Gdevelop

El programa ésta escrito en C++ y trabaja fundamentalmente con la librería gráfica SFML , añadiéndole por encima una interfaz basada en WxWidgets[[5]](#footnote-5), utiliza también TinyXMl para la gestión de mapas y Tiles.

Gdevelop trabaja en base a varios editores que abarcan todo el proceso de creación de un videojuego. Los editores con los que cuenta Gdevelop son los siguientes:

**Gestor del proyecto:** Es el editor principal del programa en el cual se indican los datos básicos del juego como pueden ser nombre o autor o pantallas de cargas. A través de este editor se accede al resto de editores funciona como entorno principal de trabajo.

**Editor de escenas:** Permite organizar cada escena, pantalla o nivel del juego de manera individual. En él se realiza por ejemplo el posicionamiento de objetos o enemigos. Cuenta con un depurador para comprobar el correcto funcionamiento de la escena y un *player* que posibilita probar la escena.

**Editor de objetos en la escena:** Posibilita la creación de los distintos tipos objetos que pueden encontrarse en una escena. Permite crear distintos tipos de objetos como podrían ser Sprites, Textos u objetos 3D. En este editor se le añaden todos los efectos tanto de físicas como de partículas o comportamientos predefinidos.

**Editor de las capas de la escena:**  Gdevelop puede trabajar con diferentes capas dentro de una escena lo que ayuda a organizar y mejorar el trabajo pudiendo por ejemplo tener una capa para enemigos, otra para objetos de fondo, otros para objetos con los que poder colisionar etcétera.

**Editor de eventos:** Gestionan los eventos de la escena y permiten realizar las funciones propias del juegos. Este editor compone toda la capa lógica del juego y en la que se va definir toda la jugabilidad del mismo.

**Banco de imágenes:** Gestiona todas las imágenes, videos y gráficos que se van a utilizar en un proyecto.

#### Clickteam Fusion.

Click Team Fusion, apartir de ahora (CTF) es un framework para la creación de videojuegos especializada en juegos 2D lanzado en el año 2013 que actualiza la herramienta de creación y programación de juegos Multimedia Fusion que fue una de los primeros programas que permitían crear juegos de manera visual aunque de una manera limitada.

Pero con la actualización de Multimedia Fusion a Click Team Fusion se añadieron toda una serie de herramientas que hicieron que del programa una potente herramienta para crear videojuegos de manera rápida y visual. Algunas de estas características fueron por ejemplo la integración del motor de físicas Box2D que le daba a las mismas un acabado profesional, nuevo sistema de objetos que hacían más sencilla la tarea la creación de los mismos y una nueva interfaz que llevaba la programación gráfica a un nuevo nivel no visto en las versiones anteriores.

Lo más destacable de CTF es su sistema de gestión de eventos, que constituye el núcleo central de todo el programa.

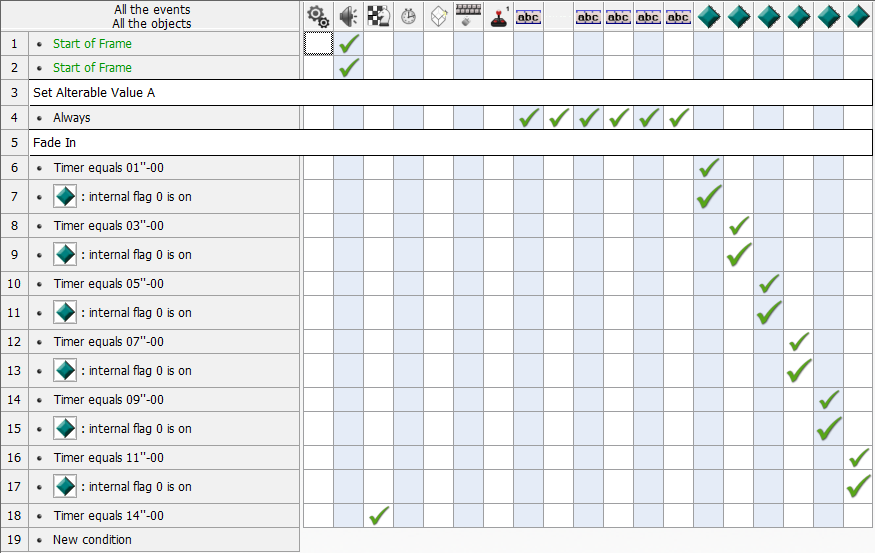


Figura 9: Sistema de eventos Click Team Fusion

#### Construct 2

Construct 2 fue lanzado en Febrero de 2011. Pensando como un motor de creación de videojuegos para usuarios sin conocimiento de programación basando en dos aspectos claves el drag and drop de elementos y en la otorgación de patrones de comportamiento de manera visual.

Figura 10: Construct 2

El programa desarrollado por Scirra Lt**\*** ésta escrito en C++ y Javacript.

#### Game Maker

Fundamentalmente divide su funcionamiento en dos grandes partes, por un lado en el sistema de eventos y comportamientos, que se compone de hojas de eventos en la que cada hoja tiene una serie de eventos y sus sentencias condicionales y una vez que se cumpla se realizará el comportamiento asignado, por otro lado en evitar la selección de instancias mediante la selección de los objetos por medio de selectores que especifica el creador en las hojas de comportamiento.

Game Maker Studio, a partir de ahora GM es un software y kit de desarrollo para la creación de videojuegos creado en **Delphi\*** nacido en el ámbito académico a finales de los noventa en Países Bajos. El profesor de la universidad de Utrecht, Mark Overmars, empezó a desarrollar una herramienta para ayudar a los estudiantes en la animación de personajes tanto de juegos como de otros ámbitos del dibujo, pero el proyecto fue creciendo y se convierto en un software para que los estudiantes pudieran crear videojuegos en los primeros momentos de su carrera con pocos conocimientos de programación. En 1999 GM dejo de ser únicamente un programa para los alumnos de Overmars y fue lanzado al mercado.

Figura 11: Game Maker

La principal característica de Game Maker es que se intenta adaptar a todo tipo de usuarios, ya que permiten realizar la implementación de las funcionalidades de dos formas completamente distintas una desde su interfaz visual haciendo uso del Drag and Drop sobre objetos y otra mediante su lenguaje de programación GML.

En 2012 Yoyo Games (compañía propietaria de GM) lanzó en paralelo a la versión estándar de Game Maker otra versión llamada Game Maker: Studio, que incorporaba cambios profundos el programa tanto a nivel visual como a bajo nivel e incluía nuevas posibilidades de exportación a plataformas y sistemas operativos. En 2015 la versión Game Maker fue retirada por Yoyo Games y la rama Game Maker Studio se convierto en la única versión del programa.

1. conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orienta-da a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. [↑](#footnote-ref-1)
2. Aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software, como pueden ser editor de código fuente, herramientas de construcción automática o compilador. [↑](#footnote-ref-2)
3. La Usabilidad es la medida de la calidad de la experiencia que tiene un usuario cuando interactúa con un producto o sistema [↑](#footnote-ref-3)
4. Esta licencia es una Licencia de software libre permisiva lo que significa que impone muy pocas limitaciones en la reutilización y por tanto posee una excelente Compatibilidad de licencia [↑](#footnote-ref-4)
5. bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++ [↑](#footnote-ref-5)