Desarrollo del juego Stacker para HTML5/Canvas

Tutor: Abraham Rodríguez Rodríguez
Departamento de Informática y Sistemas
Universidad de Las Palmas de Gran Canaria

Domingo Hernández Romano
18/11/2014

Índice de contenido

1. Introducción....3
Estructura de la memoria....4
Estado del arte....4
2. Metodología y etapas de trabajo....8
Etapas del trabajo....8
3. Competencias a cubrir....10
4. Normativa y Legislación....11
Software Libre....11
Ley de Propiedad Intelectual. 13
5. Aportaciones....13
6. Requisitos Hardware/Software....14
Lenguajes de Programación....18
Un poco de canvas....19
7. Desarrollo....21
Etapa 1: Análisis....21
Requisitos de usuario....22
Reglas del juego....23
Diagrama de casos de uso....23
Etapa 2: Diseño....31
Arquitectura del sistema....31
Diseño adaptable (Interfaces)....32
Diseño Software 38
Etapa 3: Implementación....43
-Implementando la lógica de classic Stacker....44
-Implementando la lógica de super Stacker....48
-Local Storage....54
Etapa 4: Pruebas....55
-Classic Stacker....55
-Super Stacker....57
-Pruebas visuales....59
-PhoneGap build....65
8. Guía usuario....70
9. Conclusiones y experiencia personal....74
10. Fuentes de Información....75

1. Introducción

El trabajo fin de grado "**Stacker para HTML5**" plantea dos modalidades de juego basadas en el clásico Stacker, que consiste en apilar en vertical un conjunto de figuras rectangulares alineadas horizontalmente y que a su vez describen un movimiento horizontal constante que variará según avance el juego.

La finalidad del proyecto es ofrecer dichas modalidades de juego basadas en el clásico Stacker utilizando una tecnología novedosa como lo es **Canvas de HTML5**, extendiendo así su uso a muchas de las plataformas que hoy en día están en el mercado (dispositivos móviles, tablets, etc).

Las modalidades de juego ofrecidas son las siguientes:

- Classic Stacker: Partiendo de la base del juego original, el jugador puede continuar apilando hasta que se quede sin figuras rectangulares, consiguiendo una puntuación según el nivel actual, la velocidad de movimiento y las figuras disponibles anteriormente. El jugador podrá visualizar sus mejores puntuaciones hasta el momento, dando así la posibilidad de superarse una y otra vez.

- Super Stacker: Esta modalidad de juego ofrece más dinamismo, ya que partiendo de una serie de figuras estáticas colocadas de forma estratégica en un entorno, el usuario ha de apilar una serie de figuras con formas distintas que están afectadas por fuerzas tales como la gravedad, colisiones entre objetos existentes, etc. Para ello ha sido necesario aplicar un motor físico para canvas que describiremos más adelante. El usuario ha de apilar sobre los objetos estáticos todas las figuras que se le proponen y una vez que lo ha hecho, la estructura resultante ha de aguantar un número de segundos sin que ninguno de los objetos entre en contacto con los límites del mundo planteado. Cada nivel plantea una dificultad distinta y además, cada uno de ellos puede volver a jugarse con el objetivo de superarse en cuanto al tiempo tardado en superar el nivel. El usuario tendrá la opción de visualizar los mejores de cada nivel.

Destacar que no existe en el mercado actual ninguna de estas modalidades desarrolladas para CANVAS.

Estructura de la memoria

Esta memoria intentará seguir una estructura que sea de fácil lectura y entendimiento. Es por ello que la memoria se ha dividido de la siguiente manera:

- Proporcionar una pequeña introducción que servirá para comprender mejor la envergadura del trabajo desarrollado.

- Pequeño análisis de la competencia, es decir, aplicaciones que actualmente existen y tienen similitudes con el trabajo realizado.

- Metodología y etapas de trabajo que se han afrontado a lo largo de todo el proyecto.

- Competencias a cubrir.

- Detalles de análisis, diseño e implementación. Se explicarán los puntos más importantes que han afectado al código desarrollado, así como el acoplamiento de otros frameworks al proyecto.

- Listado de casos de uso

- Detalles y manual acerca de la jugabilidad.

Estado del arte

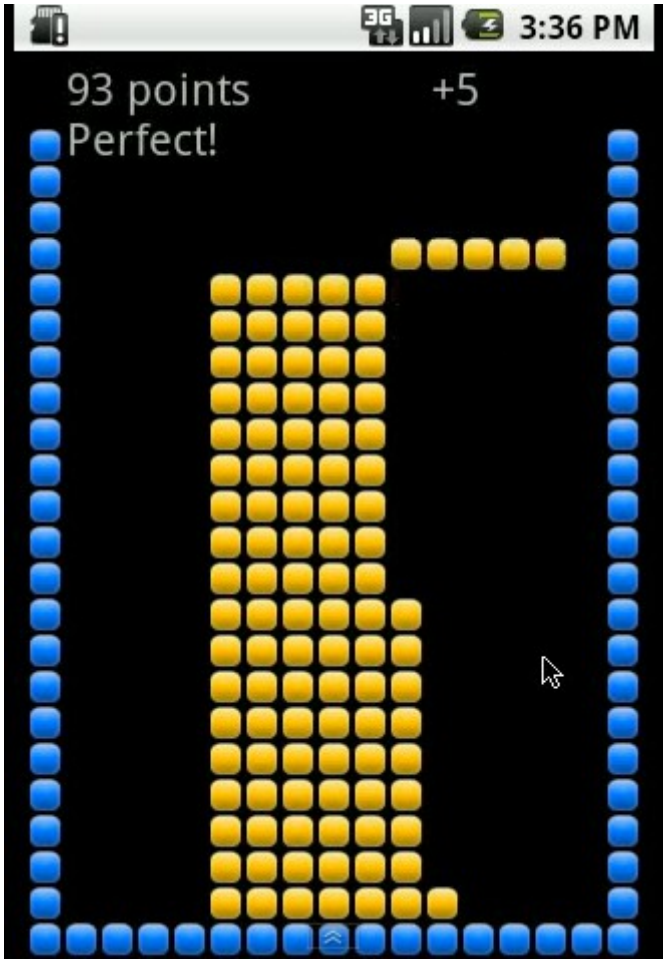
Analizando el mercado actual, se pueden encontrar aplicaciones que ofrecen una jugabilidad parecida a las modalidades desarrolladas en este proyecto.

Empezaremos analizando la modalidad classic Stacker del proyecto, buscando juegos que ofrezcan un parecido con el desarrollado en el actual proyecto.

Continuaremos buscando lo mismo para la modalidad Super Stacker.

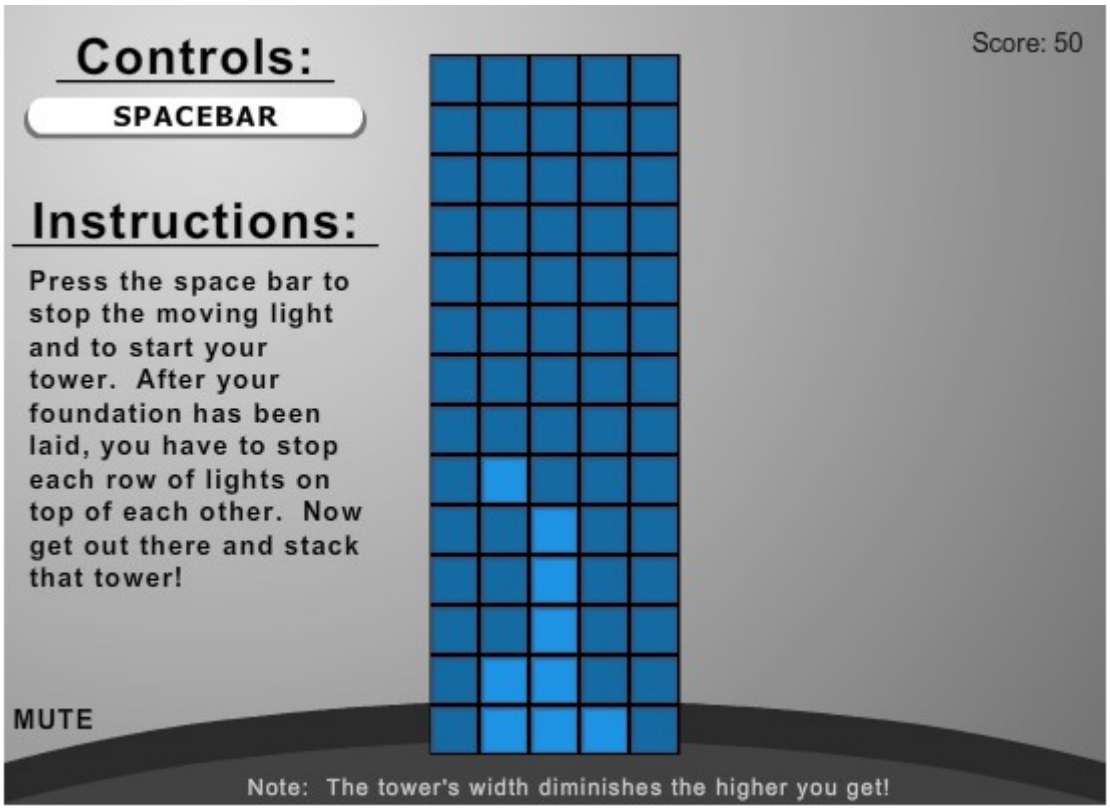
Para la modalidad classic stacker he encontrado sólo dos juegos parecidos:

Android game Stacker



Está desarrollado para la plataforma Android y se puede encontrar en **GooglePlay**. Se basa en lo mismo, es decir, el juego acaba cuando el usuario ha perdido todos los cuadrados disponibles que tiene para apilar sobre los anteriormente apilados. Sólo existe para la plataforma android.

Stacker Flash



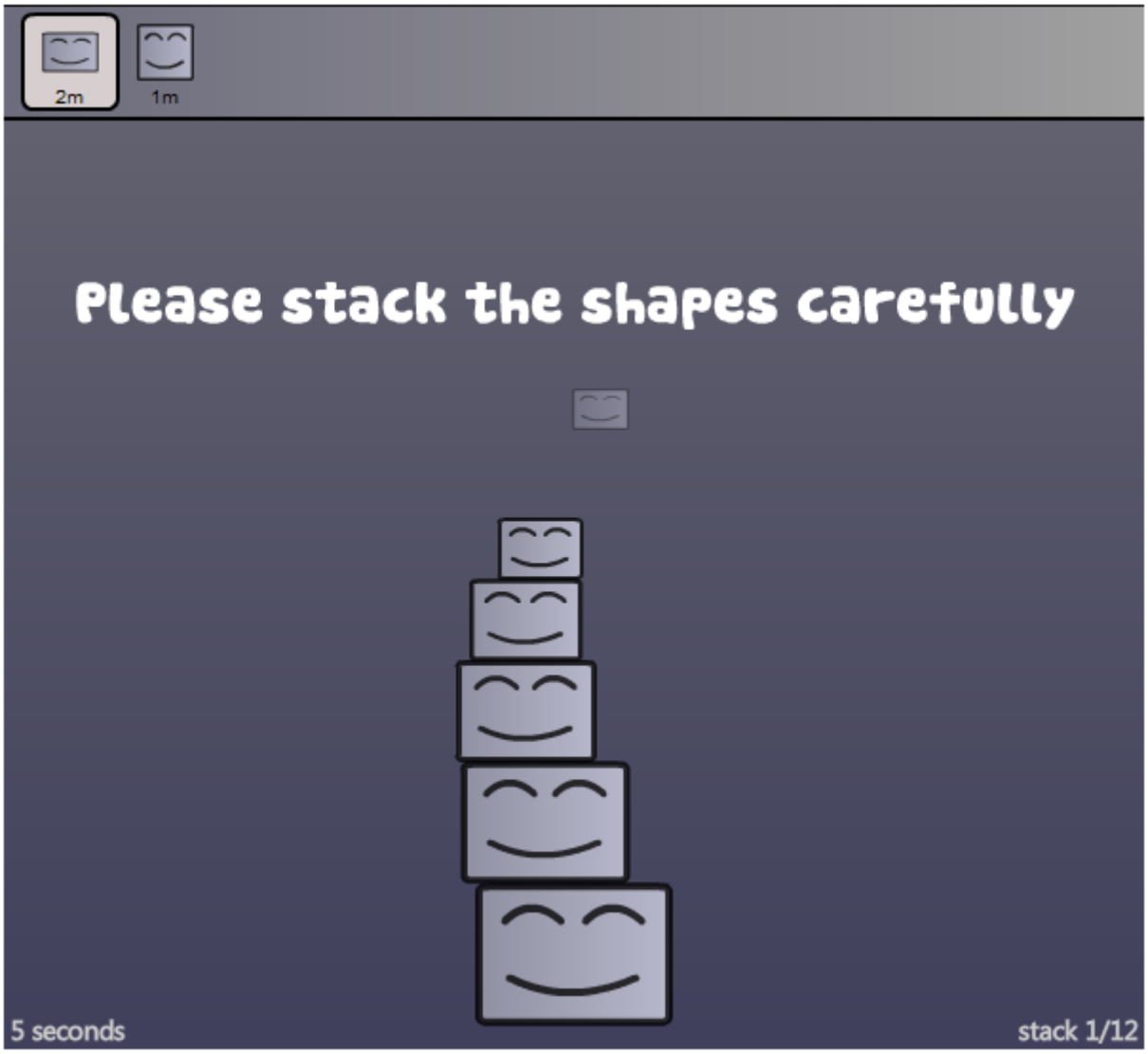
Esta basado en su totalidad al stacker original, es decir, el juego llega a su fin cuando el usuario ha alcanzado el punto más alto con al menos un cuadrado o cuando el usuario ha perdido todos los cuadrados. La velocidad de movimiento del conjunto a apilar va aumentando conforme se va alcanzando una altura mayor.

El juego esta desarrollado en flash y no he sido capaz de hacerlo reproducir en dispositivos móviles ni en tablets con los navegadores convenciones tales como chrome, firefox o android browser.

Link → **http://online.brothergames.com/flash-games/stacker.html**

Para la modalidad super stacker he encontrado sólo un juego parecido:

Super Stacker



Este juego utiliza tecnología flash para realizar las animaciones y un motor físico que permite jugar con las colisiones y la gravedad. La estructura formada por el usuario ha de aguantar un número de segundos para pasar al siguiente nivel, aumentando con ello la dificultad según se avanza.

En ninguno de los dispositivos móviles y tablets para IOS y Android que se han probado no ha funcionado, cosa que no pasa con el super Stacker desarrollado en este proyecto utilizando CANVAS.

Link→http://www.onemorelevel.com/game/super_stacker

2. Metodología y etapas de trabajo

Para el desarrollo del proyecto se ha seguido el modelo en cascada, siendo fácil su planificación desde un principio ya que se tiene claro antes de comenzar los requisitos necesarios para llevar a cabo el desarrollo.

El modelo en cascada se caracteriza por seguir un estricto orden para el desarrollo de software divididas en etapas, las cuales se comienzan de forma independiente y siempre y cuando se haya acabado la etapa anterior.

Las etapas de las que consta la metodología de desarrollo en cascada se dividen en:

- Análisis de requisitos.
- Diseño del sistema.
- Diseño del programa.
- Codificación.
- Pruebas.
- Implantación
- Mantenimiento.

De esta manera, los errores a lo largo del desarrollo afectan a cada etapa de manera individual, siendo fácil localizar en qué etapa se ha desencadenado el suceso y el impacto consecuente para su manejo, siendo incluso necesario un re-diseño que puede llegar a afectar a la codificación.

Etapas del trabajo

El actual proyecto ha constado de distintas etapas, las cuales enumeramos a continuación.

Análisis del entorno global

En esta etapa se evalúan las distintas opciones existentes en el mercado (estado del arte) además de dar comienzo a una auto-formación con respecto a la tecnología que se pretende utilizar para averiguar si es una opción a tener en cuenta o simplemente es motivo de descarte. Es necesario también investigar que herramientas existen para dicha tecnología que faciliten la implementación del proyecto.

Análisis y Diseño del proyecto

Se estudian todo lo necesario para que el usuario disfrute de una buena experiencia, estudiando posibles casos de uso y ayudándonos a elegir una interfaz amigable diseñando una serie de bocetos para cada vista.

Análisis de los recursos del software

Pequeña investigación acerca de los diferentes lenguajes de programación y de etiquetado que nos podrían ser útiles a la hora de desarrollar el proyecto de manera eficiente e intentado adaptar las etapas de análisis y diseño.

Implementación del proyecto

Se procede a la codificación una vez que se hemos completado las etapas anteriores, es decir, los requisitos, diseño, análisis y herramientas hardware/software están claros. Si alguna de las etapas anteriores no encaja en esta etapa, será necesario re-definir la etapa o etapas afectadas en cuestión.

Prueba y mantenimiento de proyecto

Las pruebas se realizan conforme el proyecto avanza con la implementación, centrándonos en los detalles que harán la aplicación su punto fuerte, tales como la jugabilidad o el responsive design. Terminada la fase de pruebas será necesario llevar a cabo el mantenimiento del proyecto, mejorando la eficiencia y calidad tanto de la aplicación resultante como del código generado.

Documentación

Una vez que todas las etapas anteriores han sido completadas, realizamos un despliegue detallado en una memoria en la cual se tratan todos los detalles que conciernen al proyecto actual, tales como casos de usos, competencias, etc.

3. Competencias a cubrir

En el siguiente punto de la memoria realizamos una justificación de las competencias que se han cubierto con este Trabajo Fin de Grado.

CII01

Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación vigente.

Esta competencia ha sido cubierta durante las distintas etapas que se han afrontado a lo largo del proyecto y que se han expuesto en el punto 2 anterior (Etapas del trabajo)

CII02

Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

La finalización de este proyecto de una manera eficiente obteniendo el producto final de la manera que se esperaba (cumpliendo todos los requisitos) y en los tiempos esperados son la prueba necesaria para dar a entender que se ha llevado una planificación coherente, además de la puesta en marcha de manera eficaz, dando así por cubierta esta competencia.

En el apartado de “Aportaciones” se realiza un análisis de del impacto económico y social.

CII04

Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.

Será necesario cubrir ciertos aspectos tales como **Ámbito geográfico, fiabilidad y entorno de pre-explotación**. Teniendo en cuenta que la aplicación resultante es accesible a través de internet desde cualquier parte utilizando un navegador web que soporte la tecnología desarrollada, su comportamiento es estable y su puesta en marcha en cualquier entorno no requiere grandes prestaciones ni software propietario, damos por cubierta esta competencia.

CII18

Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

En el apartado Normativa y Legislación haremos una pequeña descripción de las leyes que afectan a este TFG, quedando cubierta esta competencia.

TFG01

Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

La documentación, código generado y aplicación final resultante es la prueba de que esta competencia se ha logrado de manera satisfactoria cumpliendo los requisitos tecnológicos de la ingeniería informática.

4. Normativa y Legislación

A continuación tratamos una serie de normativas que afectan a este trabajo fin de grado.

Hay que resaltar que en el estado actual de la aplicación sólo se hace uso exclusivo del lado del cliente y, por tanto, no se almacena ninguna información en ningún servidor externo ni se utilizan cookies propias ni de terceros (todo dato guardado se hace utilizando **localstorage**). Por tanto, obviaremos las leyes que afectan a estas especificaciones.

Software Libre

«Software libre» es el software que respeta la libertad de los usuarios y la comunidad. En grandes líneas, significa que **los usuarios tienen la libertad para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software**. Es decir, el «software libre» es una cuestión de libertad, no de precio .

Un programa es software libre si los usuarios tienen las cuatro libertades esenciales:

•La libertad de ejecutar el programa como se desea, con cualquier propósito (libertad 0).
•La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que usted quiera (libertad 1). El acceso al código fuente es una condición necesaria para ello.

•La libertad de redistribuir copias para ayudar a su prójimo (libertad 2).

•La libertad de distribuir copias de sus versiones modificadas a terceros (libertad 3). Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello.

Un programa es software libre si otorga a los usuarios todas estas libertades de manera adecuada. De lo contrario no es libre. Existen diversos esquemas de distribución que no son libres, y si bien podemos distinguirlos en base a cuánto les falta para llegar a ser libres, nosotros los consideramos contrarios a la ética a todos por igual.

Este trabajo fin de grado se ha basado en herramientas consideradas como software libre, es decir, no es software privativo ni tiene un costo estipulado.

El software utilizado es el siguiente:

- Google Chrome
- Mozilla Firefox
- Opera Browser
- Notepad++
- Opera Mobile Emulator
- Bootstrap 3
- Box2dweb
- easejs
- phonegap
- OpenOffice

Ley de Propiedad Intelectual.

El TFG también está protegido por la Ley de Propiedad Intelectual (LPI). La **propiedad intelectual** es toda creación del intelecto humano y está regulada por el texto Refundido de la Ley de Propiedad Intelectual, aprobado por Real Decreto Legislativo 1/1996, de 12 de abril. Los derechos de propiedad intelectual protegen los intereses de los creadores al ofrecerles prerrogativas en relación con sus creaciones y también la retribución económica que les corresponde por la realización de sus obras y prestaciones.

Además la propiedad intelectual tiene que ver con las creaciones de la mente: las invenciones, las obras literarias y artísticas, los símbolos, los nombres, las imágenes y los dibujos y modelos utilizados en el comercio.

5. Aportaciones

Una de las aportaciones más importantes a las que ha contribuido este proyecto es el uso de la tecnología CANVAS para HTML5. Actualmente, el uso de tecnología flash esta limitado a muchos dispositivos que no pueden instalar dicho plugin. En este aspecto CANVAS sale ganando, ya que ofrece una potencia de visualización que no tiene nada que envidiar a la potencia que suministra la tecnología flash y no presenta problemas en cualquier dispositivo que tenga un navegador que soporte HTML5, que son prácticamente los más usados en la actualidad.

Como aportación social, el juego trata de ofrecer un sistema de entretenimiento orientado a cualquier público de manera gratuita. A partir de las distintas modalidades de juego sería posible como mejora futura ofrece una serie de niveles extras, vendiéndose como niveles complejos al alcance de pocos usuarios con el fin de promover un modelo económico.

Por último, considero importante el hecho de fomentar tanto dentro como fuera de la ULPGC el uso de nuevas tecnologías tales como HTML5 y CSS3, utilizando diseño adaptable en la medida de lo posible.

6. Requisitos Hardware/Software

Para la realización del proyecto basta con tener acceso a un ordenador. La conexión a internet no es necesaria para el funcionamiento de la aplicación, pero si para conseguir todo el software libre necesario que hará falta para llevar a cabo toda la implementación y las pruebas de la aplicación.

Es posible trabajar en cualquier sistema operativo orientado a Escritorio (cualquiera de las distribuciones de Windows, Linux y MacOSX).

Se trabajará solo con el Front-End, es decir, no será necesario instalar ningún servidor web a menos que se quiera probar la aplicación remotamente utilizando un equipo como servidor.

No será necesario ningún dispositivo hardware especial, tanto para desarrollar como para la puesta en marcha.

En lo que concierne al Software, las herramientas utilizadas han sido las siguientes:

Notepad++

Se trata de un editor de texto de código fuente libre y que soporta varios lenguajes de programación. Las últimas versiones de esta aplicación contienen herramientas muy útiles tales como el auto-relleno en lenguajes como Javascript, HTML y CSS, que serán la base de la aplicación.

Esta aplicación ha sido la utilizada para codificar prácticamente la totalidad del código generado. Ofrece un sistema eficaz de localización de métodos, búsqueda y reemplazado de palabras.

Navegadores Web(Opera, Chrome, Firefox, Explorer, Android Browser)

El navegador web será la herramienta encargada de lanzar la aplicación generada. Los distintos navegadores han servido no sólo para realizar las pruebas funcionales de la aplicación, sino que gracias a las herramientas de depuración de cada una de ellas ha permitido facilitar la implementación, tales como firebug de mozilla y herramienta de depuración de chrome/opera (inspector, consola y debugger)

Hay que mencionar que algunas versiones antiguas de algunos navegadores tales como Internet Explorer 8 no serán capaces de reproducir HTML5 y, por tanto,

no será capaz de lanzar ninguna de las modalidades de juego. A partir de la versión 10 Internet Explorer empezó a dar soporte para HTML5 y CSS3. El resto de versiones de Chrome, firefox y opera no deberían tener problemas de visualizado CANVAS, al igual que android browser.

Algunas versiones obsoletas de cualquier navegador, incluso soportando HTML5, podrían presentar problemas a la hora de mostrar ciertos estilos CSS3, afectando a la visualización de la web resultante.

Opera Mobile Emulator

Esta aplicación nos ayudará a visualizar cómo se muestra la web resultante en varios modelos distintos de tablets y móviles, emulando el navegador web opera mobile en cada uno de ellos.

Opera mobile emulator es muy parecido a rippey de Google Chrome. Ambos ofrecen una lista de dispositivos con distintos tamaños de pantalla que afectarán de una manera u otra a los estilos que aplicamos (interfaz de usuario).

Phonegap Build

Phonegap build es una aplicación basada en la nube que permitirá crear una app para Android, IOS y Windows Phone a partir de código fuente orientado a web.

Esta herramienta la utilizaremos para generar un archivo .apk que será una aplicación de Android sin necesidad de desarrollar de forma nativa en dicha plataforma.

Phonegap build permite utilizar gratuitamente,con una cuenta de usuario, su servicio de forma limitada, pudiendo trabajar sólo con un proyecto privado al mismo tiempo. En caso de ir mas allá, habría que desembolsar una cantidad económica obteniendo por ello más y mejores servicios. Para este caso sólo será necesario utilizar una cuenta con el servicio gratuito. Durante la etapa de pruebas vamos a hablar un poco más sobre este servicio.

Easeljs

EaselJS ofrece soluciones directas para trabajar con gráficos y con HTML5 Canvas. Proporciona una API que abarca muchas características de Javascript.

Easeljs ofrece una lista completa de visualización jerárquica , un modelo de interacción y las clases de ayuda para facilitar el trabajo con CANVAS mucho más fácil.

Este framework ayudará a implementar la modalidad de juego super stacker creando lo que se denomina “stage”, originando con ello un sistema que ayuda a actualizar constantemente los objetos que interactúan en el mundo creado.

En la fase de implementación se entrará en detalles, sobre todo a la hora de integrarlo con box2dweb.

Box2dWeb

Esta API será de suma importancia para realizar la modalidad de juego super stacker. Se trata de un motor físico que ayudará a simular colisiones entre objetos, gravedad, fricción o densidad de los objetos que formarán parte de el mundo que se creará.

La API ha sido portada a javascript para ser utilizada con CANVAS. Originalmente está hecha para C++, aunque también fue portada para FLASH.

Este motor físico ha sido implementado en el desarrollo de juegos como Angry Birds, el Crayon Physics Deluxe o el tiny wings.

Durante la etapa de implementación vamos a centrarnos en los detalles de box2dweb, es decir, cómo funciona y como se ha aprovechado esta API para el desarrollo de una de las modalidades de juego que se ofrecen.

