Desarrollo de un editor de sprite sheets en HTML5

Ignacio Soto Alsina

11 de Octubre del 2014

Índice de contenidos

1. Alcance del proyecto

1.1. Formulación del problema

Las facilidades que ofrece Internet han hecho que cada vez se creen más grupos de desarrolladores de videojuegos independientes, estos suelen ser estudios de gente que acaba de empezar o personas que realizan el desarrollo de videojuegos como diversión. También hay gente que sin llegar a implementar un videojuego realizan pruebas de funcionalidades que requieren tratar con gráficos.

Muchas de estas personas son programadores sin experiencia en gráficos que debido a que tienen dificultades para encontrar artistas especializados o no disponer del capital suficiente como para contratar alguno rebajan la calidad del arte final del producto. Esto comporta que sea más difícil vender sus ideas o convencer a los jugadores.

Algunas de las soluciones que toman son extraer los gráficos de algún videojuego ya existente o buscarlos por Internet. Cuando uno de estos equipos toma la decisión de hacer el juego en 2D y de buscar los gráficos por Internet, el recurso que necesita son los llamados "*Sprite sheet*". Los *sprite sheets* consisten en archivos que contienen uno o varios *sprites*.

Un *sprite* es un mapa de *bits* que se dibuja en pantalla, puede ser tanto un personaje como un mueble, una puerta, una casa, un árbol, un *ítem*...

Ilustración 1: Ejemplo de Sprite

Existen *sprites* estáticos, que son formados por una sola imagen o *sprites* dinámicos, que están formados por varias imágenes para hacer una animación.

En las *sprite sheets* se suelen poner todas las imágenes que forman parte de las animaciones (también llamadas *frames*) seguidas una detrás de otra.

Para utilizar el *sprite sheet* en el videojuego, el programador debe introducir en su código que parte de la imagen es un *sprite* específico. Si la *sprite sheet* tiene animaciones, también tiene que indicar el orden y la duración en la que su programa debe mostrar cada *frame*.

Ilustración 2: Ejemplo de sprite sheet

Buscar por Internet estos recursos hace que tengan que realizar un tratamiento del archivo antes de poder utilizarlo en su videojuego.

Uno de los problemas principales con el que los desarrolladores se pueden encontrar es que el artista añada animaciones que los desarrolladores no necesiten, si se cargan estas animaciones en el juego directamente sin tratamiento crearan una carga innecesaria.

Otro problema con el que se pueden encontrar es que el orden de las imágenes para las animaciones no sea el mismo orden en el que se muestran en el archivo, esto comporta que el desarrollador tenga que averiguar cual es el orden correcto para hacer la animación.

Este proyecto se trata de crear una herramienta online para ayudar a estos pequeños grupos con el tratamiento de los gráficos 2D para sus videojuegos.

1.2. Alcance del proyecto

Para solucionar los problemas expuestos se va a crear una herramienta online para el tratamiento de *sprite sheets* con diversas funcionalidades divididas en varias fases.

En la primera fase:

- Se creará una herramienta en la que el usuario podrá subir imágenes y recortarlas fácilmente según los sprites que vaya a utilizar para las animaciones en su videojuego.

- Se añadirá la opción de juntar archivos diferentes en un mismo archivo.

- Dispondrá de la posibilidad de que, mientras se están realizando los ajustes, el usuario pueda ir viendo en la herramienta como se verá la animación.

- El usuario podrá ajustar el tiempo entre cada imagen de la animación. Además dará la posibilidad de que dentro de una animación existan varios tiempos diferentes entre imagen e imagen.

- Se podrá modificar el color de los *pixeles* del *sprite sheet* como si una herramienta de edición de imágenes se tratara.

En la segunda fase el proyecto se enfocará en la automatización de tareas:

- Mediante algoritmos de reconocimiento de imágenes, la aplicación auto-detectará que partes de la imagen corresponden a la misma animación para ayudar al usuario en la selección.

- Se podrá auto-generar código que genere la animación mostrada en la pre visualización para algunas librerías conocidas.

- Se creará una nueva opción que permitirá comprimir el espacio vacío en la imagen.

- La aplicación podrá ajustar automáticamente el tiempo que hay entre imagen e imagen de una animación.

- Se podrá realizar ajustes en el archivo como la interpolación de imágenes

En la última fase del proyecto se ampliara la plataforma añadiendo varios aspectos sociales:

- El usuario podrá crear una cuenta para conservar los archivos que vaya modificando dentro de la plataforma.

- Existirá la posibilidad de compartir los ficheros con otros usuarios.

- Posibilidad de poner los archivos como públicos o privados y añadir una descripción para clasificar el archivo.

- Se creará un buscador para uso interno (en las imágenes privadas) por parte de cada usuario o público para buscar en las imágenes públicas de todos los usuarios.

