ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERA INFORMÁTICA

Título del proyecto:

DESARROLLO DE JUEGOS MULTIPLATAFORMA:

"DONDESTAS NICOLAS, SLIDEIT, CONDY CRUSH"

Eduardo F. Sesma Martínez

Tutor: Jesús Villadangos Alonso

Pamplona, 23 de junio de 2016

INDICE

1 INTRODUCCION

1.1 APLICACIONES BASADAS EN DISPOSITIVOS

Hoy en día, cuando utilizamos el termino aplicación lo primero que se nos viene a la cabeza es un teléfono móvil ejecutando un programa que puede ser cualquier cosa, por ejemplo, desde un juego, un chat, hasta un procesador de texto o un reproductor de música. Con esto simplemente podemos ver que la idea de aplicación ya ha dejado de referirse al típico ordenador de escritorio o portátil corriendo cualquier software, es decir actualmente existen otro tipo de dispositivos que prácticamente usamos todos los días tanto o más que el clásico ordenador.

De todo esto, surge el concepto de aplicación móvil o app, al cual nos referimos para denominar a las aplicaciones informáticas diseñadas para ser ejecutadas en teléfonos inteligentes, tabletas y otros dispositivos móviles y que permiten al usuario efectuar una tarea concreta de cualquier tipo tanto profesional, de ocio, educativas, o de acceso a servicios, etc.

Pero antes de todo este, vamos a hacer un pequeño repaso de la historia de las aplicaciones móviles, remontándonos mucho antes incluso que a la aparición de los Smartphone.

Las primera aplicaciones datan a finales de los años 90, varios ejemplos de estas son la agenda, los juegos arcade, gestión de sms, etc. Sus funciones eran muy limitadas ya que su cometido era muy especifico y sus diseños eran muy simples y elementales. Conforme avanzaba la tecnología, estas iban evolucionando, gracias a la tecnología WAP, que permitía a su vez un nuevo desarrollo de teléfonos móviles, cada vez con más funcionalidades y más potentes. Empezaron a surgir entonces nuevas empresas con nuevas ideas y modelos de teléfonos móviles, con nuevo y actualizado hardware y software con el desarrollo de potentes sistemas operativos, básicamente hablaremos de Apple y sus iPhones y Android que más tarde fue adquirido por Google. Así pues en cuanto estas dos empresas se ponen en escena es donde comienza realmente el panorama que conocemos hoy en día, con la infinidad de smartphones existentes y aplicaciones ya que el resto de empresas que trabajaban con la telefonía móvil también deciden iniciar el desarrollo de nuevas tecnologías para conseguir dispositivos cada vez más innovadores y más potentes , por ejemplo Nokia y Sony Erickson entre otras funcionaban con Symbian Os, también Black Berry, Samsung con Palm Os, etc.

Cada vez había móviles más potentes, cada vez había más modelos y más variedad de dispositivos, pero hubo un punto de inflexión donde se empezó a formar la esencia de lo que conocemos actualmente y con esto nos referimos al lanzamiento del App Store por parte Apple, una tienda online dentro del propio móvil donde se puede adquirir todo tipo de funcionalidad para tu dispositivo, desde un traductor, un juego, una aplicación de retoque fotográfico entre otros. El resto de empresas tuvo que esperar para poder participar en este mercado hasta que facilitación el acceso a los SDK (software developments kits) a todo el mundo, es aquí donde el abanico de aplicaciones se incrementa rápidamente y entran en

juego miles de nuevas empresas y es donde se empieza realmente la competencia entre unos y otros.

Llegados a este punto se pueden clasificar las aplicaciones móviles en tres tipos:

-Aplicaciones Nativas, pensadas y desarrolladas de forma específica para un sistema operativo.

-Aplicaciones Web, son las que corren sobre un navegador web del dispositivo móvil

-Aplicaciones Hibridas, combinación de las anteriores.

En este proyecto nos centraremos en estas últimas ya que toman lo mejor de las otras dos y permite el uso de tecnologías multiplataforma como html, javascript, css y además permite el uso de las características y dispositivos del teléfono. Actualmente existen varias plataformas para crear este tipo de aplicaciones pero las más conocidas probablemente son PhoneGap, Apache Cordova, y Game Maker, este ultimo más centrado en la creación de juegos y es el que utilizaremos pero lo explicaremos más adelante.

Podemos ver por lo tanto que la evolución del software se ha visto fuertemente vinculada, sobre todo en la ultima década, al diseño y desarrollo de soluciones para estos dispositivos sin dejar de lado tampoco a los computadores que han seguido de forma paralela. Así pues podemos diferenciar estos dispositivos más actuales en otro gran grupo, el de smartphones cuya función principal es la de servir como plataforma de aplicaciones que hagan provecho de las características del teléfono.

En la actualidad, los llevamos a todas partes, los utilizamos para cualquier función, trabajamos con ellos, en definitiva se han vuelto parte de nuestra vida. El hecho de pensar que tenemos una cámara, un reproductor de música, una consola de videojuegos, GPS, computadora y teléfono en el bolsillo y todo esto en un único dispositivo es un suceso que antes solo era pensable en películas de ciencia ficción pero que ahora es pura realidad, y todo esto gracias a los desarrolladores que se plantean como solucionar un problema teniendo en cuenta todos los elementos y características que formar parte de los smartphones (cámaras integradas, acelerómetros, conexión a internet, etc.) y aprovechándolas.

Como hemos mencionado anteriormente, la idea inicial de este proyecto es la de crear un software, en nuestro caso varios juegos, que puedan ser ejecutados en multitud de plataformas distintas lo que viene a equivaler a poder correr en distintos sistemas operativos móviles.

Entendemos entonces sistema operativo móvil como el sistema que controla un dispositivo móvil al igual que los ordenadores que utilizan Windows, Linux. En este ámbito los que más predominan son Android e IOS entre otros. Cabe destacar que los sistemas operativos móviles son mucho más simples y están orientados a la conectividad inalámbrica y a los formatos multimedia. La mayoría de ellos están diseñados según el modelo de capas (kernel, middleware, entorno de ejecución e interfaz grafica o de usuario).

Gracias a esta arquitectura el kernel es el núcleo que proporciona acceso a los distintos elementos hardware, ofrece servicios a las capas superiores a través de los controladores o drivers y organiza los procesos, el sistema de archivos y el acceso y gestión de memoria.

A su vez la capa de middleware se compone del conjunto de módulos que hacen posible la propia existencia de aplicaciones, es totalmente transparente para el usuario y ofrece servicios clave, como el motor de mensajería y comunicaciones, codecs multimedia, interpretes de páginas web y gestión de propio dispositivo y seguridad.

Asimismo el entorno de ejecución de aplicaciones consiste en un gestor de estas y un conjunto de interfaces programables para facilitar la creación de software.

Por último la interfaz de usuario facilita la interacción con el usuario y gracias a el diseño de lo visual. Incluye todos los componentes gráficos, pantallas, botones, listas, etc. y es el marco de interacción.

Como hemos explicado antes, conforme los teléfonos móviles aumentan en popularidad, los sistemas operativos con los que funcionan adquieren cada vez más relevancia. Podemos ver como se reparten el mercado según estadísticas del primer trimestre de este año 2016:

-Android 74%

-iOS 22%

-Windows Phone 3%

-Otros (entre los que destacan BlackBerry, Symbian, Firefox OS, Ubuntu Touch) 1%

Ahora pasaremos a contar brevemente los aspectos más importantes de estos sistemas operativos para comprender bien el objetivo final de este proyecto que repetimos es trabajar la multiplataforma de forma que el desarrollo de una aplicación o juego pueda correr indistintamente en cualquiera de los sistemas de los que hemos hablado.

Android

El sistema operativo Android es sin lugar a duda el líder del mercado móvil, está basado en Linux, originalmente fue diseñado originalmente para cámaras fotográficas, luego fue vendido a Google y modificado para ser utilizado en dispositivos móviles como los smartphones y más adelante en tablets, el desarrollador de este S.O. es Google, fue anunciado en el 2007 y liberado en el 2008, además de la creación de la Open Handset Alliance, compuesto por 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para celulares, esto le ha ayudado mucho a Google a masificar el sistema operativo, hasta el punto de ser usado por empresas como HTC, LG, Samsung, Motorola entre otros.

Android Inc., es la empresa que creó el sistema operativo móvil, se fundó en 2003 y fue comprada por Google en el 2005 y 2007 fue lanzado al mercado. Su nombre se debe a su inventor, Andy Rubin.

Las aplicaciones para Android se escriben y desarrollan en Java aunque con unas APIS propias por lo que las aplicaciones escritas en Java para PC y demás plataformas ya existentes no son compatibles con este sistema.

Una de las grandes cualidades o características de este sistema operativo es su carácter abierto. Android se distribuye bajo dos tipos de licencias, una que abarca todo el código del kernel y que es GNU GPLv2 lo que implica que su código se debe poner al alcance de todos y que todos podremos hacer con este código lo que nos parezca oportuno, modificarlo, ampliarlo, recortarlo, pero siempre estaremos en la obligación de volver a licenciarlo con las misma licencia).

iOS

iOS es el sistema operativo que da vida a dispositivos como el iPhone, el iPad, el iPod Touch o el Apple TV. Su simplicidad y optimización son sus pilares para que millones de usuarios se decanten por iOS en lugar de escoger otras plataformas que necesitan un hardware más potente para mover con fluidez el sistema operativo. Cada año, Apple lanza una gran actualización de iOS que suele traer características exclusivas para los dispositivos más punteros que estén a la venta en ese momento.