JQuery

Como bien es conocido, Jquery es un framework para javascript que facilita la lectura, escritura y modificación de los elementos del DOM, además de suministrar infinidad de funciones que ayudarán a dar forma y lógica a la interfaz del usuario.

Se utilizará en ocasiones puntuales para agilizar la codificación de la aplicación.

Bootstrap 3

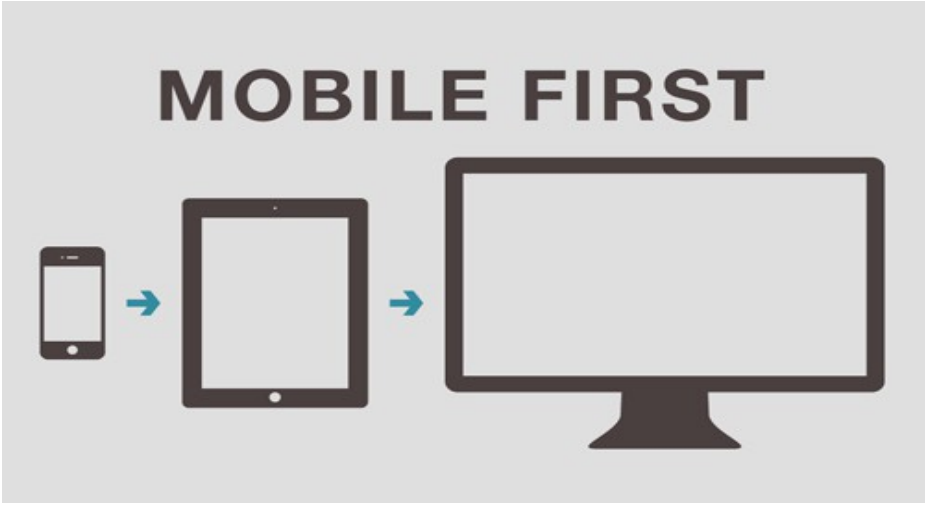
Bootstrap es un framework para CSS creado por Twitter para facilitar la maquetación web. Suministra una serie de clases que ayudarán a la creación de contenido web adaptable a dispositivos móviles.

Para esta versión de bootstrap, la filosofía seguida ha sido la de “***mobile first***”, que consiste principalmente en diseñar primero para el móvil, y luego ya adaptar ese diseño principal a la visualización en tablets, grandes pantallas,etc, mediante el responsive design.

Con esta filosofía se trata de llegar a más personas, y también tiene como uno de sus objetivos centrar el contenido del diseño en algo básico y funcional, ya que cuando perdemos hasta el 80% de la pantalla, priorizamos y optimizamos contenido al nuevo contexto.

Un ejemplo de mobile first es la nueva tienda lanzada por Amazon para el mercado indio: Junglee, ya que en la India más de la mitad de los accesos a internet se realizan a través de los móviles.

A medida que el panorama digital se hace más complejo, es necesario cambiar las estrategias para poder diseñar experiencias que se adapten y funcionen en toda la gama de dispositivos digitales, y con los conceptos de responsive design y mobile first se puede innovar en muchos sentidos..



OpenOffice

Herramienta de ofimática gratuita que ha ayudado al desarrollo de esta memoria, tanto para la generación de todo su contenido como para la creación de pdf.

Lenguajes de Programación

Javascript

El núcleo principal de la aplicación está escrito en Javascript. Con este lenguaje nos encargaremos de manejar todo lo que sucede en el entorno CANVAS en las dos modalidades de juego.

Para la modalidad classic Stacker, el código Javascript generado se ha utilizado sin la ayuda de ningún framework. Los método y las clases creadas para la correcta visualización, ejecución y funcionalidad del juego ha sido codificada utilizando javascript nativo.

Para la modalidad super stacker sí se ha utilizado APIs debido a la complejidad que planteaba la jugabilidad propuesta. El código resultante es una mezcla de métodos suministrados por las APIs easeljs y box2dWeb que han facilitado muchísimo la codificación del mismo, además de código nativo javascript y jquery.

HTML

HTML es un lenguaje de marcado cuya función es la de estructurar un documento utilizando etiquetas e incluso especificando tipos de datos, tipos de documentos y pre-carga de archivos que ayudarán a la formación de tal documento.

Su uso está muy extendido y, con la incorporación de la versión 5 ha dotado a la web de mayor potencia con etiquetas tales como <video> o <canvas>, que será la base de esta aplicación.

CSS

CSS, al igual que HTML, es un lenguaje de marcado. Su función final es la de proporcionar estilos a los elementos HTML o parecidos que forman la estructura de un documento. Juega un papel importante en la web, y no será menos para esta aplicación ya que, con el nuevo impacto de los dispositivos móviles, su uso ha cobrado muchísima importancia.

Se utilizará CSS3 en la medida de los posible para dota efectos llamativos y atractivos al usuario final, además de asegurar un uso adaptable con bootstrap 3.

Un poco de canvas

Canvas (lienzo en inglés) es un elemento **HTML** incorporado en **HTML5** que permite la generación de gráficos dinámicamente por medio del scripting. Permite generar gráficos estáticos y animaciones. Cualquier texto dentro de este elemento se mostrará solo en los navegadores que no soportan el canvas.

Fue implementado por Apple para su navegador Safari. Más tarde fue adoptado por otros navegadores, como Firefox a partir de su versión 1.5, Opera, y es adoptado por WHATWTG.

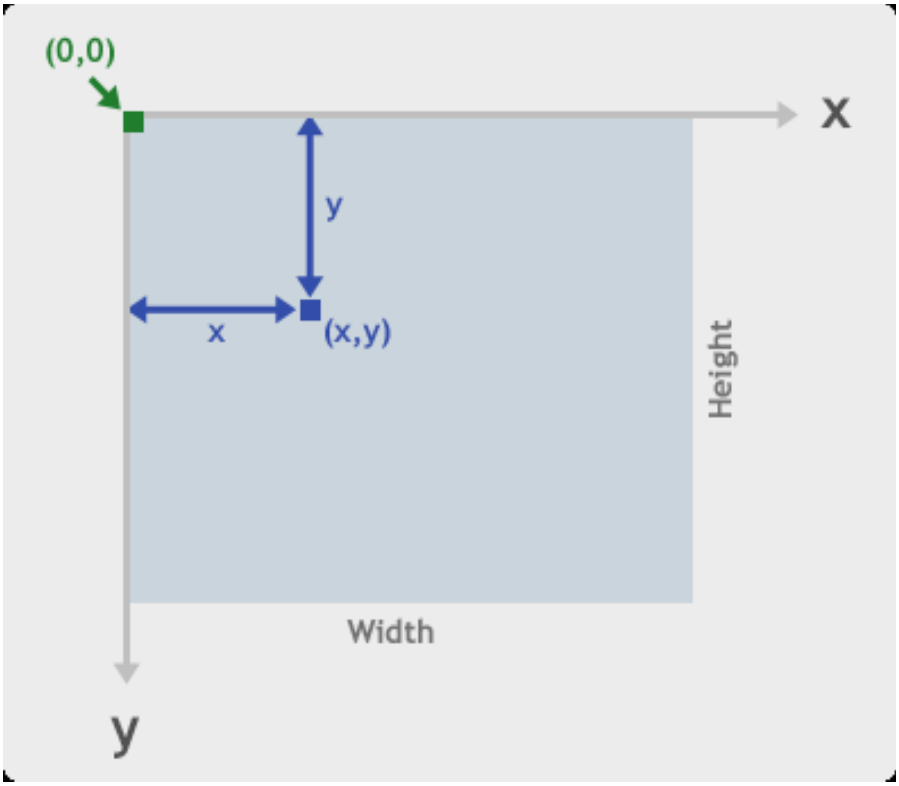
El objeto canvas puede ser accedido a través de Javascript permitiendo generar gráficos 2D, juegos, animaciones y composición de imágenes.

Usar el elemento <canvas> no es algo muy díficil pero es necesario saber y entender los aspectos básicos del HTML y JavaScript. El elemento <canvas> no está soportado en navegadores viejos, pero están soportado en la mayoría de las versiones más recientes de los navegadores. El tamaño por defecto del canvas es 300px * 150px [ancho (width) * alto (height)]. Pero se puede personalizar el tamaño usando las propiedades height y width de CSS. Con el fin de dibujar gráfico en el lienzo<canvas> se utiliza un objeto de contexto de JavaScript que crea gráficos sobre la marcha.

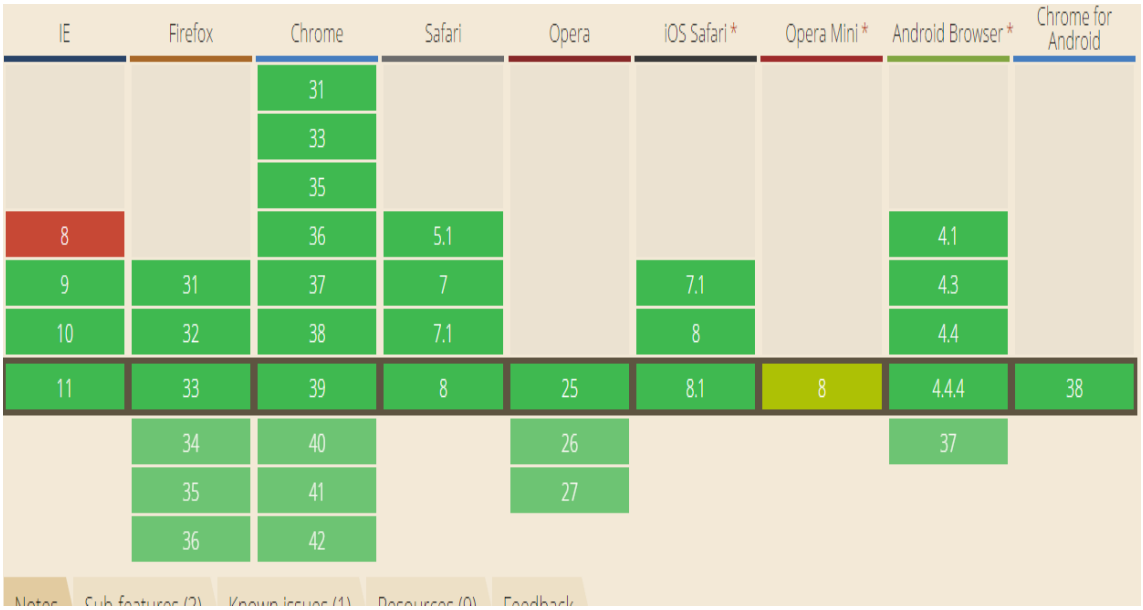
Existen infinidad de API´s para tratar con el elemento CANVAS de tal manera que es posible generar gráficos 3D, como por ejemplo K3D (http://www.kevs3d.co.uk/dev/canvask3d/k3d_test.html). Ofrece un buen rendimiento en comparación con otros elementos que son utilizados con el mismo fin, como por ejemplo la etiqueta SVG que utiliza información del DOM para generar gráficos vectoriales y tratar con ellos.

El lienzo CANVAS representa en sí un bitmap (mapa de bits) en el cual se puede dibujar a través de script. Esto hace que su curva de aprendizaje resulte un tanto costosa, ya que muchas de las figuras o formas que se quieran representar han de ser re-calculadas según coordenadas del lienzo de manera continua, sobre todo cuando se quiere llevar a cabo el desarrollo de un juego.

Otro punto a tener en cuenta es que no es posible interactuar con los elementos que se dibujan sobre el lienzo de una manera inmediata, es decir, si dibujamos un cuadrado no será posible obtener una referencia de manera sencilla hacia dicha figura, sino que será necesario trabajar con las coordenadas donde se encuentre, de tal manera que es sumamente importante entender con qué coordenadas estamos trabajando en el lienzo.



Al ser un elemento característico de HTML5, algunos navegadores no mostrarán su contenido. Podemos observar en la siguiente imagen a partir de qué versiones de navegadores se soporta el elemento. Los navegadores más utilizados a nivel mundial no deberían tener problemas a la hora de ejecutar CANVAS, como por ejemplo CHROME, que como podemos observar, soporta CANVAS en todas sus versiones de escritorio. Algunos como Internet Explorer o Firefox necesitan polyfills para simular su visuzalización.



Es importante conocer las limitaciones de CANVAS. Como hemos mencionado anteriormente, CANVAS es la opción que más se suele usar en casi todos los juegos HTML5. Es simple y rápido, sobre todo en los juegos con muchos objetos. Con todo, tiene sus limitaciones:

-Los objetos dibujados forman parte del mapa de bits del lienzo, de modo que no se pueden asociar a un controlador de eventos. Así, por ejemplo, si haces clic en un círculo específico en el lienzo, no se desencadenará ningún controlador de eventos concreto de forma nativa. Lo que sí puedes hacer es conectar un controlador de eventos a todo el lienzo; esto te permitirá calcular (mediante algunas operaciones matemáticas) si se ha hecho clic o no en un círculo determinado, pero el explorador no proporcionará este comportamiento de forma nativa.

•Cuando un objeto se mueve por el lienzo, hay que volver a dibujar todo el lienzo por cada incremento de movimiento.

•El diseño fluido es relativamente difícil: el explorador no puede ajustar el tamaño del lienzo de forma nativa en función del tamaño actual de la ventana del explorador. De todas formas, puedes recurrir a la programación para crear este comportamiento, consistente en escuchar los cambios en el tamaño de la ventana y ajustar el tamaño del lienzo en consecuencia.

•El explorador no puede convertir de forma nativa las coordenadas de pantalla al sistema de coordenadas mundial de tu juego. Así, el explorador no podrá convertir la posición de un clic del mouse (en coordenadas de pantalla) de forma nativa a un sistema de coordenadas de juego cómodo. En este caso, puedes crear ecuaciones de transformaciones.

7. Desarrollo

Este apartado vamos a dividirlo en base al modelo en cascada que hemos seguido hasta el momento, es decir, seguiremos el orden clásico del modelo en cascada : **Análisis, diseño, codificación, pruebas y mantenimiento.**

Etapa 1: Análisis

En esta etapa vamos a dejar lo más claro posible lo que el usuario necesita para disfrutar de una buena experiencia con la aplicación, es decir, haremos un ***análisis de requisitos*** y veremos a través de distintos casos de uso como ha de comportarse la aplicación, además del flujo normal a seguir según la elección.

Requisitos de usuario

- El usuario dispondrá de una pantalla en la cual podrá escoger la modalidad de juego que quiere jugar sin necesidad de login.

Classic Stacker

- Al entrar en la modalidad de classic stacker, ha de tener la opción de jugar, ver las mejores puntuaciones o volver a la pantalla principal.

- Cuando acceda a la pantalla de jugar en classic staccker, podrá volver hacia atrás, volver a la pantalla principal o jugar hasta que acabe su partida.

- Si la partida acaba, se ha de dar la opción de volver a intentarlo, de ver las mejores puntuaciones hasta el momento o de salir.

- Cuando acceda a la pantalla de mejores puntuaciones podrá visualizar las mejores puntuaciones hasta el momento además de volver hacia atrás o a la página principal

Super Stacker

- Al entrar en esta modalidad, las opciones disponibles han de ser jugar, niveles (pantalla que muestra todos los niveles que puede jugar), mejores tiempos y volver a la pantalla principal.

- La opción de jugar le permitirá empezar una partida desde el nivel 1.

- Cuando acceda a la pantalla de niveles, podrá escoger aquellos niveles que están disponibles, siendo el máximo el último al que se llegó, además de volver a las pantalla de los menús.

- Cuando acceda a la pantalla de mejores tiempos visualizará los mejores tiempos invertidos por cada nivel superado.

- Si elije la opción de jugar, ha de poder volver a intentarlo en caso de perder o pasar al siguiente nivel en caso de superar el actual.

Reglas del juego

Classic stacker : El usuario ha de apilar una fila de cuadrados que se desplazan horizontalmente sobre otra fila de cuadrados que previamente había apilado. Si el usuario no consigue apilar la totalidad de la fila, perderá la porción no apilada, quedando una fila más pequeña. Si el usuario consigue apilar al menos un cuadrado sobre la estructura total llegando a la máxima altura, se pasará al siguiente nivel y se le otorgará un cuadrado más la fila si ha perdido alguno anteriormente, además de aumentar la velocidad de desplazamiento de la fila de cuadrados que tiene que apilar. Según pasan los niveles la velocidad de movimiento aumenta y, por tanto, la dificultad de apilar. El juego acaba cuando el usuario ha perdido todos los cuadrados de la fila en movimiento.

Super stacker: Se plantea un escenario en el cual hay unos límites y una serie de figuras estáticas repartidos por alguna parte de dicho escenario. El jugador tiene que apilar todas las figuras de distintas formas que se le van planteando utilizando las figuras estáticas que existen en el escenario. Las figuras que ha de apilar son sensibles a la gravedad, colisión entre objetos, fricción y demás fuerzas que son generalmente afectadas por un objeto común. Si alguna de estas figuras entra en contacto con alguno de los limites, el jugador pierde, pudiendo volver a intentarlo. Si el jugador consigue apilar todas las figuras, la estructura resultante ha de aguantar durante un número determinado de segundos. Si lo consigue habrá superado el nivel, pudiendo pasar al siguiente aumentando con ello la dificultad de la prueba. Se contabilizará el número de segundos que tarda el usuario en apilar todas las figuras. Si es la mejor marca formará parte de la sección “mejores tiempos” para el nivel actual.

Diagrama de casos de uso

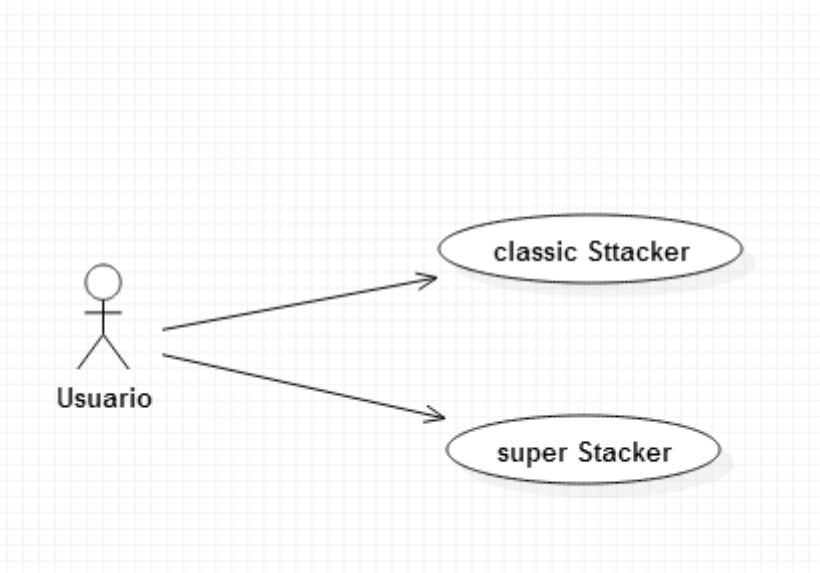
Vamos a desarrollar una serie de diagramas de casos de uso que nos van a permitir interpretar mejor la relación existente entre un usuario con las distintas partes de la aplicación. Los diagramas de casos de uso son fácilmente comprensibles. Constan de los siguientes elementos:

Actores : representan a los usuarios del sistema con un rol especifico. Según el rol que representa tendrá acceso a partes de la aplicación o no. No tiene por qué ser una persona en concreto, sino que pueden ser servicio o incluso máquinas. En la aplicación no habrá distinciones entre usuarios ni roles. La aplicación es accesible para cualquier usuario.

***Caso de uso*:** describe una funcionalidad de la aplicación.

Los siguientes casos de uso a mostrar estarán basados en el ***flujo normal de la aplicación***.

- Selección de modalidad de juego



classic Stacker:

Precondición : ninguna.

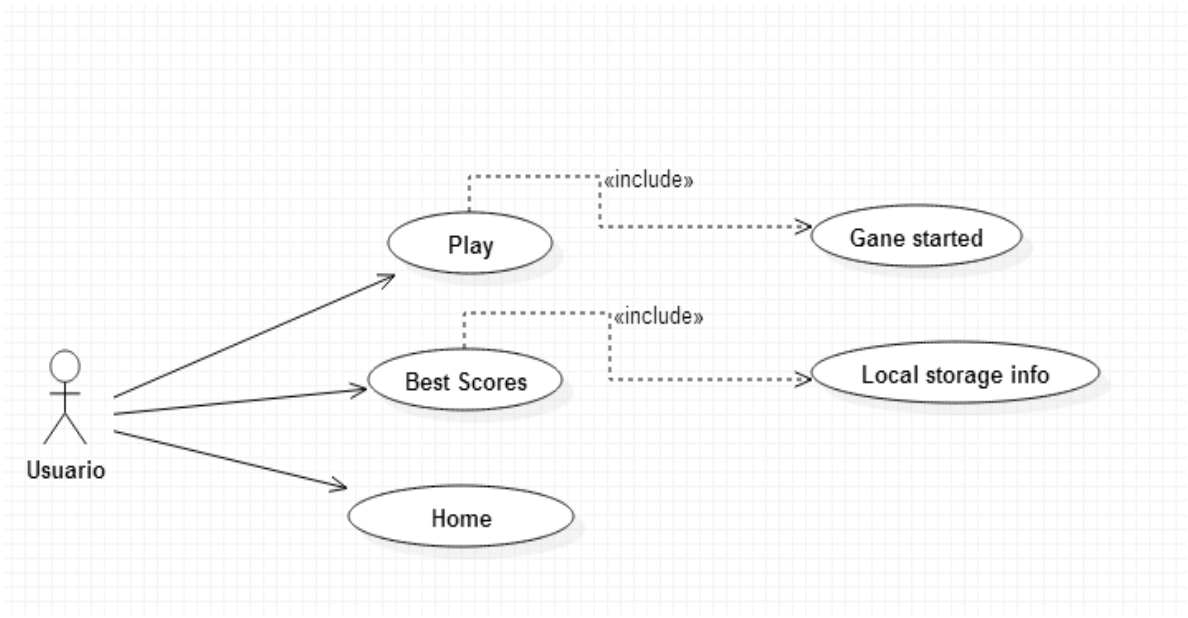
Post-Condición : El usuario accede al menú de la modalidad classic stacker.

super Stacker:

Precondición : ninguna.

Post-Condición : El usuario accede al menú de la modalidad super stacker.

- Menú modalidad classic Stacker



Play:

Precondición : Elección modalidad classic.

Post-Condición : Acceso a la pantalla de juego modalidad classic con la partida comenzada.

B*est Scores:*

Precondición : Elección modalidad classic.

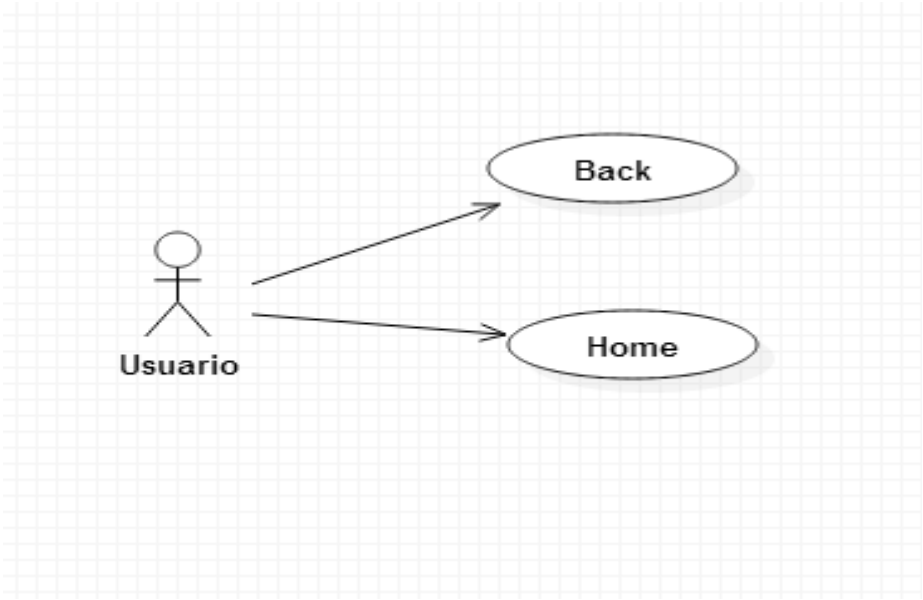
Post-Condición : Acceso a la pantalla de mejores puntuaciones. Visualización de los datos almacenados en local storage.

Home:

Precondición : Elección modalidad classic.

Post-Condición : Acceso a la pantalla principal de elección modalidad de juego.

-Jugar a classic stacker



Back:

Precondición : Elección jugar classic stacker.

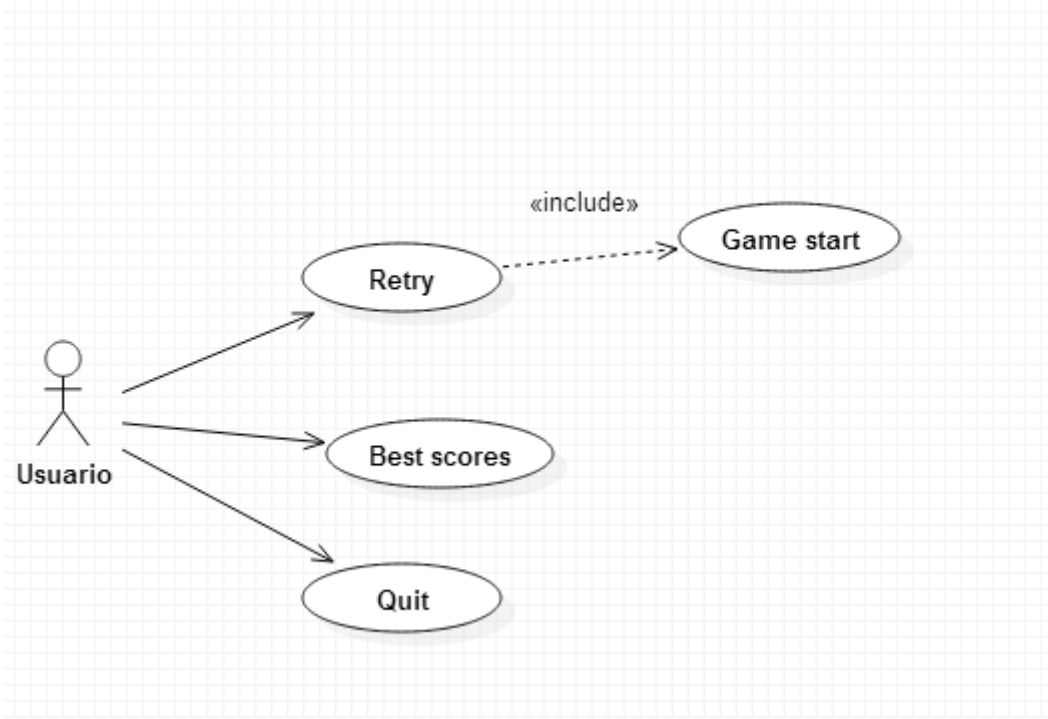
Post-Condición : Volver al menú de classic Stacker.

