Portada:

a) Títol **Desarrollo de un editor de sprite sheets en HTML5**

b) Autor Ignacio Soto Alsina

c) Data de defensa ???

d) Director i Departament del Director ALEJANDRO RÍOS JEREZ [Departamento de Ciencias de la Computación](http://www.cs.upc.edu/)

e) Titulació Grado Ingenieria Informatica

f) Especialitat Computación

g) Centre: FACULTAT D’INFORMÀTICA DE BARCELONA (FIB)

h) Universitat: UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

Contenido

[1 Introducción 3](#_Toc409395522)

[1.1 Formulación del problema 3](#_Toc409395523)

[1.2 Análisis del contexto y el estado del arte 4](#_Toc409395524)

[1.2.1 Contexto 4](#_Toc409395525)

[1.2.2 Actores implicados 5](#_Toc409395526)

[1.2.3 Estado del arte 5](#_Toc409395527)

[1.2.4 Editores de Sprite Sheet existentes 6](#_Toc409395528)

[1.2.5 Conclusiones 7](#_Toc409395529)

[2 Alcance del proyecto 7](#_Toc409395530)

[2.1.1 Objetivos del proyecto 7](#_Toc409395531)

[2.2 Posibles obstáculos y soluciones 9](#_Toc409395532)

[2.2.1 Tiempo de desarrollo 9](#_Toc409395533)

[2.2.2 Incompatibilidad entre navegadores 9](#_Toc409395534)

[2.2.3 Aglomeración de usuarios 9](#_Toc409395535)

[2.2.4 Bugs 9](#_Toc409395536)

[2.3 Método de trabajo 9](#_Toc409395537)

[2.4 Herramientas de seguimiento 10](#_Toc409395538)

[2.5 Método de validación 10](#_Toc409395539)

[2.6 Cambios en el proyecto respecto al alcance inicial 10](#_Toc409395540)

[2.6.1 Funcionalidades no implementadas 10](#_Toc409395541)

[2.6.2 Funcionalidades añadidas 10](#_Toc409395542)

[3 Gestión del proyecto 11](#_Toc409395543)

[3.1 Planificación del proyecto 11](#_Toc409395544)

[3.1.1 Duración 11](#_Toc409395545)

[3.1.2 Planificación temporal 11](#_Toc409395546)

[3.2 Plan de acción 15](#_Toc409395547)

[3.3 Presupuesto y sostenibilidad 15](#_Toc409395548)

[3.3.1 Introducción 15](#_Toc409395549)

[3.3.2 Recursos 15](#_Toc409395550)

[3.3.3 Sostenibilidad 18](#_Toc409395551)

[3.4 Identificación de leyes y regulaciones 19](#_Toc409395552)

Resumen

Una *Sprite Sheet* es un recurso utilizado para optimizar la carga de imágenes, normalmente es un recurso utilizado en animaciones o en videojuegos en 2D. Estas *Sprite Sheets* pueden estar compuestas por uno o más *Sprites* que son imágenes que pueden representar personajes u objetos. Además, si estos Sprites están animados, estarán compuestos por varios *Frames* que son las imágenes parciales que formaran parte de una animación.

Para realizar estas animaciones correctamente se precisa de una información que indica en qué posición esta cada *Frame*, y cuando ha de aparecer. Esto se puede programar manualmente o utilizar un archivo de *Meta Data* guardando esta información.

En este proyecto se realiza una aplicación para navegador que permite al usuario subir imágenes que contengan *Frames* para poder unirlos en *Sprites* y después poder exportar toda esta información en forma de *Sprite* *Sheet* con, o sin *Meta* *Data*.

# Introducción

## Formulación del problema

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 además ninguna de ellas permite compartir archivos entre ellos. En el apartado 1.2.4 se habla sobre varias de estas herramientas.

Existen dos roles diferenciados con intereses en esta plataforma. Por un lado está el artista creador de la *Sprite* *Sheet* y por el otro está 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 además 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 específico. Para ello utilizan herramientas como pueden ser email, dropbox, mega,... Hay muchas herramientas disponibles para compartir archivos, aunque no es objetivo de este documento, ni analizar las existentes.

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 el apartado 1.2.4 o la herramienta desarrollada en este proyecto.

## Análisis del contexto y el estado del arte

### Contexto

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 las llamadas “*Sprite Sheet*”. Las *Sprite Sheets* consisten en archivos que contienen uno o varios *Sprites*.

Un S*prite* 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...

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 una *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.*

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.

### Actores implicados

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

#### 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.

#### 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.

#### 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.

#### 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.

### Estado del arte

#### 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.

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. Aunque es una solución buena, no es una solución extendida.

#### Librerías de desarrollo de videojuegos

Los programadores utilizan librerías de desarrollo para realizar el código de los programas. Algunas librerías enfocadas al tratamiento de los videojuegos como clanlib permiten introducir toda la meta data de la *Sprite Sheet* en un archivo en formato *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*.

En algunas librerías como en *SFML* o *SDL* 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.

### Editores de Sprite Sheet existentes

Existen multitud de editores de *Sprite Sheets* para ayudar en la tarea de la edición de *Sprite*. Tal y como observó Xianze Lin en su estudio todos editores parecen compartir varias funcionalidades como la de auto detectar con un clic encima de un *Frame* de qué tamaño es ese *Frame* especifico. Esto es muy útil en frente a los editores de imágenes 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 imágenes 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. Las plataformas analizadas se indican en la Tabla 1

|  |
| --- |
| Sprite Vortex [10] |
| *Shoebox* [11] |
| Texturepacker [12] |
| DarkFunction Editor [13] |

Tabla 1. Editores de Sprite Sheets analizados

#### Sprite Vortex

Es una herramienta que permite importar una *Sprite Sheet* o crear una *Sprite* *Sheet* a partir de varias imágenes 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

Esta herramienta tiene la opción de coger los *Frames* de una animación *Gif* y crear una

*Sprite Sheet* con ellos. También permite crear *Sprite* a partir de pantallazos fotografías.

Además dispone de la opción de juntar varias imágenes en un *Sprite Sheet*.

Esta herramienta sin embargo carece de opciones interesantes como la *Onion Skinning* o ver la vista previa de las animaciones.

#### Texturepacker

Otro editor que se ha utilizado es Texturepacker, 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 *Sprite* 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

Este editor dispone de la posibilidad de hacer *Onion Skinning*, visualización previa la animación. En este editor también se puede auto seleccionar el *Frame* de un *Sprite Sheet* haciendo doble clic 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 *Gif* 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.

### 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. Además 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 únicamente con una de ellas, si no que se necesitaría utilizar todas ellas para obtener todas las funcionalidades de este proyecto.

Además 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, sí que ayudara en muchos aspectos al aunar todo en una sola plataforma.

# Alcance del proyecto

## Introducción

Este proyecto consiste en el desarrollo de una aplicación online de código libre que permita al usuario generar *Sprite Sheets* de manera rápida y cómoda. Además permitirá ajustar parámetros como la velocidad entre *Frames* o el desplazamiento entre ellos si no son de la misma medida. Además el usuario podrá ver como quedara la animación de los Sprites que desee mediante una vista previa.

## Objetivos del proyecto

Al realizar la identificación de funcionalidades de este proyecto se dividieron en tres grupos o fases diferenciados. El primer grupo consiste en las funcionalidades básicas que el programa debería tener para que un usuario pueda realizar el objetivo planteado de manera manual. En el segundo grupo, las funcionalidades se centran en facilitar el trabajo al usuario intentando automatizar el mayor número de tareas posibles. El tercer grupo se trataba de una serie de mejoras opcionales y la posibilidad de volver más social la plataforma pudiendo compartir archivos o guardar *Sprites* de manera online, esta fase sin embargo no estaba planteada como obligatoria para dar por finalizado el proyecto.

### Objetivos del Primer grupo de funcionalidades

Este grupo consiste en las funcionalidades básicas que un usuario de la plataforma podría necesitar para introducir una imagen, crear *Sprites* y exportar tanto la Meta Data como un *Sprite Sheet* reducido para poder utilizarlo posteriormente.

Las funcionalidades de este grupo son los siguientes:

* El usuario podrá subir imágenes y recortarlas fácilmente según los *Sprites* que vaya a utilizar para las animaciones en su videojuego.
* La plataforma tendrá la opción de juntar imágenes de diferentes archivos en una sola *Sprite Sheet*.
* 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.
* Permitir modificar el color de los pixeles del *Sprite Sheet* como si una herramienta de edición de imágenes se tratara.

### Objetivos del segunda grupo de funcionalidades

En el segundo grupo, las funcionalidades se centran en la automatización de tareas, como puede ser la detección de Frames de una imagen o la generación de código para algunas librerías de programación.

Las funcionalidades del segundo grupo son:

* 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 vista previa para librerías de programación.
* Se creará una nueva opción que permitirá comprimir el espacio vacío en la imagen.
* La aplicación tendrá una opción para poder ajustar automáticamente el tiempo que hay entre imagen e imagen de una animación.

### Objetivos del tercer grupo de funcionalidades

El tercer grupo de funcionalidades se trata de mejoras opcionales que no afectan a la finalización del proyecto y solo se planteó realizar como ampliación si las funcionalidades anteriores estaban finalizadas antes de la fecha de entrega. Estas se centran sobre todo en darle un enfoque más social a la plataforma pudiendo guardar imágenes en un servidor y poder compartirlas o incluso trabajar colaborativamente con otros usuarios.

Estas funcionalidades son las siguientes:

* Permitir al usuario crear una cuenta para conservar los archivos que vaya modificando dentro de la plataforma.
* La plataforma dispondrá de la posibilidad de compartir los ficheros con otros usuarios.
* Se podrá poner los archivos como públicos o privados y añadir una descripción para clasificar el archivo.
* Se podrán buscar las creaciones creadas por los usuarios siempre que sean públicas, y en el caso de ser privadas solo aparecerán para el propietario.
* El programa dispondrá de función colaborativa, se trata de la posibilidad de que varios usuarios puedan editar la misma imagen para crear animaciones en grupos de trabajo.

## Posibles obstáculos y soluciones

### 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.

### 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.

### 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.

### 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.

## 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.

## 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 G*ithub* para hacer el seguimiento del código. Este servicio ofrece un repositorio de código y además dispone de un servicio de tickets que se puede utilizar para comprobar que se cumplan los requisitos del proyecto.

## Método de validación

Mediante el seguimiento de los tickets en G*ithub* 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.