Anteriormente denominado iPhone OS y fue creado por Apple. Realmente es un derivado de Mac OS X que se desarrollo originalmente para el iPhone y se lanzó en el año 2007, aumentando así el interés en este con la aparición de iPod Touch e iPad, que son dispositivos con las capacidades multimedia del iPhone pero sin la capacidad de hacer llamadas telefónicas. En si su principal revolución es una combinación casi perfecta entre hardware y software, el manejo de la pantalla multi-táctil que no podía ser superada por la competencia hasta el lanzamiento del celular Galaxy S I y II por parte de Samsung.

Las aplicaciones para estos dispositivos se programaban hasta no hace mucho en lenguaje Objective C, no obstante la propia empresa Apple ha creado un nuevo lenguaje llamado SWIFT con el cual quieren dar más facilidad, agilidad y rapidez al desarrollo de estas aplicaciones.

Windows Phone

Anteriormente llamado Windows Mobile es un sistema operativo móvil compacto desarrollado por Microsoft, se basa en el núcleo del sistema operativo Windows CE y cuenta con un conjunto de aplicaciones básicas, actualmente va por la versión 10. Está diseñado para ser similar a las versiones de escritorio de Windows estéticamente y existe una gran oferta de software de terceros disponible para Windows Mobile, la cual se puede adquirir a través de la tienda en línea Windows Marketplace for Mobiles.

Para poder desarrollar aplicaciones para este sistema se usa el lenguaje de programación C# o Visual Basic .NET a través de la propia plataforma de Microsoft Visual Studio.

BlackBerry 6

BlackBerry es un sistema desarrollado por Research In Motion el cual fue presentado en el WES 2010 junto con un video promocional donde se muestra algunas novedades.

La mejora sustancial de este sistema se encontrara en la incorporación de touchscreen (Pantalla Táctil) en los dispositivos, aunque los equipos que cuenten con un TouchPad o TrackPad clásico también podrán ejecutarlo ya que ejerce casi la misma función. La empresa se ha enfocado también en la parte multimedia hacia el usuario, sin dejar a un lado la parte profesional, también se muestra la integración de las redes sociales y la mensajería instantánea.

Symbian

Fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia como la más importante, Sony Ericsson, Samsung, Siemens, BenQ, Fujitsu, Lenovo, LG, Motorola, esta alianza le permitió en un momento dado ser unos de los pioneros y más usados.

El objetivo de Symbian fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm OS ,el Smartphone de Microsoft. Técnicamente, el sistema operativo Symbian es una colección compacta de código ejecutable y varios archivos, la mayoría de ellos son bibliotecas vinculadas dinámicamente (DLL) y otros datos requeridos, incluyendo archivos de configuración, de imágenes y de tipografía, entre otros recursos residentes.

Symbian se almacena, generalmente, en un circuito flash dentro del dispositivo móvil. Gracias a este tipo de tecnología, se puede conservar información aun si el sistema no posee carga eléctrica en la batería, además de que le es factible reprogramarse, sin necesidad de separarla de los demás circuitos.

Las aplicaciones compatibles con Symbian se desarrollan a partir de lenguajes de programación orientados a objetos como C++, Java y Visual Basic, entre otros.

Firefox OS

Este es un sistema operativo móvil, basado en HTML5 con núcleo Linux, para smartphones y tabletas. Es desarrollado por Mozilla Corporation bajo el apoyo de otras empresas como Telefónica y una gran comunidad de voluntarios de todo el mundo.

Este sistema operativo está enfocado especialmente en los dispositivos móviles incluidos los de gama baja. Está diseñado para permitir a las aplicaciones HTML5 comunicarse directamente con el hardware del dispositivo usando JavaScript y *Open Web APIs*. Ha sido mostrado en smartphones y Raspberry Pi, compatibles con Android.

Empresas como LG Electronics, ZTE, Huawei y TCL Corporation se han comprometido a la fabricación de dispositivos con Firefox OS.

Ubuntu Touch

Esta distribución de Ubuntu es un sistema operativo móvil basado en Linux. Es desarrollado por Canonical Ltd.. Presentado el 2 de enero de 2013 al público mediante un anuncio en la web de Ubuntu, culmina el proceso de Canonical de desarrollar una interfaz que pueda utilizarse en ordenadores de sobremesa, portátiles, netbooks, tablets y teléfonos inteligentes.

Una de sus características más destacadas es su pantalla de inicio sin sistema de bloqueo/desbloqueo que funciona con un nuevo sistema de gestos, y que se aprovecha para mostrar notificaciones.

1.2 MULTIPLATAFORMA

Un factor importante en la actualidad es la portabilidad, es decir hacer un software que se pueda adaptar a varias plataformas o sistemas operativos distintos independientemente de su estructura y contenido y sin que esto de lugar a generar muchas aplicaciones iguales para hacer lo mismo en diferentes sistemas, es decir crear una aplicación útil para cualquier entorno.

Como ya hemos comentado en los puntos anteriores, en este proyecto, esta parte adquiere especial relevancia ya que nos queremos centrar en desarrollar un juego que sea capaz de ejecutarse en cualquier dispositivo, ya sea un móvil iPhone, Android o Windows Mobile, incluso a través de un navegador web o como un juego de escritorio de cualquier computador y todo esto haciendo un único diseño y desarrollo del juego, sin tener que implementarlo en cada lenguaje especifico.

Vamos a explicar con más detalle lo que entendemos por multiplataforma, definiéndose este concepto como un atributo conferido a programas informáticos y conceptos de computo que son implementados e interoperan en múltiples plataformas informáticas.

Este puede dividirse en dos tipos, uno que requiere una compilación individual para cada plataforma que le da soporte, y el otro se puede ejecutar directamente en cualquier plataforma sin preparación especial.

Realmente podemos acotar el termino plataforma a lo que entendemos como una combinación de hardware y software para ejecutar aplicaciones informáticas. Una plataforma puede ser descrita simplemente como un sistema operativo o una arquitectura de ordenador o incluso podría ser la combinación de ambos. Así pues podemos observar la existencia de dos clases:

-Plataformas de hardware: Se refieren a la arquitectura del ordenador o procesador, por ejemplo los CPUs x86 x86-64 constituyen una de las más comunes arquitecturas de computadores en uso de ordenadores de propósito general, estas maquinas suelen ejecutar una versión de Microsoft Windows aunque también pueden ejecutar otros sistemas operativos como Linux o Mac Os X. Los smartphones poseen una arquitectura ARM y corren los sistemas operativos que hemos descrito en puntos anteriores.

-Plataformas de software: Se refieren al sistema operativo o entorno de programación, aunque mas comúnmente se refiere a la combinación de ambos. Una notable excepción a esto es Java, que utiliza un sistema operativo independiente de la máquina virtual para cada código compilado, conocido como bytecode. Ejemplos de estas son Android, Java, Linux, iOS, Mac OS, Microsoft Windows, etc.

Si nos fijamos bien en el punto anterior, nos encontramos con que Java es la única excepción de que sin ser un sistema operativo es una plataforma de software. El lenguaje Java requiere de una maquina virtual en la que se ejecuta todo el código. Esto permite que el mismo ejecutable pueda correr en todos los sistemas apoyado por el software de Java a través del uso de una maquina virtual (JVM) de forma que estos ejecutables no se ejecutan de forma nativa en el sistema operativo. Así pues gracias a la Java Virtual Machine, somos capaces de de obtener acceso a servicios relacionados con el propio S.O como el disco I/O o acceso a red.

En definitiva para que un software pueda ser considerado multiplataforma , debe ser capaz de funcionar en más de una arquitectura de ordenador o sistema operativo, pero esto es una tarea que consume mucho tiempo ya que los diferentes sistemas operativos tienen distintas interfaces de programación de aplicaciones (API).

El hecho de que un determinado sistema operativo se pueda ejecutar en diferentes arquitecturas de computadora no quiere decir que el software escrito para ese sistema operativo automáticamente funcione en todas las arquitecturas que soportan el sistema operativo, asimismo un programa que se escriba en un popular lenguaje de programación, no tiene por qué funcionar en todos los sistemas operativos que soporten la programación de dicho lenguaje o incluso en el mismo sistema operativo en una arquitectura diferente.

Un caso particular de aplicaciones multiplataforma son las aplicaciones web o webapps de las que hemos hablado anteriormente. Estas como ya sabemos son descritas como multiplataforma ya que se pueden acceder a ellas desde cualquiera de los diversos navegadores web en cualquier sistema operativo ya sea de computador de escritorio o móvil. Tales aplicaciones generalmente emplean la arquitectura de sistema cliente-servidor y pueden variar ampliamente en complejidad y funcionalidad lo que complica la capacidad de la multiplataforma.