Home:

Precondición : Elección jugar classic stacker.

Post-Condición : Acceso a la pantalla principal de elección modalidad de juego.

-Game Over classic Stacker



Retry:

Precondición : La partida ha terminado.

Post-Condición : La partida empieza desde el principio.

Best Scores:

Precondición : La partida ha terminado.

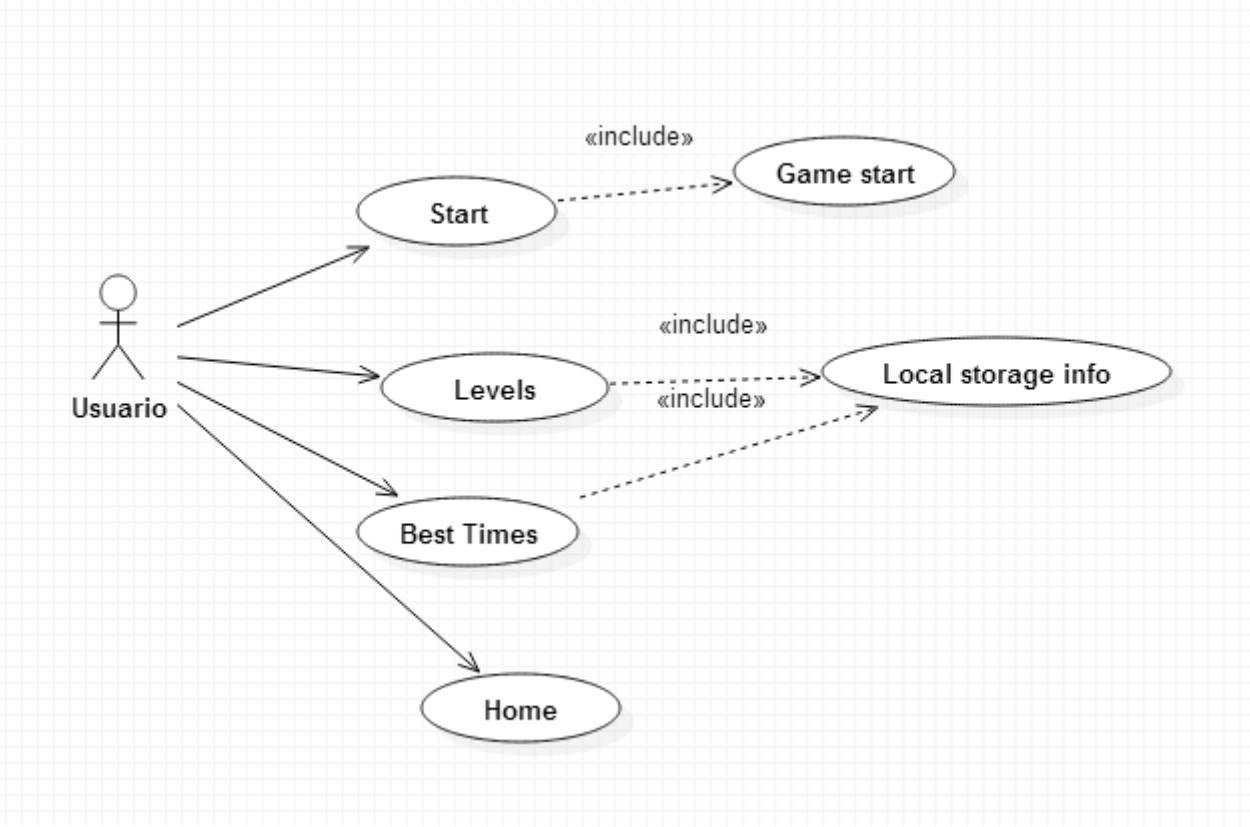
Post-Condición : Acceso a la pantalla de mejores puntuaciones.

Quit:

Precondición : La partida ha terminado.

Post-Condición : Acceso a la pantalla principal de elección modalidad de juego.

-Menú super Stacker



Start:

Precondición : Acceder al menú super stacker.

Post-Condición : La partida empieza desde el principio (level 1).

Levels:

Precondición : Acceder al menú super stacker.

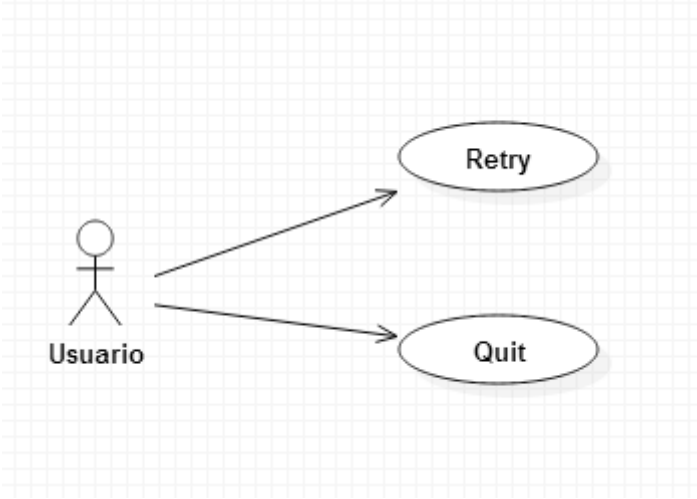
Post-Condición : Acceso a la pantalla de elección de niveles hasta el nivel mas alto conseguid hasta el momento.

Best times:

Precondición : Acceder al menú super stacker.

Post-Condición : Acceso a la pantalla principal de elección modalidad de juego.

-Menú partida perdida



Retry:

Precondición : La partida ha terminado.

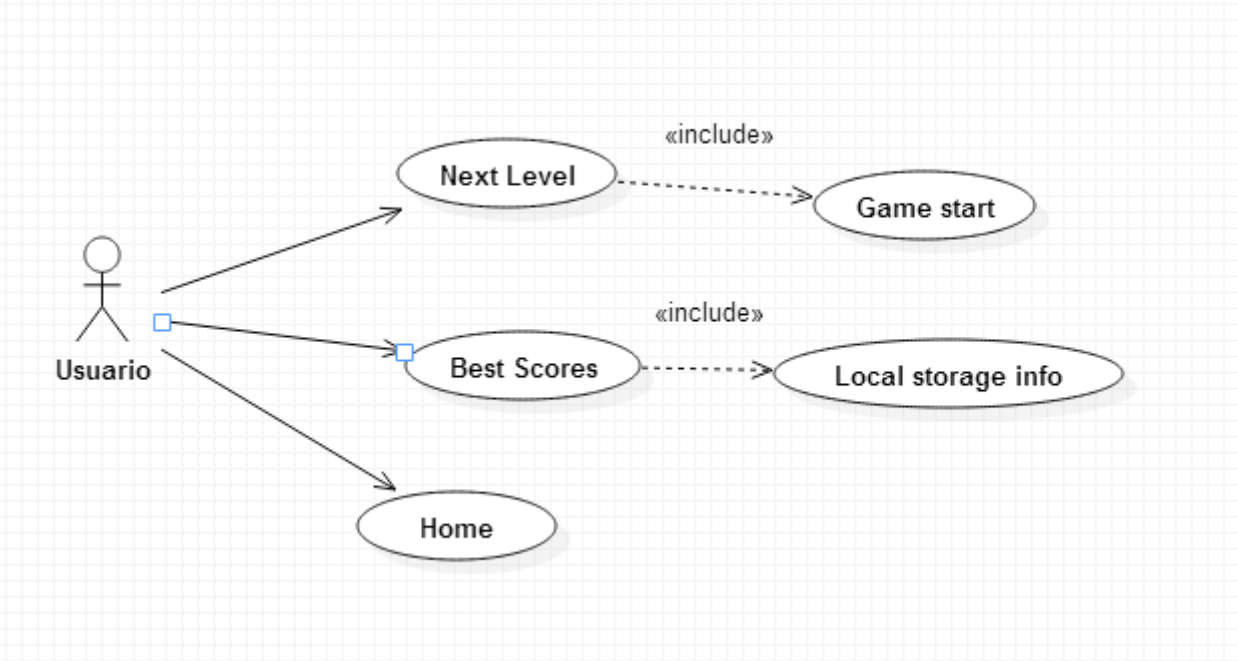
Post-Condición : La partida empieza desde el principio.

Quit:

Precondición : La partida ha terminado.

Post-Condición : Acceso a la pantalla principal de elección modalidad de juego.

-Menú partida ganada



Next level:

Precondición : La partida ha terminado.

Post-Condición : La partida comienza con otro nivel distinto al actual, siempre y cuando haya un siguiente nivel disponible.

Best Scores:

Precondición : La partida ha terminado.

Post-Condición : Acceso a la pantalla de mejores tiempos por nivel.

Quit:

Precondición : La partida ha terminado.

Post-Condición : Acceso a la pantalla principal de elección modalidad de juego.

Etapa 2: Diseño

Una vez que la etapa de análisis ha quedado clara(requisitos de usuario, casos de uso, etc) pasamos a la fase de diseño, en la cual hablaremos de la arquitectura apropiada para este caso, además de proponer diseños visuales que ayudarán a entender mejor el desarrollo y la finalidad de la aplicación.

Arquitectura del sistema

La arquitectura de la aplicación esta basada puramente en el cliente, es decir con el estado actual de la aplicación, sólo será necesario que el usuario tenga en su poder los archivos correspondientes al proyecto para poner en marcha la aplicación, sin necesidad de realizar petición de datos hacia un servidor externo que modifique el flujo normal según los datos que recibe.

Es importante resaltar que lo lógico en todas las aplicaciones web es que llegue al usuario a través de un servidor, y en este caso puede ser así, realizando pues una petición HTTP desde un cliente a un servidor que le devolverá los archivos necesarios, destacando pues una arquitectura cliente-servidor.

Con respecto a la definición de arquitectura cliente/servidor se encuentran las siguientes definiciones:

- Cualquier combinación de sistemas que pueden colaborar entre si para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde esta ubicada.
- Es una arquitectura de procesamientos cooperativo donde uno de los componentes pide servicios a otro.
- Es un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red.
- El término cliente/servidor es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o más programas: una

aplicación y un servicio que lo soporte.

• IBM define al modelo Cliente/Servidor. "Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización , en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes', resultan en un trabajo realizado por otros computadores llamados servidores.

• "Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información".

Destacar que la aplicación de este proyecto no tendrá lógica de negocio ni existirá codificación Back-End y la utilización de un posible servidor será para que nos proporcione la web con todo lo que conlleva. En la parte del cliente se encuentra toda la lógica de las modalidades de juego, la generación de niveles, la lógica del motor físico, estilos, etc. Esto ha sido así por el empleo de CANVAS, que requiere un lenguaje del lado del cliente como lo es Javascript.

Diseño adaptable (Interfaces)

En esta etapa vamos a plantear un diseño de interfaz de usuario que resulte lo mas amigable y sencillo posible, orientada a cualquier público que desee jugar a cualquiera de las modalidades ofrecidas.

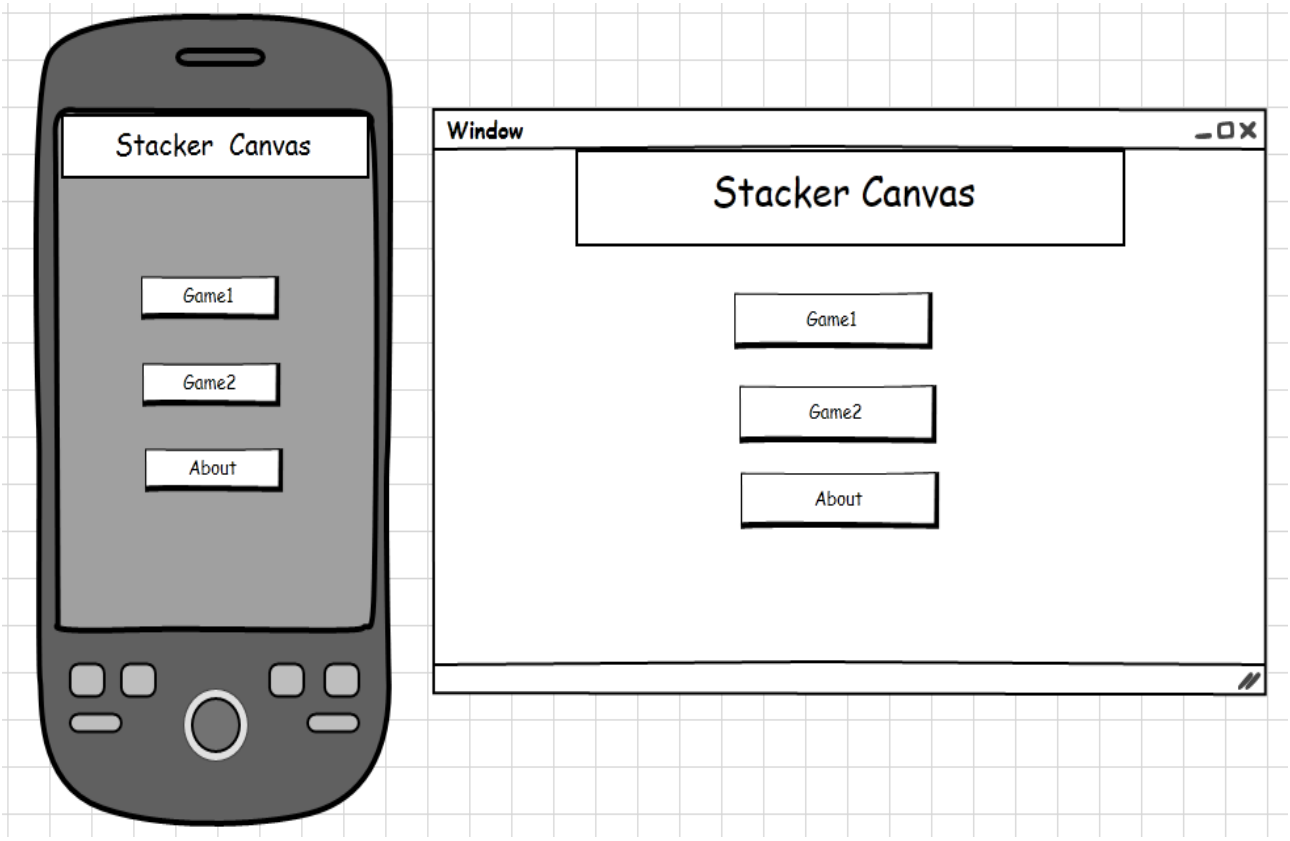
Para ello, hay que tener en cuenta que sea cual sea el tipo de usuario, éste puede acceder al contenido de la aplicación a través de cualquier dispositivo, ya sea a través del PC-Desktop, smartphone, tablets, etc, debido al auge de los sistemas operativos en los teléfonos móviles entre otras cosas. Es por ello que será necesario realizar un diseño de la aplicación teniendo en cuenta el tamaño de la pantalla del cliente solicitante, ya que no se visualizará de la misma manera en todos los dispositivos al variar el tamaño de su pantalla.

Basándonos en lo anterior, realizaremos unos bocetos que mostraremos a continuación teniendo en cuenta los tipos de pantallas más pequeños y partir desde ese punto hacia las pantallas más grandes. Esta filosofía también se conoce como ***mobile-first***, existiendo diferentes frameworks que trabajan con esta filosofía, como por ejemplo Bootstrap 3.

-Mockups

A continuación, vamos a mostrar algunos bocetos que servirán para basarnos en el resto de interfaces de la aplicación, ya que tendrán una estructura similar:

-Pantalla menú principal



Esta pantalla será la primera que vea el usuario por primera vez.

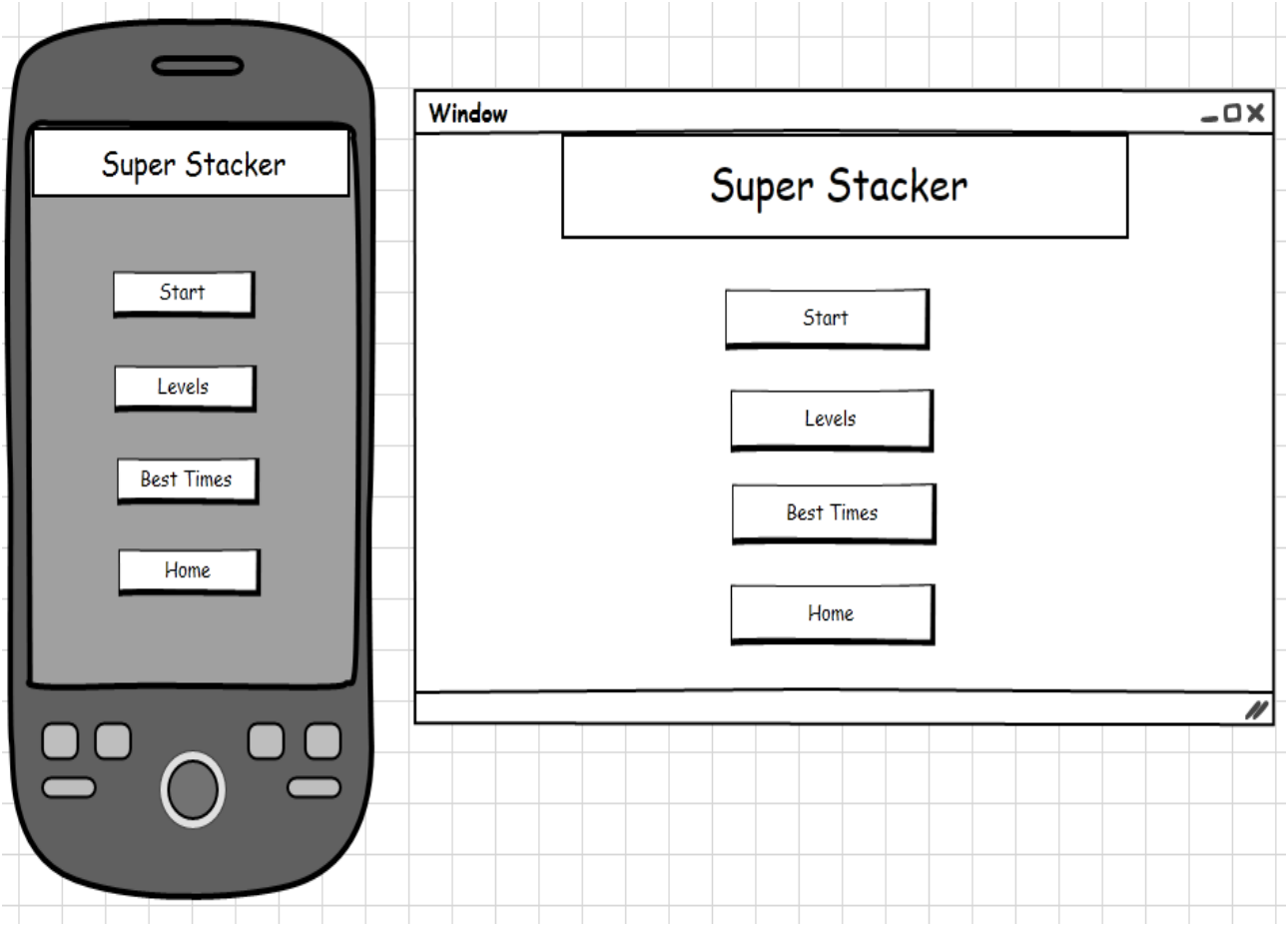
Se trata de hacerla lo más sencilla e intuitiva posible y, a simple vista, no debería haber grandes diferencias entre la versión PC y la versión móvil. Se intenta evitar el scroll en ambos casos.

Se puede observar hacía donde lleva cada opción:

Game 1 desplegará las opción de la modalidad 1, pasando los mismo con la modalidad 2.

About llevará a una pantalla de información acerca de la aplicación actual, como por ejemplo información sobre fecha y creación de la aplicación.

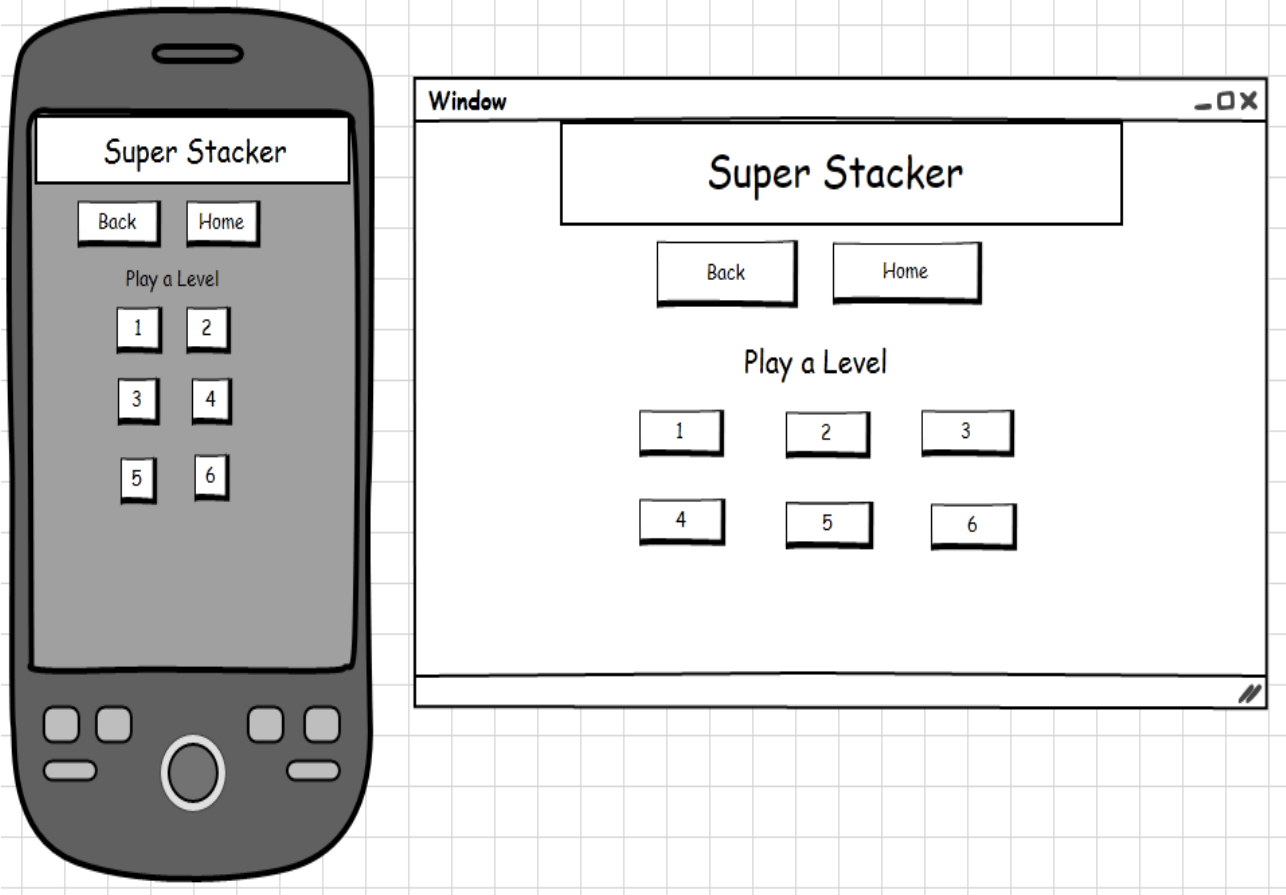
-Pantalla menú Super Stacker



Esta pantalla tampoco presenta grande cambios con respecto a la anterior en las 2 versiones.

Se muestran más opciones dependiendo de la modalidad escogida. Destacar también que los tamaños de las fuentes han de variar según el dispositivo que muestra la aplicación.

-Pantalla LEVELS Super Stacker

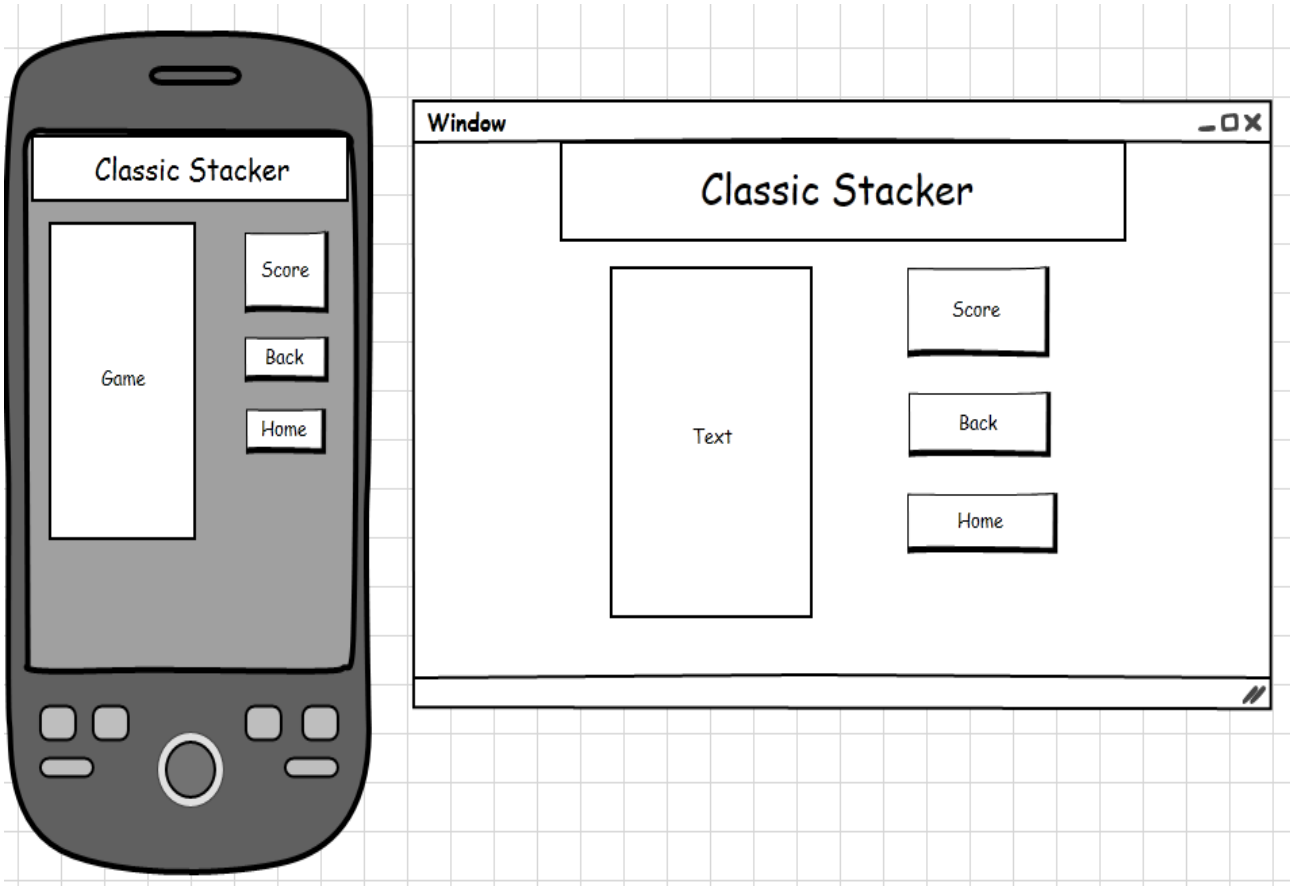


En esta pantalla si podemos observar algunas diferencias entre ambas versiones. La versión PC, al tener una pantalla más ancha, permite mostrar más niveles por filas de manera que la interfaz no pierda calidad.