- Función colaborativa, es la posibilidad de que varios usuarios puedan editar la misma imagen para crear animaciones en grupos de trabajo.

1.3. Posibles obstáculos

1.3.1 Tiempo de desarrollo

Debido a la cantidad de características que tiene el proyecto existe la posibilidad de no poder realizar todas en el tiempo que dura el proyecto.

Se ha establecido un orden de prioridades según la utilidad de la funcionalidad respecto a resolver el problema propuesto. Si existe algún desvío importante respecto a la planificación durante el desarrollo de la aplicación, se negociará con el director del proyecto la importancia de cada funcionalidad.

1.3.2 Incompatibilidad entre navegadores

Existe la posibilidad de que los usuarios utilicen un navegador que no soporte ciertas funcionalidades necesarias para el proyecto.

Se avisará al usuario con que navegadores se ha testeado la aplicación y esta funciona correctamente. Así, podrá utilizar uno de esos navegadores si la aplicación no funciona correctamente en el suyo.

1.3.3 Aglomeración de usuarios

Al tratarse de una herramienta online, es posible que conforme avance el proyecto más usuarios utilicen la plataforma. Esto puede hacer que el servidor utilizado inicialmente no tenga capacidad suficiente para soportar a estos usuarios.

En este caso se parara la plataforma durante unas horas y se trasladara a un servidor con más capacidad para soportar futuros usuarios.

1.3.4 Bugs

Dado que la aplicación utilizará una herramienta externa (el navegador) existe la posibilidad de que una funcionalidad testeada en una versión de un navegador deje de funcionar debido a una actualización que haya hecho la empresa propietaria del navegador en su producto.

Se añadirá en la aplicación un formulario para enviar errores, que recogerá ciertos datos como el navegador utilizado junto a su versión.

1.4. Método de trabajo

Se seguirá una metodología basada en *SCRUM*. Esta metodología se caracteriza en realizar un desarrollo incremental de las funcionalidades.

Se ha dividido el proyecto en varios objetivos pequeños que conformarán entregas.

Con cada entrega se comprobará con el director del proyecto que el objetivo de esa

entrega se haya cumplido correctamente según lo esperado y se decidirá cuál será la siguiente entrega.

Conforme avance el proyecto se pueden realizar cambios en el proyecto entre entregas si se presenta alguna necesidad nueva o un cambio en el mercado.

1.5. Herramientas de seguimiento

Se utilizarán las herramientas *online google drive* y *google calendar* para el seguimiento del proyecto con el tutor.

También se utilizará la herramienta *github* para hacer el seguimiento del código. Este servicio ofrece un repositorio de código y ademas dispone de un servicio de tickets que se puede utilizar para comprobar que se cumplan los requisitos del proyecto.

1.6. Método de validación

Mediante el seguimiento de los tickets en github se irá comprobando la validez de las implementaciones.

Cada vez que se lleve a cabo la implementación de una funcionalidad importante se subirá también a un servidor al que el director del proyecto tendrá acceso.

Se contactará con el director del proyecto, este validará que la funcionalidad es correcta y se llegara a un acuerdo sobre la siguiente funcionalidad a implementar.

2. Planificación del proyecto

2.1. Duración

2.1.1 Duración estimada

El proyecto tiene una duración aproximada de 3 meses. El proyecto se empieza a desarrollar el 17 de septiembre y acabará el 30 de noviembre.

2.1.2. Consideraciones

El uso de las metodologías ágiles para este proyecto hace que la planificación pueda variar de un *sprint* a otro. Esta es una aproximación inicial de lo que se hará en cada *sprint* pero es posible que la distribución de tareas cambie con el paso del tiempo.

Otra consideración importante es que aunque en la planificación temporal no se especifica, en el tiempo estimado para cada característica se esta teniendo en cuenta el análisis, el diseño, la implementación y la comprobación de validez de cada una de ellas.

2.2. Planificación temporal

2.2.1 Planificación y viabilidad del proyecto

Esta fase pertenece al curso de Gestión de Proyectos y es la fase en la que se generará el documento para cumplir el hito inicial del proyecto.