Cabe mencionar que multiplataforma es un término que también puede aplicarse a los videojuegos, ya que a la vez que existen diferentes consolas con sus particularidades como pueden ser PlayStation 4, Xbox360, Wii también existen ordenadores dedicados a la tarea de jugar e incluso dispositivos móviles. Esto es un dato interesante ya que como sabemos este proyecto se centra en la creación de varios juegos simples que podrán ser ejecutados en móviles y en un PC.

En casi cualquier ámbito de programación que escojamos, tenemos infinidad de herramientas, lenguajes y entornos disponibles que podemos elegir para desarrollar nuestro software. Toda esta maraña de opciones hace que a veces sea muy complicado decantarnos por una u otra. En nuestro caso, definiremos nuestro proyecto para el desarrollo de móviles y tablets aunque a su vez permitiremos su ejecución en computadores corrientes. Llegados a este punto es cuando realmente se valora que vamos a utilizar y como, ya que incluso aun queriendo desarrollar un aplicación multiplataforma podríamos perfectamente realizar dos implementaciones nativas distintas y hacer uso de sus plataformas por ejemplo utilizar Swift y Xcode para iOS y Java con Android Studio para Android o también C# con Visual Studio para Windows Phone o Windows 8 pero como hemos dicho, nuestra meta principal es hacerlo con una única implementación por lo cual esta opción queda descartada.

Tenemos otras herramientas multiplataforma que compilan cualquier desarrollo a código nativo, la más conocida y utilizada es Xamarin pero como la aplicación a desarrollar es un juego ésta nos es poco útil y nos encontramos con otras como Game Maker la cual ha sido la elegida.

Además existen otras herramientas basadas en HTML que también podríamos haber utilizado por los conocimientos previos que tenemos de los lenguajes influyentes en ellas utilizando PhoneGap o Apache Cordova.

Finalmente se eligió el ya mencionado Game Maker con el afán de aprender a usar esta plataforma, previamente desconocida y su lenguaje de programación denominado GM, aunque también se valoro seriamente la opción de hacerlo en PhoneGap por los conocimientos previos de HTML5 y CSS pero al final se decidió aprender cosas nuevas para que puedan aportar más en nuestra formación y conocimiento.

1.3 OBJETIVOS

En este proyecto, perseguimos varias metas, la primera de ellas es la propia motivación por crear una aplicación con una única implementación que sea capaz de ser ejecutada en multitud de dispositivos sin importar el sistema operativo y arquitectura que presenten. Esta situación parece idílica, no obstante conforme se ha ido desarrollando el proyecto, nos hemos encontrado con varias dificultades y problemas típicos que presenta este tipo de programación que realmente dependen de las plataformas donde se ejecutan, de lo que deducimos que no todo iba a ser tan ideal como en un principio se planteaba. Esto a su vez nos crea un gran reto de aprender a resolver todas estas cuestiones y nos dota de la motivación necesaria para enfrentarnos a ello.

Por otro lado, pretendemos también aprender a usar nuevas plataformas de programación desconocidas previamente y nuevos lenguajes de programación que además difieren en parte de los típicos lenguajes de programación como Java en que es un lenguaje orientado tanto a objetos como a eventos y además incluye mecanismos de desarrollo de juegos basados en componentes de los mismos (sprites, rooms, backgrounds, game-objects) con los cuales nunca antes habíamos trabajado.

Además existen sucesos que nos han llamado la atención, como es el caso del famoso juego Flappy Birds que arraso en los markets móviles haciendo ganar mucho dinero a su desarrollador. Tal fue la repercusión que su creador se vio presionado a retirar el juego del mercado por el revuelo que causó y la gente empezó a enloquecer tanto por querer conseguir este juego que incluso se vendían móviles con esta aplicación instalada por precios que rozaban los 1000 euros, sin importar el dispositivo en el que estaba instalado. La simple idea de un juego que incorporaba elementos de otros juegos, en este caso un protagonista con forma de pájaro de otro gran juego llamado Angry Birds, un escenario inspirado en otro juego Super Mario Bros que incluía la temática y paisajes de este, y una funcionalidad muy simple con la que bastaba en tocar la pantalla para superar los obstáculos que presentaba el juego, fueron los 3 elementos principales que bastaron para repercutir en la escena de las aplicaciones móviles. Por supuesto esto nos hace ver la posibilidad de explotación de este tipo de aplicaciones por el tema económico ya que en el caso de esta aplicación el propio autor confirmo que la desarrollo mediante herramientas multiplataforma.

Todos estos aspectos nos incentivaron para iniciarnos en este mundo de la programación multiplataforma y tras varios ensayos y pruebas, una organización de Pamplona nos contactó para desarrollar un juego de este tipo, lo que nos permitió a su vez modelar y desarrollar ideas de terceros bajo su supervisión y control dándonos así un proyecto más serio y riguroso.

En definitiva nuestra meta final es el conjunto de poder aprender a desarrollar pequeñas aplicaciones, juegos y distintos software adaptable a cada sistema para obtener mayor versatilidad con el fin de ampliar el campo de adquisición de estas aplicaciones, sin hacer trabajos redundantes y esfuerzos de adaptación, así como la responsabilidad de llevar a cabo la creación de un producto para otra entidad ajena y poder darnos a conocer.

1.4 PROYECTO CONDY CRUSH SARE

Como hemos mencionado anteriormente, SARE, una asociación de prevención del sida situada en Pamplona en el barrio de Rotxapea, nos contacta debido a que están interesados en crear un juego que tenga que ver con la temática de prevención del sida. Inspirándose en el famoso juego Candy Crush Saga, nos sugieren la idea de un juego basado en la forma de juego de este pero con sus propias variantes.

Nos cuentan que están llevando a cabo un proyecto global en el cual quieren incluir el juego que desean desarrollar, este proyecto engloba diversos trabajos concernientes a artes (contenedores pintados expuestos por muchas calles de Pamplona), atención y apoyo a personas que necesiten información en lo relativo a las enfermedades venéreas, embarazo y prevención de los mismos, aplicación de carácter general para ayudar a los usuarios, asimismo pagina web y la creación de juegos con los que poder aprender a evitar situaciones no deseadas relacionadas con estas temáticas.

Tras varias reuniones y mediante la mediación de otros desarrolladores decidimos entonces formar parte de este proyecto y accedemos a desarrollar el llamado juego Condy Crush de momento para incorporarlo en su página web pero sin dejar de lado la posibilidad de explotar el resto de plataformas en un futuro no muy lejano.

2 ANALISIS Y DESCRIPCION

En este apartado describiremos el análisis de requisitos y diseño del juego Condy Crush que se nos ha requerido. Dicho proyecto se ha ejecutado basándonos en la metodología de proceso unificado (UP). No obstante, antes de esto, explicaremos los juegos previos que se utilizaron como punto de partida y fueron la base para poder llegar al producto solicitado.

2.1 PROTOTIPOS DE APRENDIZAJE

2.1.1 DONDESTAS NICOLAS

Tras la inesperada aparición del juego que hemos comentado anteriormente Flappy Birds y ver la posibilidad de sacar rendimiento a este tipo de juegos y aplicaciones se nos despierta el interés y decidimos inmiscuirnos en este mundo.

Como primera toma de contacto con la multiplataforma, decidimos crear un pequeño juego tipo Donde esta Wally, pero utilizando como personaje a "El pequeño Nicolás" aprovechando el tirón mediático de este personaje español. Con el fin de aprender a manejar un nuevo entorno de desarrollo con su propio lenguaje de programación y ver la capacidad de monetizar este tipo de juegos cada vez mas descargados empezamos a plantear el juego.

La idea básica es que aparezca un escenario el cual está compuesto por una fotografía o imagen donde aparecen muchas personas, y entre ellas camuflar a nuestro personaje principal con el objetivo que el jugador lo descubra y pueda seguir encontrándolo en sucesivos niveles. Es un juego bastante sencillo pero está dotado de las funcionalidades básicas y necesarias que queremos aprender a desarrollar en esta plataforma.

Así pues los requisitos que nos imponemos son bastante sencillos:

- El juego deberá ser soportado para múltiples plataformas (nuestro interés principal)

- Deberá componerse de varios escenario, un personaje principal y elementos disuasorios que dificulten la aparición del personaje.

- Estos escenarios podrán ser la misma imagen o fotografía sin que el lugar de aparición del personaje principal se repita.

- Se necesitara un sistema de puntuación para poder medir los aciertos de los jugadores.

- Se necesitara también un contador de tiempo para limitar la estancia de un jugador en un mismo escenario a 20 segundos. Al terminar este tiempo se reiniciara el juego teniendo que volver a empezar desde el principio.