La versión móvil ha de tener menos opciones de nivel por fila ya que, si la pantalla es aún mas pequeña de la que el ejemplo muestra, las filas podrían tener problemas de visualización y posicionamiento.

Así pues, reducimos el tamaño de los contenedores y de las fuentes en versión móvil para evitar un posible descuadre entre opciones de niveles.

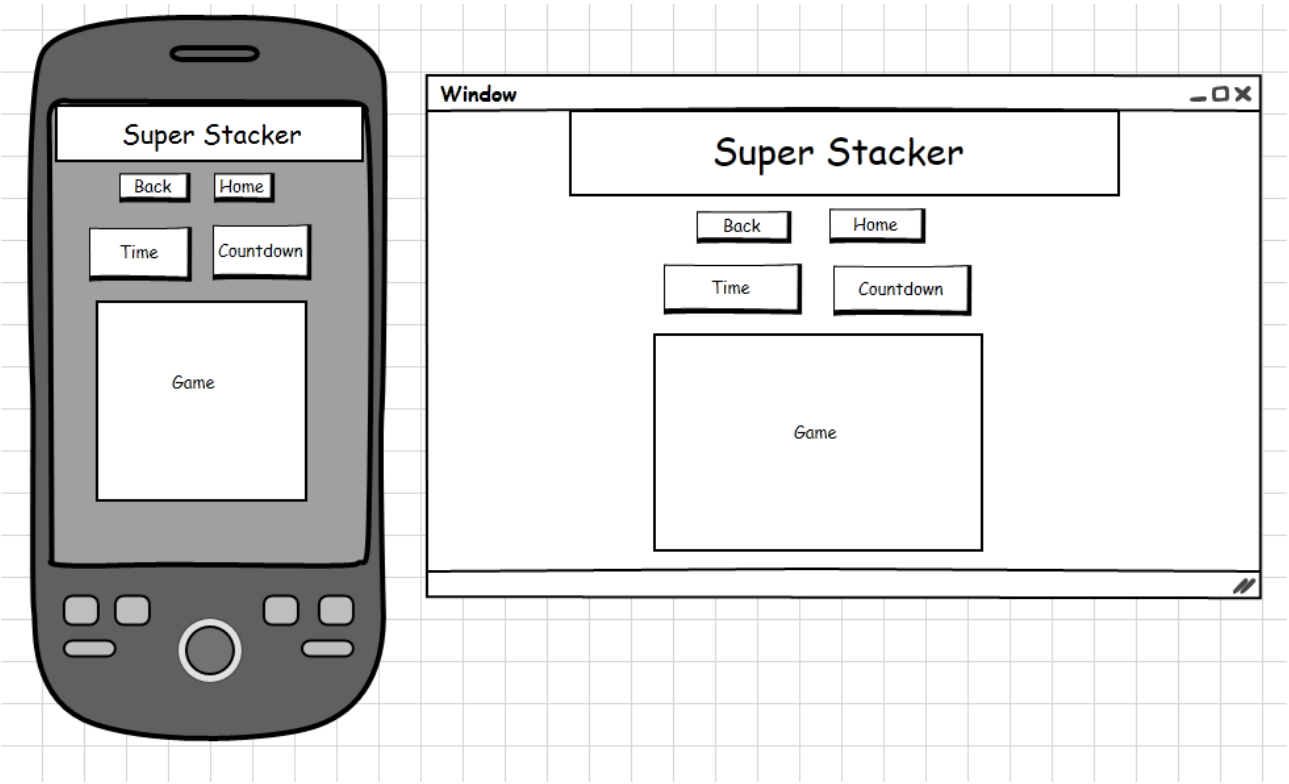
-Pantalla Play Classic Stacker



Esta pantalla deberá mantenerse lo más parecida posible tanto en versión móvil como en versión PC. Hay que destacar que el contenedor CANVAS está constantemente repintando el estado de la partida y puede ser sensible al re-escalado en caliente de la aplicación, aunque siempre se intentará trabajar con tamaños porcentuales.

Des esta manera, hemos de estar preocupados por las fuentes y sus contenedores para que se adecuen según el tamaño de la pantalla actual.

-Pantalla Play Super Stacker



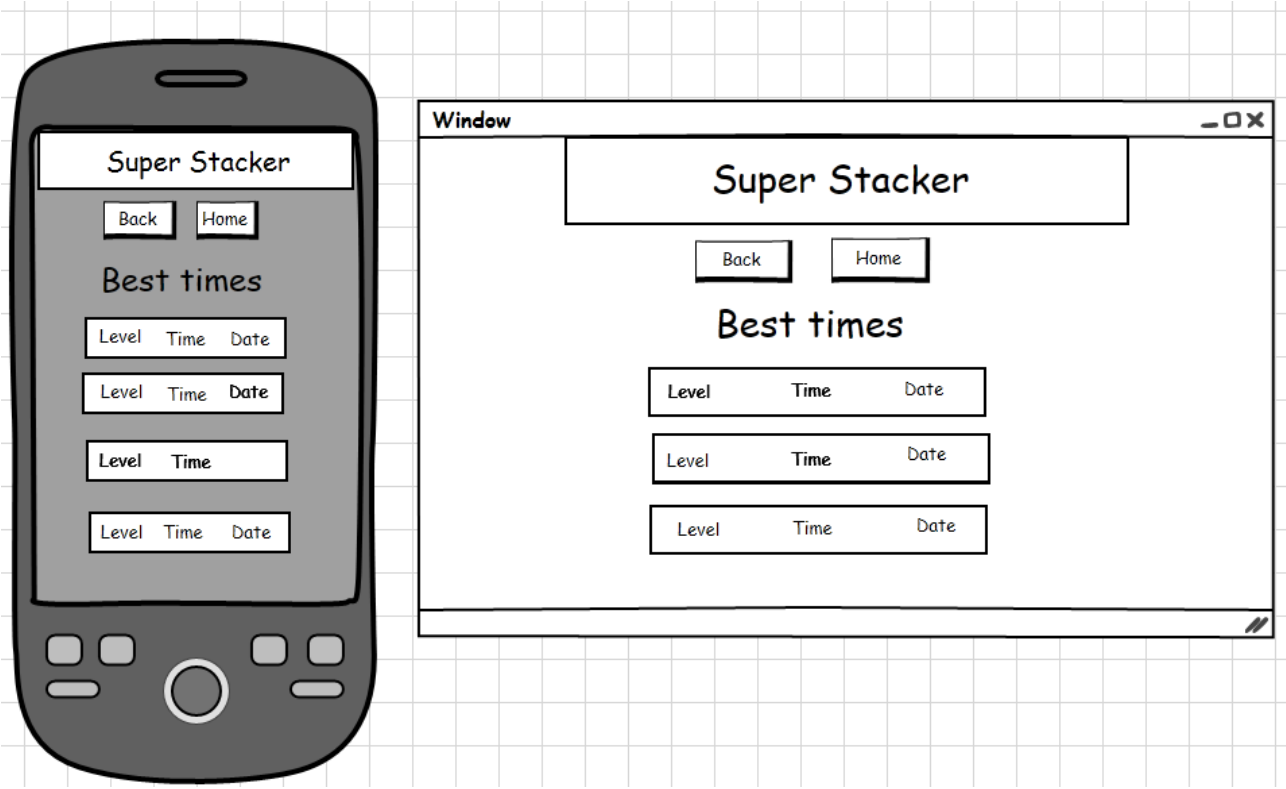
Esta pantalla tiene cierta complejidad a la hora de aplicar estilos y posicionamiento sobre ellos. Hay que tener muy en cuenta el elemento CANVAS, aunque realmente son dos por detalles de implementación que explicaremos mas adelante.

Visualmente han de mantener un parecido en ambas versiones, aunque el tamaño del lienzo (CANVAS) ha de re-calcularse aplicando un tamaño fijo según el ancho de la pantalla que estamos tratando.

El contenedor countdown sólo aparecerá cuando el usuario haya apilado todas las figuras propuestas para realizar una cuenta atrás. Luego ha de desaparecer. Lo colocamos en los bocetos para tener en cuenta a la hora de estilos y posicionamiento.

Destacar también la aparición de un contenedor indicando si el usuario ha ganado o ha perdido (comentado en el apartado de análisis – casos de usos), con las opciones correspondientes según el estado obtenido.

-Pantalla Best Times Super Stacker



Observamos con esta pantalla como han de visualizarse los mejores tiempos para super stacker. Para classic stacker, la pantalla de mejores puntuaciones será muy parecida a esta.

Los estilos se intentarán mantener en ambas pantallas.

Diseño Software

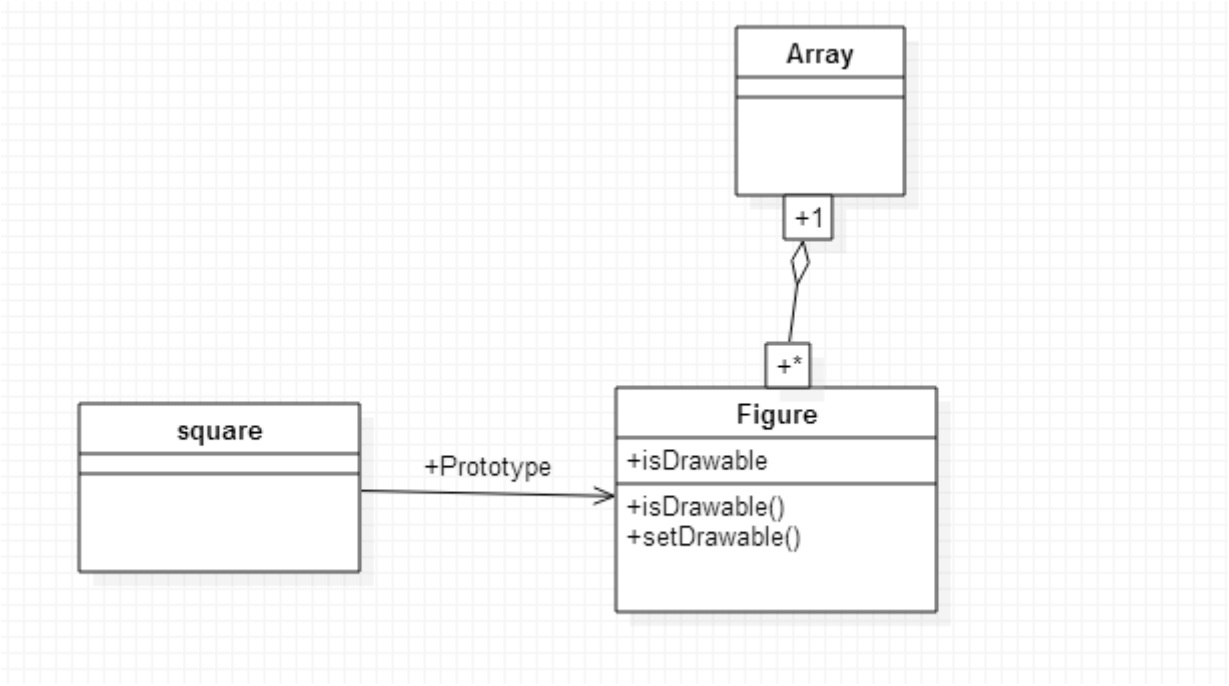
En este apartado vamos a explicar las bases necesarias para trabajar con una serie de estructuras que nos ayudarán a diseñar la arquitectura de la aplicación para cada una de las modalidades. Durante la etapa de implementación veremos algunos detalles y además, será necesario entender como funcionan algunos de los frameworks utilizados.

-Diseñando classic stacker

Empezaremos diseñando esta modalidad de juego.

Recordemos que esta modalidad consiste en apilar una fila horizontal de cuadrados en movimiento sobre otra fila de cuadrados que ya forman parte del mapa (ver reglas en el apartado de análisis). Como toda la lógica del juego va a depender de un mapa en forma de cuadrícula y una fila de cuadrados moviéndose a través de ellos, resulta una buena idea diseñar una matriz en la cual cada elemento de ésta contenga un objeto que representará un cuadrado del mapa que dibujaremos en CANVAS.

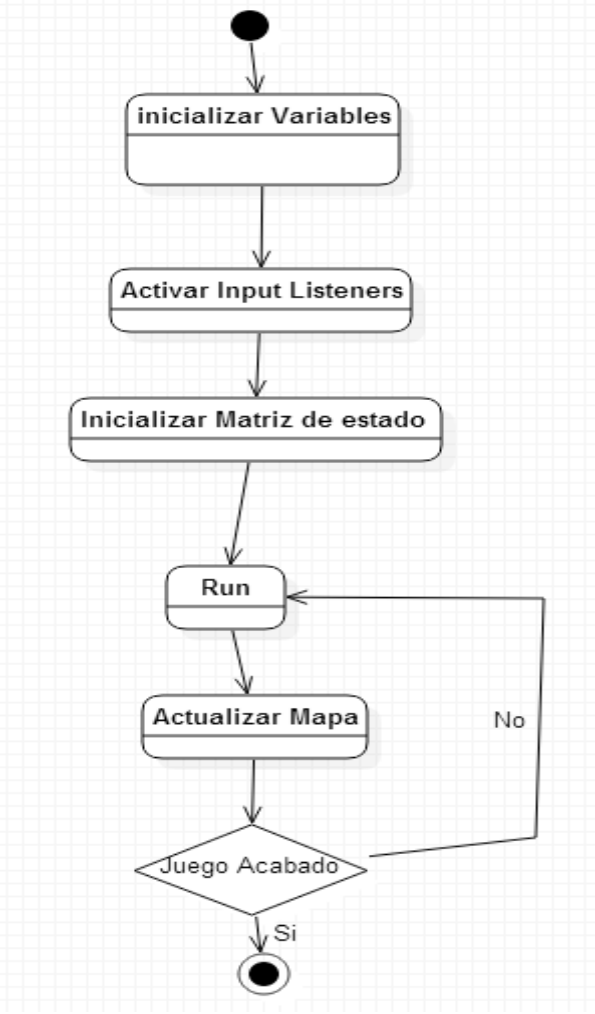
Así pues, quedaría representado de la siguiente forma teniendo en cuenta que vamos a codificar con JAVASCRIPT, utilizando prototypes.



Utilizaremos este diseño para mantener constantemente el estado de la partida actualizado.

En el sencillo diagrama podemos observar como “Square” será un prototipo de “Figure”, heredando así sus atributos que nos servirán para saber si un cuadrado ha de pintarse en CANVAS o no. Los métodos nos ayudarán a ejecutar tal acción. El array estará compuesto por todas los cuadrados que formarán parte del juego. Serán tantos ARRAY como columnas tenga el mapa de juego.

La lógica del juego seguirá el siguiente patrón.



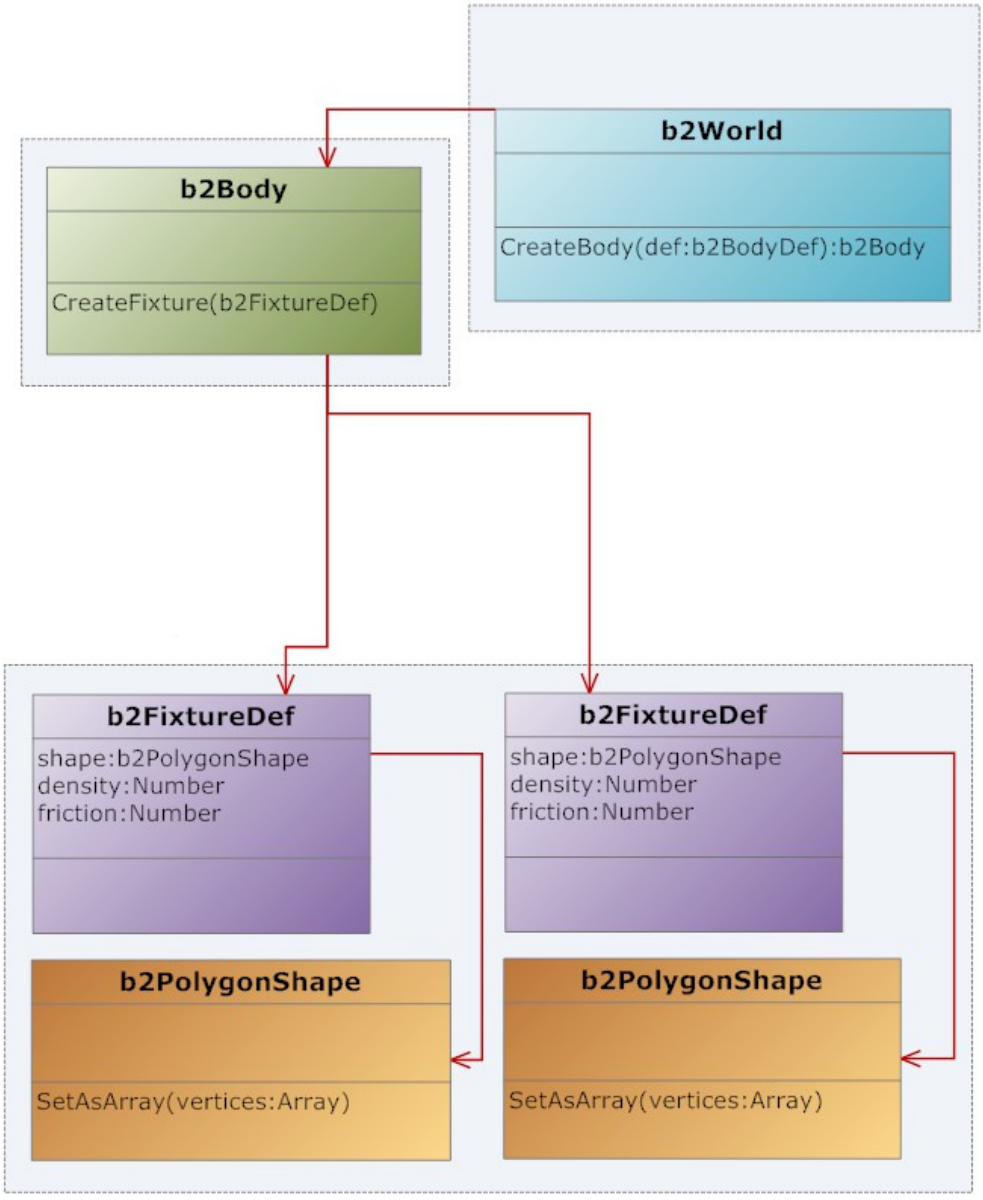
Actualizar jugará un papel muy importante, ya que no sólo dibujará el estado del mapa, sino que comunicará cuando han de insertarse o eliminarse de la matriz de estado los objetos correspondientes. Dentro del bucle principal habrá que estar pendientes de los eventos de entrada del usuario, tales como el evento “mosedown”, y actuar según el estado del mapa al pulsar. Según se compruebe el estado, el juego habrá acabado o no.

-Diseñando super stacker

Este apartado necesitará explicar, además de las clases propias utilizadas, cómo trabajan los frameworks implementados tales como box2dweb y easeljs.

El mundo que se va a crear ha de estar afectado por un entorno físico simulado y para ello es necesario aplicar un motor físico que nos ayude a simular estos efectos. Box2dweb implementará toda esta función, pero es necesario entender cómo esta diseñado ya que gran parte de la codificación tendrá presente clases y métodos de este framework.

Vamos a ver los principios básicos de **Box2dWeb**



Es importante entender el diagrama anterior para poder codificar acoplando box2dweb.

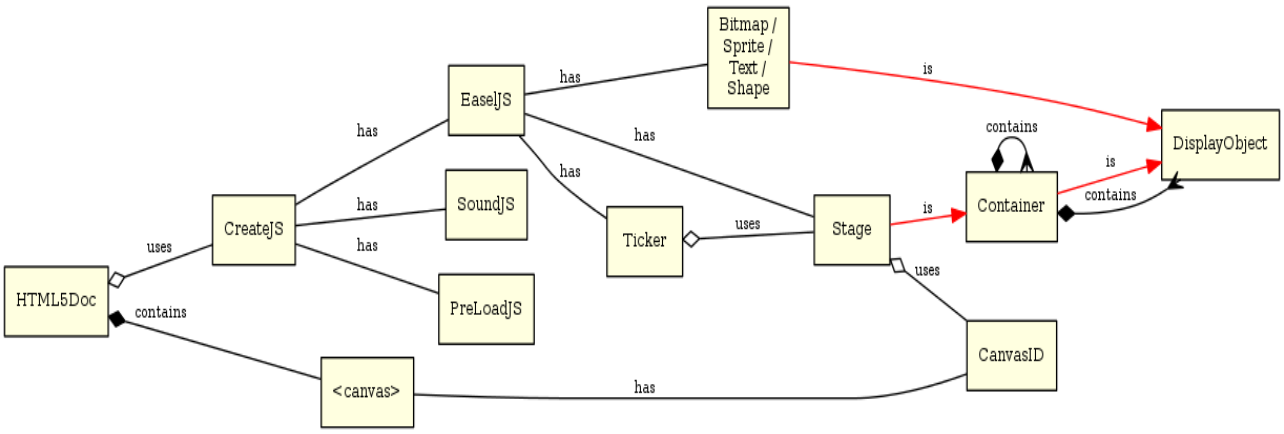
Lo primero que hay que tener en cuenta es "**b2world**". Esta clase de box2dweb va a permitir crear un entorno que contendrá las figuras que van a estar afectadas por las fuerzas simuladas. Esta clase se configurará de primera mano a través de su constructor, especificándole la fuerza gravitatoria que afectarán a los objetos al caer (cuanto mayor sea el parámetro más fuerza ejercerá sobre los objetos que estén en altura o en movimiento) . Es importante resaltar que esta clase va a depender en cierta medida de otra clase llamada "**b2DebugDraw**", en la cual se especificará la escala del mundo que vamos a pintar y donde lo vamos a pintar (en nuestro caso será un elemento CANVAS).

La siguiente clase a tener en cuenta es " **b2Body**", que serán las figuras que formarán parte del mundo. El cuerpo podrá ser estático o dinámico. Si es estático, las fuerzas, colisiones y demás no tendrán efecto sobre este, mientras que los objetos dinámicos si estarán afectados por este tipo de fuerzas y colisiones. Los objetos b2Body podrán tener formas distintas gracias a la clase "**b2FixtureDef**", en la cual se especificará si se desea pintar un polígono o una figura circular. La instancia de b2body dependerá de b2fixture para mostrarse en el mapa con la forma deseada.

Estas clases y algunas otras serán de suma importancia a la hora de implementar el mundo que queremos simular.

Pero... ¿como conseguimos que el mundo que generamos interactúe con fluidez? Para eso **easeljs** no servirá de mucha ayuda.

Vamos a visualizar el siguiente diagrama para intentar comprender mejor el funcionamiento de easeljs:



Lo primero que se ha de tener en cuenta es que con las nuevas versiones de HTML5 y canvas, la generación de gráficos tanto 2D como 3D han cobrado más importancia. Es por ello que los métodos clásicos setInterval y setTimeout de javascript son un poco menos eficientes para este tipo de características de una aplicación. Easeljs ofrece facilidades para implementar la generación de un "Stage" con fluidez basándose en la función requestanimationframe de javascript.

La clase "**CreateJs**" nos va a permitir crear lo que se denomina un "**Stage**", que será el enlace con el elemento canvas para trabajar con este. Con la instancia de esta clase podremos configurar un **Ticker**, que será la función que se ejecutará constantemente y que se utilizará como bucle principal. El Ticker dependerá de los fps que queremos para el juego.

Con estos easeljs y box2dweb realizaremos la totalidad de esta modalidad de juego y los métodos necesarios para facilitar la creación de figuras dinámicas, estáticas o kinetic shapes (describen un movimiento constante).

La lógica de esta aplicación dependerá de toda la información suministrada por box2dweb, es decir, ¿cuándo un cuerpo dinámico entra en contacto con uno de los límites? ¿cómo colocar los cuerpos a lo largo del mundo? Todo esto y más ha de reflejarse en métodos de fácil manejo para facilitar la codificación tanto actual como futura.

Etapa 3: Implementación

Después de finalizar con las etapas anteriores y teniendo claro el diseño del software que vamos a codificar, pasamos a la etapa de implementación que , a su vez, se ha divido en dos partes.

Destacar que durante toda la implementación se ha intentado codificar de la manera más limpia posible, utilizando y desarrollando métodos cortos y de fácil entendimiento. También se ha intentado mantener una estructura de directorios ordenada dividiendo código JAVASCRIPT, HTML, CSS y carpeta de imágenes por separado.

En la fase de mantenimiento se podrá refactorizar lo mejor posible el código final para ayudar a facilitar futuros cambios, mejoras para la aplicación o localización de errores.

-Classic stacker

Primeramente vamos a explicar como están organizados los diferentes archivos de esta modalidad de juego.

Directorio classicStacker:

css: Contiene el archivos styles.css que contiene todas las etiquetas que afectan a esta parte del juego.

Img: Directorio que contendrá imágenes según se necesiten para esta modalidad.

Js: Directorio que contiene el fichero gameLogic.js, que se encargará de dar toda la lógica necesaria para que el juego funcione como ha de hacerlo.

Ficheros html: Serán ficheros que afecten sólo a esta modalidad de juego, como por ejemplo game.html, que contiene el elemento canvas que será utilizado para visualizar el juego.

-Implementando la lógica de classic Stacker

Cuando el usuario accede a game.html da comienzo una serie de acciones que llevan a mostrar y preparar el juego para proceder a interactuar con él.

En primer lugar, podemos observar que el fichero game.html cargará el juego cuando su carga haya finalizado.

```
<body onload="init()">
```

La función *init* a la que hace referencia la encontramos en gameLogic.js. Esta función será la encargada de poner en marcha todo lo necesario.

Echándole un vistazo a esta función, vemos que se encarga de inicializar todas las variables globales que son necesarias para el funcionamiento del juego además de preparar ciertas estructuras. Se activarán también los listener para los eventos que sean necesarios y se inicializará un intervalo que hará que el juego se actualice cada cierto número de milisegundos.

```
function init(){
    initVariables();
    numFillRectsActual = numFillRectsIni;
    canvas=document.getElementById('canvas');
    $("#gameScore").text("Score : ");
    if(canvas == null) return;
    rectWidth = canvas.width/numRectsX;
    rectHeight = canvas.height/numRectsY;
    ctx=canvas.getContext('2d');
    enableInput();
    initStatusMatrix();
    gameRefreshInterval = setInterval(function(){requestAnimationFrame(run);},gameRefreshTime);
}
```

Vamos a desglosar un poco a continuación a lo que estamos haciendo referencia con la imagen anterior. Lo primero que observamos es el método *initVariables*, que inicializará todas las variables con sus parámetros por defecto.