## Cambios en el proyecto respecto al alcance inicial

Como ya se ha explicado en el apartado 2.5 durante las iteraciones se han ido estableciendo cambios en las necesidades del proyecto.

### Funcionalidades no implementadas

A continuación se explican las funcionalidades no implementadas debido a que al ir avanzando el proyecto había un desvío importante según la planificación inicial como para implementarlas.

* Una de las funcionalidades que se han dejado como futura implementación es la de editar los pixeles como si un editor de imágenes se tratara. Aunque sería una funcionalidad interesante, cuando se utiliza el editor, las imágenes ya están acabadas de editar además de la existencia de múltiples herramientas para editar imágenes que serían mejor que utilizar la modificación pixel a pixel que se propuso en la planificación inicial.
* Tampoco se implementara en el contexto de este TFG las funcionalidades de la socialización de la plataforma. Esto además permitirá que la herramienta se pueda ejecutar localmente sin internet, ya que no requerirá un servidor al solo necesitar un intérprete de JavaScript, HTML y CSS.

### Funcionalidades añadidas

También se han añadido algunas funcionalidades que se han visto interesantes mientras se realizaba el proyecto y no se habían tenido en cuenta al inicio.

* Doble clic para detectar un *Frame*. Al seleccionar Frames a mano en *Sprite Sheets* muy pobladas se hacía costoso para el usuario seleccionar correctamente los márgenes del *Frame*. Como se iba a implementar igualmente una manera de detectar todos los *Frames* de la imagen se podía provechar para añadir esta funcionalidad fácilmente.
* Exportar a GIF. Esto añade una funcionalidad al proyecto en la que se pueden realizar animaciones para páginas webs que no tengan que ver con la programación de videojuegos y en ese sentido esta fuera del contexto de este proyecto. Sin embargo tiene su utilidad para hacerse una idea de cómo quedara la animación y poderla compartirla con otras personas sin tener que escribir un programa de demo para ello.

# Dependencias externas

## Introducción

En este apartado se explican las dependencias a otros proyectos, en ellos se explica el funcionamiento de las dependencias y la razón que justifican la utilización en el proyecto.

## Navegadores

### Explicación

Para reproducir html5 y JavaScript se utilizan los navegadores. Existen muchos navegadores y la aplicación debería ser compatible entre todos ellos. Sin embargo las diferentes implementaciones que hacen las compañías entre navegadores hacen que a veces el código no se interprete igual en todos ellos.

### Uso y justificación

Durante el desarrollo del proyecto se ha utilizado sobretodo Google Chrome. Se han ido probando las funcionalidades también en Internet Explorer y en Firefox. En todos ellos la aplicación funciona, sin embargo en Internet Explorer la imagen que se importaba al Canvas perdía calidad. Aunque se ha intentado en la medida de lo posible que la experiencia del usuario sea la misma para todos los navegadores, es posible que algunos elementos se muestren diferente de algún navegador a otro. Durante el proyecto siempre se le ha dado prioridad a Google Chrome por una simple preferencia de comodidad en el momento de debugear la aplicación mediante la consola de desarrollo de Google Chrome.

## Jquery

### Explicación

Jquery es una librería que ayuda a que las aplicaciones sean compatibles entre varios navegadores. Además hace el uso de funcionalidades Ajax de manera más simple. Es una librería bastante extendida y es poco probable que introduzca errores en la aplicación. Además tiene bastante documentación y una comunidad grande detrás que hace poco probable que dejen de dar soporte en poco tiempo.

### Uso y justificación

En el proyecto solo se ha utilizado esta librería ya que hace que la compatibilidad entre navegadores más fácil de programar. Se utiliza para seleccionar elementos visuales como botones o los Canvas y para la lógica de ocultar y mostrar elementos según el modo en el que se encuentre la aplicación.

## SweetAlert

### Explicación

Esta librería se utiliza para modificar los mensajes de alerta que crea el navegador por defecto por unos con un aspecto visual más atractivo.

### Uso y justificación

Aunque es una herramienta no muy conocida su autor la actualiza bastante a menudo. Además se trata de una librería de código abierto alojada en Github con lo cual si hubiera algún bug se podría solucionar aunque el autor dejara de dar soporte. En el proyecto se utiliza para mostrar los diálogos popups de alerta o de confirmación cuando algo se ha realizado correctamente. Se ha utilizado por ahorrar tiempo de desarrollo para crear estos diálogos.

## JsGif

### Explicación

Se trata de una librería que transforma animaciones hechas en HTML5 a GIF, está basada en otra librería llamada AS3GIF y es de código abierto.

### Uso

Se ha utilizado para la opción de exportar los GIF’s. En un primer momento se planteó la opción de exportar las animaciones a diversos formatos, entre ellos GIF aunque no era necesario para el proyecto. Al intentar exportarla como Canvas no existía la posibilidad de exportarlo como animación así que se planteó la posibilidad de quitar esta funcionalidad. Sin embargo gracias a esta librería se ha podido realizar esta tarea de manera fácil sin que ocupara mucho tiempo y así se ha podido conservar esa funcionalidad para el programa final.

## BootStrap

### Explicación

Bootstrap es un Framework que se utiliza para añadir compatibilidad entre navegadores de escritorio, de tabletas y de móviles independientemente de su medida. Además añade una serie de estilos predefinidos a los elementos HTML mediante CSS.

### Uso

Aunque el proyecto no es compatible con móviles o tabletas se ha utilizado este Framework para mejorar el aspecto visual de la aplicación. Con este Framework nos aseguramos que en los navegadores de escritorio que tengan diferentes resoluciones de pantalla la aplicación se redimensionara automáticamente.

## Clanlib SDK

### Explicación

Clanlib SDK es un Framework de desarrollo de videojuegos para C++. Tiene compatibilidad para varios tipos de sistemas operativos y es de código abierto. Se encarga de aspecto como las visualizaciones en pantalla o el input del usuario. También dispone de módulos para el uso de programas en red, para el manejo de timers y la gestión de sonido.

### Uso

No se utiliza en el proyecto para nada, sin embargo el proyecto exporta información en XML preparada para la lectura en esta librería. De esta manera al exportar desde la aplicación la animación solo se necesitan unos pocos comandos para tenerla funcionando en un programa hecho con este Framework. Por lo tanto si la comunidad que mantiene esta librería decidiera cambiar el formato de estas animaciones y no lo hiciera compatible con versiones anteriores este programa perdería esta funcionalidad.

# Interfaz de usuario

## Introducción

En esta sección se explicara la interfaz del usuario y todas las funcionalidades de las que la aplicación dispone, el programa tiene varios modos de trabajo pero hay una zona común a todos ellos.

Se puede dividir el programa en tres partes: El *modo General*, el *modo Sprite* y el *modo Frame.* Aunque en ningún momento se cambie de ventana al pasar entre modos, sí que cambian las funcionalidades que le damos al área principal de trabajo y además se ocultan o muestran botones diferentes según las necesidades de cada modo. En los apartados siguientes se explicara en detalle cada modo.

## Zonas comunes

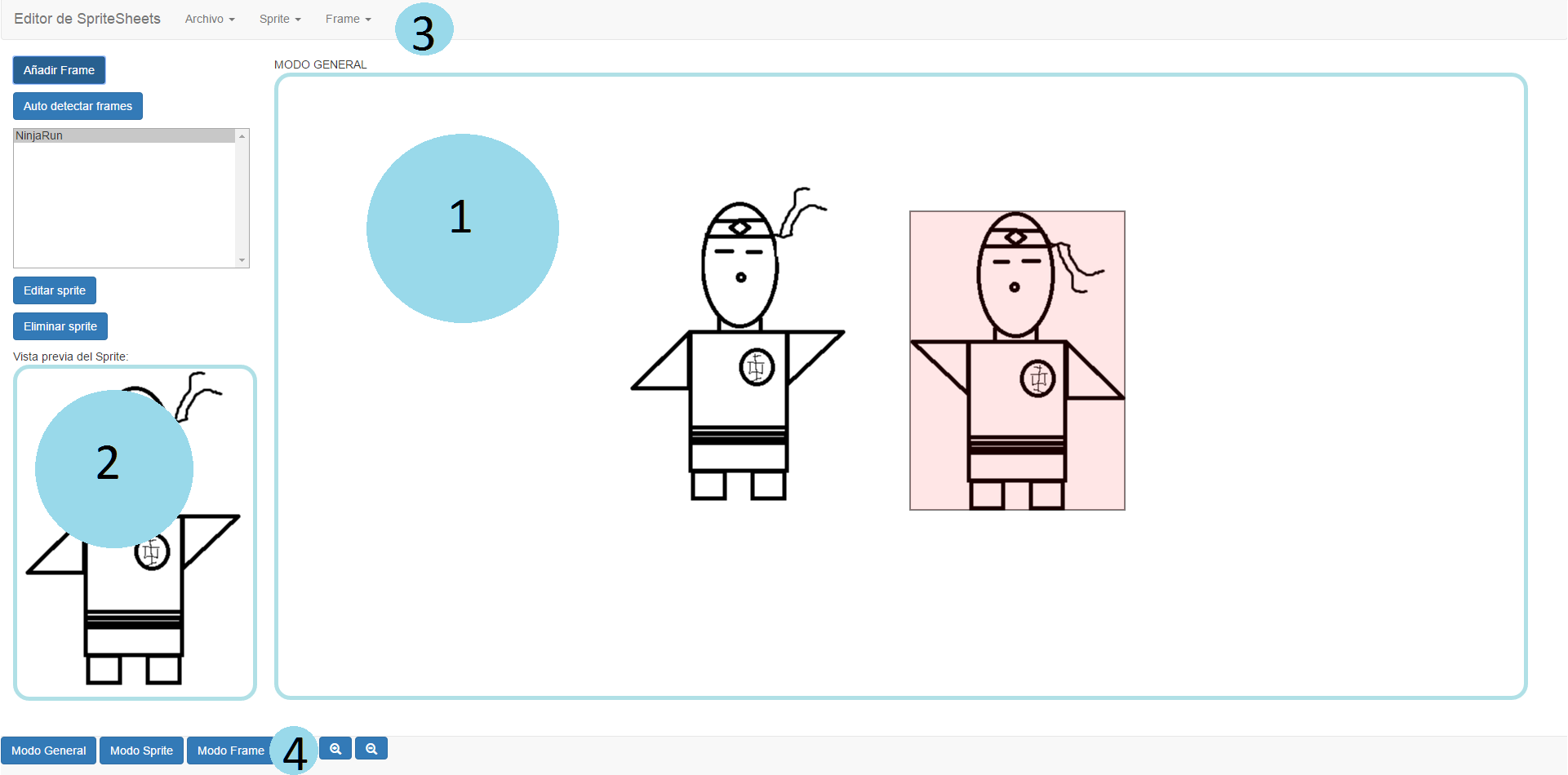


Ilustración 1. Área común de la aplicación

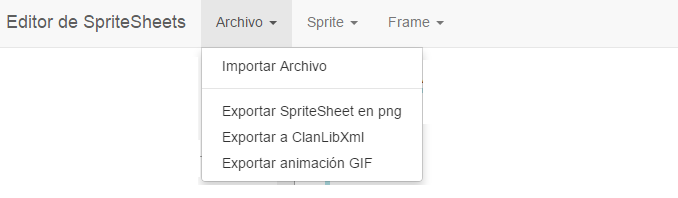
1. Área de trabajo principal: Es donde el usuario puede editar o seleccionar los *Sprites* o *Frames.* Aunque este área es común para todos los modos ira cambiando su funcionamiento según estos.
2. Vista previa del *Sprite*: Se utiliza para mostrar la vista previa de las animaciones, este mostrara siempre el Sprite que tengamos seleccionado o en el que estemos trabajando en ese momento.
3. Barra superior: En la barra superior de la aplicación podemos encontrar varias opciones. De izquierda a derecha nos encontramos con el menú archivo, el menú Sprite y el menú Frame.
   1. Menú Archivo: En este menú podemos encontrar las opciones siguientes:

Ilustración 2. Menú Archivo

* + 1. Importar Archivo: Esta opción sirve para llevar al area de trabajo principal una imagen externa al programa, una vez en el area de trabajo principal podremos trabajar con ella por ejemplo seleccionando *Sprites.* Podemos trabajar con imágenes diferentes conservando los *Sprites* ya creados.
    2. Exportar SpriteSheet en png: Esta opción nos permite extraer en una imagen PNG con todos los frames de todas las animaciones que hayamos seleccionado en el programa. El programa intentara comprimir el espacio que ocupa esta imagen de la mejor manera que pueda calcular.
    3. Exportar a ClanLibXml: Esta opción extraera una imagen PNG (igual que en la opción exportar Sprite Sheet) y un archvio *XML* llamado resources.xml que contiene información sobre la posición de todos los Frames y Sprites para su uso en programación de videojuegos mediante la libreria ClanLibXml
    4. Exportar animación GIF: Mediante esta opción obtendremos una animación en formato GIF del *Sprite* que tengamos seleccionado en ese momento en el menu de *Sprites* si estamos en el modo General, o del *Sprite* con el que estamos trabajando si estamos en cualquier otro modo.
  1. Menú Sprite: Desplegando este menú obtendremos las siguientes funcionalidades:

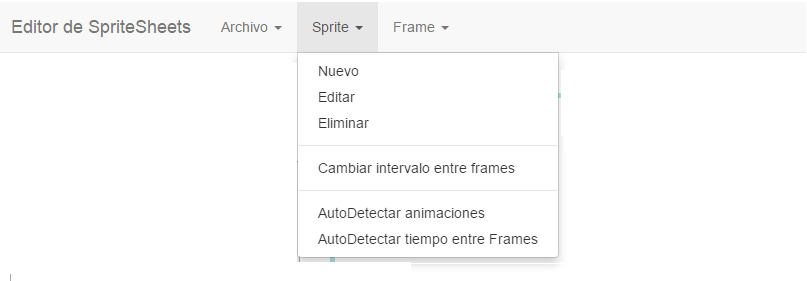


Ilustración 3. Menú Sprite

* + 1. Nuevo: Si seleccionamos esta opción nos aparecerá un dialogo para introducir el nombre a un nuevo *Sprite*. Todos los *Sprites* tienen que tener nombres diferentes. Mediante la introducción de diferentes *Sprites* podremos controlar que *Frames* pertenecen a cada animación.
    2. Editar: Si tenemos seleccionado un *Sprite*, mediante esta opción accederemos al modo de edición de *Sprite* para poder editar sus parámetros o modificar sus *Frames*
    3. Eliminar: Mediante esta opción eliminaremos el *Sprite* que hayamos seleccionado, también eliminaremos todos los *Frames* que se hayan introducido en ese *Sprite*
    4. Cambiar Intervalo entre Frames: Mediante esta opción cambiaremos el tiempo que pasa desde un Frame a otro. Si utilizamos esta opción se establecerá para todos los Frames el mismo tiempo.
    5. AutoDetectar animaciones: Mediante esta opción el programa intentara buscar dentro de la imagen que tengamos en el área principal todos los *Frames* que existan, después los introducirá en diferentes *Sprites* según su parecido. Al pulsar esta opción nos pedirá un valor entre 0-100 que sirve para indicarle al programa que porcentaje de diferencias tendrá un *Frame* para formar parte de un mismo *Sprite* respecto a otro. Hay que destacar que esta opción solo se puede utilizar si la imagen tiene transparencias.
    6. AutoDetectar tiempo entre Frames: Esta opción permitirá al programa intentar dar detectar cuantos milisegundos deben pasar desde un Frame a otro. Para ello calcula cuales son las diferencias respecto al siguiente *Frame*. Tiene dos opciones:
       1. *Más rápido si son iguales*: cuanto más diferente sea el Frame más velocidad le dará.
       2. *Más rápido si son diferentes*: esta opción hará que si tenemos Frames parecidos estos se muestren más rápidos.
  1. Menú Frame: Al seleccionar opciones de este menú deberemos tener un *Sprite* seleccionado y nos referiremos siempre a los *Frames* de ese *Sprite*

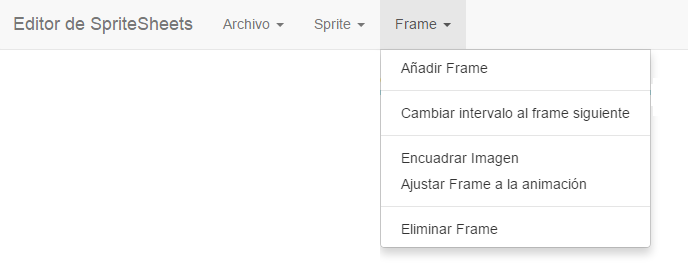


Ilustración 4. Menú Frame

* + 1. *Añadir Frame:* Esta opción creara un nuevo Frame con el recuadro que tengamos seleccionado en el área de trabajo si estamos en Modo General.
    2. *Cambiar intervalo al Frame siguiente:* Mediante esta opción cambiaremos el tiempo en ms en los que se verá el Frame seleccionado durante la animación.
    3. *Encuadrar Imagen:* En esta opción pasaremos al modo *Frame* con el que hayamos seleccionado, y nos permitirá mover dentro del encuadre del *Frame*.
    4. *Ajustar Frame a la animación:* Con esta opción pasaremos al modo *Frame.* Nos permitirá mover el *Frame* respecto a los demás Frames de la animación.
    5. *Eliminar Frame:* Mediante esta opción eliminaremos el *Frame* seleccionado.

1. Botones de cambio de modo: En la esquina inferior izquierda se encuentran varios botones para cambiar de un modo a otro, sin embargo para poder pasar al modo *Sprite* se tiene que haber seleccionado un *Sprite* anteriormente, y para pasar al modo Frame tiene que haberse seleccionado un *Frame* de un *Sprite*.

## Modo General

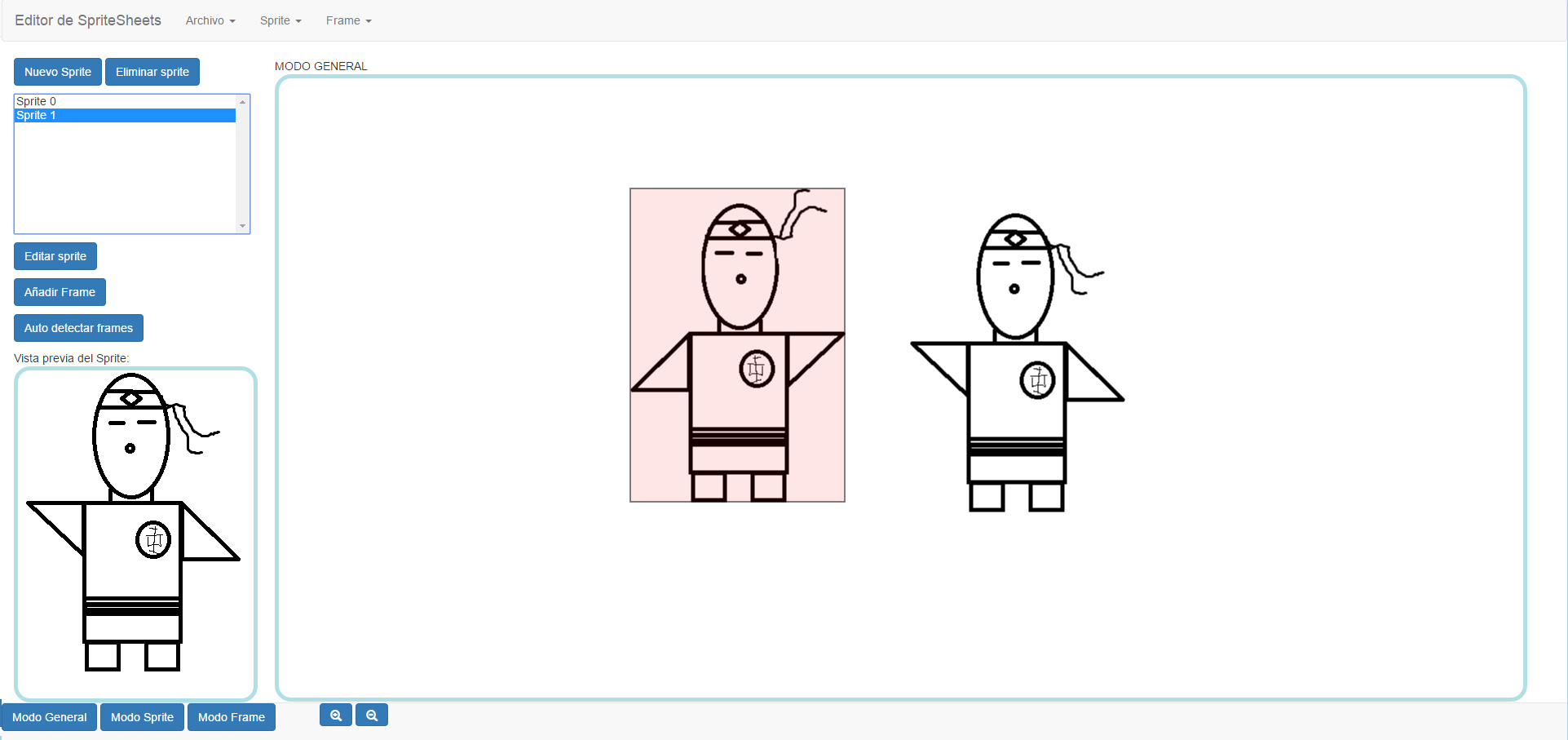


Ilustración 5. Modo General

Mediante este Modo podremos crear Sprites y añadirles Frames seleccionándolos en el área principal. Para ello antes tendremos que importar una imagen mediante el menú archivo, como se ha explicado en el apartado anterior.