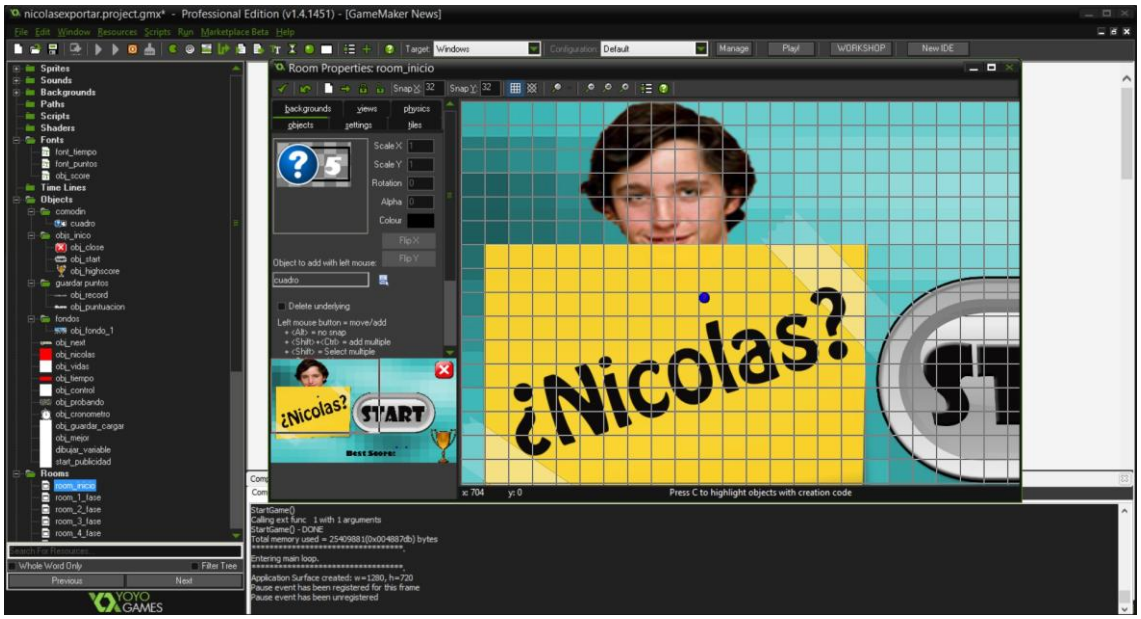
- Deberá existir además un modulo de intentos lo cual contaremos como "vidas" del jugador siendo el máximo de estas 3 y el mínimo 0. Si este contador de vidas llega a 0 el jugador deberá empezar desde el principio.

- Se proveerá también de un sistema de comodines con los que el jugador pueda saltar a un nivel sin haber encontrado al personaje en el nivel anterior. El jugador dispondrá de 5 comodines posibles, una vez agotados estos deberá empezar desde el principio.
- Existirá a su vez una pantalla principal donde se recogerá la puntuación mas alta obtenida por el jugador.

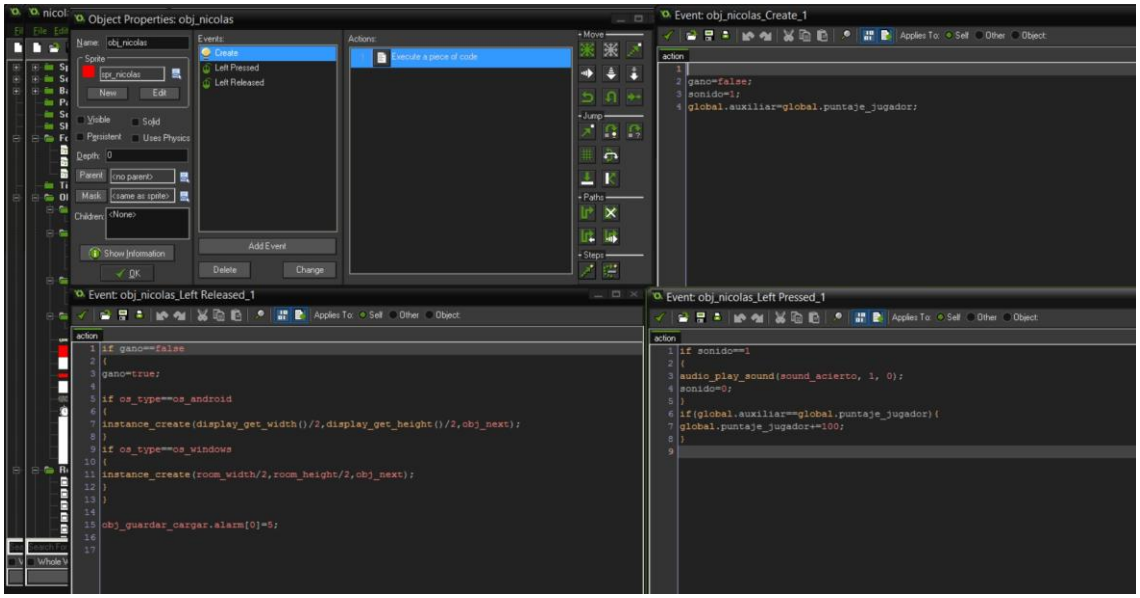
-Esta pantalla anterior deberá contener un botón de inicio para empezar a jugar y un botón de cierre para poder salir del juego.

Tras el estudio previo de requisitos, nos ponemos a implementar el juego sin necesidad de seguir con el diseño formal, natural de la metodología UP, ya que simplemente se creara un prototipo para iniciarnos en todo este mundo.

Ahora detallaremos imágenes de como ha quedado el prototipo tras su desarrollo y trozos de código y la disposición de todos los elementos que conforman la plataforma y dan vida al juego de ¿Donde está el Pequeño Nicolás?

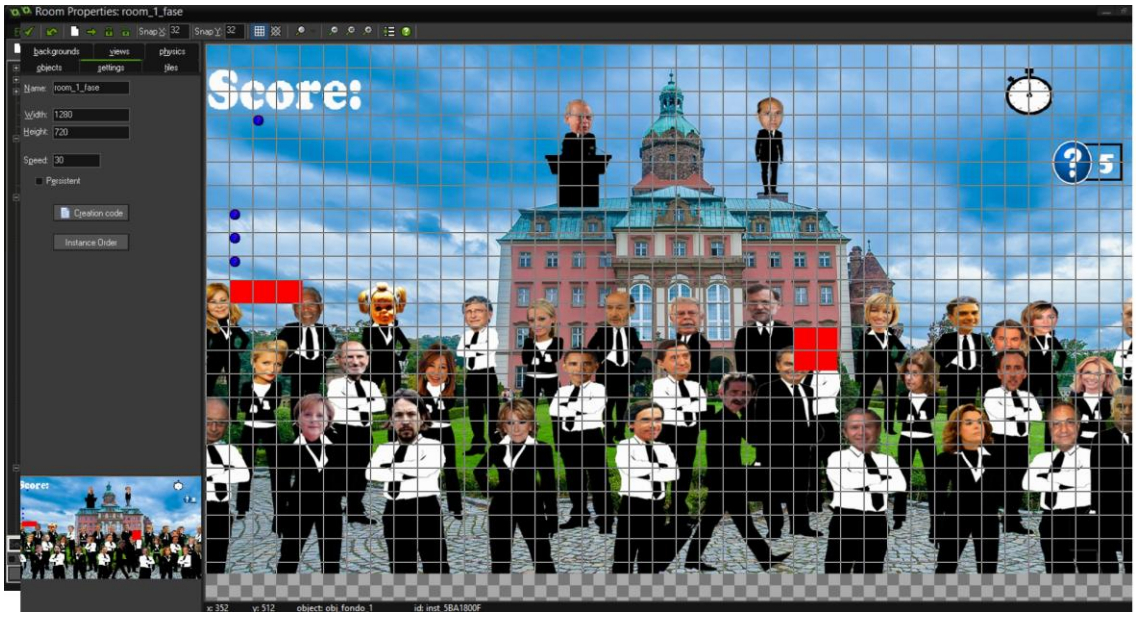


En esta imagen podemos apreciar en el panel izquierdo varios de los elementos que logran formar el juego. Los más importantes son los que están dentro de la carpeta objects ya que como su nombre indica son los objetos que dan la funcionalidad al juego, cada uno de ellos tiene su propio comportamiento y dota al juego de distintas características. Podemos apreciar también la carpeta rooms donde están los escenarios los cuales a su vez incorporan distintos backgrounds y muchos de estos objetos que hemos comentado antes.