Podemos ver un conjunto de variables que serán importantes para el desarrollo del juego sobre el elemento canvas. Vemos que la variable *ctx* obtiene una referencia al contexto en 2D del lienzo. Esto nos servirá para dibujar sobre el canvas de la referencia las figuras que queramos, en este caso cuadrados. Dibujar formas sobre el lienzo no resulta difícil. Podemos ver un ejemplo a continuación.

```
function paint(ctx,numFillRects){

    ctx.beginPath();
    ctx.fillStyle = "#113D99";
    ctx.fillRect(0,0,canvas.width,canvas.height);
```

Con la referencia *ctx* mencionada anteriormente dibujamos una figura rectangular con relleno (*FillRect* y su relleno, que será lo especificado en *fillStyle*). Se partirá del punto (0,0) y el tamaño será el total del elemento canvas (*canvas.width y canvas,height*).

Después de explicar un poco como pintar formas sobre el lienzo, vamos a destacar tres puntos importantes de la función anterior. La función *enableinputs* va a establecer un listener sobre canvas cuando el usuario pique sobre este.

La función *initStatusMatrix* inicializará una matriz de elementos con el objeto diseñado en la etapa 2.

Este objeto queda codificado de la siguiente manera:

```
function Figure(){
    this.drawable = false;
}

Figure.prototype.setDrawable = function(drawable) {
  this.drawable = drawable;
};

Figure.prototype.isDrawable = function() {
  return this.drawable;
```

Por lo que la matriz de elementos quedará :

```
function initStatusMatrix (){
    gameStatusMatrix = new Array();

    for(var i=0;i<numRectsX;i++){
        gameStatusMatrix[i] = new Array();
        for(var j=0;j<numRectsY;j++){
            gameStatusMatrix[i][j] = new Figure();
        }
    }
}
```

Observamos que cada elemento de la matriz tendrá una instancia de “Figure”, de tal manera que cuando se quiera obtener la información del estado del mapa tengamos la variable global *gameStatusMatrix* a nuestra disposición. El tamaño de la matriz será el especificado en *numRectsX* y *numRectsY,* que vienen a indicarnos el número de cuadrados del mapa que habrá de alto y de ancho.

El siguiente punto a tener en cuenta es la función *run*, que es la que está afectada por el intervalo que encontramos al final de la función *init.*

La función *run* se encargará de llamar a la función *paint,* que será la encargada de pintar el mundo según lo que encuentre en la matriz de estado.

```
function run(){
    checkMove(numFillRectsActual);
    paint(ctx,numFillRectsActual,posActual);
    refreshReset();
}
```

Por último, resaltar para esta modalidad el evento "*onmousedown*" que tendremos en escucha para saber si el juego ha de continuar repintando o de lo contrario el juego ha acabado.

```
function enableInput(){
    if(canvas == null ) return;
    canvas.addEventListener('mousedown',mouseDownEvent,false);
}

function mouseDownEvent(){
    checkGameStatus(numFillRectsActual);
    saveFillRectsToPaint(numFillRectsActual);
    run();

}

function saveFillRectsToPaint(numFillRects){
    var firstFillRectX = getFirstFilRectPosX();
    var j = getFirstFilRectPosY();

    for(var i=firstFillRectX;i<firstFillRectX+numFillRects;i++){
        gameStatusMatrix[i][j].setDrawable(true);
    }
    actualRow++;
}
```

Con esta imagen vemos un ejemplo de cómo al detonar el evento se comprueba el estado del juego y se actualiza la matriz de elementos que simulan el mapa actual del juego. Si el jugador se ha quedado sin cuadrados para apilar, el juego ha acabado. La función *checkGameStatus* se encarga de comprobarlo y de modificar la puntuación. También comprueba si se ha pasado al siguiente nivel, con lo cual la

velocidad del intervalo actual del juego aumenta.

-Implementando la lógica de super Stacker

Vamos a explicar algunos detalles importantes acerca del desarrollo de super stacker.

En primer lugar explicaremos los archivos importantes que han afectado al desarrollo.

Directorio superStacker:

Contiene los directorios *css, img y js*, los cuales cumplen la misma función que describimos anteriormente para classic stacker.

Si nos centramos en el directorio js, encontraremos lo siguientes ficheros:

main.js: Contiene toda la lógica que afecta al juego super stacker.

box2dweb.js: Archivo que contiene todas las clases y métodos necesarios para simular el mundo planteado durante el análisis.

easeljs.js: Fichero necesario para poder implementar el framework easeljs que nos ayudará a trabajar con el entorno canvas de una manera sencilla.

Figure.js: Este fichero contiene una clase creada a partir de box2dweb y de easeljs que facilitará la creación de un cuerpo dinámico con una imagen ajustada a su altura y anchura.

preload.js: Archivo que se cargará primeramente para comprobar que todas las imágenes necesarias para mostrar durante el juego se cargan correctamente, ya que de no ser así algunas propiedades no serían posible obtenerlas a través de javascript, siendo bastante perjudicial para el desarrollo del juego.

Levels/level.js: Este archivo contiene una serie de métodos que ayudarán a crear y colocar objetos estáticos en una posición concreta y especificar qué objetos dinámicos tendrán que ser apilados.

A continuación vemos algunos detalles de implementación que ayudarán a comprender un poco mejor lo que se ha hecho.

Lo primero de todo es partir del fichero game.html que podremos encontrar en el directorio superStacker.

```
<body onload="load()">
```

Encontraremos que después de la carga de los elementos del fichero html se ejecutará la función load. Esta función se encuentra en el fichero preload.js y se encargará de precargar todas las imágenes necesarias para el juego. En la siguiente imagen podemos ver un ejemplo:

```
function loadImages (){
    var img=new Image();
    img.src="soccer.png";
    img.onload = function(){
        numImagesLoaded++;
    }

    var img=new Image();
    img.src="tesoro.png";
    img.onload = function(){
        numImagesLoaded++;
    }

    var img=new Image();
    img.src="brownDegrad.jpg";
    img.onload = function(){
        numImagesLoaded++;
    }

    var img=new Image();
    img.src="triangle.png";
    img.onload = function(){
        numImagesLoaded++;
    }
```

Cuando todas las fotos se hayan cargado se ejecutará entonces la función init que se encuentra en main.js

Hay que destacar del fichero main.js que para facilitar el acceso a las clases de box2dweb hemos creado una lista de referencias hacia las clases que vamos a utilizar a lo largo del código.

```
var box2d = {
    b2Vec2 : Box2D.Common.Math.b2Vec2,
    b2BodyDef : Box2D.Dynamics.b2BodyDef,
    b2Body : Box2D.Dynamics.b2Body,
    b2FixtureDef : Box2D.Dynamics.b2FixtureDef,
    b2Fixture : Box2D.Dynamics.b2Fixture,
    b2World : Box2D.Dynamics.b2World,
    b2MassData : Box2D.Collision.Shapes.b2MassData,
    b2PolygonShape : Box2D.Collision.Shapes.b2PolygonShape,
    b2CircleShape : Box2D.Collision.Shapes.b2CircleShape,
    b2DebugDraw : Box2D.Dynamics.b2DebugDraw,
    b2Listener : Box2D.Dynamics.b2ContactListener
};
```

Teniendo estas referencias, nos centramos en explicar como empieza a formarse el mundo que queremos.

```
function init(){

    resetVariables();
    stage = new createjs.Stage(document.getElementById("canvas"));
    setupPhysics();
    stageOnMouseDown();
    createjs.Ticker.addListener(this);
    createjs.Ticker.setFPS(60);
    createjs.Ticker.useRAF = true;
    world.SetContactListener(listener);

    loadBasicShapesLevel();
    loadListLimits();
    loadShapesLevel();
    repaintBasicShapes();
    showListShapesToStack();
    drawLimits();
    countActualLevelSeconds();

}
```

Vamos a resaltar los puntos más importantes de esta función, que es la que hará que todo se ponga en marcha para empezar a interactuar.

Primeramente se entiende que hay que inicializar las variables con los valores que queremos por defecto. La variable “Stage” que podemos observar y que se inicializa utilizando la clase de easeljs “stage” para acceder al elemento canvas.

A partir de aquí utilizaremos algunas funciones tales como setupPhysics que nos ayudarán a crear un mundo b2world con unas características adecuadas a los intereses del juego.

```
function setupPhysics(){

    world = new box2d.b2World(new box2d.b2Vec2(0,30),true);
    var debugDraw = new box2d.b2DebugDraw();
    debugDraw.SetSprite(debug.getContext('2d'));
    debugDraw.SetDrawScale(SCALE);
    debugDraw.SetFillAlpha(0.7);
    //debugDraw.SetFlags(box2d.b2DebugDraw.e_shapeBit | box2d.b2DebugDraw.e_joinBit);
    world.SetDebugDraw(debugDraw);
}
```

Es importante observar como con easeljs en la función *init* utilizamos algunos métodos que nos proporciona para facilitarnos la creación de un entorno adecuado y lógico para el objetivo que buscamos. Especificamos con createjs un Ticker que tendrá 60 fps según indicamos y haremos uso del RAF (requestAnimatioFrame), permitiendo mejorar el rendimiento de las texturas sobre canvas. Como hemos especificado un ticker, utilizaremos la función tick de easeljs para controlar la lógica del juego en cada momento. Será el bucle principal, y como hemos especificado, se ejecutará a 60 fps.

```
function updateCanvas(){

    if(stage == null || world == null) return;
    stage.update();
    world.DrawDebugData();
    world.Step(1/60, 10, 10);
    world.ClearForces();

    if(canvas.clientWidth != actualWidht){
        repaintAll();
    }
}

function tick(){
    updateCanvas();
}
```

Todo lo que viene a continuación en la función init se hace con el fin de inicializar todos lo objetos estáticos y dinámicos del mundo. Los objetos que van a tener una imagen van a inicializar un objeto de tipo Figure con propiedades de box2dweb y de easeljs. Utilizaremos easeljs para tratar las imágenes.

```
function Figure(fixDef,shapeSize,image){

    if(image == null)
        this.view = new createjs.Bitmap("");
    else{
        this.view = new createjs.Bitmap(image);
    }

    this.scaleImage(shapeSize);

    this.fixDef = fixDef;
    this.bodyDef = null;

    this.view.body = null;
    this.view.onTick = tick;

}
```

Y las propiedades que va a tener.

```
Figure.prototype.addToStage = function(){
    if(this.bodyDef == null) return;

    this.view.body = world.CreateBody(this.bodyDef);
    this.view.body.CreateFixture(this.fixDef);
    stage.addChild(this.view);
}

Figure.prototype.getFixDef = function(){
    return this.fixDef;
}

Figure.prototype.setBodyDef = function(bodyDef){
    this.bodyDef = bodyDef;
}

Figure.prototype.setImage = function(image){
    this.view.image = image;
}

Figure.prototype.scaleImage = function(shapeSize){

    if(this.view == null) return;

    var imageWidth = this.view.image.width;
```

Como vemos, esta clase dispone de unos métodos para incluirle el tipo de figura que queremos crear (dinámica o estática), sus dimensiones y la imagen si queremos que tenga alguna.

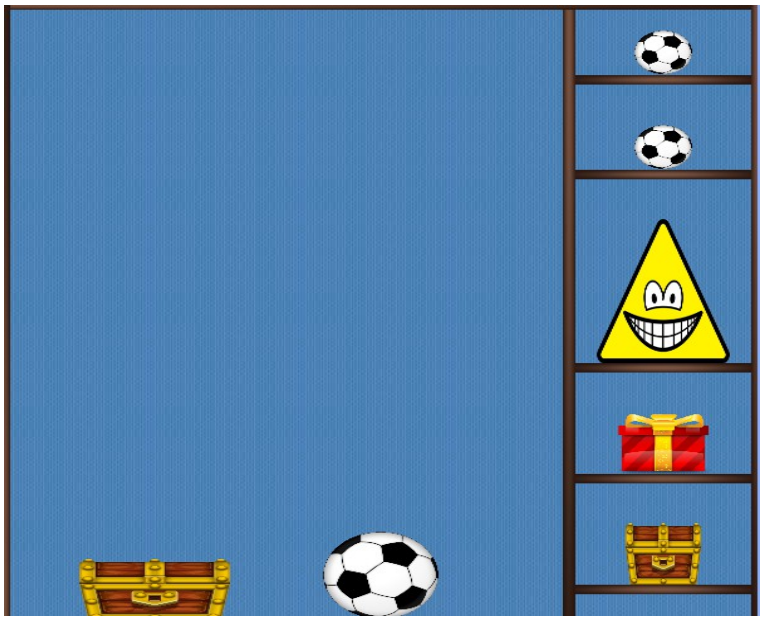
A partir de aquí, utilizaremos esta clase para crear los límites del juego, los objetos estáticos y los objetos dinámicos.

Un buen ejemplo para ver cómo invocamos a esta función y cuándo es a la hora de crear niveles, por lo que extraemos un ejemplo del fichero levels.js dentro del directorio levels.

```
shapeSize = calculatePoints(8,8);
shapeSize.y = shapeSize.x;
fixDef = createCircleShape(shapeSize.y);
positionShape = calculatePoints(50,99);
positionShape = getPointsForGroundPosition(shapeSize,positionShape);
arrayBasicShapesLevel1.push({shape : new Figure(fixDef,shapeSize,"img/soccer.png"),
```

Teniendo en cuenta esto, podemos utilizar de manera más sencilla los restantes métodos que nos harán falta para llegar hasta el resultado final, es decir, aplicar lógica del juego según colisiones entre objetos, repintado según el movimiento del ratón sobre el lienzo, etc.

El resultado final sería algo parecido a esto, es decir, figuras con imágenes que tenemos que ir apilando sobre otras figuras estáticas.



Las imágenes que están a la derecha son las que se han de apilar sobre las que están más a la derecha (pelota y baúl). Cuando el ratón se mueva por el lienzo, la siguiente figura a apilar seguirá el movimiento del ratón hasta hacer click, que será

cuando la figura se desprenda del ratón y caiga desde el punto en el que se soltó según la gravedad configurada para este mundo.

-Local Storage

Para ambas modalidades es importante almacenar en algún lugar los logros conseguidos por el usuario y poder visualizarlos cuando desee.

Para ello, utilizaremos la característica que proporciona HTML5 llamada local storage. Esta característica permite guardar información de forma local por dominio. Se puede guardar entre 5MB y 10 MB de información, superando el límite de 4 KB de las cookies.

Para esta aplicación se ha usado tanto en classic stacker para almacenar la puntuación obtenida por el jugador como en super stacker, almacenando el mejor tiempo de cada nivel.

Podemos observar un ejemplo de uso cuando queremos guardar una puntuación al almacenamiento local para luego acceder desde cualquier sitio web de nuestro dominio.

```
function addToBestScores(){
    var bestScores;

    if(typeof localStorage["bestScores"] == "undefined"){
        bestScores = new Array();
    }
    else{
        bestScores = JSON.parse(localStorage["bestScores"]);
    }
    bestScores.push(gameScore);
    localStorage["bestScores"] = JSON.stringify(bestScores);
}
```

La clave para acceder a los datos en este caso será “bestScores”. En este espacio guardaremos un objeto de tipo array que contendrá las mejores puntuaciones hasta el momento. En el extracto de código obtenido vemos que si el espacio que queremos utilizar no está definido, aprovechamos para crear un nuevo objeto de tipo array. Si esta definido extraemos los datos que se encuentran en este. Hay que observar que la información en localstorage se guarda como string, por tanto hay que parsearla a la hora de recoger los datos como a la hora de guardarlos si no es un objeto de tipo string.

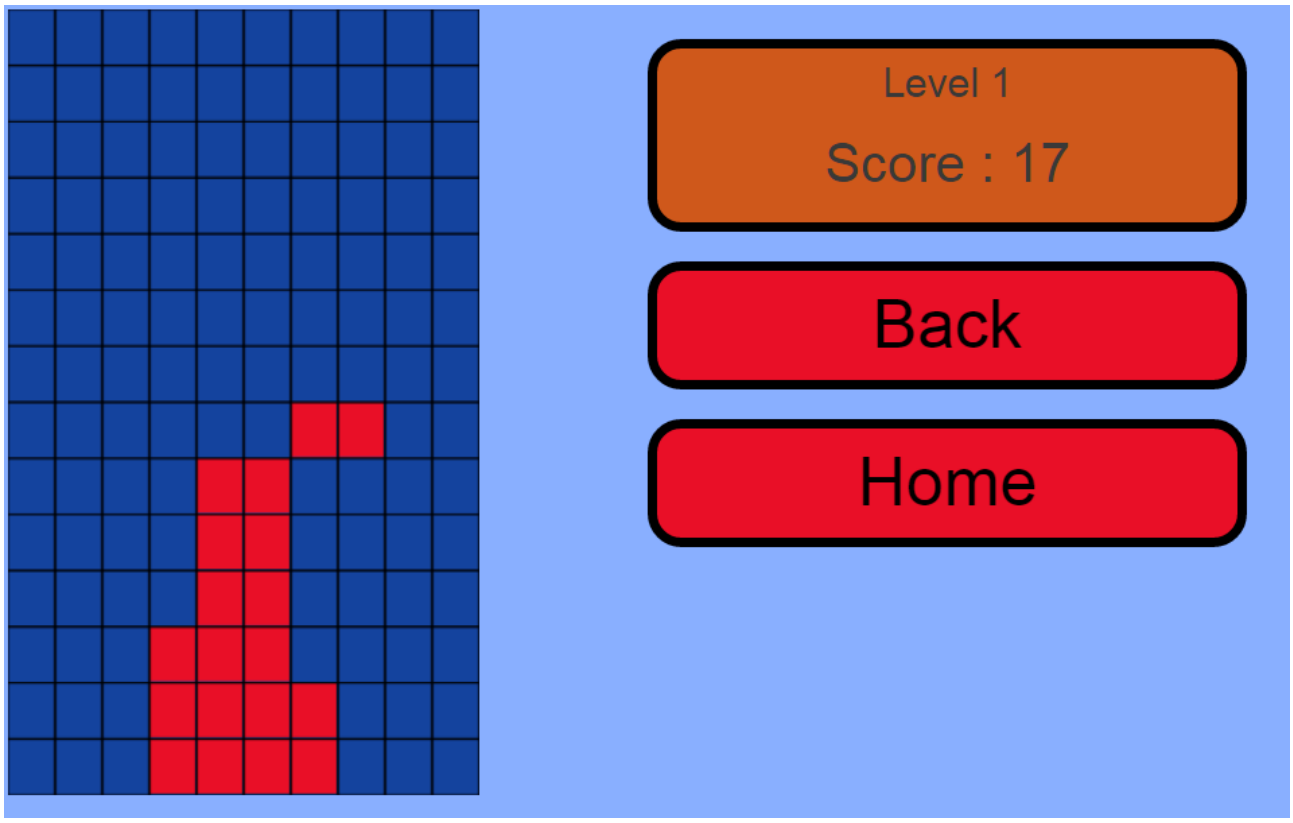
Etapa 4: Pruebas

Una vez que terminado la etapa de implementación pasamos a la etapa de Pruebas.

Vamos a realizar una batería de pruebas e iremos mostrando si la funcionalidad del juego cumple su objetivo para las dos modalidades.

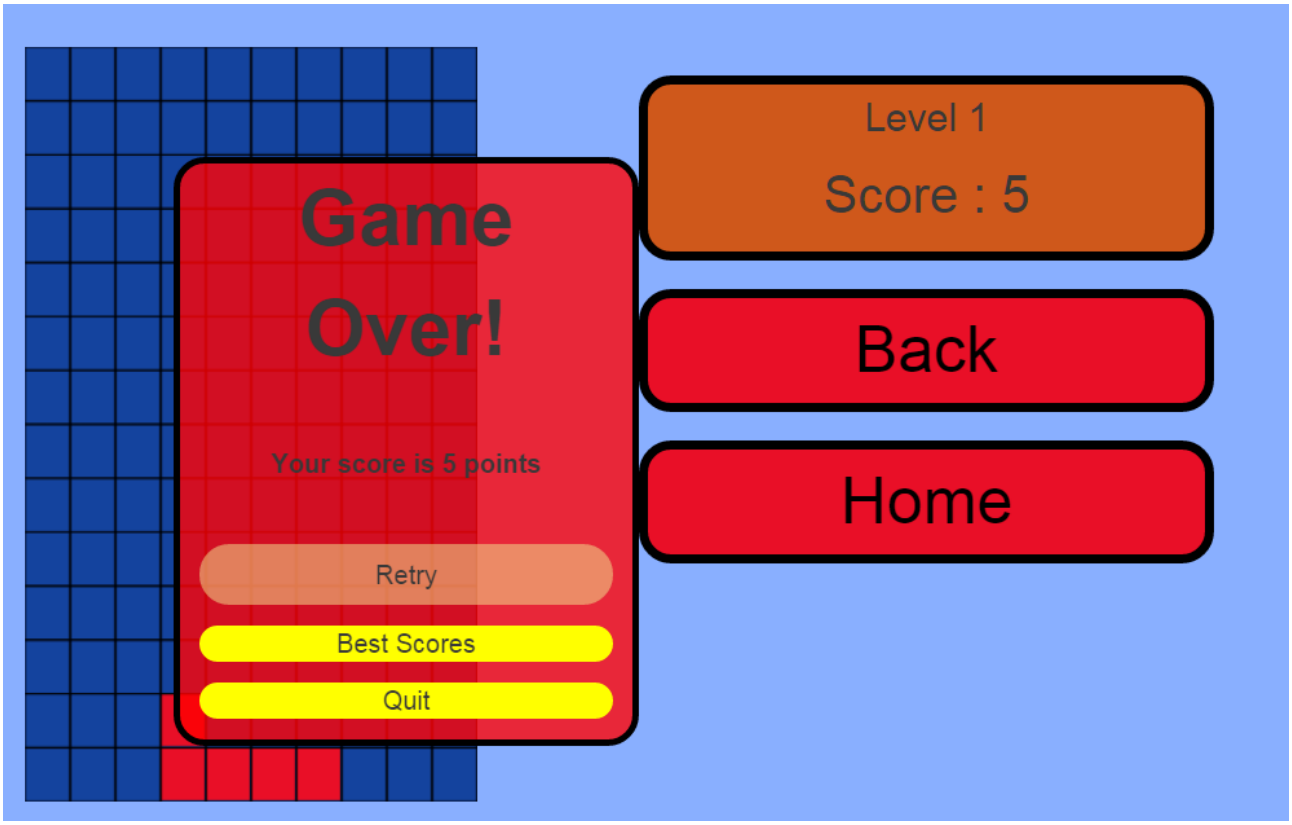
-Classic Stacker

En primer lugar vamos a probar si el juego consigue apilar de manera correcta, es decir, pierde porciones si no están alineadas y el juego continua si tiene cuadrados disponibles.

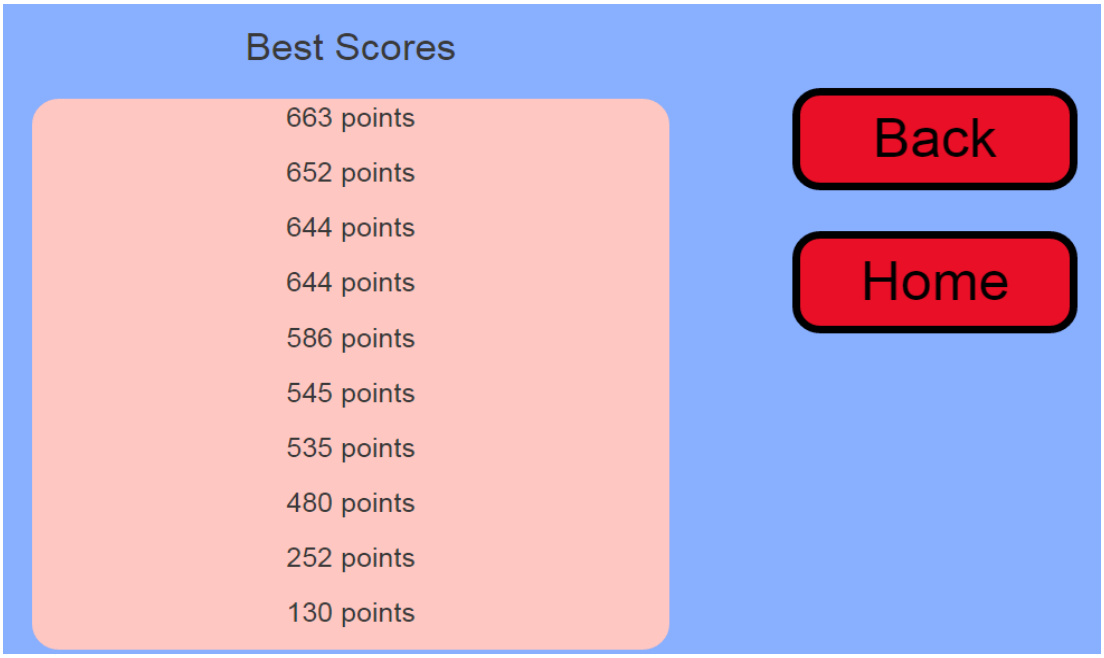


Nos fijamos en varios detalles. Primeramente que el usuario a ido perdiendo tamaño conforme a avanzado porque no ha consigo apilar completamente la fila con respecto a la fila anterior. Cuando lo ha hecho, ha podido proseguir sin problema con la fila. Otro detalle es la puntuación. La puntuación utiliza la siguiente fórmula para calcula los puntos : (puntos actuales + cuadrados disponibles * nivel actual). Podemos observamos que la puntuación se está almacenando bien, ya que en la imagen vemos 17 cuadrados apilados * 1, que es el nivel acttual. Los dos cuadrados últimos están en movimiento y por tanto no se suman al marcador hasta que se consigan apilar.

Vamos a probar si el juego finaliza de manera correcta cuando el usuario no tiene mas cuadrados que apilar y si da la opción a volver a jugar.