Para crear un *Sprite* hemos de hacer clic en la opción *Nuevo Sprite,*  se abrirá un dialogo preguntando por el nombre del *Sprite,* una vez creado, este aparecerá en el listado de *Sprites.*

Mediante el botón *Eliminar Sprite* se podrá eliminarel *Sprite* que este seleccionado en el *Listado de Sprites.*

La siguiente opción es el *Listado de Sprites*, este es el menú mediante el cual se podrá ver cuantos *Sprites* se han creado además de poder seleccionarlos. Al seleccionar un *Sprite* también se cambiara la imagen de Vista previa para mostrar la animación de ese *Sprite* con todos los *Frames* introducidos.

Mediante *Editar Sprite* el programa pasara al modo Sprite con el que hayamos seleccionado en el *listado de Sprites.*

*Añadir Frame* añadirá un Frame al Sprite que tengamos seleccionado en ese momento. Para ello se tendrá que seleccionar mediante un recuadro en el área de trabajo principal

La última opción del menú lateral se trata de la opción de *Auto detectar Frames,* esta opción recorrerá toda el área de trabajo intentando descubrir los *Frames*, después los añadirá en *Sprites* diferentes según las diferencias que encuentre entre ellos.

Por ultimo están los iconos de la *lupa* en el área inferior de la zona de trabajo. Mediante estos botones se puede ampliar la imagen que existe en el área de trabajo.

El funcionamiento del *área de trabajo principal* en este modo es para seleccionar los *Frames* de imágenes externas al programa una vez la imagen haya cargado en el área principal (mediante la opción importar archivo como se ha explicado en el punto 4.2).

*Seleccionar Sprite en el área de trabajo:* se puede seleccionar un Sprite haciendo clic con el botón izquierdo y arrastrándolo alrededor del área deseada. Si la imagen tiene bits de transparencia también se podrá hacer doble clic en el área dibujada del *Frame* y el programa hará el encuadre automáticamente.

*Desplazar imagen de área de trabajo* Haciendo clic con el botón derecho del ratón y desplazándolo por la pantalla se podrá mover la imagen cargada en el programa si esta fuera demasiado grande.

## Modo Sprite

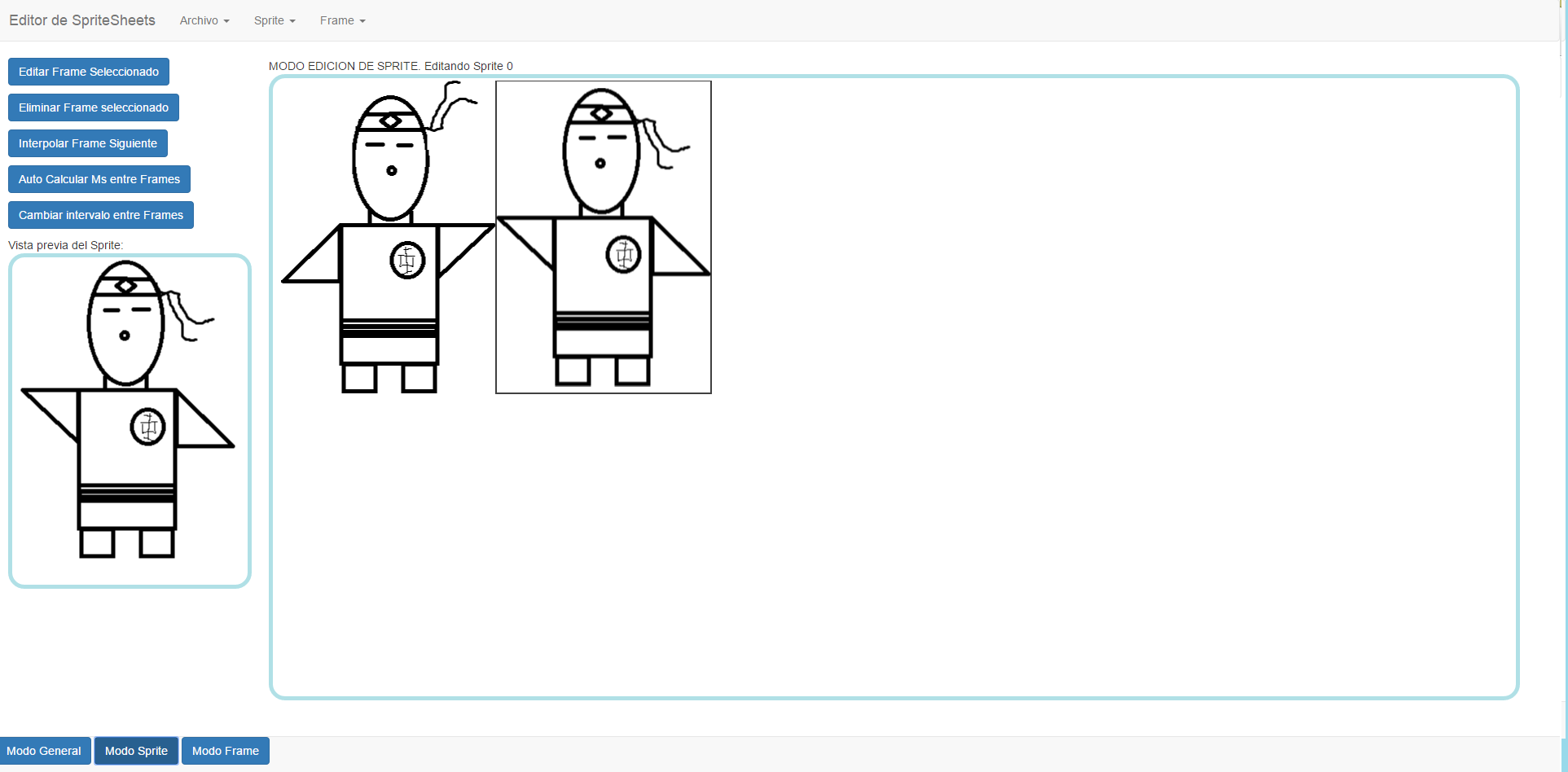


Ilustración 6. Modo Sprite

Editar Frame seleccionado

Eliminar Frame seleccionado

Interpolar Frame siguiente

Auto calcular Ms entre Frames

Cambiar Intervalo entre Frames

Área de trabajo

## Modo Frame

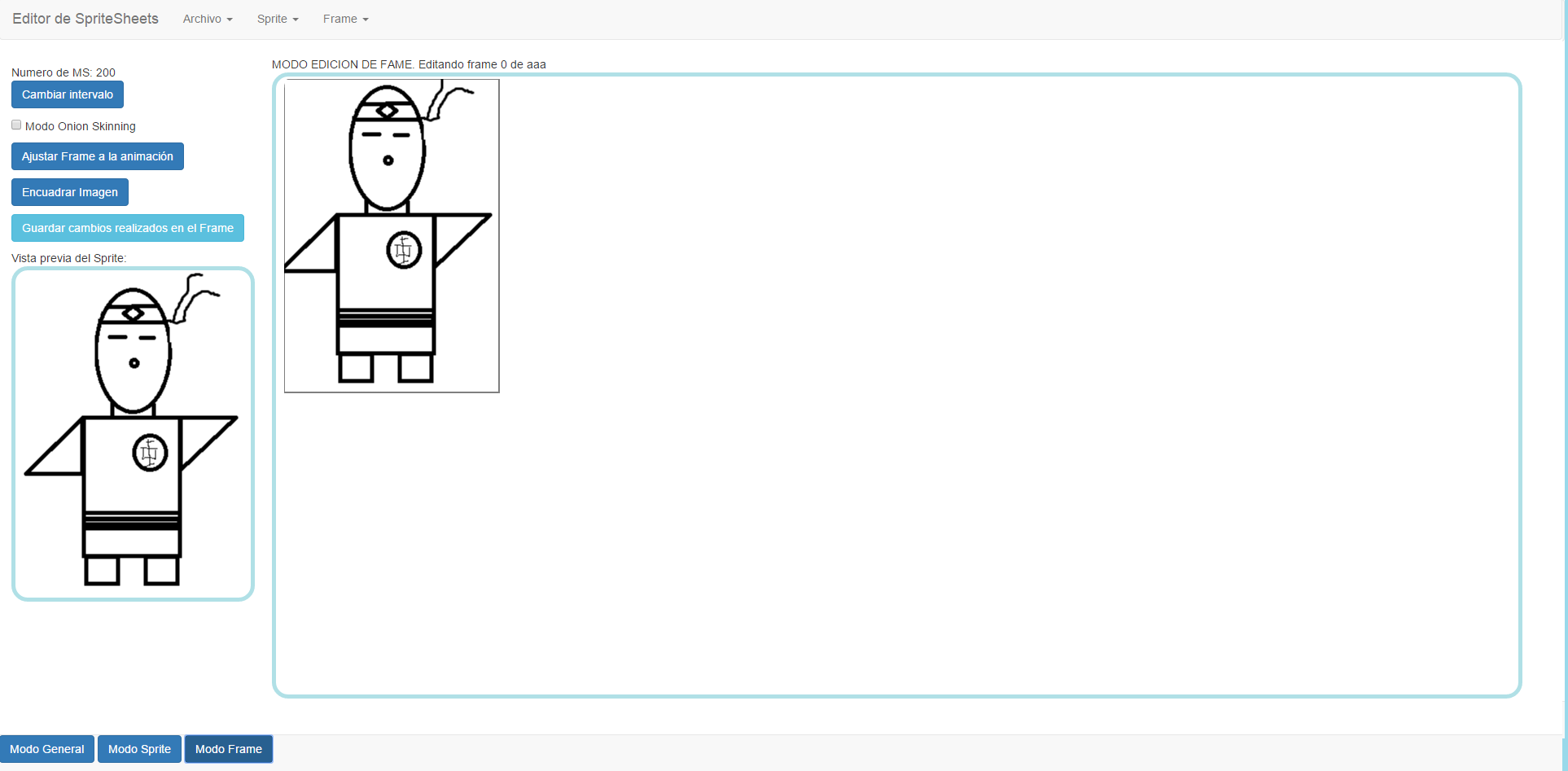


Ilustración 7. Modo Frame

Cambiar Intervalo

Modo Onion Skinning

Ajustar Frame a la animación

Encuadrar la imagen

Guardar

# Implementación

## Introducción

En esta sección del documento se explicara la estructura del programa así como la implementación de algunas funcionalidades interesantes a nivel de código.