Consta de las siguientes etapas:

Nombre	Fecha inicial	Fecha final
Alcance	08/09/14	12/09/14
Planificación temporal	12/09/14	16/09/14
Gestion Económica y sostenibilidad	16/09/14	21/09/14
Presentación preliminar	21/09/14	28/09/14
Contextualización y bibliografía	28/09/14	03/10/14
Documento final	03/10/14	12/10/14

Tabla 1: Etapa planificación y viabilidad del proyecto

2.2.2 Primera iteración

En esta primera fase se implementarán las caracteristicas del editor de *sprite sheets* necesarias para que el usuario pueda elegir cómo serán las animaciones de sus *sprites.*

Se ha dividido esta iteración en dos etapas:

Nombre	Fecha inicial	Fecha final
Implementación de marcaje de frames	17/09/14	27/09/14
Cargar y visualizar imagen		
Hacer zoom de la imagen cargada		
Permitir al usuario marcar un rectángulo que contenga un *frame* del *sheet*		
Acumular los *frames* marcados para crear un *sprite*		
Ajustar de forma manual la imagen dentro del *frame* para que queden todos centrados		
Permitir poner un *frame* sobre otro con transparencia para facilitar la tarea anterior		
Visualización de la animación	27/09/14	04/10/14
Visualizar el *sprite* generado a partir de los *frames* seleccionados (un *frame* al lado del otro)		
Visualizar la animación que el usuario ha seleccionado		
Darle al usuario la posibilidad de especificar los milisegundos entre *frame* y *frame* (para todos los *frames* o de uno en uno)		

Tabla 2: Etapa primera iteración

Hay que destacar que en el caso de que se produzca algún desvío en la planificación, no se podrá iniciar la etapa *"visualización de la animación"* sin haber completado la etapa *"implementación del marcaje de frames"*.

2.2.3 Segunda iteración

Se ha dividido esta iteración en dos bloques:

Nombre	Fecha inicial	Fecha final
Exportación	05/10/14	11/10/14
Exportar animaciones para librerías como *ClanLib Game SDK* en formato *XML*		
Exportar animaciones de un sprite específico en formato *GIF*		
Exportar un sprite sheet en formato PNG		
Automatización	11/10/14	23/10/14
Autodetectar que *sprites* forman parte de una animación		
Optimizar el espacio vacío entre dos *sprites*		
Cálculo automático del número de milisegundos entre *frames*		
Cálculo automático de los *frames* que hay entre dos *frames*		

Tabla 3: Etapa segunda iteración

Una vez acabada la fase de la exportación, la aplicación ya será utilizable por cualquier usuario que quiera un editor de *sprite sheets* y se habrá conseguido alcanzar la primera fase del alcance del proyecto. Por esta razón se ha decidido que en la planificación temporal la *exportación* se realice antes que la *automatización* aunque no exista ningún prerrequisito entre ellas.

2.2.4 Tercera iteración

Los bloques que se implementarán en esta iteración son los siguientes:

Nombre	Fecha inicial	Fecha final
Socialización plataforma	23/10/14	09/11/14
Crear cuenta de usuario para conservar ficheros		
Posibilidad de compartir ficheros con otras personas		
Poder guardar las creaciones como publicas		
Buscador de creaciones públicas mediante *tags*		
Posibilidad de editar los *sprite sheets* entre varias personas a la vez		
Posibilidad de crear *sprite sheets* en grupos de trabajo (funcionalidad colaborativa)		
Ampliación funcionalidades	09/11/14	11/11/14
Mejoras en el buscador		
Modificar partes del *sprite sheet* (como puede ser modificar el color de un *pixel*, o el color de varios *pixeles*)		

Tabla 4: Etapa tercera iteración

En esta iteración se implementarán todas las características que añaden un aspecto social a la plataforma ademas de permitir guardar ficheros en la nube. También se implementarán en este bloque las ampliaciones que añaden valor a la plataforma pero no son necesarias para su objetivo inicial.

2.2.5 Final de proyecto

Las etapas de las que consta esta iteración son:

Nombre	Fecha inicial	Fecha final
Testing y ajustes	11/11/14	20/11/14
Documentación	20/11/14	28/11/14
Preparación presentación	28/11/14	01/12/14
Presentación	01/12/14	05/12/14

Tabla 5: Etapa final del proyecto

La última fase se trata de la preparación para la presentación del TFG ante el tribunal. Se han puesto unas fechas orientativas y se ha dejado un margen (la etapa *Testing y ajustes*) para comprobar que toda la aplicación funcione correctamente, en caso de desvío de alguna de las iteraciones anteriores se utilizaría parte de esta etapa para compensar este desvío.

2.3. Tiempo estimado

Etapa	Horas
Planificación y viabilidad del proyecto	90
Primera iteración	78
Segunda iteración	84
Tercera iteración	108
Final de proyecto	108
Horas totales	468