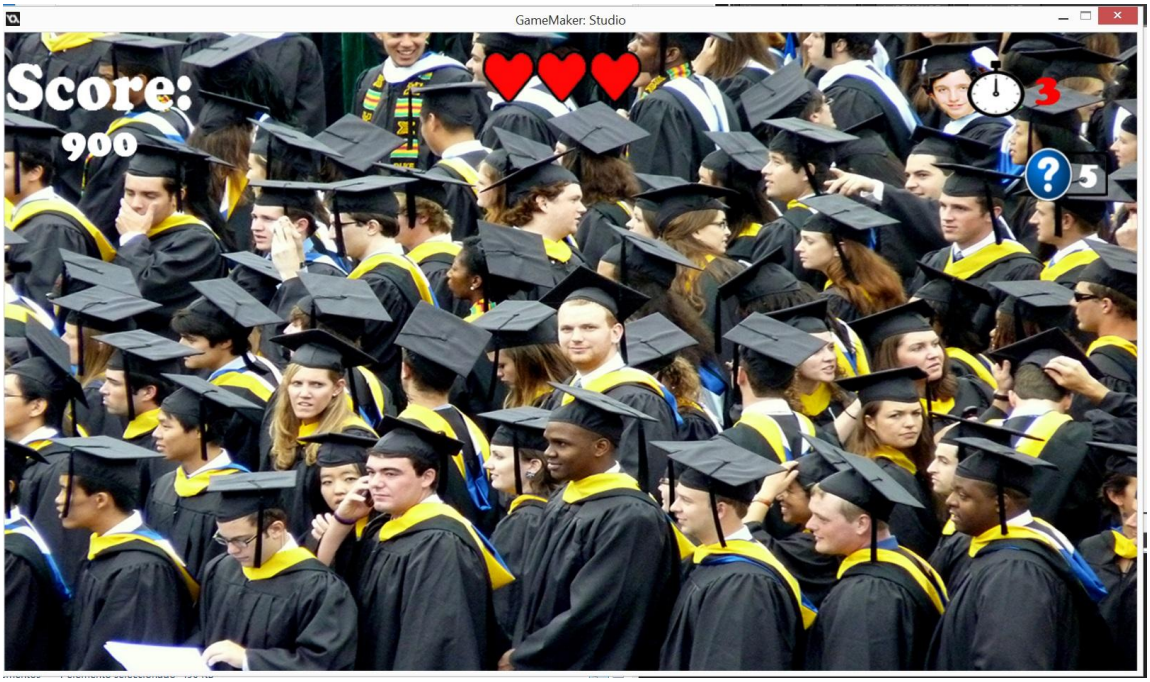


Aquí podemos observar el comportamiento de uno de los objetos, en este caso el que controla el comportamiento del personaje principal. En si su funcionalidad es muy sencilla y se compone de tres eventos. A la hora de la creación de este objeto se inicializan una serie de variables, en el evento del click izquierdo del ratón, el cual es el mismo que al tocar la pantalla en un dispositivo táctil realiza una serie de comprobaciones y crea un botón para pasar al siguiente nivel, esto es así porque una vez clickemos en el querrá decir que hemos encontrado al personaje y este nivel ya habrá terminado. Por último a la hora de soltar el botón o levantar el dedo de la pantalla reproduce un sonido de acierto y suma la puntuación al jugador.

En los códigos anteriores, más concretamente en los dos eventos relacionados con el click, se puede ver cómo hacemos una distinción entre si el sistema operativo es Android o Windows, esto es debido a que a la hora de pintarse la interfaz funcionaba mal si no se hacia la comprobación por el tema de pantallas ya que no es lo mismo una pantalla de un dispositivo móvil que la de un ordenador. No obstante esto no siempre es necesario ya que hay objetos más sensibles que otros y muchos no necesitan esta especificación.



En la imagen anterior nos encontramos con el diseño de uno de los escenarios lo que llamaremos rooms. Se puede ver como posee un fondo el cual es una fotografía hecha en Photoshop con un conjunto de personajes políticos. Además se aprecian el resto de elementos que no son más que los objetos que hemos mencionado anteriormente. Podemos ver el objeto cronometro, el objeto comodines, el objeto score y el resto de objetos que aparecen representados con unos pixeles rojos o azules pequeñitos son elementos de control entre otros que previamente no poseen una representación propia sino que se pintan en tiempo de ejecución.



Estas últimas imágenes son el reflejo final de como quedan los escenarios implementados a modo de prueba y se puede ver como interaccionan todos los objetos entre sí. Cada uno de ellos es una room y se va pasando de unas a otras conforme el juego avanza.

Por último, nos queda comentar acerca de este juego que se intento monetizar incorporándole anuncios en forma de banner de la empresa LeadBolt ya que tras intentarlo con adMob la propia de Google nos advirtieron de que existía un conflicto por la temática de la aplicación. Tras varios meses en los markets y sin poder sacar provecho del juego se desechó la idea y decidimos seguir creando otro tipo de juegos y continuar aprendiendo.

2.1.2 SLIDEIT

Nuestro segundo prototipo de aprendizaje es un juego de pensar y agudizar el ingenio, que se basa en resolver y adquirir capacidades de pensamiento para superar ciertos retos. El juego típico en el que tienes varios palitos dispuestos en distintas formas y tienes que pasar por todos ellos sin levantar el dedo del papel o en este caso de la pantalla. A este juego le llamaremos SlideIt.

Decidimos cambiar a este otro tipo de juego tras el fracaso del anterior, y entonces nos ponemos a analizar los requisitos del mismo.

-El juego deberá constar de un escenario en el que se dispondrá de los elementos con los que jugar. Asimismo el escenario deberá estar inspirado en una temática , en este caso en concreto deberá parecer con matices de maderas, tornillos, etc. con el fin de dar un aspecto bonito al juego.

- Se crearan varios objetos los cuales serán los encargados de llevar la cuenta de si se han pasado por ellos una única vez o no.

- Se necesitara crear un efecto para darnos cuenta de los lugares por los que nos hemos desplazado en la pantalla con el fin de poder identificar si hemos pasado por los objetos anteriores o no. Asimismo este efecto no será permanente pero si lo suficientemente duradero para otorgar fluidez al juego.

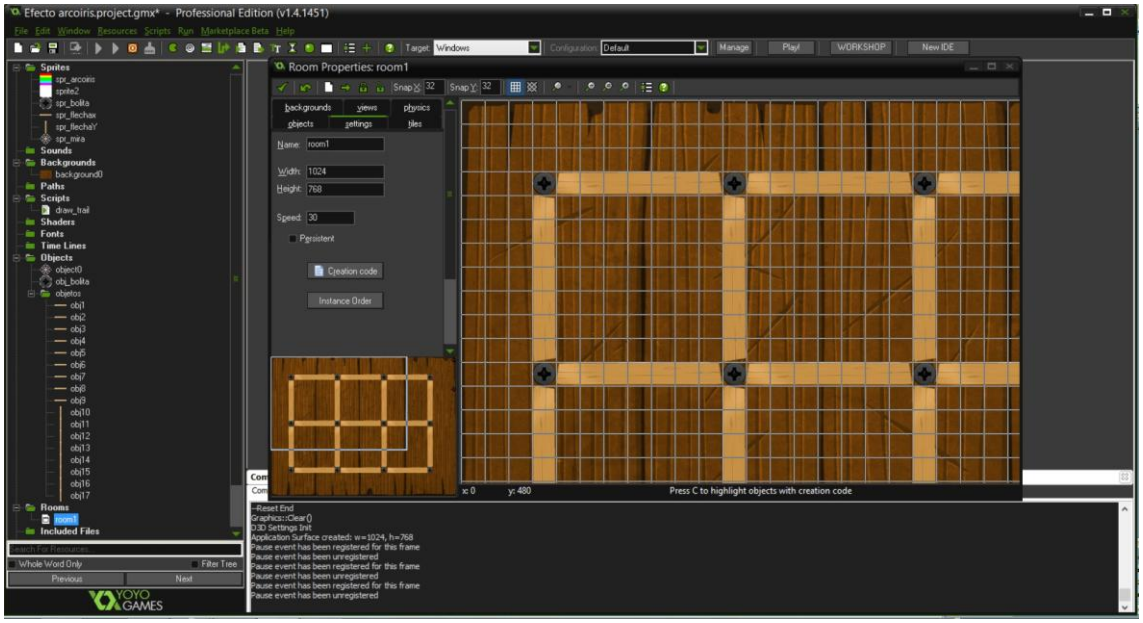
- Deberá implementar un sistema de puntuación.

- El cambio de nivel se producirá tras superar el nivel de forma correcta o en su defecto si esto no se consiguiera tras efectuar 10 intentos.

- Consideramos un intento como el proceso desde que el usuario clica y mantiene el click o posa el dedo sobre la pantalla hasta que liberemos la tecla del ratón o levantemos el dedo de la pantalla.

- El propio escenario nos advertirá si hemos logrado llevar a cabo el reto de forma correcta.

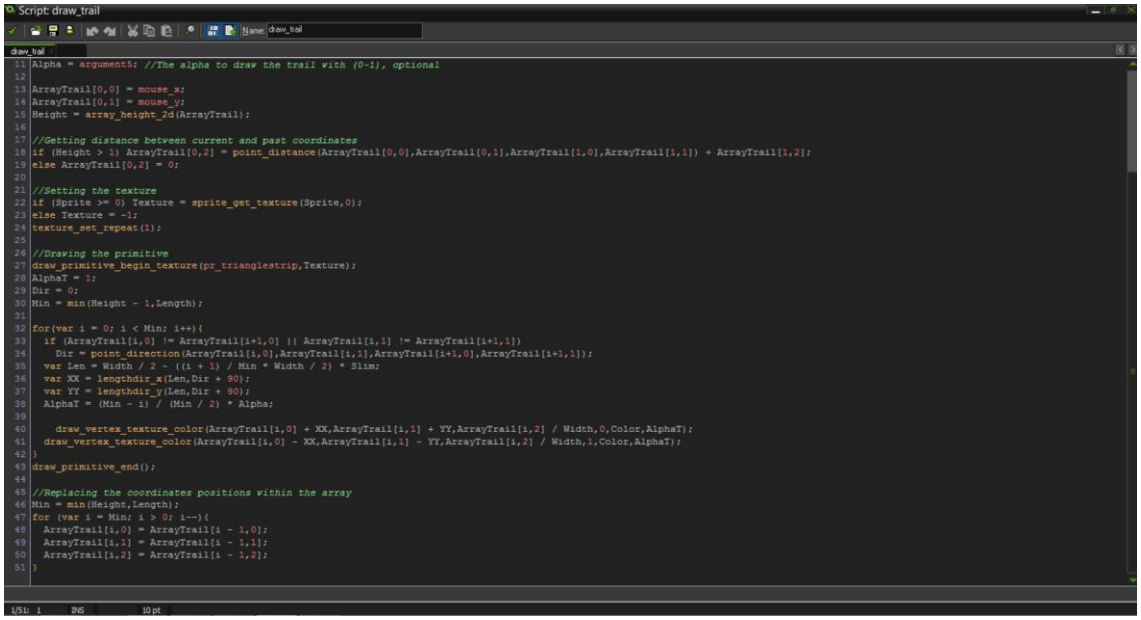
Tras la descripción de todo lo anterior nos ponemos manos a la obra y empezamos a desarrollar la aplicación en la misma plataforma que el juego anterior, Game Maker.