## Estructura del programa

El programa está dividido en varias carpetas:

*Carpeta CSS:* se incluyen aquí las hojas de estilo utilizadas para la personalización del aspecto visual de la aplicación. En este proyecto solo se han usado para definir el recuadro alrededor de los Canvas utilizados.

*Carpeta ext:* En esta carpeta se han incluido todas las librerias externas a la aplicación (bootstrap, jsgif, sweetalert y jquery). No se ha modificado ninguna parte del código de esta parte.

*Carpeta inc*: En esta carpeta se han incluido todos los archivos de implementación de los objetos que utilizara la aplicación. En la sección 5.2.1 se explica más detalladamente estas clases.

*Carpeta resources:* Esta carpeta existe para poner recursos varios como imágenes a mostrar durante la ejecución del programa. Actualmente solo consta de White.png que se trata de una imagen precargada al iniciar el programa.

*Carpeta raíz:* En la raíz encontraremos el archivo index.html que equivale a la capa de presentación de la aplicación. En este archivo se indica mediante código HTML los elementos de los que dispondrá la página principal. También se encarga de cargar todos los archivos necesarios para la ejecución del programa (las librerías y los archivos js de la aplicación). También está en la raíz el archivo mainSpriteSheetEditor.js, este archivo contiene toda la lógica relacionada con los botones y la visualización e interacción del usuario con la aplicación.

### Clases creadas

#### Clase Point.js

Se trata de una clase sencilla que guarda dos variables x,y para el manejo de posiciones, tanto en los *Canvas* como en las *Sprite Sheets*, se usa en bastantes partes del proyecto y es considerada un tipo de variable como puede ser un entero o un *string*. Esta clase no tiene ninguna funcionalidad interesante a destacar y únicamente se usa como estructura.

##### Variables

*x:*

Representa la coordenada x del punto

*y:*

Representa la coordenada y del punto

##### Funciones

*reset():*

Acción que sirve para un punto como inválido (que no guarde ninguna coordenada)

*defined() @returns {boolean}:*

Función que devuelve un boleano indicando si el punto esta inicializado o no.

#### Clase Frame.js

Guarda la información que representa un *Frame* de un *Sprite.* Esta se compone de un objeto imagen que guarda la información de ese *Frame* específico, del parámetro *width* que representa la amplitud de esa imagen y del parámetro *height* que representa la altura. Ademas este objeto contiene las funciones para comparar y calcular la diferencia entre dos *Frames* utilizadas tanto para auto detectar animaciones como para auto detectar el intervalo de ms entre los *Frames*.

##### Variables

*image:*

Contiene el elemento DOM de la imagen que representa este *Frame*.

*width:*

Indica la amplitud de la imagen de este *Frame*

*height:*

Indica la altura de la imagen de este *Frame*

##### Funciones

*getImageFrame()@returns {HTMLElement}:*

Devuelve el elemento DOM de la imagen que representa este Frame.

*eval (pixelOrig, imageDataOrig, pixelDest, imageDataDest) @returns {number}:*

Evalúa el parecido del pixelOrig (objeto del tipo Point) de imageDataOrig (estructura getImageData del objeto context) con el pixelDest(objeto del tipo Point) de imageDataDest(estructura getImageData del objeto context). Para realizar esto, por cada canal de la imagen (RGB y ALFA) hace una resta del punto origen al destino. Por ultimo suma sus valores absolutos y devuelve esta diferencia en un valor entre 0 y 1.

*isPointAround(point,dataOrig,dataDest) @returns {number}:*

Devuelve la diferencia (usando la función eval) respecto al punto point (objeto del tipo Point) de dataOrig (estructura getImageData del objeto context) con los puntos de dataDest(estructura getImageData del objeto context) en la misma posición y en los pixeles que rodean esa posición.

*compareWithCanvas(canvas,x,y)* *@returns {number}:*

Compara el *Frame* actual desplazado en la posición x y (valor numérico) respecto a la imagen pintada en un canvas, utilizando la función isPointAround para todos sus puntos, los suma y devuelve este valor en un rango entre 0 y 1.

*compareWithFrame(frameA,frameB,xA,yA,xB,yB) @returns {number}:*

Compara frameA y frameB (objetos del tipo Frame) desplazados en xA,yA (para el caso del frameA) y xB yB (para el caso de frameB. Utilizando las funciones de evaluar diferencias antes comentadas.

#### Clase Sprite.js

Un objeto de esta clase representa el *Sprite,* es decir la animación que se quiere realizar. Además esta clase también es la encargada de interpolar los *Frames* mediante la función *interpolateNextFrame* y de calcular los Ms entre *Frame* y *Frame* cuando el usuario selecciona esta opción. Estas funcionalidades se explicaran más detalladamente en sus respectivos apartados.

##### Variables

*Framelist:*

Es una lista de objetos del tipo *Frame,* representan los *Frames* de la animación

*Pos:*

Es una lista de objetos del tipo *Point* que representan el desplazamiento respecto a la visualización del resto de *Frames* de la animación. Cada posición de esta lista representa al *Frame* de la misma posición en *FrameList*.

*TimeMs:*

Se trata de una lista de números que representan el tiempo que estará el *Frame* durante la animación.

*maxWidth:*

La máxima amplitud que puede llegar a tener teniendo en cuenta todos sus *Frames*

*maxHeight:*

La máxima altura que puede llegar a tener teniendo en cuenta todos sus *Frames*

*defaultMs:*

Es una variable que contiene el valor por defecto que se introducirá en los *Frames* (en realidad es una constante pero por el funcionamiento de JavaScript se ha declarado como variable del objeto).

Las variables restantes se dividen en dos bloques, se han declarado dos intervalos para ir haciendo las animaciones sin bloquear el programa. Esto se ha hecho así para simular el funcionamiento de hilos de ejecución (ya que en JavaScript no existen pero sí que existen los timeInterval[[1]](#footnote-1)).

*Variables Vista previa Canvas:*

Los objetos para la animación del Canvas se utilizan para animar la vista previa del Sprite seleccionado en el navegador del usuario, contiene las siguientes variables:

*timeInterval*:

Evento que interrumpe cada X milisegundos (los indicados en timeMS del anterior *Frame* para mostrar el siguiente *Frame*. Este intervalo ejecuta la función *doPaintAnimation* para ello.

*canvasAnimation:*

Es el *Canvas* donde la función *doPaintAnimation* pintara el *Frame*

*nAnimation:*

Es un contador que indica cual es el *Frame* que tiene que pintar *doPaintAnimation*

*Variables Animación GIF:*

De manera similar a como funciona la animación en la vista previa del *Canvas* para la exportación se utiliza otro timeInterval. Esto se hace de esta forma porque al no existir los hilos, en el momento de exportación el programa se bloqueaba durante toda la animación.

*timeCanvasToExport*:

Se trata del intervalo que se utiliza para simular un hilo de ejecución. Utiliza la función doPaintExportAnimation para pintar en el *Canvas*

*canvasTmpExportAnimation:*

Es el *Canvas* donde la animación se pintara.

*nExport:*

Es el contador que indica cual es el *Frame* que tiene que pintar en el momento de la interrupción.

*Encoder:*

Es el objeto del tipo jsGIf que se utiliza para realizar la exportación de canvas a GIF.

*encoderEnd:*

Es un boleano de estado que indica el momento que el programa haya recorrido toda la animación para acabar el proceso.

##### Funciones

La mayoría de estas funciones son de manejo de la estructura y la otra parte son gets y sets. O son poco interesantes de explicar Además están autodocumentadas en el código. Sin embargo se explicara el funcionamiento de las funciones para las animaciones y como se realiza el cálculo de los milisegundos entre Frame y Frame.

Animación del Canvas de la vista previa

Para las animaciones se utilizan cuatro funciones. Estas funciones son las siguientes:

*paintAnimation(canvas):*

Esta función se encarga de comprobar que exista un Frame en su listado de Frames y de inicializar el Canvas donde se pintara la animación (que es el Canvas que se ha pasado por su parámetro). Por ultimo llama a doPaintAnimation.

*doPaintAnimation():*

Esta función controla cual es el siguiente Frame a pintar en el canvas y se encarga de llamar a la función paintNextFrame para que lo haga. Ademas se autollama mediante un timeout de tantos milisegundos como hayan de pasar en la información introducida en el frame. Esto hace que la ejecución solo pare si se elimina este intervalo.

*stopAnimation():*

Esta función se encarga de borrar el intervalo definido para que la llamada no se vuelva a ejecutar y la animación pare.

*paintNextFrame(ctx, nFrame, transparency, canvas)*

Es la función que se utiliza para pintar el *Frame* nFrame en el *Canvas* inicializado por ctx. El parámetro transparency se utiliza para pintar las transparencias de ese color en lugar de utilizar el parámetro alfa. Este parámetro, para la animación del canvas no se utiliza y se pasa como null.

Exportación GIF

Para exportar a GIF el proceso es semejante a la animación del canvas, sin embargo se utilizan otros parámetros en el momento de pintar. Esto es asi debido a que el canal alfa no existe en GIF y solo se utiliza un color como alfa. Es por ello que el canvas se ha de pintar con ese color cuando en a imagen el valor de alfa este pintado. A diferencia de la animación de la vista previa, en la exportación solo se pintara una vuelta de la animación. Las funciones que se utilizan para la exportación son las siguientes (ademas de paintNextFrame explicada en la animación de la vista previa con el parámetro transparency inicializado a un color):

*exportToGif():*

Igual que paintAnimation esta función se encarga de inicializar todos los parámetros que se utilizaran en el proceso y de lanzarlo. A diferencia de paintAnimation el Canvas se crea en esta función. Además también inicializa el objeto JSGIF y le indica el color que usara para las transparencias.