Tabla 6: Tiempo de proyecto

2.4. Diagrama de Gantt

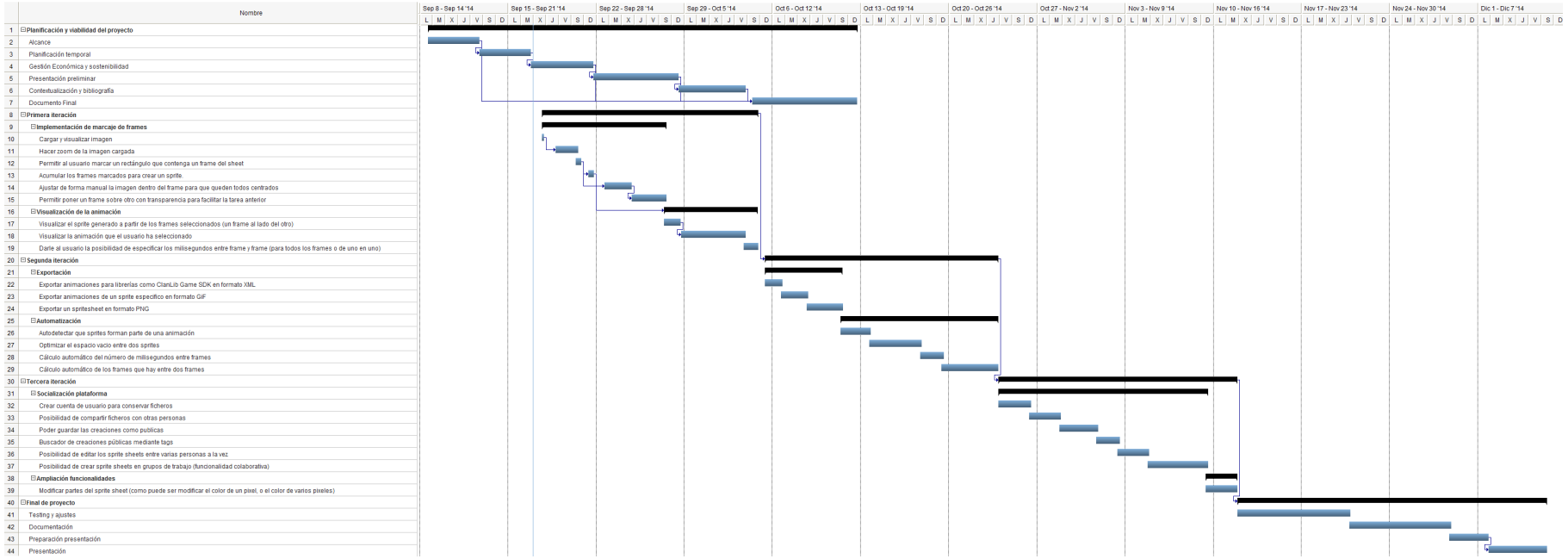

Ilustración 3: Diagrama de Gantt

2.5. Plan de acción

Como se ha explicado en el punto 1.2 se están utilizando metodologías ágiles para este proyecto. En caso de tardar menos tiempo del previsto en alguna iteración, se comenzarán a implementar características de la siguiente iteración en el orden establecido. En el caso contrario, se retrasará la implementación de iteraciones siguientes. Se ha dejado una etapa en la iteración final llamada *testing* y ajustes para compensar este tipo de desvíos.

Al final de cada iteración habrá una reunión con el director del proyecto para analizar el progreso del proyecto y confirmar que todo esta siendo implementado según lo esperado.

2.6. Recursos

Hardware:

- *MSI GE60*
- *Hosting* con las siguientes características:
  - Velocidad de la red: *10 mbps*
  - RAM: *16GB*
  - Procesador: *Intel Xeon E3-1230*
  - Sistema operativo: *CentOS 6.2*

Software:

- *Microsoft Windows 8.1 Professional*
- *OpenOffice 4*
- *XAMPP v.3.2.1*
- *Navegador Chorme Versión 37.0.2062.120 m*
- *Navegador Firefox v 31*
- *Internet Explorer v 11.0.9600.17107*

Humanos:

- 1 Persona con disponibilidad semanal de 36 horas.

3. Presupuesto y sostenibilidad

3.1. Introducción

En este apartado se realizará una aproximación del coste del proyecto. Se ha intentado trabajar con herramientas de *software* libre en la medida de lo posible. Dado que este proyecto no va a generar beneficios al tratarse de *software* libre no se desea generar gastos innecesarios.

3.2. Recursos Humanos

El proyecto estará desarrollado por una persona. Se ha estimado el coste dividiendo las tareas que debe realizar según su rol. En la tabla 7 aparecen estas tareas desglosadas según el diagrama de Gantt.