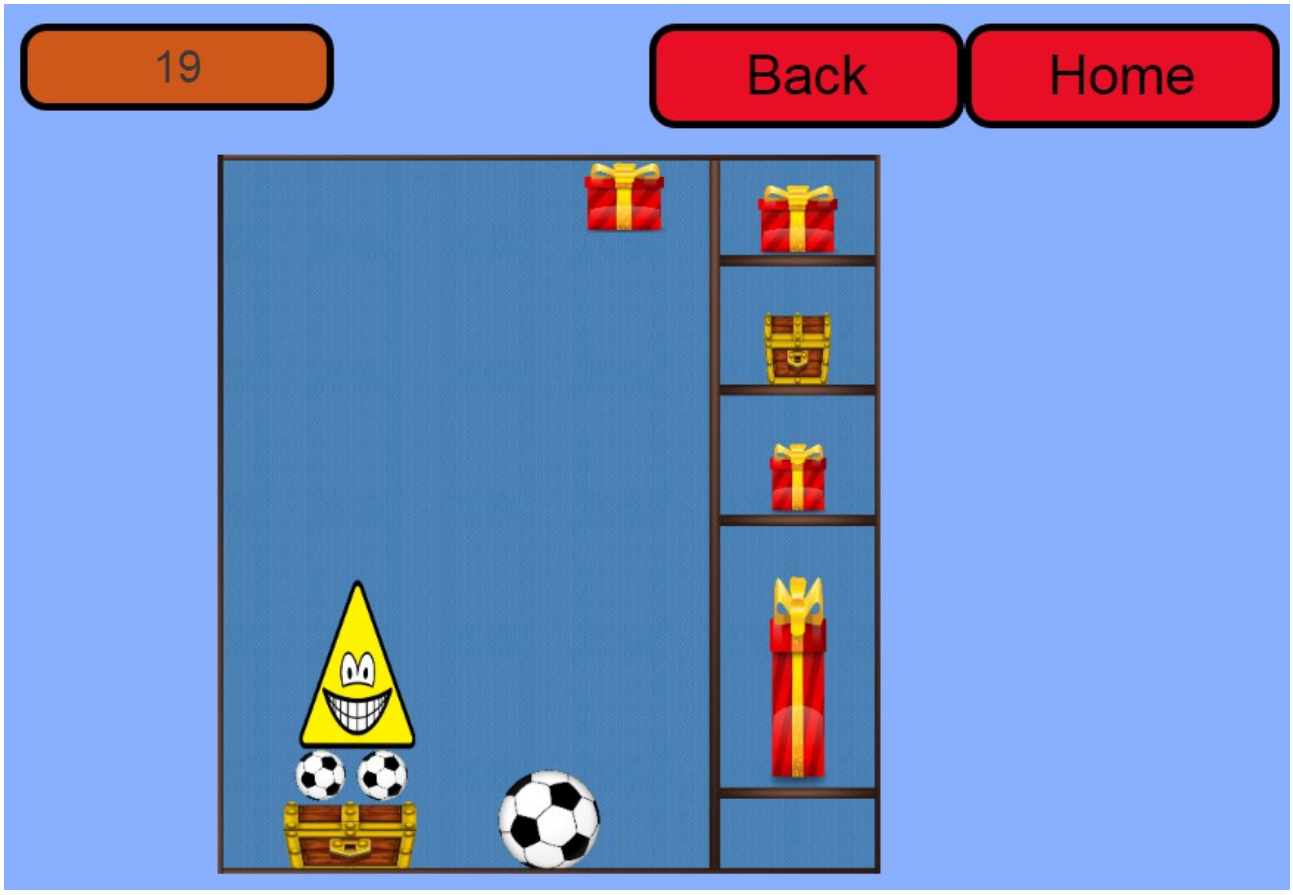


Observamos como su marcador es de 5 puntos y no ha conseguido apilar todos los cuadrados. Muestra correctamente la ventana de opciones que le permite, entre otras cosas, volver a intentarlo. Si nos dirigimos ahora a la pantalla de “Best Scores”, nuestra puntuación no debería aparecer ya que no se encuentra dentro de las mejores puntuaciones obtenidas hasta la fecha.



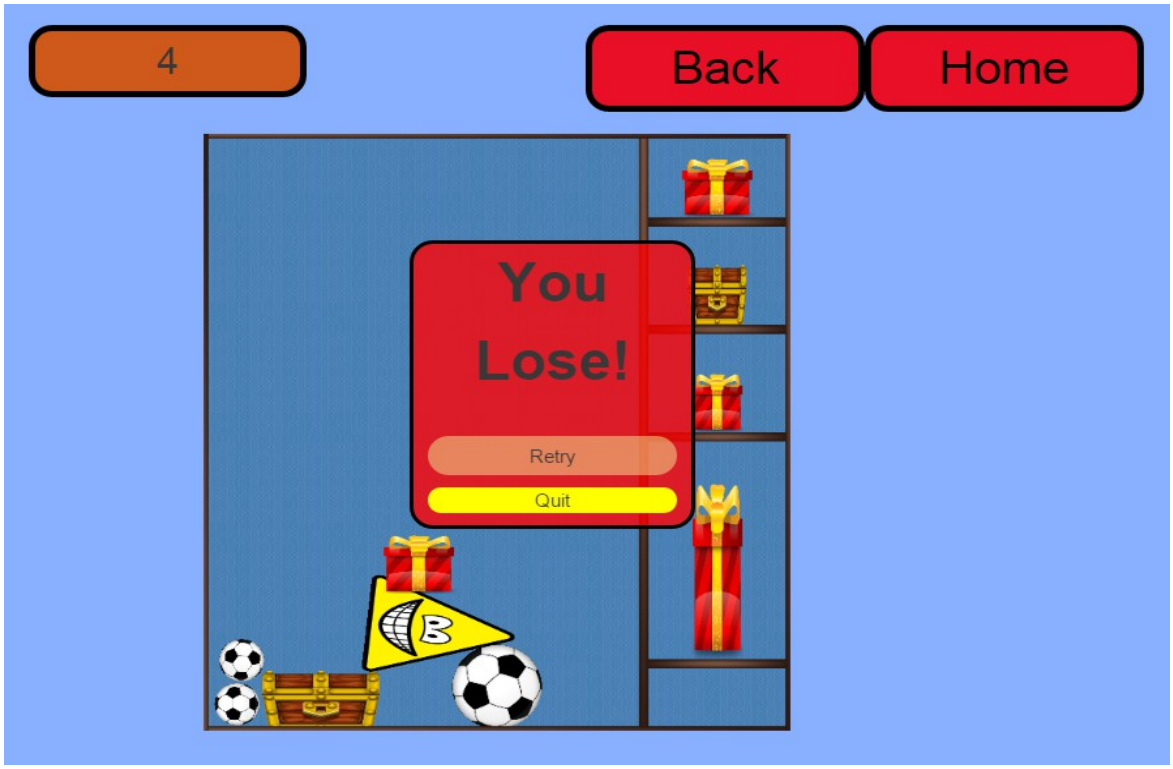
-Super Stacker

La batería de pruebas para super stacker seguirá los mismos pasos que las pruebas para classic stacker. Primeramente nos fijaremos si el juego continua normalmente si ninguna de las figuras dinámicas entra en contacto con alguno de sus límites y además, visualizaremos el tiempo transcurrido y las figuras restantes por apilar.



Han transcurrido segundos, se han apilado 3 objetos y el siguiente a apilar sigue el movimiento del ratón hasta pulsar sobre el lienzo. El juego no ha acabado ya que ninguno de los objetos dinámicos a entrado en contacto con alguno de los límites. A la derecha podemos observar que se han ido restando los objetos que ya han sido apilados y se muestran los que quedan por apilar. Por tanto, esta prueba la damos por válida.

Veamos ahora si el juego acaba cuando se entra en contacto con algún limite.



Efectivamente, uno de los objetos dinámicos ha caído y ha entrado en contacto con el suelo y por tanto, ha perdido. Vemos que se le da la posibilidad de volver a intentarlo. Comprobamos ahora si el juego termina cuando el usuario logra completar el objetivo de apilar y mantener unos segundos la estructura.

Todos los objetos dinámicos están apilados, no queda ninguno en la parte derecha que indique que nos falta alguno. Además, ninguno de los objetos dinámicos ha entrado en contacto con algún límite y, después de pasar unos segundos, la estructura ha aguantado. En este aspecto, la funcionalidad es válida para super stacker.

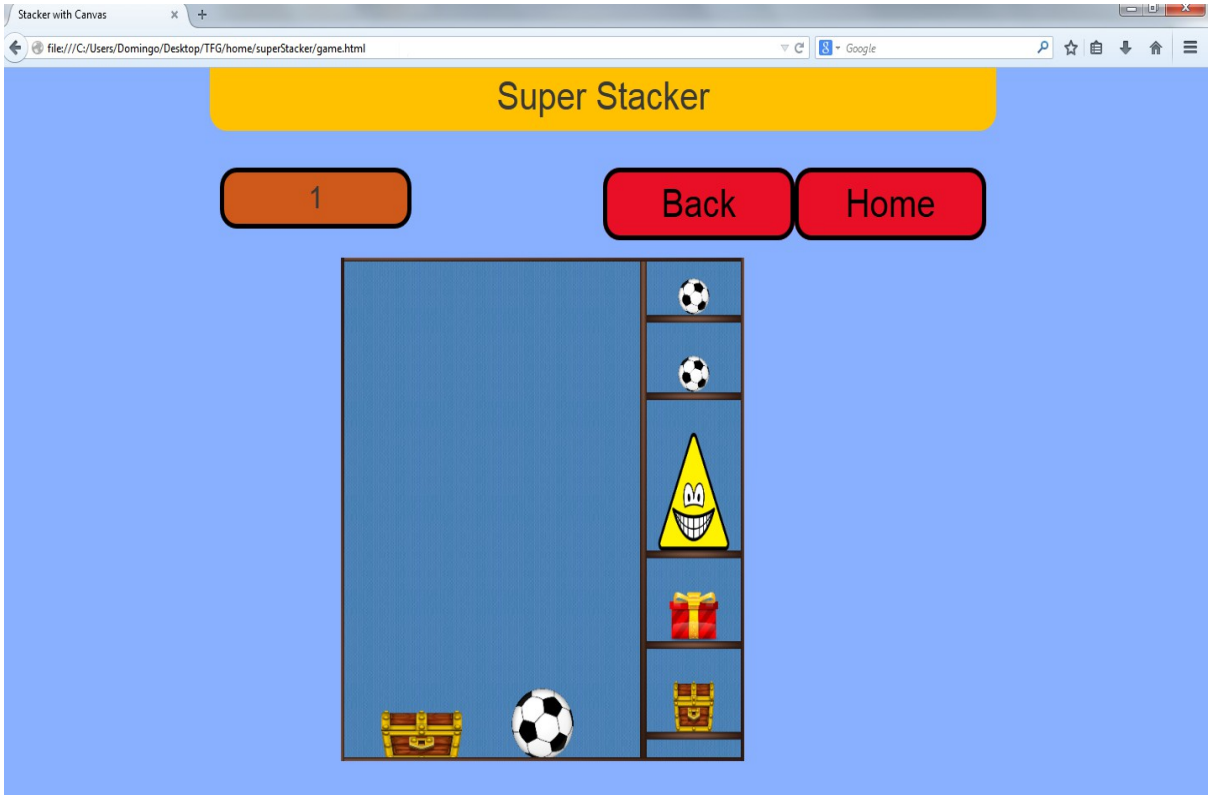
-Pruebas visuales

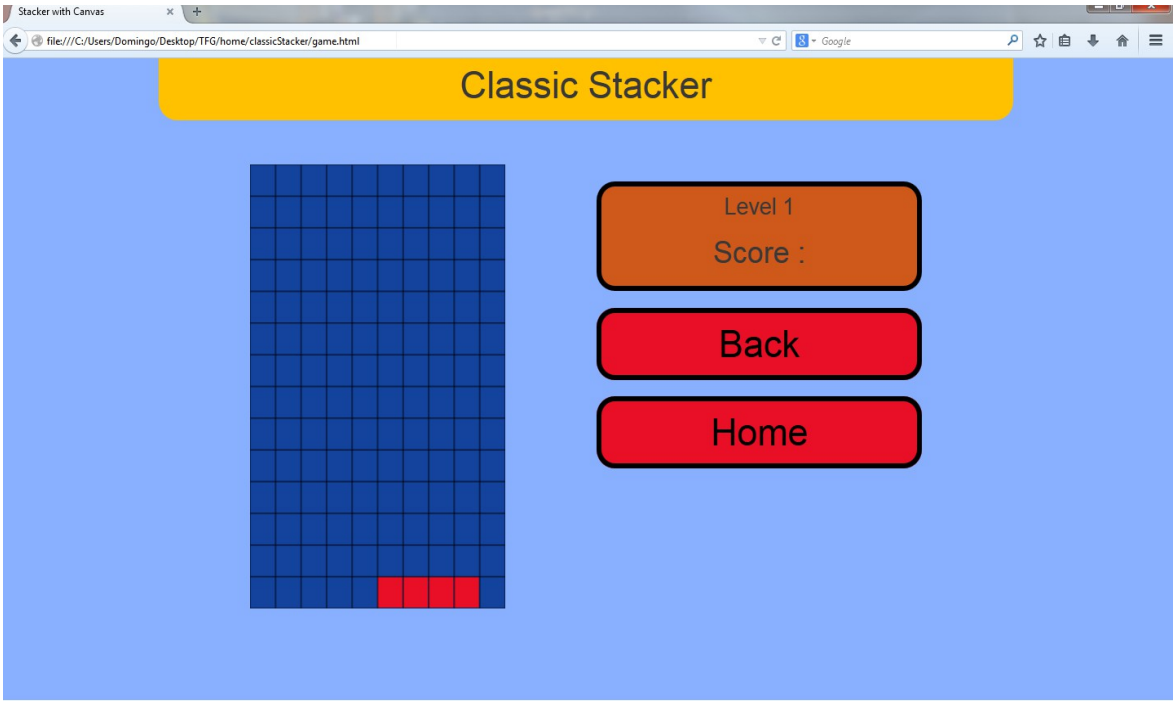
Vamos a realizar a continuación una serie de pruebas para PC y para móviles en distintos navegadores para comprobar la calidad de los estilos, es decir, de asegurarnos de que nuestra aplicación web es capaz de adaptarse sin problemas a los distintos tipos de navegadores y distintos tipos de pantalla.

Hay que resaltar que algunos navegadores antiguos no cumplen los estándares actuales de CSS y algunos ni siquiera son capaces de interpretar etiquetas HTML5 tales como CANVAS, con lo que no podrán disfrutar de la experiencia.