*doPaintExportAnimation():*

Esta función se encarga (al igual que doPaintAnimation) de calcular cual es el siguiente frame a pintar y de inicializar el timeout para volverse a llamar. A diferencia de doPaintAnimation, llamara a la función paintNextFrame con un valor de transparency diferente de null. Ademas cuando el vector de Frames haya acabado dejara de ejecutarse.

*stopExportToGif():*

Función que se utiliza para parar el intervalo definido y parar la exportación.

*getResultGif():*

Al ser llamada esta función devuelve los datos que contienen la información del GIF ya acabado o -1 si el proceso todavía no ha finalizado.

Interpolar Frame

Para esta funcionalidad lo que se hace es insertar un *Frame* nuevo entre dos *Frames* ya existentes. Para ello se utiliza una función de interpolación lineal que calcula la diferencia entre dos pixeles (sus colores RGB y el canal alfa) en un punto X.

El valor de tiempo inicial para realizar el cálculo es de 0, el valor final es el tiempo que tarda en mostrar el siguiente *Frame* y por último, la imagen interpolada se situara en el tx = (tf - t0)/2.

Para realizar el cálculo correctamente, se pintan los dos *Frames* en un *Canvas* añadiendo el desplazamiento (la x y la y de su posición en la animación). El nuevo canvas será del tamaño máximo de la altura y amplitud entre los dos *Frames.*

Por último se cambian los valores de t0 a la mitad (para mantener la misma velocidad en la animación) y se guarda e valor de tx en el vector de *Frames* de la animación.

La función que se utiliza para hacer esto es la siguiente:

*interpolateNextFrame(no):*

Crea un nuevo Frame interpolando el Frame no con no+1 si este existe o vuelve sin hacer nada si no hay siguiente *Frame.*

Calcular tiempo en ms automáticamente

Esta funcionalidad utiliza la función *compareWithCanvas* explicada en el apartado 5.2.1.2.2 para calcular la diferencia entre el *Frame* n y el *Frame* n+1*.* Además si se encuentra con dos *Frames* iguales los eliminara y los tratara como uno solo que dura más tiempo. Por ultimo utilizara una de las dos funciones elegidas como parámetro para darle un valor al tiempo según la diferencia. Esto se ha hecho como parámetro porque según lo que el creador de la animación quiera mostrar, intentara buscar efectos diferentes (por ejemplo mostrar un *Sprite* acelerado, o al contrario mostrarlo por el contrario más pausado). A continuación se explican las funciones:

*autoTuneTimeMs(usedFunc):*

Esta es la función que itera sobre cada *Frame* mirando las diferencias entre el *Frame* n y el n + 1. Después llama a la función pasada por el parámetro usedFunc para dar un valor al tiempo.

*autoTuneCalcFastDiffFunc(dif, count):*

Función que calcula el tiempo de un Frame basándose en dif (la diferencia con el siguiente frame) de manera que cuanto más diferente sea un *Frame* con el siguiente pase más rápido. Count es un parámetro que cuenta el número de veces que ha aparecido el mismo *Frame* (y se ha eliminado al estar repetido)

*autoTuneCalcSlowDiffFunc(dif, count):*

Esta función realiza los cálculos necesarios para crear el efecto contrario a *autoTuneCalcFastDiff*, es decir, que cuanto más diferentes sean dos *Frames* los muestre mas pausados.

#### Clase SpriteSheet.js

Esta clase es la que representa toda la información que guarda el programa. Representa el *Sprite Sheet* que finalmente se querrá exportar. Maneja toda la lógica en el momento de crear un *Frame* o un *Sprite* (controla que no se repitan, ejecuta sus operaciones, decide que *Sprite* se ha de pintar en la vista previa (el que el usuario haya seleccionado), guarda información sobre con que *Sprite* y con que *Frame* se están trabajando y además es la clase encargada de exportar el PNG y el XML como se explicara más adelante.

##### Variables

La clase SpriteSheet.js Consta de las siguientes variables:

*spriteList:* Es una lista donde se guarda cada *Sprite* que contendrá la *Sprite Sheet*.

*wMax:* En esta variable se guarda el valor de la amplitud del Frame más amplio de todo el *Sprite Sheet.*

*hMax:* En esta variable se guarda el valor de la altura del *Frame* más alto de todo el *Sprite Sheet.*

*oldAnimation*: Esta variable contiene un identificador de cuál era el *Sprite* que se estaba usando en la animación de la vista previa por si se necesita enviar una señal de parar para poner otro *Sprite* en la vista previa.

*frameSelected*: Esta variable guarda que *Frame* es el que hay seleccionado.

*spriteSelected:* Esta variable guarda cual es el *Sprite* que hay seleccionado.

*frameMetaData:* Se trata de una lista en la que cada posición contiene una estructura sobre cómo se situara cada *Frame* en el momento de hacer la exportación. Esta estructura contiene la siguiente información:

*IndexSprite:* Es la posición del vector *SpriteList* en la cual está situado el *Frame* actual

*IndexFrame:* Es el número del *Frame* de la animación del *Sprite IndexSprite* sobre el cual se está guardando la información

*Pos:* Es una variable del tipo *Point* que indica donde se posicionara el *Frame*

*Width:* Variable que contiene la amplitud que ocupa el *Frame* en la *Sprite Sheet*

*Height:* Variable que contiene la altura que ocupara el *Frame* en la *Sprite Sheet*

##### Funciones

Como funcionalidades interesantes de esta clase existen las funciones que crean el *Sprite Sheet* y su *Meta Data*. Se ha demostrado que el algoritmo de ordenar elementos de distintos tamaños en un área infinita de manera óptima es un algoritmo NP-HARD. Se podría llegar a una solución óptima utilizando un algoritmo de backtracking hasta recorrer todos los estados posibles. Sin embargo esto ocuparía un tiempo que no es admisible para el uso de una aplicación web. Así que se han intentado varias soluciones hasta encontrar alguna que sea una aproximación que de un resultado aceptable.

*getSpriteSheet()@returns {HTMLElement}:*

Esta función devuelve un canvas de todos los *Frames* organizados según la función utilizada en organizeFunction.

*getClanLibXML()@return{String}:*

Esta función devuelve el texto que representa el archivo XML para la librería *ClanLib* con toda la información de en qué posición se encuentra cada *Frame* de cada *Sprite* dentro de la *Sprite Sheet* organizada según la función organizeFunction.

*organizeFunction(type):*

Esta función es la encargada de ordenar la *Sprite Sheet, según el parámetro indicado en type utilizara una función u otra.*

*organizeSpriteSheetHorizontalOnly():*

Esta función se trata de una primera implementación bastante simple que solo organiza los *Frames* poniéndolos uno seguido de otro sin ningún tipo de limite. No tiene ningún interés

*organizeSpriteSheetShelf(maxWidth):*

Esta función aplica un algoritmo por niveles con una amplitud máxima fijada y con altura fija. En este algoritmo primero se ordena de mayor altura a menor altura los elementos y se intentan colocar en ese orden. Si al introducir el elemento este no cabe (debido a la limitación de la amplitud), se creara un nuevo nivel. Los próximos elementos intentaran introducirse en todos los niveles ya existentes creados de esta forma.

El resultado de este algoritmo, aunque crea un *SpriteSheet* bastante bueno si los *Frames* tienen la misma medida, no es del todo bueno si existe algún elemento con mucha diferencia respecto al resto.

*organizeSpriteSheetLeftTop():*

Este algoritmo intenta ordenar de una manera rápida descubriendo primero cual es la mejor solución con una altura fija.

En la primera vuelta del algoritmo se fija esta altura al elemento más alto, y se intentan poner los elementos lo más a la izquierda posible. Si un elemento cabe debajo de otro lo colocara allí, si no ampliara la amplitud disponible hasta que se hayan colocado todos los elementos. En ese momento se define la mejor solución como la altura y la amplitud que nos ha dado esta primera vuelta.

En ese momento se reduce la amplitud disponible y se intentan volver a colocar todos los elementos, si caben se volverá a reducir hasta que los elementos no quepan.

Si los elementos no caben lo que se hará es ampliar la altura disponible y volver a intentarlo hasta que se puedan volver a colocar todos los elementos.

En el momento en que la amplitud disponible para colocar los *Frames* sea de la misma amplitud o menor que el *Frame* más amplio del *Sprite Sheet* el algoritmo habrá finalizado y se podrá devolver cual es el área más pequeña que permite alojar los *Frames.*

Para hacer los cálculos más rápidos sin tener que intentar introducir todos los elementos hay las optimizaciones de no comprobar áreas que sean mayor que el área mínima que hemos encontrado hasta el momento, cuando esto pase restaremos la amplitud directamente sin comprobar si los elementos cabían o no.

Otra optimización es la de que si el área que vamos a comprobar es más pequeña que la suma de áreas de los elementos que vamos a introducir en ella, se amplía directamente la altura. Ya que no habrá manera posible de introducir estos elementos haciendo que quepan en ella.

Para introducir los elementos dentro del área disponible lo que se hace es crear una matriz de una celda con la amplitud y altura que queremos probar. Cada celda de la matriz, tendrá una altura y una amplitud y tendrá un valor indicando si está ocupada por parte de un *Frame* ono. Si el elemento cabe dentro de la celda y no está ocupada, se dividirá la matriz en una fila y una columna de manera que la celda antigua tenga el tamaño justo para que el elemento introducido quepa en ella (y la nueva fila y columna tendrán el resto de pixeles disponibles).

El siguiente elemento se introducirá comprobando primero si cabe en las columnas de la izquierda. Si no es capaz de introducirlo pero tiene más área disponible ocupando una de las celdas de abajo o de la derecha que estén libres entonces las utilizara y las marcara como ocupadas. En cualquier otro caso intentara introducirlas en las siguientes columnas.

Este algoritmo genera *Sprite Sheets* de una manera bastante rápida y con un resultado bastante bueno para todos los *Frames* probados. Así que finalmente se ha decidido que el programa funcionara con este algoritmo.

#### mainSpreetSheetEditor.js

En esta clase existen dos funciones interesantes a nivel algorítmico. La primera de ellas se trata de la función que calcula el doble clic. La segunda se trata de la función que se utiliza para auto detectar las animaciones, que de hecho se base en la anterior. A continuación se explicara las dos funciones. Para las dos funcionalidades es necesario que el área de trabajo contenga pixeles transparentes, ya que es la manera que utiliza el programa para detectar si un *Frame* está separado de otro.