Rol	Tarea	Horas aproximadas	Coste por hora	Coste
Project Manager	Planificación y viabilidad	90 h	30 € / h	2.700,00 €
	Primera Iteración	8 h	30 € / h	240,00 €
	Segunda Iteración	8 h	30 € / h	240,00 €
	Tercera Iteración	8 h	30 € / h	240,00 €
	Final del proyecto	60 h	30 € / h	1.800,00 €
	Total	174 h	30 € / h	5.220,00 €
Software Developer	Planificación y viabilidad	0 h	15 € / h	0,00 €
	Primera Iteración	70 h	15 € / h	1.050,00 €
	Segunda Iteración	76 h	15 € / h	1.140,00 €
	Tercera Iteración	100 h	15 € / h	1.500,00 €
	Final del proyecto	48 h	15 € / h	720,00 €
	Total	294 h	15 € / h	4.410,00 €
Total		468 h		9.630,00 €

Tabla 7: Recursos Humanos

3.3. Hardware

El proyecto requiere del uso de *hardware*. Concretamente un ordenador para desarrollar tanto el programa como la documentación y un servidor para acceder a la aplicación online, la página web estará alojada en un servidor gratuito y por esta razón su coste será de 0 €. En la tabla 8 se incluyen los gastos que el *hardware* comporta.

Producto	Precio	Unidades	Vida útil	Total amortización
MSI GE60	885,00 €	1	3 Años	62,53 €
Hosting	0,00 €	1	3 Años	0,00 €
Total	885,00 €			62,53 €

Tabla 8: Hardware

3.4. Software

Para realizar el proyecto se necesitará *Software* específico, se ha optado por utilizar *software* libre en la medida de lo posible para poder ahorrar costes. En la tabla 9 se detalla el *software* utilizado.

Producto	Precio	Unidades	Vida útil	Total amortización aproximada
Windows 8.1 Pro.	60,00 €	1	3 Años	4,23 €
OpenOffice 4	0,00 €	1	N/A	0,00 €
XAMPP	0,00 €	1	N/A	0,00 €
Chrome	0,00 €	1	N/A	0,00 €
Firefox	0,00 €	1	N/A	0,00 €
Internet Explorer	0,00 €	1	N/A	0,00 €
Notepad++	0,00 €	1	N/A	0,00 €
Total	60,00 €			4,23 €

Tabla 9: Software

3.5. Gastos generales

En este apartado se incluyen los gastos más genéricos como el consumo de internet o de la luz al tener el ordenador encendido mientras se realiza el proyecto. No se incluyen gastos de local al no disponer de ninguno.

Tampoco se incluyen impuestos relacionados con la actividad económica ya que, al tratarse de un proyecto de *software* libre, no se obtendrán ganancias. No se incluye el consumo del servidor al estar incluido en la cuota del contrato con el servicio de *Hosting*.

En la tabla 10 se encontrará una explicación detallada de estos gastos.

Concepto	Coste por hora	Total amortización aproximada
Consumo Energetico Pc	0,022 €/h	10,29 €
Telefono (conexión a internet)	0,18 €/h	84,24 €
Total		94,53 €

Tabla 10: Gastos Generales

3.6. Riesgos

Se ha de tener en cuenta el riesgo de que el *hosting* contratado no sea suficiente para alojar la plataforma. Esto puede ocurrir si la página empieza a generar visitas que consuman todo el ancho de banda disponible gratuitamente. El mismo servicio de *hosting* dispone de ampliaciones.

Otro riesgo es el no poder acabar el proyecto en el tiempo estimado. En el diagrama de Gantt esta contemplado este riesgo y hay un cierto margen que se restaría a la etapa de *testing y ajustes* para finalizar el proyecto.

En la tabla 11 se explican los gastos por riesgos.

Riesgo	Precio	Vida útil	Probabilidad	Total
Hosting 1000 GB	6,99 €/mes	1 mes	10,00%	2,80 €
Hosting 2000 GB	13,99 €/mes	1 mes	5,00%	2,80 €
Hosting 3000 GB	20,99 €/mes	1 mes	1,00%	0,84 €
Total				6,44 €

Tabla 11: Riesgos

3.7. Coste del proyecto

En la tabla 12 se muestra un resumen de todos los apartados anteriores junto a la suma del coste del proyecto. Se ha decidido fijar un nivel de contingencia para el proyecto de un 15 %.

Riesgo	Total
Recursos humanos	9.630,00 €
Hardware	62,53 €
Software	4,23 €
Gastos Generales	94,53 €
Contingencia (15%)	1.468,69 €
Riesgo	6,44 €
Total	11.266,42 €

Tabla 12: Coste del proyecto

3.8. Sostenibilidad

3.8.1 Económica

Este proyecto es sostenible económicamente, se han evaluado los costes de recursos materiales y humanos necesarios para la realización del proyecto.

Al ser un proyecto *software* en el que la mayoría de herramientas que se utilizan son de *software* libre, el coste de los recursos materiales lo hace muy competitivo la única manera que existiría de ahorrar costes sería realizarlo en menos tiempo.

3.8.2 Social

Este proyecto es un proyecto enfocado hacia la comunidad de desarrolladores de videojuegos con poca capacidad económica. Al ser un proyecto *online* tendrá acceso todo el mundo, pero habría que realizar tareas de localización de idioma para que lo pudieran utilizar completamente sin saber español.