En la fotografía anterior se observa la disposición de los componentes. Podemos ver los sprites principales de la aplicación, el background, un script que previamente no habíamos usado nunca en el cual desarrollamos una funcionalidad aparte la cual usan los objetos, los propios objetos que como vemos son los que a su vez pintan el resto del escenario por el sprite que llevan asociado y la room donde se incluyen todos los antes mencionados.

Primeramente comentaremos el nuevo elemento que hemos usado para este juego, el script draw_trail, lo que nos viene a ofrecer la funcionalidad de simular el efecto de estela al desplazar el dedo por la pantalla.

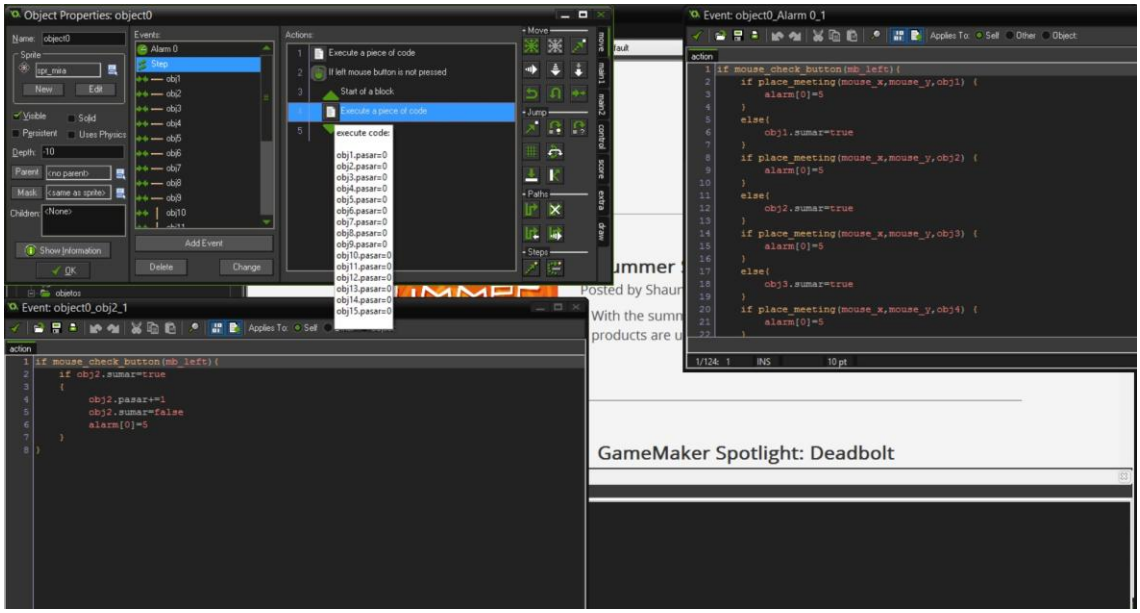
Esto se consigue realizando cálculos mediante la captación de las coordenadas x e y aplicando distintas funciones propias de la plataforma y la transformación de estas en pixeles que simulan un arco iris y se van refrescando conforme se va produciendo el movimiento.




```
Alpha = argument5; //The alpha to draw the trail with (0-1), optional

ArrayTrail[0,0] = mouse_x;
ArrayTrail[0,1] = mouse_y;
Height = array_height_2d(ArrayTrail);

//Getting distance between current and past coordinates
if (Height > 1) ArrayTrail[0,2] = point_distance(ArrayTrail[0,0],ArrayTrail[0,1],ArrayTrail[1,0],ArrayTrail[1,1]) + ArrayTrail[1,2];
else ArrayTrail[0,2] = 0;

//Setting the texture
if (Sprite >= 0) Texture = sprite_get_texture(Sprite,0);
else Texture = -1;
texture_set_repeat(1);

//Drawing the primitive
draw_primitive_begin_texture(pr_trianglestrip,Texture);
AlphaT = 1;
Dir = 0;
Min = min(Height - 1,Length);

for(var i = 0; i < Min; i++){
  if (ArrayTrail[i,0] != ArrayTrail[i+1,0] || ArrayTrail[i,1] != ArrayTrail[i+1,1])
    Dir = point_direction(ArrayTrail[i,0],ArrayTrail[i,1],ArrayTrail[i+1,0],ArrayTrail[i+1,1]);
  var Len = Width / 2 - ((i + 1) / Min * Width / 2) * Slim;
  var XX = lengthdir_x(Len,Dir + 90);
  var YY = lengthdir_y(Len,Dir + 90);
  AlphaT = (Min - i) / (Min / 2) * Alpha;

    draw_vertex_texture_color(ArrayTrail[i,0] + XX,ArrayTrail[i,1] + YY,ArrayTrail[i,2] / Width,0,Color,AlphaT);
  draw_vertex_texture_color(ArrayTrail[i,0] - XX,ArrayTrail[i,1] - YY,ArrayTrail[i,2] / Width,1,Color,AlphaT);
}
draw_primitive_end();

//Replacing the coordinates positions within the array
Min = min(Height,Length);
for (var i = Min; i > 0; i--){
  ArrayTrail[i,0] = ArrayTrail[i - 1,0];
  ArrayTrail[i,1] = ArrayTrail[i - 1,1];
  ArrayTrail[i,2] = ArrayTrail[i - 1,2];
}
```

En definitiva con todo esto se consigue que dibuje la trayectoria que ha seguido nuestro dedo por todo el screen.

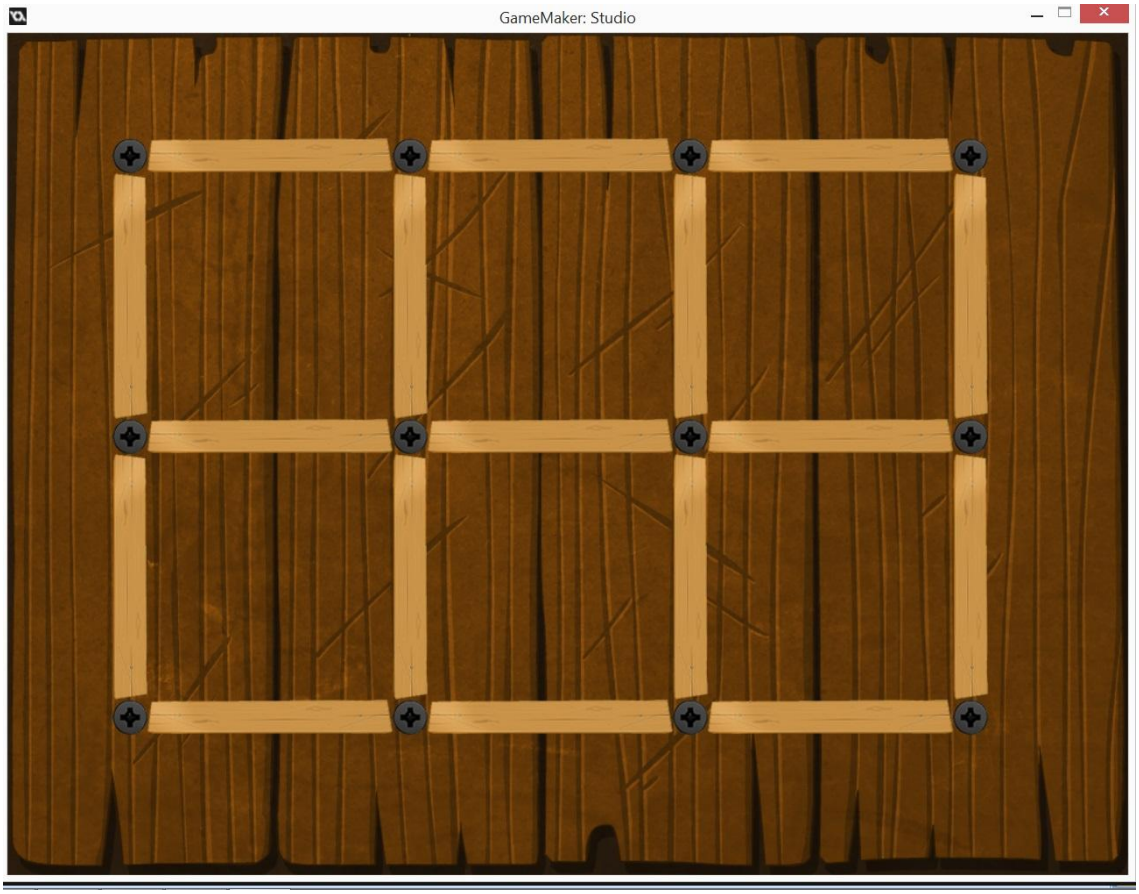
Por otra parte, tenemos un objeto principal el cual podemos decir que es una extensión de nuestro dedo en la pantalla o del ratón al hacer el click que es el que controla estos eventos.