##### Variables

*selectionTL:*

Se trata de la variable que indica el punto superior izquierdo de una selección.

*selectionBR:*

Se trata de una variable que indica el punto inferior derecho de una selección.

*puntosVisitados:*

Es una matriz del tamaño de la imagen que indica si un punto se ha visitado o no. Se utiliza para no repetir comprobaciones.

*dataImage:*

Es una estructura que incluye la información sobre la imagen introducida en el canvas. Esta estructura incluye su altura, su anchura, y una lista en la que cada elemento representa el valor que tiene un canal (Rojo, Azul, Verde o Transparente) en un pixel, para el algoritmo solo se utilizara la información del canal transparente.

*listOfFrames:*

Se trata de una estructura temporal utilizada para guardar los Frames encontrados en un *Sprite* *Sheet* mediante el algoritmo de auto detección. Esta estructura será la misma que utilizara luego el algoritmo de detectar animaciones para introducirlas en su *Sprite* correspondiente.

##### Funciones

*getSelectionAroundPoint(point, imageData):*

Esta función inicializara la matriz de puntos visitados a falso y después visitara todos los puntos alrededor de la posición *Point* en la imageData.

En primer lugar se marcara el pixel inicial como visitado y se procederá a mirar los pixeles adyacentes. Si el pixel no está marcado como visitado y no es transparente hemos encontrado un elemento. Sin embargo este elemento podría no ser del Frame, así que también hay que comprobar que los pixeles vecinos a este pixel encontrado estén marcados. Si es adyacente a un pixel ya encontrado y el área con este pixel es más grande que el área máxima que estábamos comprobando, la ampliaremos y seguiremos buscando. En cualquier caso, marcaremos el pixel como visitado.

Este algoritmo se ira repitiendo hasta que en una de las vueltas no encontremos un pixel sin marcar que cumpla las condiciones. En ese momento habremos encontrado el rectángulo que encuadra el *Frame* alrededor del punto inicial.

*autoDetectFrames():*

Para esta función se ha creado una estructura de áreas a comprobar. Esta estructura se trata de un vector que contiene dos puntos representando la esquina de arriba a la izquierda del área a comprobar y la esquina de abajo a la derecha.

Esta función recorrerá el área de la imagen desde la posición inicial indicada en el vector de áreas hasta la posición final. Si no encuentra ningún *Frame* no transparente en ese espacio el algoritmo habrá finalizado (o si hay más áreas pasara a la siguiente).

En el momento en que encuentre un punto que no sea transparente realizara una llamada a la función *getSelectionAroundpoint* con el punto encontrado para descubrir cuál es el recuadro que contiene el *Frame*.

A partir de ese momento lo que se hace es comprobar si el *Frame* esta completamente dentro del área a comprobar, si es asi se crean dos nuevas áreas, una a la derecha de la imagen con su altura, y otra en la parte de abajo ocupando todo la amplitud a comprobar. Además se limitara el área actual a la altura del *Sprite* para no solapar las nuevas áreas creadas.

*autoDetectAnimations(percentDifference):*

# Gestión del proyecto

## Introducción

En esta sección se explicaran todos los conceptos relacionados con la gestión del proyecto. Se hablara de la planificación y de cómo ha ido variando mientras se avanzaba en el proyecto. También se explicara en esta sección el presupuesto y como se ha obtenido este presupuesto. Por ultimo hay un apartado dedicado a las leyes que podrían afectar a este proyecto y se tienen que tener en cuenta si se quisiera implantar la solución en un entorno de producción.

## Planificación del proyecto

Aunque se ha seguido el orden de la planificación del proyecto inicial, ha habido una desviación en el tiempo de desarrollo.

En la planificación inicial se propuso que el proyecto duraría 3 meses y que se realizaría entre el 17 de septiembre al 30 de noviembre. Debido a haber subestimado el tiempo que llevaría realizar la asignatura de gestión de proyectos y otras obligaciones, se ha tenido que ralentizar la realización de este TFG.

En primer lugar se ha retrasado un mes debido a que no se comenzó a realizar la implementación de código hasta no estar acabado el módulo de GEP.

En segundo lugar, el no conocer completamente la tecnología al empezar a desarrollar ha hecho que el tiempo estimado que se propuso fuera incorrecto.

A continuación se explicará con detalle la nueva planificación final indicando que partes se han modificado respecto al inicial.

### Duración

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

Finalmente el proyecto se ha alargado varios meses más. Concretamente el proyecto se ha dado por finalizado el día 16 de Enero quedando algunas funcionalidades por implementar en un futuro debido a que eran partes que ya se establecieron al principio que serían ampliaciones opcionales.

Hay que tener en cuenta que aunque en la planificación temporal no se especifica, en el tiempo estimado para cada característica se está teniendo en cuenta el análisis, el diseño, la implementación y la comprobación de validez de cada una de ellas.

### Planificación temporal

A continuación se detallan todas las etapas por las que ha ido pasando el proyecto y se especifica si ha habido algún cambio en algún momento.

#### Etapa de Planificación y viabilidad del proyecto

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

No hay cambios respecto a la etapa planificación y viabilidad del proyecto y se cumplió en el tiempo previsto.

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 |
| Gestión 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 2. Etapa planificación y viabilidad Del proyecto

#### Primera iteración

En esta primera fase se estableció inicialmente que se implementarían las características del editor de *Sprite Sheets* necesarias para que el usuario pueda elegir cómo serán las animaciones de sus *Sprites*.

Se desplazó la fecha inicial de implementación debido a la infravaloración del tiempo que ocuparía el módulo GEP.

Además debido a otras obligaciones y la subestimación del tiempo de implementación al desconocer la tecnología utilizada, se fue retrasando la fecha final hasta el día 24/12/2014.

Se dividió en las siguientes dos etapas:

|  |  |  |
| --- | --- | --- |
| Nombre | Fecha Inicial | Fecha Final |
| Implementación marcaje de *Frames* | 13/10/14 | 24/12/2014 |
| Cargar y visualizar imagen | | |
| Hacer zoom de la imagen cargada | | |
| Permitir al usuario marcar un rectángulo que contenga un *Frame* del *Sprite* *Sheet* | | |
| Acumular los *Frames* marcados para crear un *Sprite* | | |
| Ajustar la forma manual de 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 | 27/11/14 | 24/12/2014 |
| Visualizar el *Sprite* generado a partir de los *Frames* seleccionados (un *Frame* al lado del otro) | | |
| Visualizar la animación que el usuario a seleccionado | | |
| Darle al usuario la posibilidad de especificar los mili-segundos entre *Frame* y *Frame* (para todos los *Frames* o de uno en uno) | | |

Tabla 3. Etapa primera iteración

#### Segunda iteración

Una vez acabada la fase de la exportación, la aplicación ya era utilizable por cualquier usuario que quisiera un editor de *Sprite Sheets* y se habría conseguido alcanzar la primera fase del alcance del proyecto. Por esta razón se decidió que en la planificación temporal la exportación se realizara antes que la automatización aunque no existiera ningún prerrequisito entre ellas.

En el hito intermedio se tuvo que retrasar las fechas de inicio y las fechas de fin pero se mantuvo la duración aproximada de esta iteración.

Al no haber tenido en cuenta las fechas navideñas esto hizo que la etapa se alargara del 10/01/2015 (fecha que se estableció en el hito intermedio) al 15/01/2015.

|  |  |  |
| --- | --- | --- |
| Nombre | Fecha Inicial | Fecha final |
| Exportación | 24/12/2014 | 04/01/2015 |
| Exportar animaciones para librerías como *Clanlib Game SDK* en formato *XML* | | |
| Exportar animaciones de un *Sprite* especifico en formato *GIF* | | |
| Exportar un *Sprite Sheet* en formato *PNG* | | |
| Automatización | 04/01/2015 | 15/01/2015 |
| Auto detectar que *Sprites* forman parte de una animación | | |
| Optimizar el espacio vacío entre dos *Sprites* | | |
| Cálculo automático del número de mili-segundos entre *Frames* | | |
| Cálculo automático de los *Frames* que hay entre dos *Frames* | | |

Tabla 4. Etapa segunda iteración

#### Tercera iteración

Esta iteración se trataba de una ampliación de características para añadir un aspecto social a la plataforma además de permitir guardar ficheros en la nube. Como ya se explicó en el documento inicial, estas eran características que añadían valor a la plataforma pero que no son necesarias para el problema planteado inicialmente y se tratan de tareas opcionales.

Debido a la desviación en el proyecto se decidido en el momento de hacer el hito intermedio no implementar esta parte a no ser que se hubiera sobrestimado el tiempo restante hasta la finalización del proyecto, en cuyo caso se hubiera realizado una reunión con el director del proyecto para elegir cuál de las siguientes características se implementarían para la plataforma.

Debido a que esto no ha ocurrido se deja como posibles implementaciones en un futuro pero no se consideran parte de la plataforma y en caso de realizarlas se trataría de otro nuevo proyecto compatible con la aplicación presentada en este documento.

Las funcionalidades que no han sido implementadas para este proyecto son las siguientes:

|  |  |  |
| --- | --- | --- |
| Nombre | Fecha inicial | Fecha final |
| Socialización plataforma | No aplica | No aplica |
| Crear cuenta de usuario para conservar ficheros | | |
| Posibilidad de conservar 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 | No aplica | No aplica |
| Mejoras en el buscador | | |
| Modificar partes de un *Sprite Sheet* (como puede ser modificar el color de un pixel, o el color de varios pixeles) | | |

Tabla 5. Etapa tercera iteración

#### Final de proyecto

La última fase se trata de la preparación para la presentación del TFG ante el tribunal.

Inicialmente se pusieron unas fechas orientativas y se dejó 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. Hay que recordar que aunque ese margen de tiempo quedara en 0 días no significaría que no existiera el testing porque aunque no están especificado en la planificación temporal todas las etapas tienen incluidas el análisis, diseño, implementación y comprobación de validez para cada una de ellas.