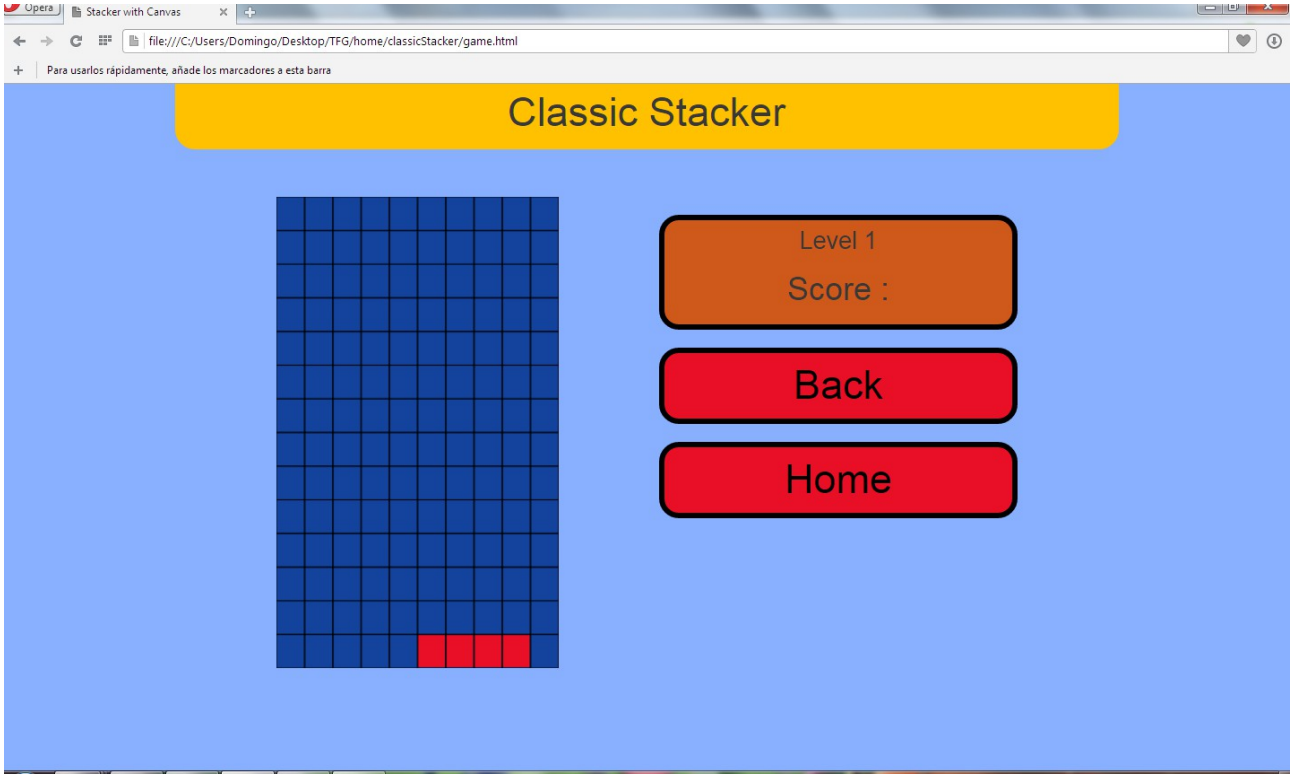
-Pruebas navegadores PC

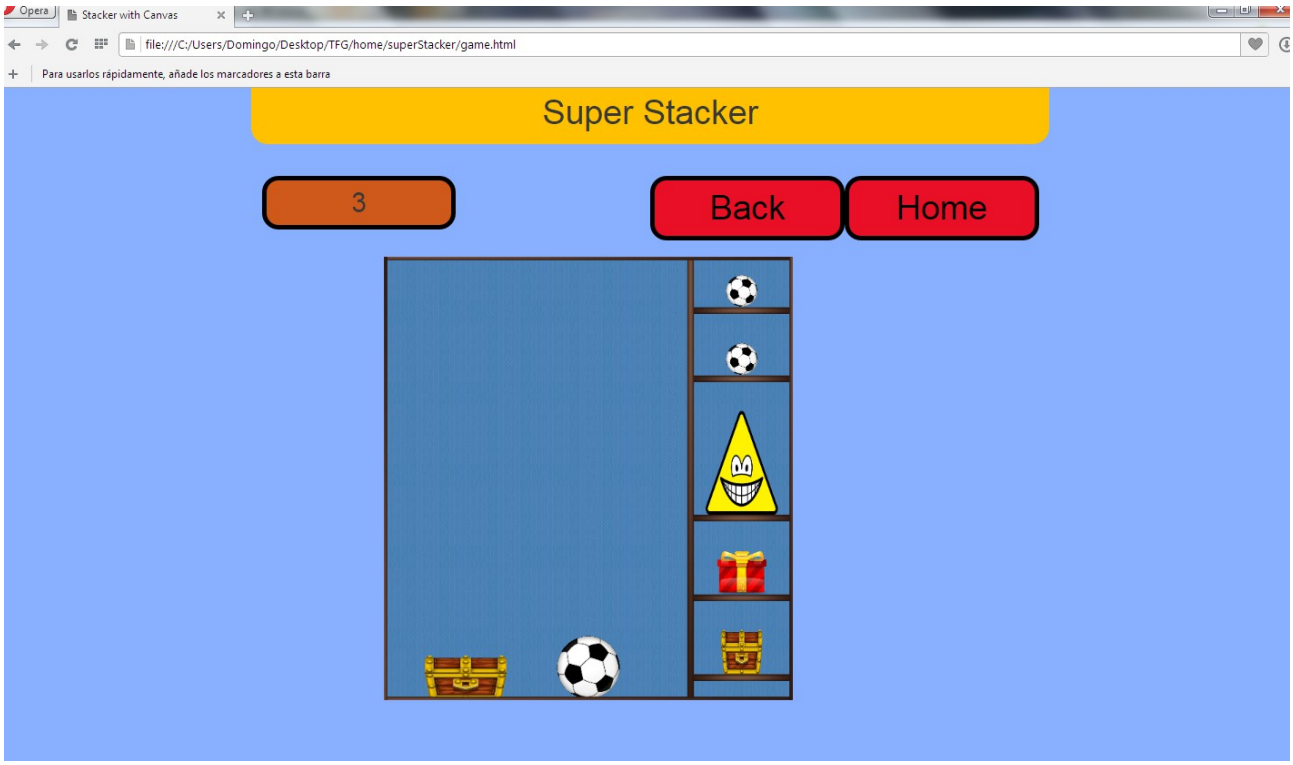
Navegador Mozilla firefox



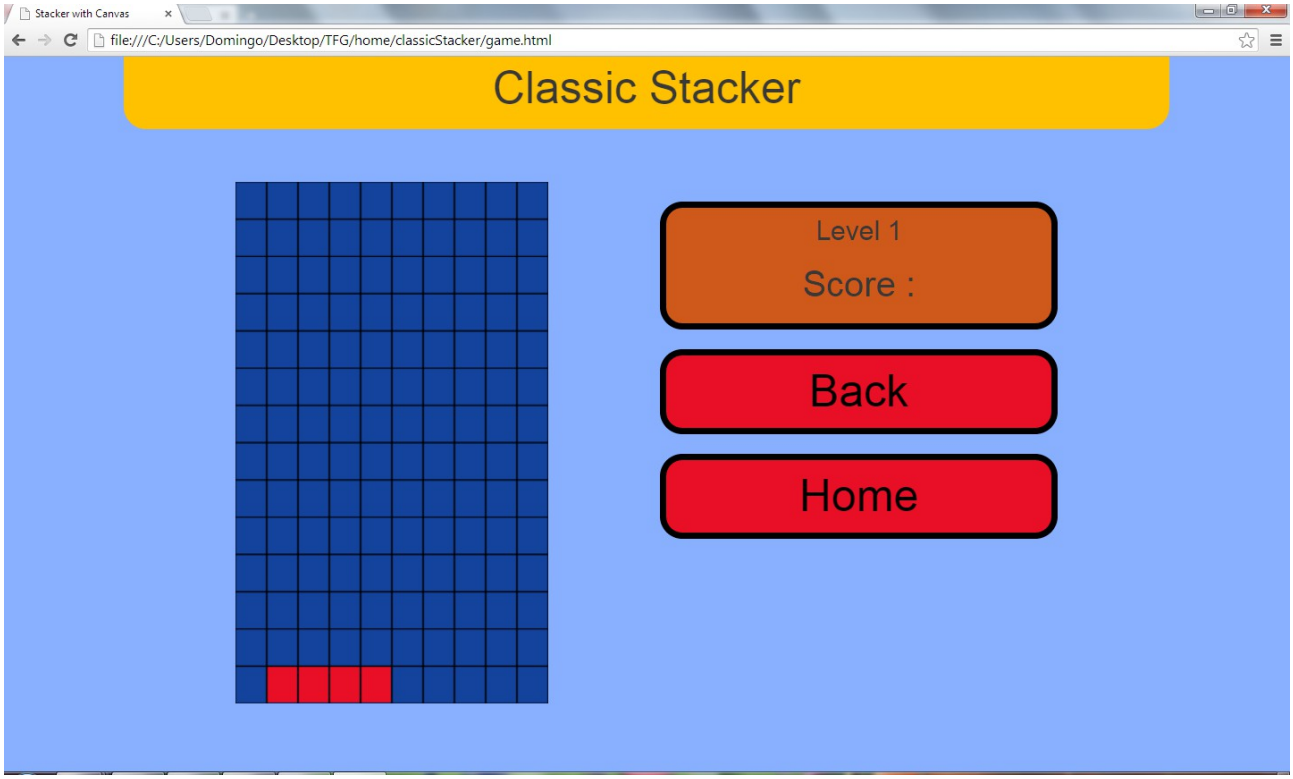


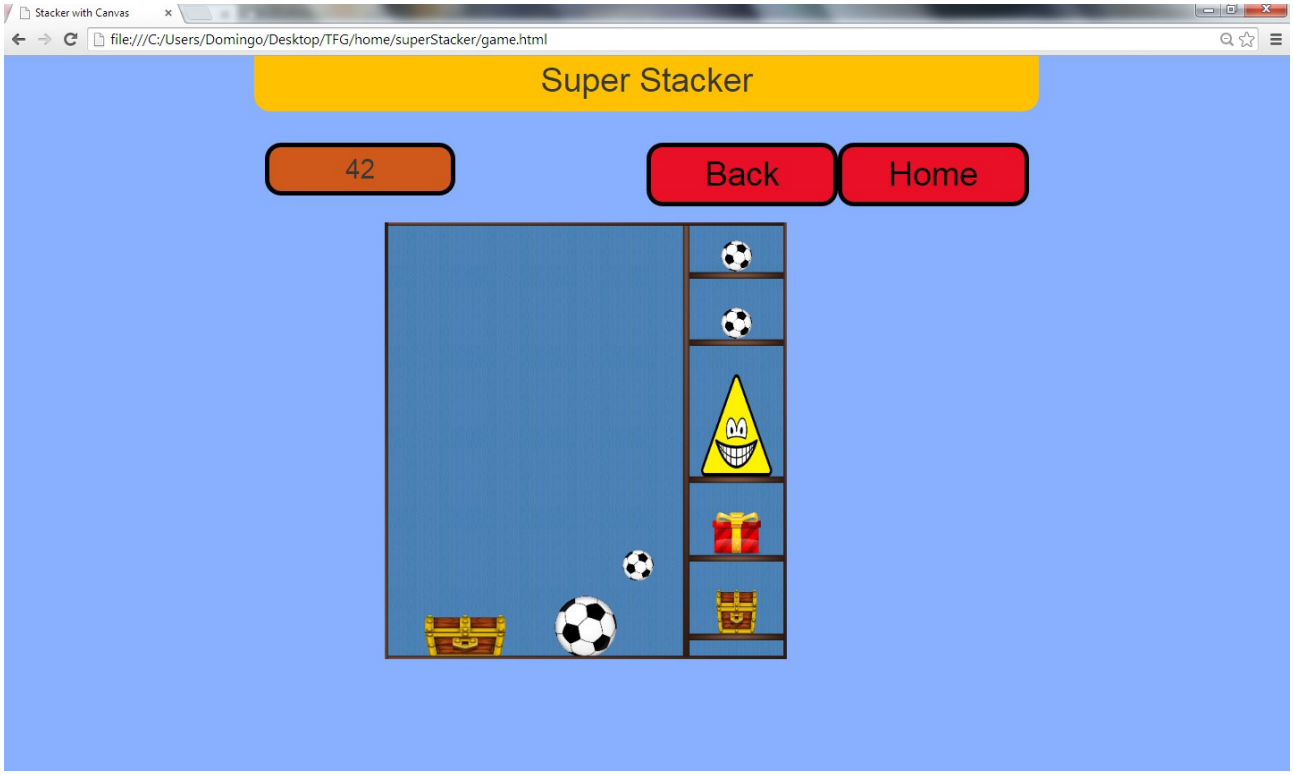
Opera Browser



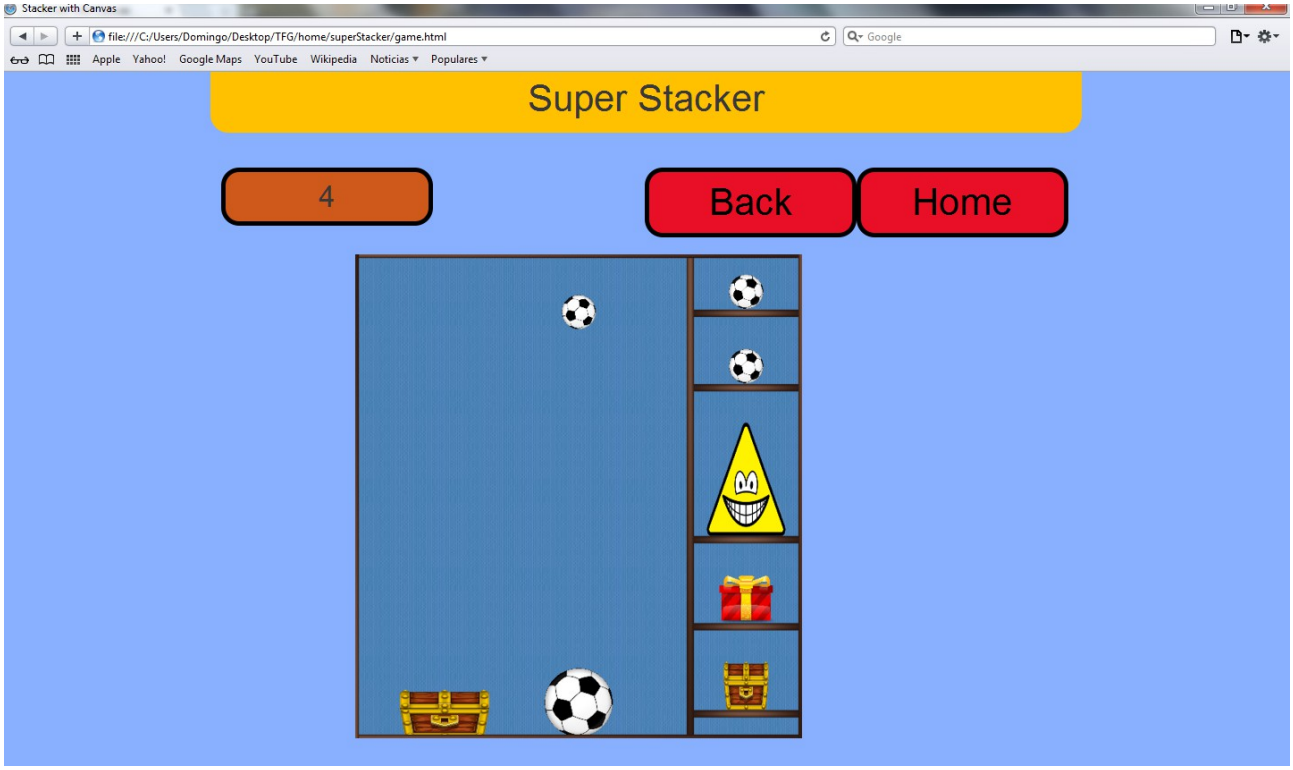


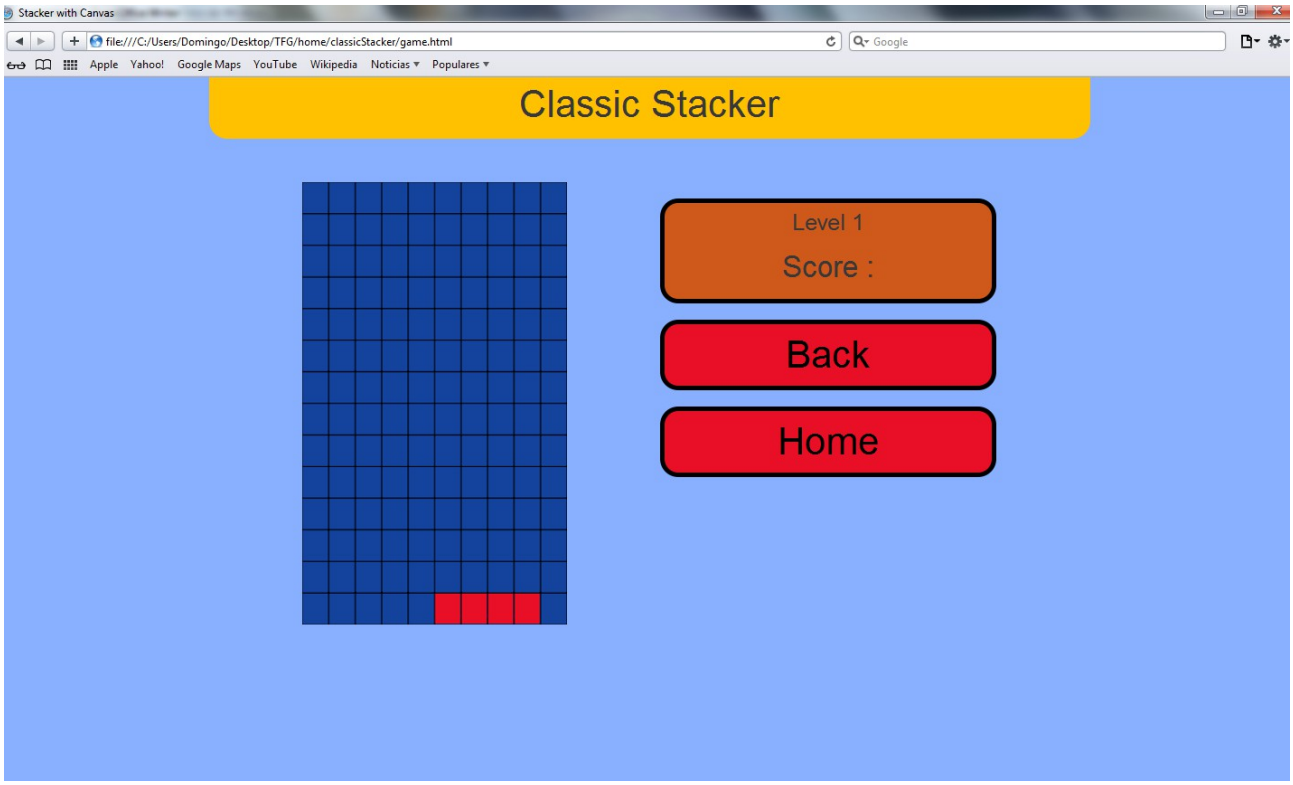
Google Chrome





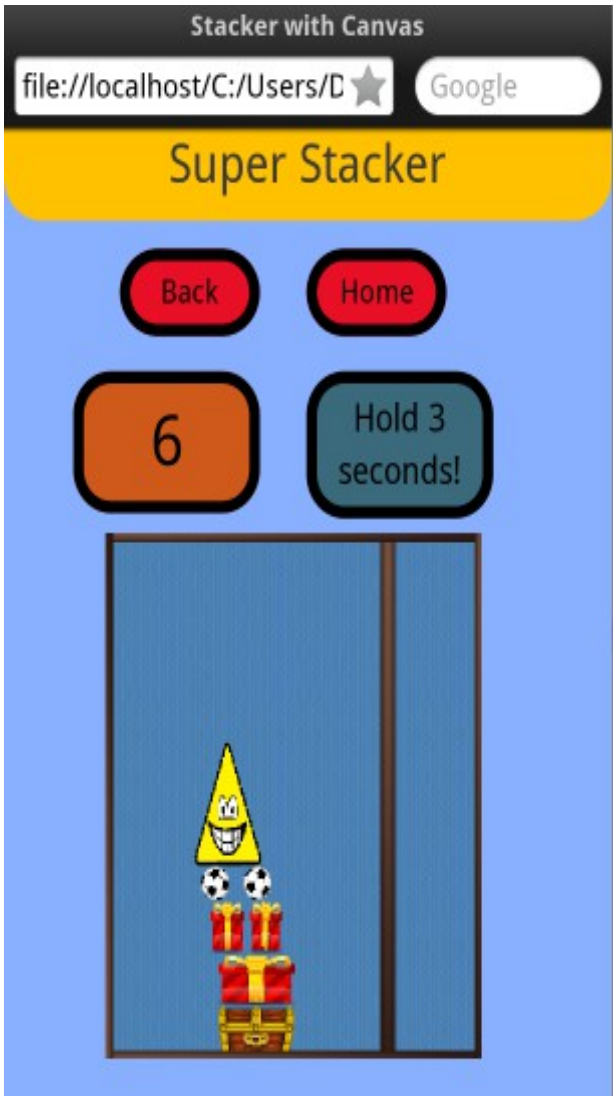
Safari

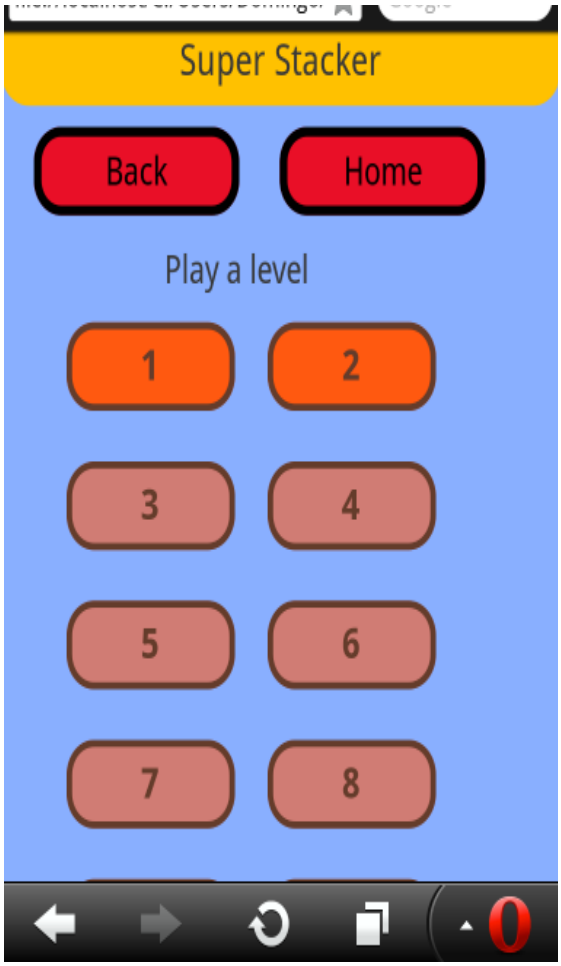

Stacker with Canvas
file:///C:/Users/Domingo/Desktop/TFG/home/superStacker/game.html
Apple Yahoo! Google Maps YouTube Wikipedia Noticias Populares
Super Stacker
4
Back
Home

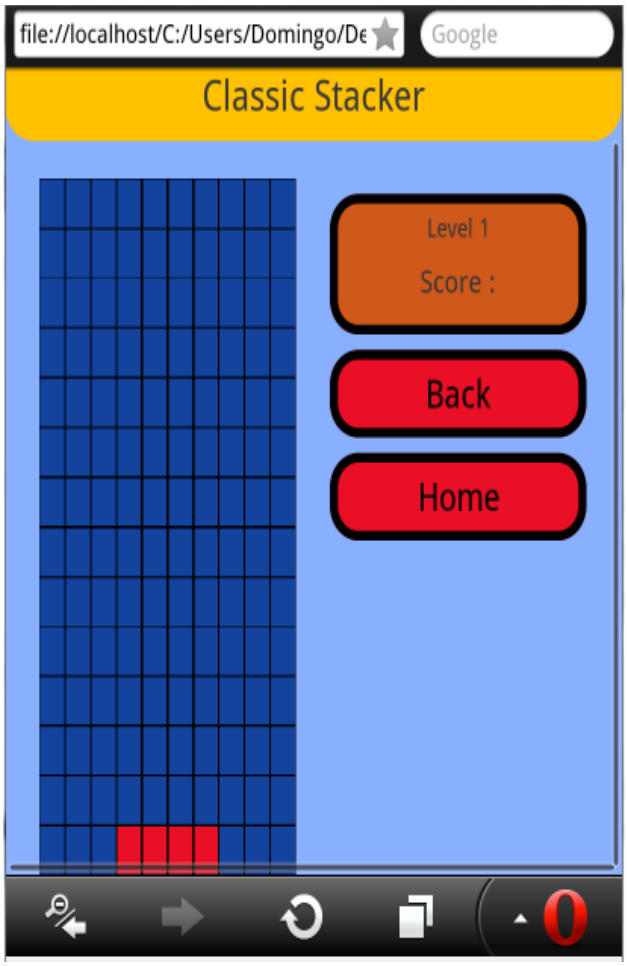

Stacker with Canvas
file:///C:/Users/Domingo/Desktop/TFG/home/classicStacker/game.html
Apple Yahoo! Google Maps YouTube Wikipedia Noticias Populares
Classic Stacker
Level 1
Score :
Back
Home

-Pruebas navegadores Móvil

El siguiente paso de las pruebas será comprobar cómo será la visulización en los dispositivos móviles. Para ello, vamos a utilizar una aplicación que simula varios dispositivos distintos con distinta anchura de pantalla para ver como se verán las pantallas (**Opera Mobile Emulator)**.







En las imágenes podemos observar cómo se han respetado los diseños planteados durante la etapa 2, los mencionados tanto en las versiones PC como en las versiones móvil.

-PhoneGap build

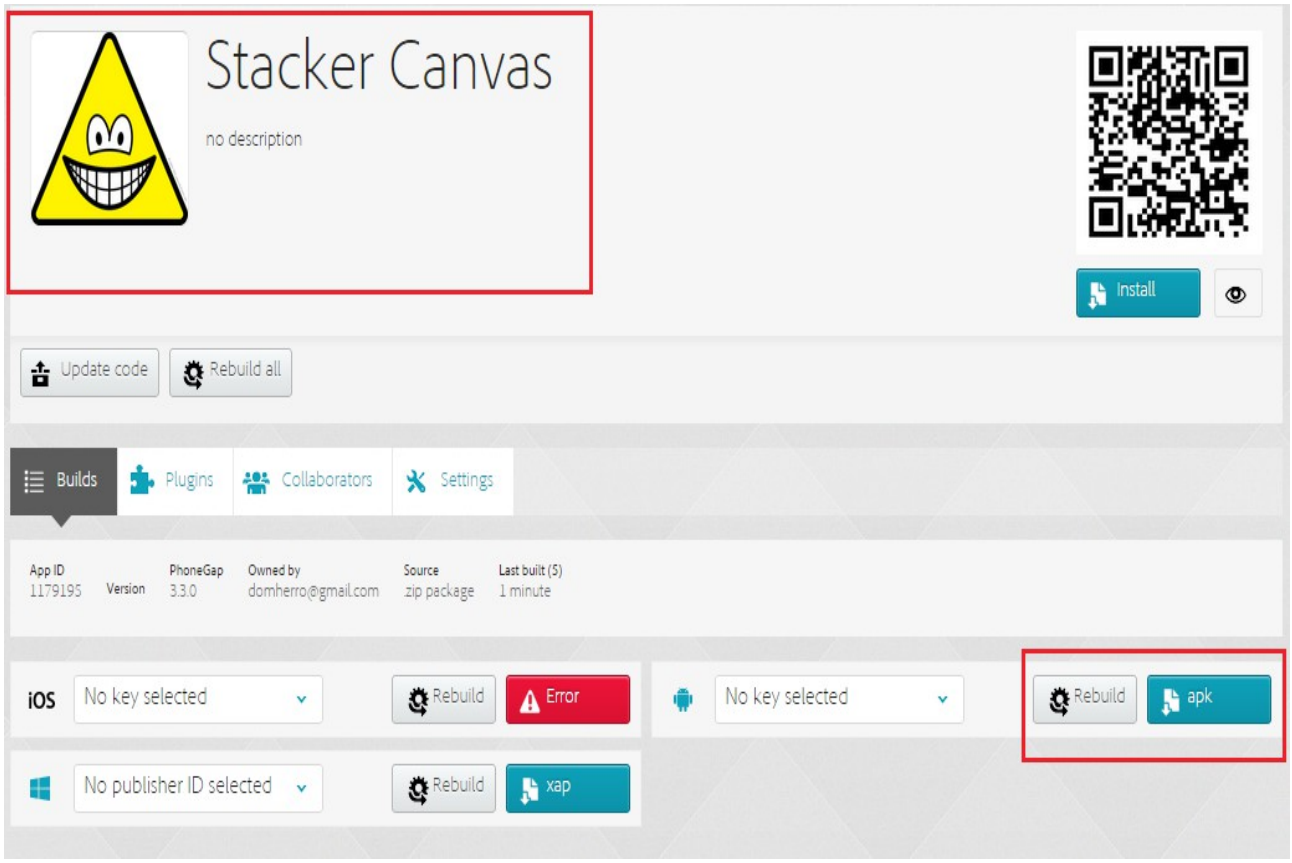
PhoneGap build nos va a permitir crear una app basándonos en las tecnologías web.

Esta aplicación web la podemos encontrar en phonegap.com (Adobe).

PhoneGap Build es sin lugar a dudas un sistema muy bien construido, gracias al uso de la nube, donde podemos convertir nuestras aplicaciones web en móviles, si bien el framework trabaja con siete plataformas, PhoneGap Build sólo compila para seis: iOS, Android, BlackBerry OS, Symbiam, Web OS y Windows Phone, integrado hace muy poco. Para Bada no es posible compilar las aplicaciones con PhoneGap Build, pues en la versión 1.7.0 del framework es la que viene con pleno soporte para la plataforma de Samsung. Una vez nos hemos registrados en PhoneGap Build,

podemos crear nuestras aplicaciones, donde tenemos la posibilidad de enviar el código a través de un repositorio git, o bien en un archivo .zip que este alojado en nuestra computadora. Es importante destacar que este proyecto web debe tener un archivo index.html, pues este es el *main* o inicio de la aplicación, es posible darle el nombre e icono que va a usar la aplicación al ser instaladas en los diferentes sistemas operativos móviles.

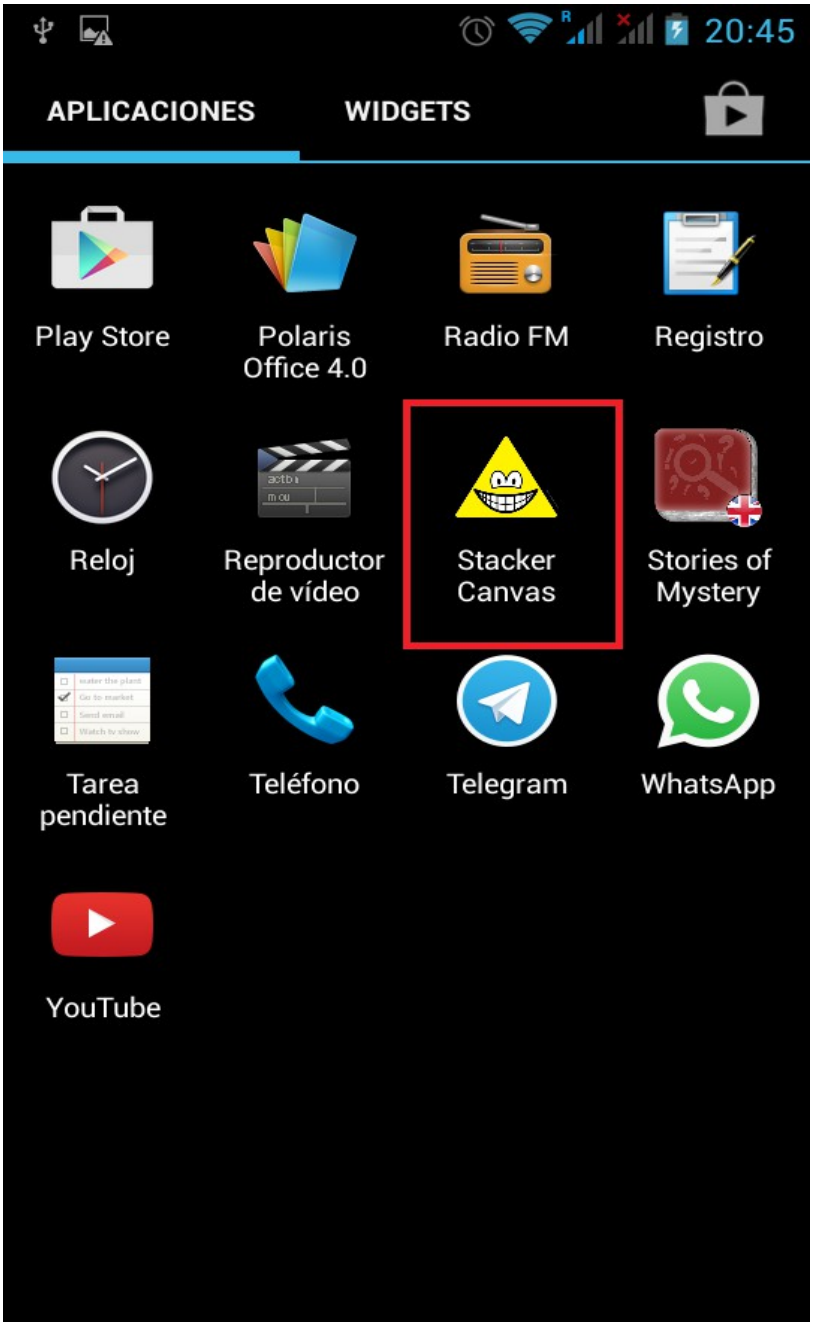
A continuación, vamos a ver una imagen de la interfaz de phonegap build.



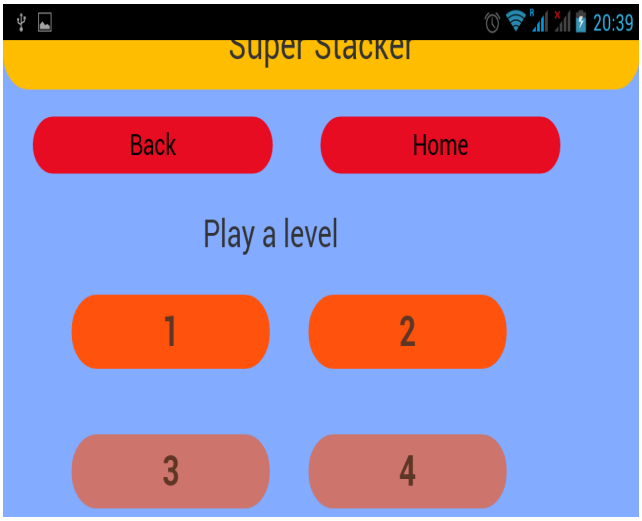
Como podemos observar, el proyecto subido tiene el nombre de STACKER CANVAS con un icono personalizado. Las opciones más importantes se encuentran en la parte inferior de la imagen, en la cual nos podemos percatar de las plataformas ofrecidas para generar las apps que deseemos. Es posible incluso instalar la aplicación a través de código QR. Si optamos por generar un paquete de instalación para la plataforma android, simplemente se ha de escoger dicha opción. Tardará unos segundos, dependiendo del tamaño de la aplicación, y nos ofrecerá obtener el archivo generado que instalaremos en el dispositivo móvil con el sistema android.

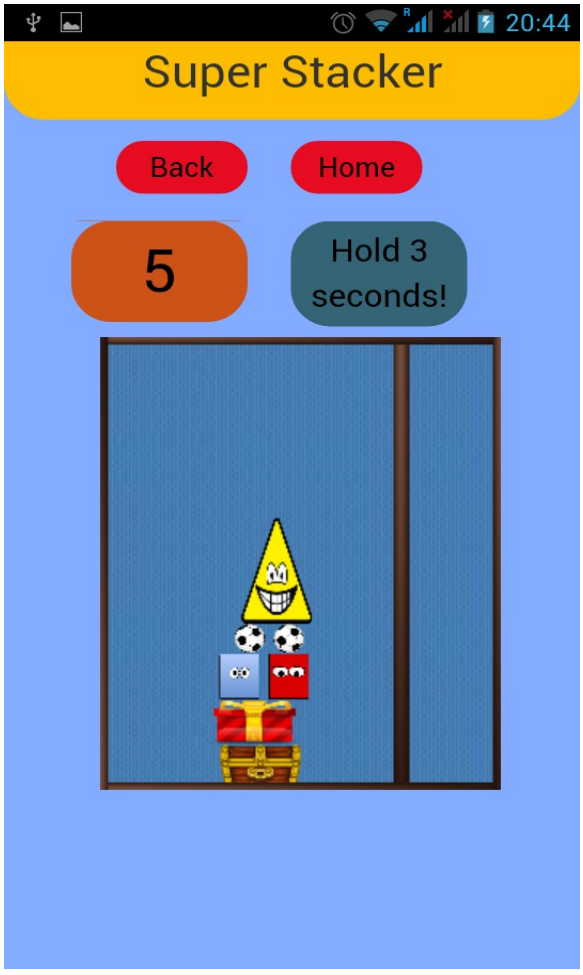
A continuación, vamos a mostrar imágenes de varias pantallas de la aplicación obtenidas a través de un teléfono móvil con sistema operativo android y con la apk instalada y generada a través de phonegap build.