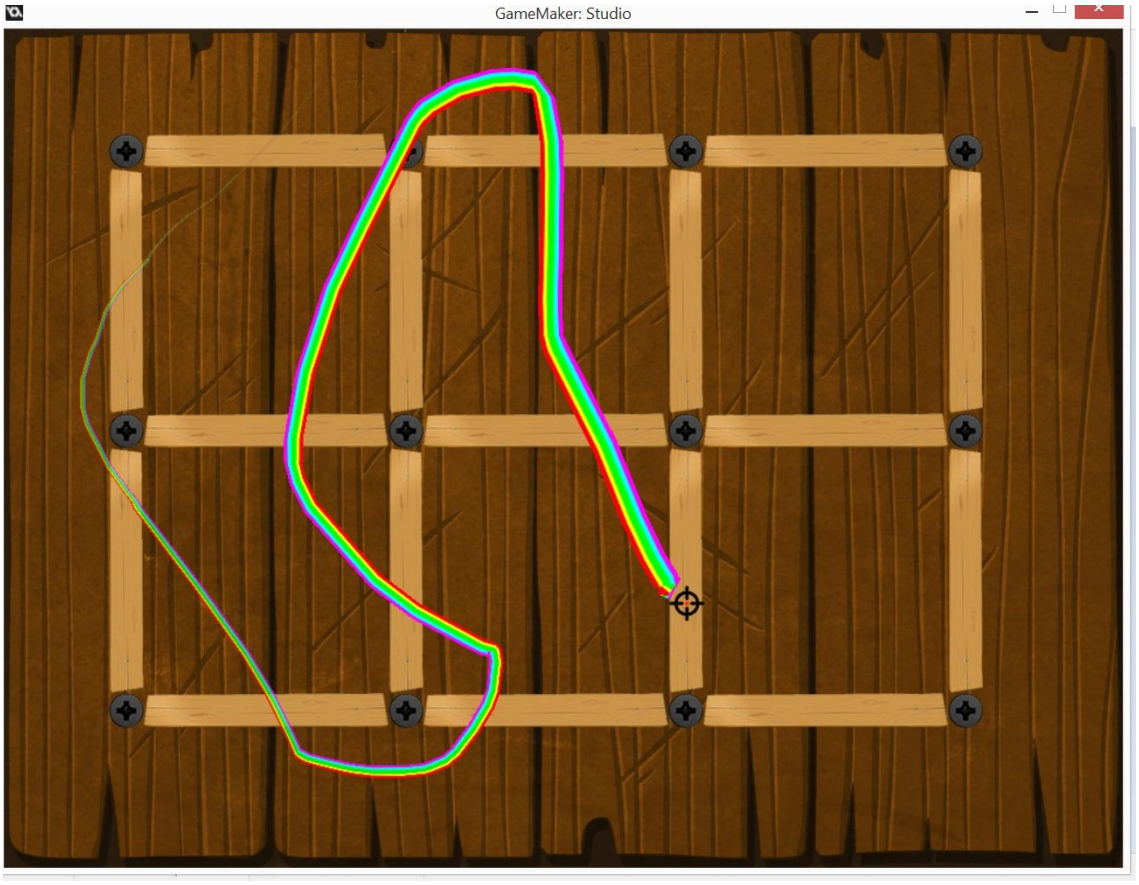


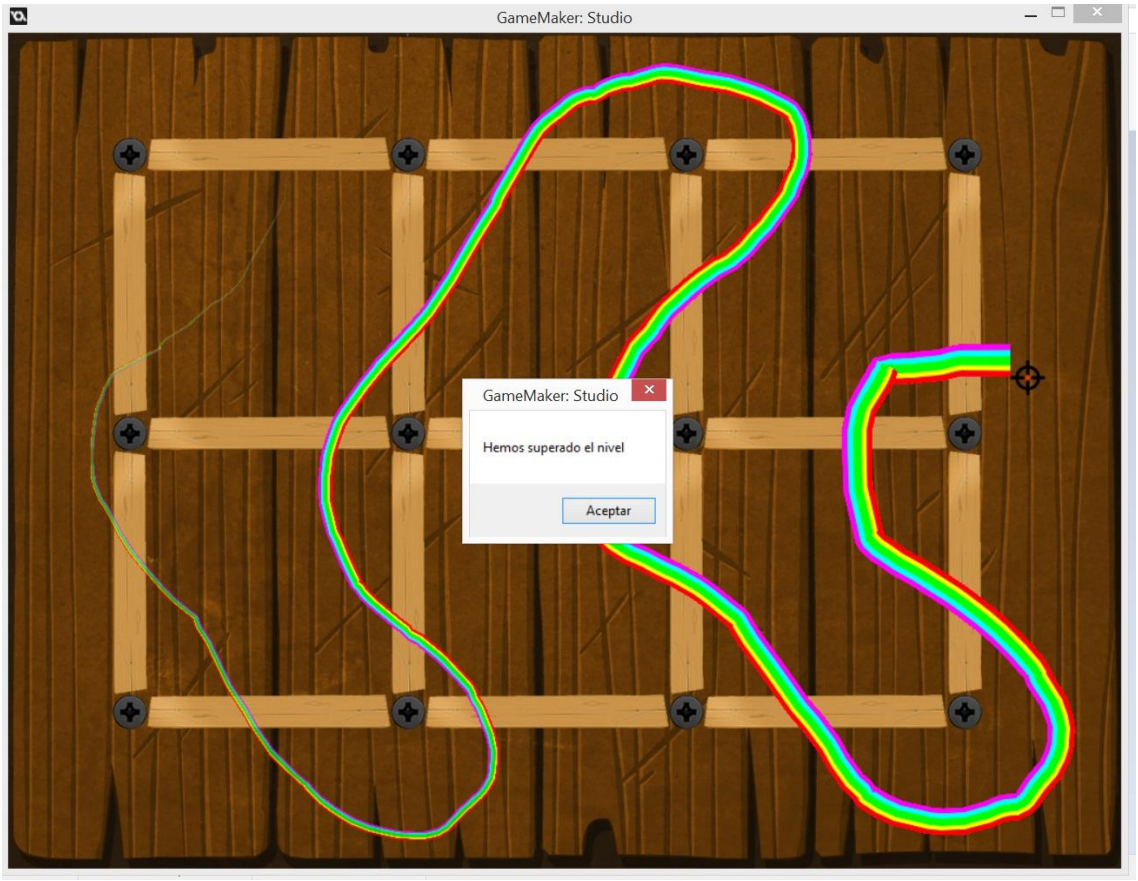
En la imagen anterior se puede ver como el denominado objeto0 o objeto de origen tiene una serie de eventos los cuales básicamente son una alarma con la que se controla la velocidad del juego o fps (frames per second), el código asociado a esta es el que aparece a la derecha de la imagen y simplemente comprueba si esta pulsado el botón de click o en su defecto si tenemos el dedo sobre la pantalla y comprueba en cada instante del juego si hay colisión de este objeto con el resto de objetos que representan los palos por los que hay que pasar. Esto se ayuda a su vez del evento step que es el que realmente controla cada instante del juego, las alarmas en definitiva sirven para cooperar con este otro evento y lanzar señales de entrada al juego para observar posibles cambios.

Por otra parte tenemos también los eventos de colisión propios de este objeto con el resto de objetos, cuyo código podemos ver en la parte inferior izquierda y que simplemente detecta que si nuestro objeto principal choca contra uno de estos y si estamos habilitados para llevar el conteo (llevado a cabo por la alarma) suma 1 en una variable propia del resto de objetos para indicar que hemos pasado por ellos.

Por último tenemos el resto de objetos cuya representación son los famosos palos o palitos de los que hemos estado hablando y simplemente sirven para ayudar a dibujar la interfaz y para que el objeto0 o objeto de control compruebe si esta chocando con ellos o no.