Gracias a esta plataforma el público objetivo tendrá más facilidades a la hora de tratar con los gráficos en sus videojuegos y ahorrarán tiempo que podrán utilizar en otras tareas de sus proyectos.

Se podría pensar que esta plataforma perjudicaría el sector de los artistas, sin embargo este planteamiento es erróneo, ya que esta plataforma puede ayudar a empujar la carrera de artistas pocos conocidos si aprovechan las capacidades sociales de la plataforma para compartir sus obras.

3.8.3 Ambiental

En este punto se puede decir poco sobre este proyecto. El único recurso necesario para el proyecto es la energía que consumen los ordenadores (tanto del ordenador utilizado para desarrollar, como la del servidor).

Al utilizarse un *hosting* compartido, la energía consumida será la misma tanto si existiera el proyecto como si no existiera.

Si se decidiera finalizar el servicio sería suficiente con desalojar la página web del dominio y no generaría un gasto ambiental extra.

3.8.4 Matriz de sostenibilidad

Se incluye en el documento la tabla 13 indicando las puntuaciones de la planificación.

¿Sostenible?	Económica	Social	Ambiental
Planificación	Viabiliad Económica	Mejora en la calidad de vida	Análisis de recursos
Valoración	9	9	10

Tabla 13: Matriz de sostenibilidad del TFG

4. Contexto y bibliografía

4.1. Contexto

El contexto de este trabajo final de grado es el de crear una herramienta para ayudar a desarrollar videojuegos en 2D más rápidamente.

Concretamente esta herramienta existirá para agilizar el traspaso de gráficos entre un artista y un programador.

Actualmente existen varias plataformas que pueden ayudar a los desarrolladores y a los artistas con el tratamiento gráfico, pero todas estas plataformas necesitan instalarse en los ordenadores y ademas ninguna de ellas permite compartir archivos entre ellos. En el apartado 4.3.2. se habla sobre varias de estas herramientas.

Existen dos roles diferenciados con intereses en esta plataforma. Por un lado está el artista de la *sprite sheet* y por el otro esta el desarrollador del videojuego. Estos roles tienen intereses y necesidades muy diferenciados al no trabajar de la misma forma durante el desarrollo de un videojuego.

A día de hoy los artistas utilizan herramientas especializadas para trabajar con los gráficos de manera cómoda para ellos y ademas con calidad. Cuando necesitan exportar estos gráficos a un juego adaptan las imágenes que tienen para encajarlas con el proyecto del videojuego que tienen que utilizar, modificando por ejemplo la resolución de la imagen o la cantidad de colores.

Una vez acabado el gráfico los artistas los publican en Internet o los envían a un desarrollador determinado si es para alguien especifico. Para ello utilizan herramientas como pueden ser el email, *dropbox*, *mega*,... Hay muchas herramientas para compartir archivos y no se pretenden documentar, ni analizar en este documento.

Una vez los desarrolladores reciben los gráficos que han realizado los artistas, lo que quieren es poder introducirlos en el código. Normalmente han hablado ya con el artista para que pongan las imágenes de una forma y tamaño determinado, si no es así realizan ellos mismos estos cambios mediante algún editor de imágenes o alguna de las herramientas analizadas en este documento.

4.2. Actores implicados

A continuación se va a detallar los *stakeholders* interesados en el proyecto.

4.2.1. Desarrollador de la plataforma

El desarrollador de la plataforma tiene como objetivo finalizar el proyecto en el plazo estimado y que la plataforma cumpla las expectativas de los diferentes actores.

4.2.2. Tutor del proyecto

Su rol es supervisar que el proyecto cumpla con los objetivos propuestos y que siga una la planificación. Además guiará y ayudará al desarrollador a llevar el proyecto adelante.

4.2.3. Programadores de videojuegos o demos

Los programadores son personas que necesitan que la herramienta aporte rapidez en el momento de utilizar los sprites dentro de su aplicación. Necesitarán que sus programas puedan importar toda la *meta data* de la manera más automatizada posible. También necesitarán que la herramienta tenga un buscador que les permita encontrar lo que necesitan cómoda y rápidamente.

4.2.4. Artistas

Necesitaran que sea una herramienta con facilidad de uso. Que les permita dar retoques finales para añadir *meta data* si la imagen no la tiene. Necesitarán poder compartir sus obras con otros autores o con programadores y obtener visibilidad.

4.3. Estado del arte

4.3.1. Definiciones

En esta sección se van a definir algunos conceptos de importancia para el proyecto.

4.3.1.1. Sprite

Originalmente un sprite se refería a un tipo de mapa de bits dibujado en la pantalla de ordenador por *hardware* gráfico especializado sin cálculos adicionales de la CPU.