Se desplazó esta etapa debido al retraso de las otras etapas. Finalmente la etapa de ajustes se ha utilizado para mejorar el aspecto visual de la aplicación. También se han modificado las fechas de preparación de la documentación teniéndolas antes de lo marcado durante la etapa del hito intermedio

A continuación la Tabla 5 indica las fechas finales de esta etapa teniendo en cuenta todos estos cambios:

|  |  |  |
| --- | --- | --- |
| Nombre | Fecha inicial | Fecha final |
| Testing y ajustes | 15/01/15 | 19/01/15 |
| Documentación | 15/01/15 | 19/01/15 |
| Preparación presentación | 19/01/15 | 26/01/15 |
| Presentación | 26/01/15 | 30/01/15 |

Tabla 7. Etapa final del proyecto

#### Tiempo total

|  |  |
| --- | --- |
| Etapa | Horas |
| Planificación y viabilidad del proyecto | 90 |
| Primera iteración | 78 |
| Segunda iteración | 84 |
| Tercera iteración | 108 |
| Final del proyecto | 108 |
| Horas Totales | 468 |

Tabla 6. Horas del proyecto

#### Diagrama de Gantt

A continuación en la Ilustración 1 se puede observar un diagrama detallado con las fechas explicadas en los anteriores apartados.

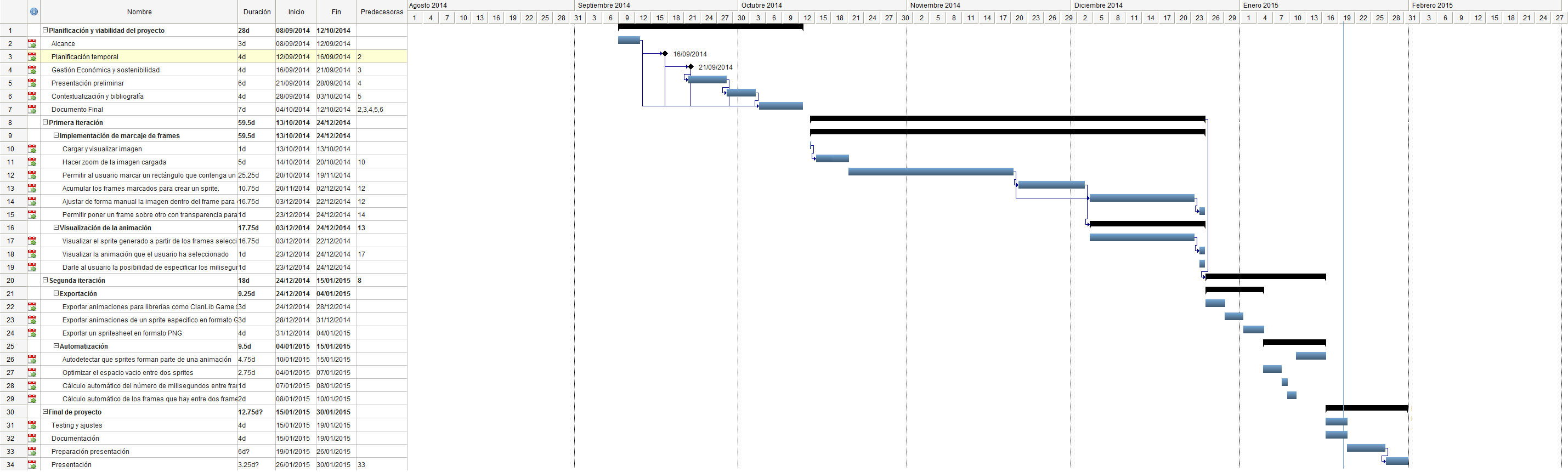


Ilustración 8. Diagrama de Gantt

## 

## Plan de acción

Como se ha explicado en el apartado 2.3 en este proyecto se utilizan metodologías ágiles. La idea que se iba a seguir en caso de tardar menos tiempo del previsto en alguna iteración, era comenzar a implementar características de la siguiente iteración en el orden establecido. En el caso contrario, se tenía pensado retrasar la implementación de iteraciones siguientes. Se dejó una etapa en la iteración final llamada testing y ajustes para compensar este tipo de desvíos.

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

Debido a los retrasos respecto a la planificación propuesta inicialmente, estas reuniones se fueron retrasando. En el hito intermedio se decidió cambiar las reuniones, en lugar de esperar a que una fase acabara, el director del proyecto y el alumno se ponían en contacto mediante email cada vez que una funcionalidad intermedia de una fase era implementada para darla por validada o no según los requisitos iniciales propuestos.

## Presupuesto y sostenibilidad

### Introducción

En este apartado se explicara detalladamente el 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.

Cabe destacar que aproximadamente las horas dedicadas a cada tarea son las mismas y el proyecto se ha desplazado por haber dedicado 21 horas semanales en lugar de las 31 que se tenían previstas.

Además inicialmente se planteó la necesidad de un servidor sobre todo por las funcionalidades de la tercera fase. Como estas no se han implementado el servidor no es necesario, pero no comporta ningún gasto extra, en primer lugar porque el *hosting* que se había buscado era gratuito. En segundo porque en el caso de haberlo contratado de pago, se habría contratado al poner la página en producción, y esto solo habría ocurrido finalizada la tercera fase.

### Recursos

Durante el desarrollo de la plataforma, se han utilizado los siguientes 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
* Open Office 4
* XAMPP v.3.2.1
* Navegador Chrome Versión  39.0.2171.99 m
* Navegador Firefox v 33.1.1
* Internet Explorer v 11.0.9600.17107

Humanos:

* 1 Persona con disponibilidad semanal de 21 horas.

En los siguientes puntos se detallara el coste de los recursos utilizados

#### 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 6 aparecen estas tareas desglosadas según el diagrama de Gantt.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rol | Tarea | Horas Aproximadas | | Coste 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 8. Recursos humanos

#### 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 7 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 9. Hardware

#### Software

Para realizar el proyecto se necesita Software específico, se ha optado por utilizar software libre en la medida de lo posible para poder ahorrar costes. En la Tabla 8 se detalla el software utilizado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Producto | Precio | Unidades | Vida Útil | Total amortización |
| 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 10. Software

#### 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 9 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 | 885,00 € | 94,53 € |

Tabla 11. Gastos Generales

#### 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 10 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 | 885,00 € | 94,53 € |  | 6,44 € |

Tabla 12. Riesgos

#### Coste del proyecto

En la Tabla 11 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 13. Coste del proyecto

### Sostenibilidad

#### 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.

#### 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.

#### 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.

#### Matriz de sostenibilidad

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

Se ha reducido la puntuación de consumo de recursos al haber planteado inicialmente la necesidad de un servidor. Sin embargo la del coste final no cambia y por eso se ha introducido un 10.

En adaptación a los cambios de escenarios se ha introducido una puntuación de 0 ya que en la planificación inicial las funcionalidades no implementadas ya estaban pensadas como opcionales y quedaban fuera del proyecto para darlo como finalizado.

|  |  |  |  |
| --- | --- | --- | --- |
| ¿Sostenible? | Económica | Social | Ambiental |
| Planificación | Viabiliad Económica | Mejora en la calidad de vida | Análisis de recursos |
| Valoración | 9 | 9 | 10 |
| Resultados | Coste final versus previsión | Impacto en entorno social | Consumo de recursos |
|  | 10 | 10 | 8 |
| Riesgos | Adaptación a cambios de  escenario | Daños sociales | Daños ambientales |
|  | 0 | 0 | 0 |
| valoración total | 56 |  |  |

Tabla 14. Matriz de sostenibilidad del TFG

## Identificación de leyes y regulaciones

Debido a que la plataforma no conserva ningún dato, no es necesario contemplar leyes de protección de datos.

Sin embargo, si se llegara a implantar la tercera iteración, al disponer de información de usuarios e incluso sus creaciones se debería de cumplir la LOPD (Ley Orgánica de 12 de protección de datos) y se debería estudiar también la ley de propiedad intelectual.

También se debería cumplir, al implementar esta iteración, la ley de cookies.

#### LOPD

Para cumplir esta ley se debe en primer lugar informar a los usuarios que información estamos recopilando sobre ellos. Además se le debe ofrecer al usuario la opción de darse de baja si no desea que sigamos teniendo su información. En el caso de la plataforma que se quiere implementar solo se necesitaría su email y un nombre de usuario para el registro. Si en un futuro se decidiera la necesidad de más información, estos se deberían añadir al aviso.

Otras partes de esta ley que se deberían tener en cuenta para este proyecto son la necesidad de establecer una serie de medidas de seguridad en la página como pueden ser hacer copias de seguridad, cambiar las contraseñas de acceso a los datos una vez al año, garantizar que no se puede obtener información del usuario mediante algún fallo de programación.

Por último, se tendría que registrar el fichero de datos en la AEPD (Agencia española de protección de datos).

#### Ley de propiedad intelectual

En lo que respecta a las creaciones de los usuarios se debería tomar una decisión. La intención de la página web es en primer lugar que los usuarios puedan compartir sus obras con otros usuarios, aunque la web no tiene ningún interés en estas obras sí que es interesante que los usuarios puedan utilizarlas sin tener problema alguno en lo referente a la ley de propiedad intelectual.

Una solución sería que los usuarios al subir una imagen, cedieran los derechos a la página web u obligar que sus creaciones fueran obras de dominio público con alguna licencia tipo *Copyleft* o *Creative* *Commons*. Otro problema podría ser que los usuarios subieran contenido protegido. Para proteger a la web de esto, se debería responsabilizar a los usuarios de todo el contenido que suban y se les tendría que avisar que el incumplimiento de esto implicaría perder las creaciones que hayan subido a la página.

#### Ley de cookies

Esta ley obliga a toda página web que utilice cookies que informe al usuario de su existencia y cuál es su uso, además debe permitir desactivarlas y explicar cómo hacerlo.

Para la página web se utilizarían únicamente para conservar la sesión del usuario una vez hecho el *login*. Se debería indicar que son cookies técnicas para el funcionamiento interno de la web, y que el usuario no podría hacer *login* si las desactiva.

Glosario

*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. 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.

*Frame*

*Sprite:* Originalmente 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.

*Meta Data:* 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 cuál es el tamaño de cada *Frame*.

*SCRUM*

G*ithub*

*Clanlib*

*XML*

Json

*SFML*

*SDL*

*Onion Skinning*

*Framework*

*tags*

*Canvas*

# Bibliografía

[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 http://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 animaciones 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.

[14] Agencia Española de protección de datos. URL <http://www.agpd.es/portalwebAGPD/index-ides-idphp.php> Accedido el 21 de diciembre de 2014

1. Estos intervalos lo que hacen es interrumpir la ejecución y ejecutar una función al pasar los segundos indicados. [↑](#footnote-ref-1)