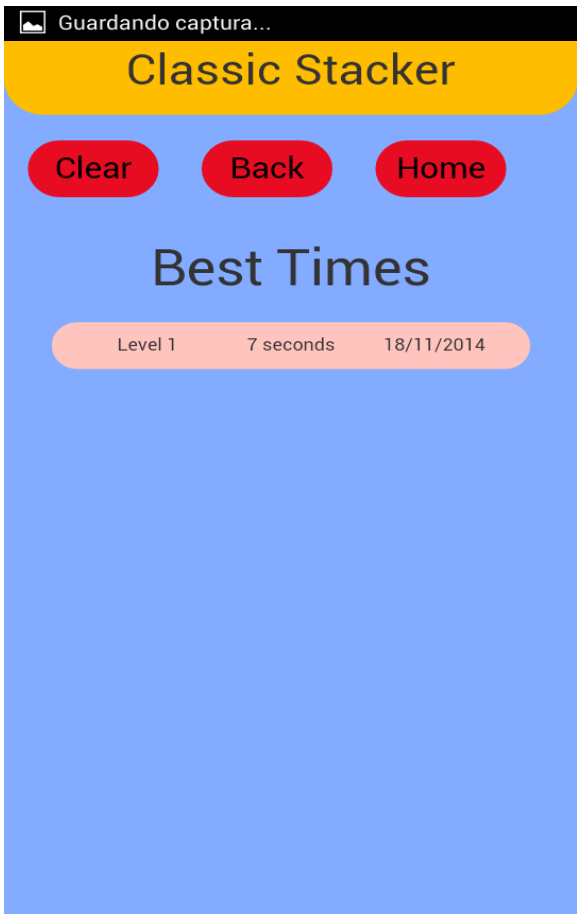
Primeramente hacemos llegar hasta el teléfono móvil la apk generada. Nos colocamos en la ruta donde se ha colocado el archivo apk y pulsar sobre el archivo para que el sistema lo instale. Una vez que finalice, nos ha de aparecer en la lista de aplicaciones del teléfono

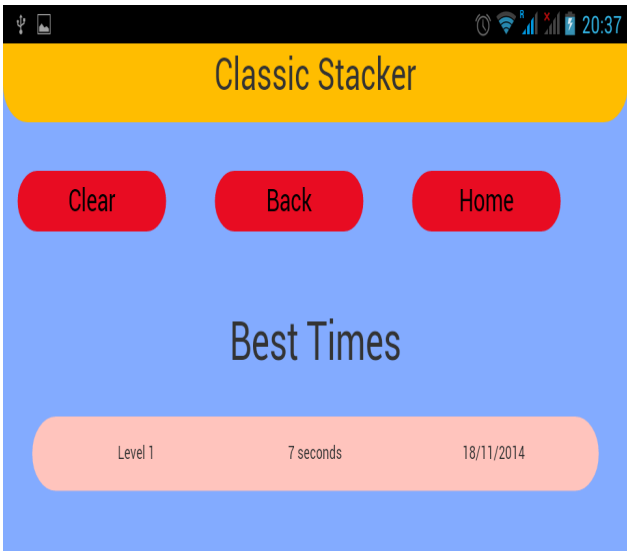


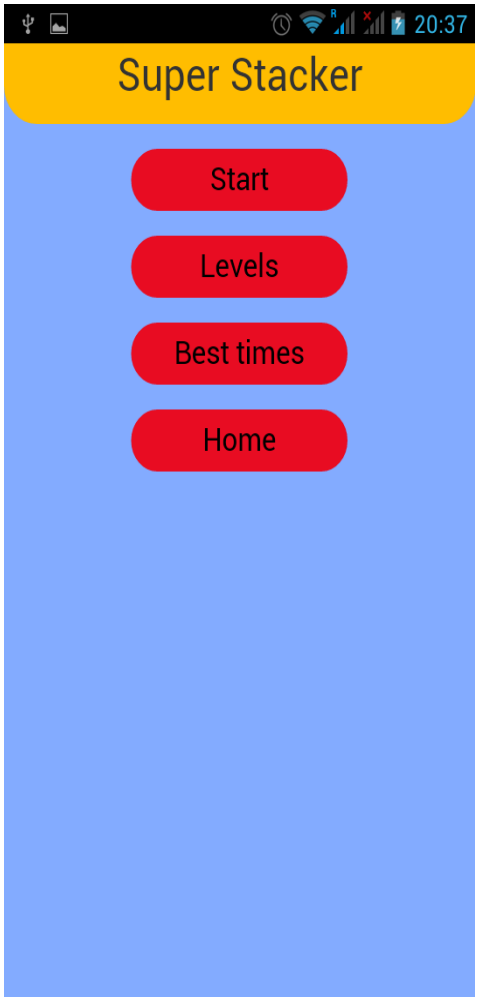
A partir de ahora ya es posible utilizar Stacker canvas como si fuera una app de android. En las siguientes imágenes podemos ver las pantallas más importantes de la aplicación y su correcto funcionamiento (portrait y landscape).

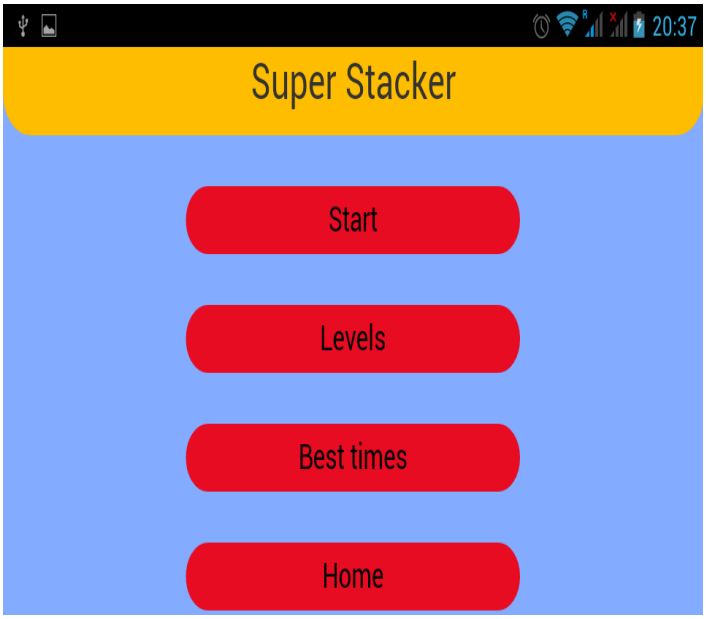


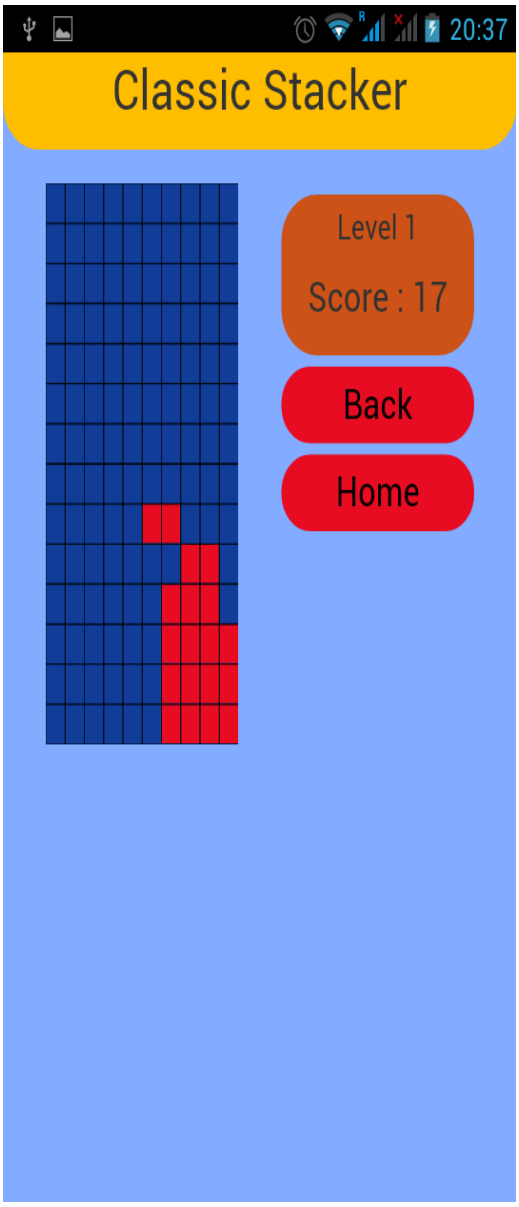


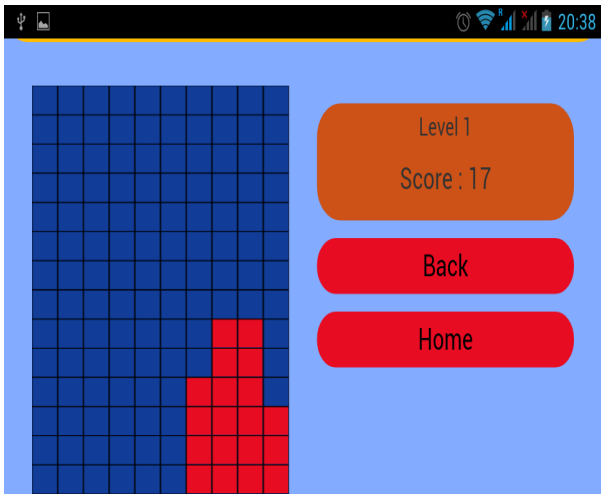

Guardando captura...
Classic Stacker
Clear
Back
Home
Best Times
Level 1
7 seconds
18/11/2014

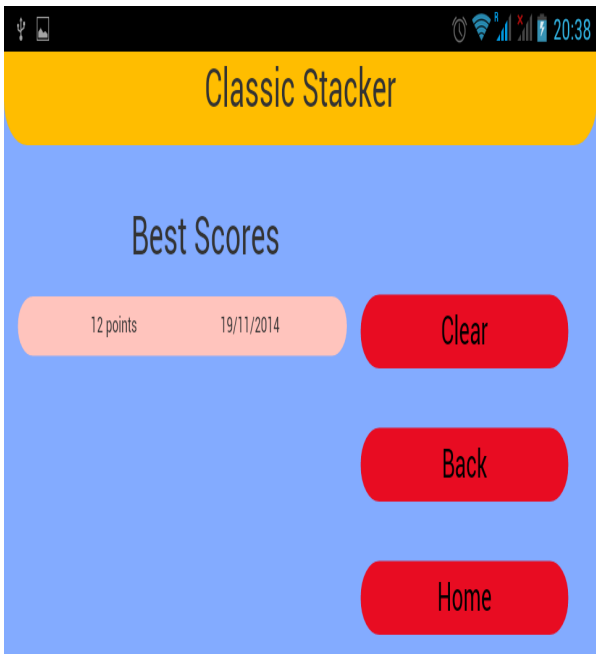

20:37
Classic Stacker
Clear
Back
Home
Best Times
Level 1
7 seconds
18/11/2014


20:37
Super Stacker
Start
Levels
Best times
Home


20:37
Super Stacker
Start
Levels
Best times
Home


20:37
Classic Stacker
Level 1
Score : 17
Back
Home

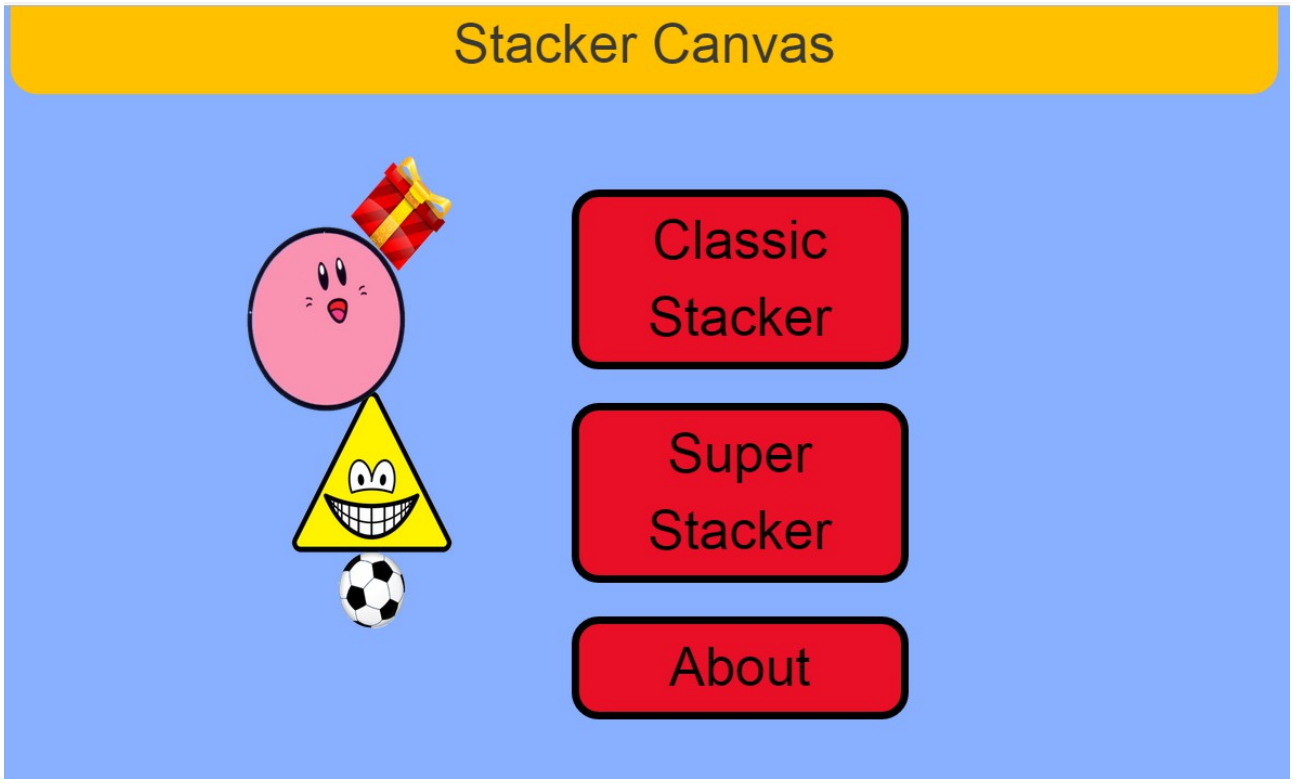

20:38
Level 1
Score : 17
Back
Home


20:38
Classic Stacker
Best Scores
12 points
19/11/2014
Clear
Back
Home

8. Guía usuario

A continuación vamos a desarrollar una pequeña guía de usuario en la cual mostraremos los aspectos más importantes de la aplicación y hacia donde lleva cada elección ofrecida. Resaltar que la web se ha hecho de una manera sencilla para ayudar al usuario a encontrar lo que busca de una manera rápida.

-Pantalla Principal

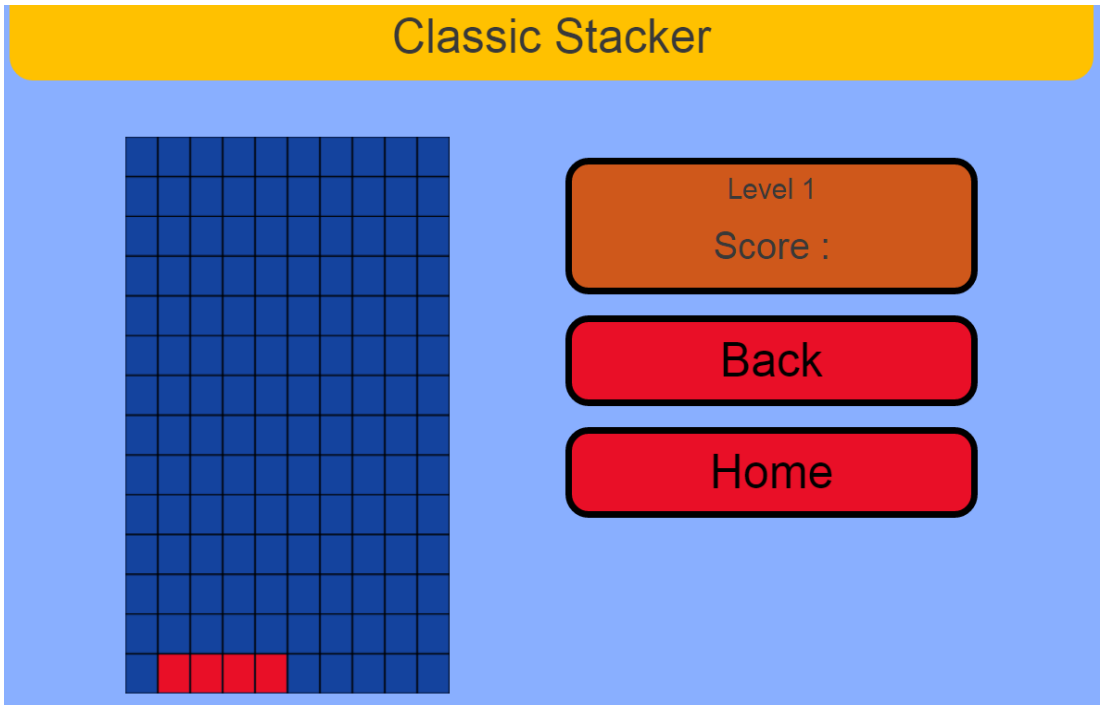


Este será el aspecto que tendrá la pantalla principal a la que se debería acceder por primera vez . En esta pantalla, como se puede observar, se da a elegir la modalidad de juego.

-Classic Stacker

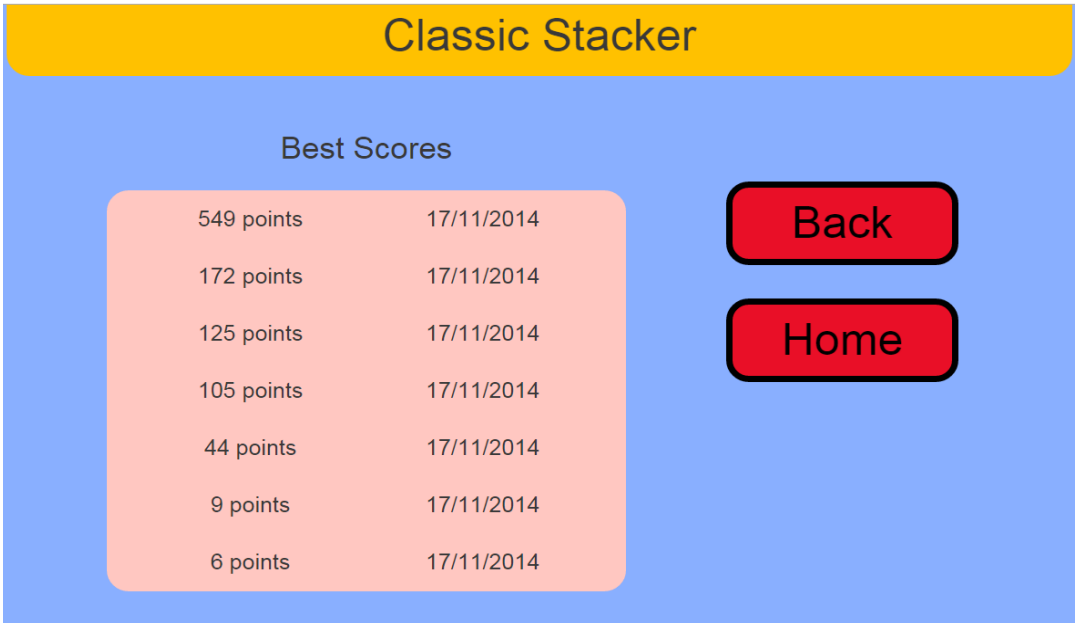
Si se escoge la modalidad de juego classic stacker verá la pantalla en la cual se da la posibilidad de jugar, ver las mejores puntuaciones conseguidas hasta el momento almacenadas en el navegador o volver a la página principal.

-Play Classic Stacker



Pantalla sencilla en la cual se puede comenzar el juego de esta modalidad. Aquí se decide cuando parar la fila de cuadrados roja que se está moviendo horizontalmente de izquierda a derecha. Pulsando sobre el lienzo, hará que la fila se pare y el juego continúe según las reglas (Ver apartado Reglas).

-Best Scores Classic Stacker

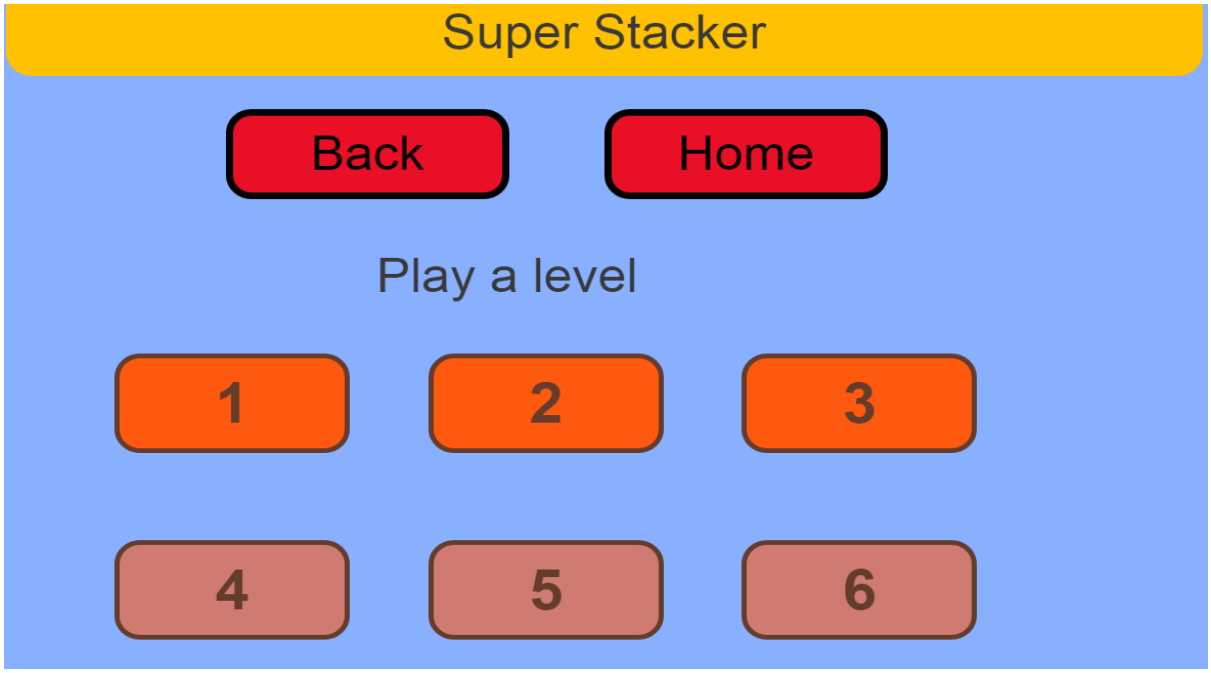


Cuando se ha jugado a classic stacker, las puntuaciones al terminar la partida se van almacenando en un espacio reservado del navegador web. En esta pantalla que mostramos podemos ver las mejores puntuaciones conseguidas hasta el momento (máximo 9).

-Super Stacker

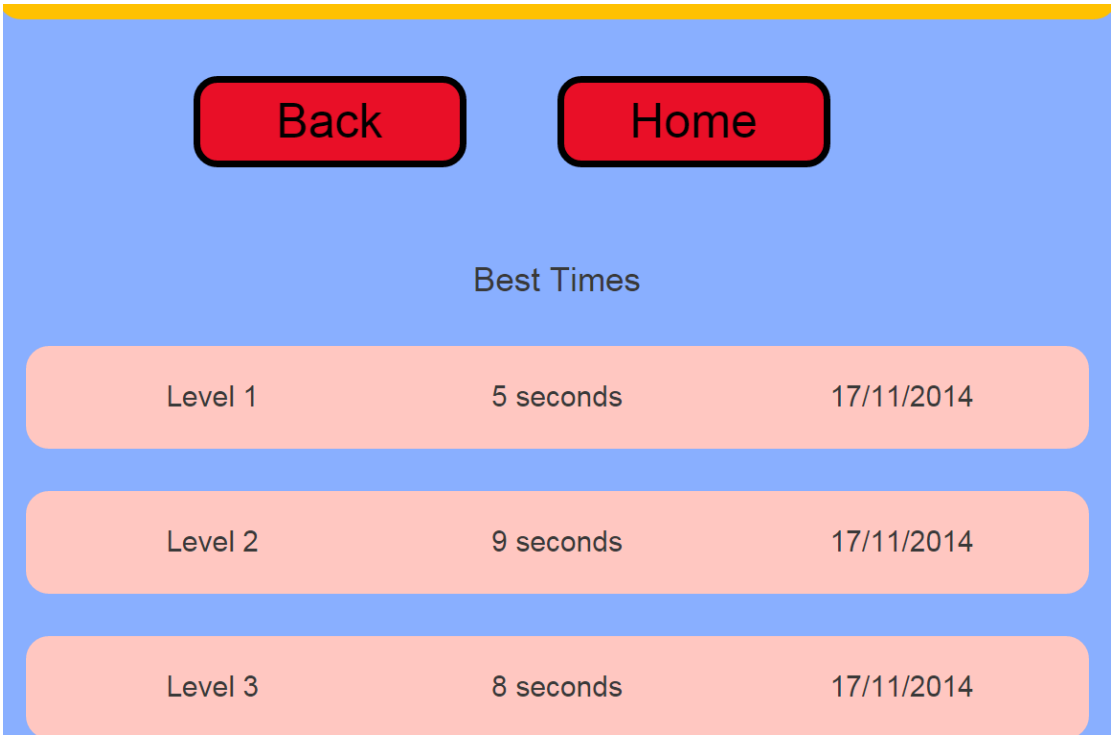
En la imagen vemos el menú que se mostrará cuando se accede a la modalidad de juego super Stacker. “Start” empezará la partida desde el nivel 1. “Levels” permitirá escoger el nivel que se quiera jugar, dando a elegir como máximo el último que se consiguió alcanzar. El resto de niveles hay que desbloquearlos. “Best Times” va a mostrar por cada nivel tiempo que menos se ha tardado en superar la prueba.

-Levels



La pantalla “Levels” va a mostrar con un color más llamativo los niveles a los cuales se puede acceder de manera directa. En la imagen, el nivel máximo que se puede escoger es el nivel 3, mientras que el resto de niveles aparecerá con un color más oscuro. Los niveles se desbloquean cuando se ha llegado hasta el.

-Best Times



En esta pantalla se pueden visualizar los mejores tiempos de cada nivel, así como la fecha del logro.

9. Conclusiones y experiencia personal

La realización del trabajo fin de grado me ha permitido, además de obtener una gran cantidad de conocimientos, poner en práctica muchos de los conocimientos aprendidos durante todos estos años en la Universidad de Las Palmas de Gran Canaria. Es importante resaltar que la rama de intensificación escogida durante el último año del graduado (Tecnologías de la Información) me ha permitido implantar de una manera más rápida la aplicación, ya que esta basada puramente en contenido web. No obstante, siempre he intentado seguir un orden y una metodología que asignaturas de otras ramas como Ingeniería de software I y II me han ayudado muchísimo.

Uno de los aspectos más importantes de este proyecto ha sido el uso de nuevas tecnologías y de nuevas herramientas de software que ayudan a generar y mantener una gran cantidad de aplicaciones basadas en otras tecnologías. La utilización de HTML5 con CANVAS ha sido un descubrimiento por mi parte, ya que esta tecnología puede aportar a la web un sinfín de nuevas oportunidades de manera nativa sin necesidad de instalar software de terceros como pueda ser flash. Además, la utilización de Local Storage abre las puertas a un almacenamiento local potente y flexible, pudiendo en unos años sustituir el uso de cookies.

Otros aspectos a destacar incluyen el uso de frameworks en la aplicación tales como bootstrap 3 para garantizar un diseño adaptable. El acceso a la web a través de un gran número de dispositivos, todos ellos distintos con dimensiones de pantalla diferentes, ha dado pie a que la maquetación se haya que tener muy en cuenta a la hora de realizar una página web.

Así mismo, utilizar un motor físico para una de las modalidades de juego ha supuesto un duro trabajo puesto que era algo nuevo para mi, pero finalmente ha resultado muy gratificante ver el resultado obtenido y la lógica aplicada hasta llegar a simular un mundo con las reglas que se pretendían.

Finalmente, comentar que la jugabilidad y disponibilidad de la aplicación desarrollada siempre se pueden mejorar, proponiendo sistemas tales como un creador de niveles, el uso de un servidor externo para almacenar logros a nivel general o dar la posibilidad de desbloquear niveles especiales a cambio de una pequeña cantidad económica.

10. Fuentes de Información

http://www.createjs.com/#!/EaselJS

http://getbootstrap.com

http://jquery.com

http://stackoverflow.com

https://code.google.com/p/box2dweb/

http://www.youtube.com/watch?v=tQRiuxJZ_d0

http://phonegap.com

http://notepad-plus-plus.org

http://juegos.canvas.ninja

http://www.jonnystromberg.com/box2d-javascript-tutorial

http://box2d-js.sourceforge.net/index2.html

https://www.apple.com/es/safari/

http://www.opera.com/es/developer/mobile-emulator

https://www.openoffice.org/es/