Con el paso del tiempo este término se extendió a cualquier mapa de bits que se dibuja en pantalla.[1]

4.3.1.2. Sprite sheet

Una *sprite sheet* (o *strip*) es una imagen no animada de gran tamaño en la que aparecen muchas imágenes de un personaje (o varios) mostrando todos los frames de sus animaciones.[2]

La idea principal de utilizar *sprite sheets* es que al cargar una imagen y mostrar solo parte de esas imágenes a modo de ventana se gastan menos recursos que cargando las imágenes una a una. Esta práctica está cayendo en desuso debido a que cada vez los ordenadores son más potentes. Pero últimamente se ha vuelto a utilizar en *css* para ahorrar el gasto de ancho de banda al hacer peticiones a un servidor.[3]

4.3.1.3. Meta data

En este proyecto, el programador del videojuego estará sobretodo interesado en conseguir la meta data de la imagen para poder trabajar de la manera más cómoda posible.

En el contexto de este proyecto con meta data nos referimos a la información de un *sprite sheet* indicando la posición de un *frame* de un *sprite*, cual es el orden de las animaciones y cual es el tamaño de cada *frame*.

El problema principal de esta información es que los artistas no la suelen incluir o la incluyen de una manera visual (y los programadores tienen dificultades para trabajar con eso). Existen estudios que proponen una alternativa, es incluir la *meta data* dentro de la imagen modificando el valor de alguno de los *pixeles*[4]. Aunque es una solución buena, no es una solución extendida.

4.3.1.4. Librerías de desarrollo

Los programadores utilizan librerías de desarrollo para realizar el código de los programas. Algunas librerías enfocadas al tratamiento de los videojuegos como *clanlib*[5] permiten introducir toda la *meta data* de la *sprite sheet* en un archivo *xml* ahorrando mucho tiempo al programador.

Otra manera de crear videojuegos es *online*, en *html5* se puede utilizar *json* o *xml* para indicar la *meta data* de los *sprite sheets*[6]

En algunas librerías como en *sfml*[7] o *sdl*[8] el procedimiento es hacer que los *frames* de la *sprite sheet* tengan el mismo tamaño. Mediante código se le dice al programa que muestre mediante una ventana un *frame*, en cada iteración se va moviendo el *frame* que muestra consiguiendo así la animación.

4.3.2. Editores de sprite sheet existentes

Existen multitud de editores de *sprite sheets* para ayudar en la tarea de la edición de *sprites.* Tal y como observó *Xianze Lin* en su estudio[4] todos editores parecen compartir varias funcionalidades como la de autodetectar con un click encima de un frame de que tamaño es ese frame especifico. Esto es muy útil en frente a los editores de imagenes convencionales como *Microsoft Paint* en los que tienes que arrastrar un recuadro.

Otra funcionalidad que explica que comparten es la *Onion Skinning,* esto se trata de poner todas las imagenes una encima de la otra para poder ajustar más fácilmente el *frame* actual respecto al anterior.

Por último otra funcionalidad que comparten es la posibilidad de extraer la *meta data* en formato *XML*, aunque al parecer cada herramienta lo hace con un protocolo diferente.

A continuación se hará un análisis de varios editores destacando sus características y sus deficiencias.

Plataformas analizadas
Sprite Vortex
Shoebox
Texturepacker
DarkFunction Editor

Tabla 14: Plataformas analizadas

***Sprite Vortex*[10]**

Es una herramienta que permite importar una sprite sheet o crear una *sprite sheet* a partir de varias imagenes y exportar la *meta data* en un archivo xml o en una animación *gif*.

Se pueden seleccionar los *frames* de la animación e indicarle la posición dentro de la animación.

Aunque permite borrar todos los *pixeles* de un color (ponerlos como transparencia) esta herramienta no permite más modificaciones sobre un *sprite sheet* una vez creado.

***Shoebox*[11]**

Esta herramienta tiene la opción de coger los *frames* de una animación *gif* y crear una *sprite sheet* con ellos. También permite crear *sprites* a partir de pantallazos fotografias. Ademas dispone de la opción de juntar varias imagenes en un *sprite sheet*.

Esta herramienta sin embargo carece de opciones interesantes como la *onion skinning* o ver la previsualización de animaciones.

***Texturepacker*[12]**

Otro editor que se ha utilizado es *texturepacker*[12], se trata de un editor que permite crear *sprite sheets* a partir de imágenes con la cualidad de poder exportar la *meta data* a múltiples *frameworks* conocidos en el mundo de la programación de videojuegos.

Otra funcionalidad a destacar es la de crear alias en la *meta data* cuando encuentra dos o más imágenes iguales en varias animaciones poder utilizar solo un *frame* repetido.

El problema de este editor es que solo sirve para juntar varios *sprites* en una *sprite sheet* y optimizar el espacio. No permite algunas funcionalidades del proyecto como añadir más *sprites* a un *sprite sheet* o quitar *sprites* de una *sprite sheet*.

***DarkFunction Editor*[13]**

Este editor dispone de la posibilidad de hacer *onion skinning*, previsualizar la animación. En este editor también se puede auto seleccionar el *frame* de un *sprite sheet* haciendo doble *click* y elegir que parte del *sprite sheet* se va a utilizar para descartar el resto.

Por último este editor también permite exportar las animaciones a *xml* o crear *gifs* con ellas.

A diferencia del proyecto, este editor no permite añadir *sprites* sueltos en una *sprite sheet* ya creada, aunque permite crear una nueva a partir de varios *sprites* sueltos.

4.3.3. Conclusiones

Tal y como se ha comprobado existen varias herramientas en el mercado de *software* libre que cubren muchos de los aspectos que se desea realizar en el proyecto. Ademas estas herramientas disponen de funciones interesantes que en un principio no estaban planteadas, pero que tras analizarlas se ha decidido implementar en el proyecto por ser útiles.

Se podría llegar a realizar todas las tareas que con este proyecto se pretenden implementar utilizando varias herramientas ya existentes. Sin embargo, no se podrían realizar estas tareas unicamente con una de ellas, si no que se necesitaría utilizar todas ellas para obtener todas las funcionalidades de este proyecto.

Ademas ninguna de estas herramientas es *online*, con lo cual se dificulta el acceso a ellas. Y en algunos casos ni si quiera son multiplataforma.

Tampoco existe la opción de compartir las creaciones en ninguna de las herramientas, como si existirá en la plataforma que se pretende desarrollar. Teniendo que usar algún servicio de archivos *online,* enviarlo a través de correo o mediante algún medio de almacenaje físico.

Por lo tanto, esta herramienta, aunque no es necesaria para el desarrollo de videojuegos, pues ya existen formas de realizar estos tratamientos, si que ayudara en muchos aspectos al aunar todo en una sola plataforma.

5. Bibliografia

[1] Wikipedia, definición de sprite. URL http://es.wikipedia.org/wiki/Sprite_(videojuegos). Accedido en 2 de octubre 2014

[2] Comunidad GameMaker, glosario. URL http://www.comunidadgm.org/manual_GM/Glosario.htm . Accedido en 2 de octubre 2014.

[3] Marc Climent Articulo blog Ahorrando ancho de banda con sprites y CSS. URL http://blog.climens.net/2008/03/27/ahorrando-ancho-de-banda-con-sprites-y-css . Accedido en 2 de octubre de 2014.

[4] Lin, Xinze. *Web-based Sprite Sheet Animator for Sharing Programmatically Usable Animation Metadata.* Uppsala University, Disciplinary Domain of Science and Technology, Mathematics and Computer Science, Department of Information Technology 2013. Suecia 2013. URL http://www.diva-portal.org/smash/record.jsf?pid=diva2:612988 . Accedido en 2 de octubre de 2014.

[5] Biblioteca de desarrollo clanlib. URL http://www.clanlib.org . Accedido en 2 de octubre de 2014.

[6] Colt McAnlis,Peter Lubbers,Brandon Jones,Andrzej Mazur,Sean Bennett,Bruno Garcia,Shun Lin,Ivan Popelyshev,Jon Howard,Ian Ballantyne,Takuo Kihira,Jesse Freeman,Tyler Smith,Don Olmstead,Jason Gauci,John McCutchan,Chad Austin,Mario Andres Pagella,Florian d'Erfurth,Duncan Tebbs. *HTML5 Game Development Insights* paginas 99-102. Apress 2014. ISBN: 978-1-4302-6697-6.

[7] Biblioteca de desarrollo sfml. URL http://www.sfml-dev.org . Accedido en 2 de octubre de 2014.

[8] Biblioteca de desarrollo SDL. URL https://www.libsdl.org . Accedido en 2 de octubre de 2014.

[9] Artur Moreira, Jan Haller, Henrik Vogelius Hansson. *SFML Game Development.* Packt Publishing 2013. ISBN: 978-1-84969-684-5

[10] Sprite vortex, creador de animaciónes basadas en sprite. URL http://spritevortex.codeplex.com . Accedido en 2 de octubre de 2014.

[11] Shoebox, aplicación relacionada con juegos y ui. URL http://renderhjs.net/shoebox . Accedido en 2 de octubre de 2014.

[12] Texturepacker, empaquetador de sprites. URL https://www.codeandweb.com/texturepacker . Accedido en 2 de octubre de 2014.

[13] Dark function, Empaquetador y animador de sprites. URL http://darkfunction.com/editor . Accedido en 2 de octubre de 2014.

6. Apéndice

6.1. Índice de tablas

6.2. Índice de ilustraciones