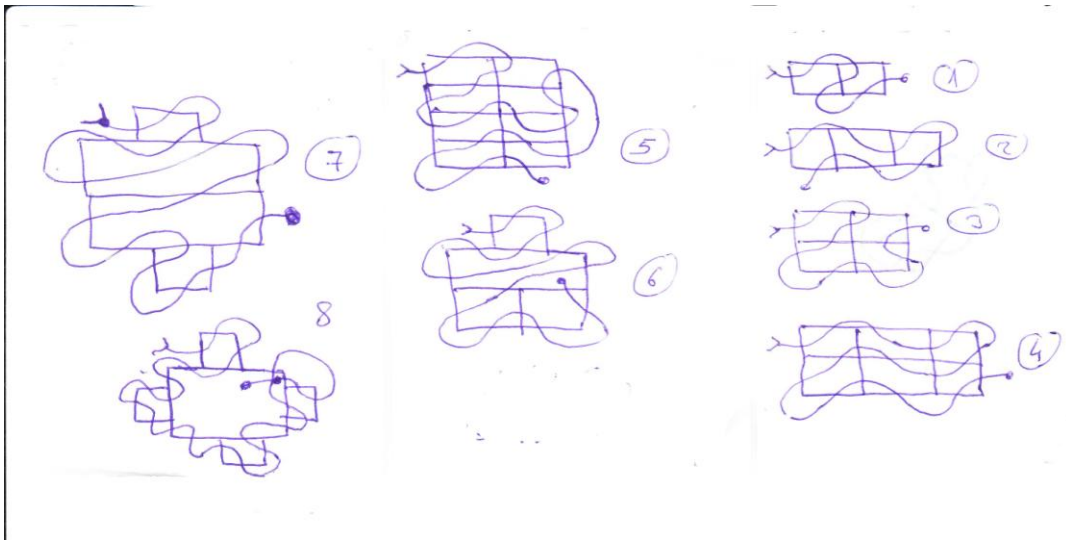






En las 3 pantallas anteriores podemos ver como se lleva a cabo el juego y como es la interacción del usuario con el mismo.

Llegados a este punto estamos obligados a mencionar que esta aplicación se quedo en este punto y no se pudo continuar realizando mas escenarios, ni el resto de controles que habíamos pensado como el sistema de puntuación y de intentos, debido a que enseguida nos contactaron para desarrollar el grueso de este proyecto, el juego Condy Crush, no obstante se pensaron y se diseñaron mas niveles los cuales podemos ver en los siguientes bocetos.



2.1.3 CONDY CRUSH

Tras el diseño e implementación de los dos prototipos de los juegos explicados anteriormente y tras la pequeña promoción que se hizo de ellos en ellos en nuestros círculos sociales, nos contactan desde una asociación de Pamplona dedicada a la prevención e información de las enfermedades venéreas llamada SARE y situada en el barrio de la Rotxapea. Nos comentan que están envueltos en un proyecto que se compone de varios subproyectos de distintas disciplinas y cuyo propósito es concienciar a la población acerca de los riesgos de del virus del V.I.H y nos proponen desarrollarles un juego basado en el mítico juego Candy Crush Saga, pero obviamente, enfocado y ambientado en la temática en la que querían reparar.

Así pues, tras presentarnos y tener varias reuniones, pudimos analizar los requisitos que debía cumplir dicho juego.

2.1.3.1 ANALISIS DE REQUISITOS

2.1.3.1.1 REQUISITOS FUNCIONALES

1) La aplicación deberá poder ser utilizada bajo las distintas plataformas y sistemas operativos, como Android, Windows, Windows Mobile, Navegadores web, iOS, etc.

2) El juego será muy similar al archiconocido juego Candy Crush Saga, en nuestro caso se cambiaran las piezas de fruta con las que se juegan en este por preservativos de colores, incidiendo así en la temática de prevención del SIDA.

3) Existirán a su vez casillas no permitidas, con las que el usuario no pueda interactuar. Estas serán las que representan el virus del V.I.H.

4) Todas estas piezas aparecerán repartidas por la pantalla formando así un tablero con el que el usuario interaccionara.

5) Al pulsar sobre un preservativo (ya sea haciendo click con el ratón o con el dedo en un dispositivo móvil) este queda seleccionado y se intercambiara su posición por la de otro preservativo.

6) Solo existen cuatro movimientos permitidos para hacer el intercambio, esto son las direcciones arriba, abajo, izquierda y derecha.

7) A su vez este desplazamiento entre las piezas del tablero solo será permitido si va a provocar un cambio en el tablero, con esto nos referimos a la generación de al menos una fila o una columna de preservativos del mismo tipo.

8) Cuando se formen filas y/o columnas de tres o más elementos del mismo tipo se crea lo que llamamos una línea y esta deberá hacer que desaparezcan todas las fichas involucradas en ella.

9) Tras la destrucción de las líneas, el tablero deberá recomponerse, para ello primero, se necesita que el resto de preservativos que quedan se desplacen hacia abajo rellenando los huecos si existieran y después generar nuevas fichas para volver a llenar el tablero de piezas.

10) La generación de cada línea otorgara al jugador una puntuación de diez puntos, por lo tanto el juego necesitaran un modulo de puntuaciones, llevando también la cuenta de la máxima puntuación obtenida por un jugador con el fin de que siempre se pueda superar.

11) Si tras el proceso de reordenación y relleno del tablero se vuelven a generar líneas automáticamente se obtendrá una puntuación extra de cincuenta puntos además de los diez puntos de cada línea.

12) El sistema de puntuación guardara la información de cada jugador internamente.

13) Existirán veinte niveles distintos que se diferenciaran por la posición en el tablero de las casillas no permitidas generando así distintos escenarios.

14) A su vez se necesitara de un temporizador que limitara la estancia y juego en cada nivel, incrementándose en diez segundos cada nivel superior.

15) El juego deberá tener una pantalla principal la que contara con la máxima puntuación obtenida y un botón para empezar a jugar.

16) Tras agotarse el tiempo, se deberá poder elegir entre repetir el nivel, pasar al siguiente o ir directo a la pantalla principal.

17) Cuando alcancemos el ultimo nivel, tras superarlo nos aparecerá una pantalla final donde nos aconsejara el uso del preservativo y un botón para volver al inicio del juego.

2.1.3.1.2 REQUISITOS NO FUNCIONALES

1) La aplicación trabajará con un volumen de información muy pequeño, así que se espera que la información almacenada sea lo más sencilla y clara posible.

2) Tanto las imágenes como los audios deberán ser lo más livianos posibles para no aumentar en gran cantidad el tamaño de la app.

3) Las pantallas, deberán ajustarse a las dimensiones y características de cualquier dispositivo.

4) Las pantallas deberán ser lo más intuitivas posibles y fáciles de usar.

5) El programa no debe bloquearse de forma que la ejecución del mismo sea fluida y sin errores.

7) Se utilizará una temática relacionada con los preservativos, virus y enfermedades venéreas para la ilustración de las pantallas, utilizando imágenes propias y prediseñadas basadas en dicha materia, de forma que el uso de la aplicación evoque el sentido que queremos transmitir.

8) Deberán aparecer efectos estéticos como GIFS o transiciones visuales que doten al juego de una mejor presentación.

2.1.3.2 DIAGRAMAS

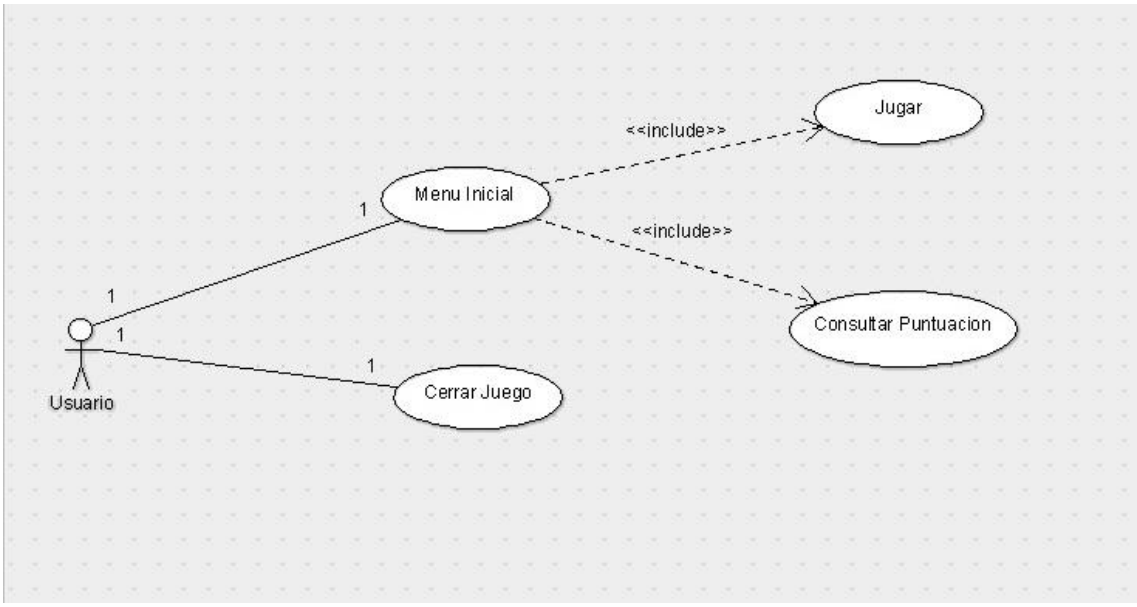
2.1.3.2.1 CASOS DE USO

Siguiendo la metodología de Proceso Unificado (UP/RUP) los requisitos especificados en el apartado anterior, deben ser analizados y convertidos en Casos de Uso. Los actores que se han detallado son los usuarios finales de la aplicación.

CASO DE USO: INICIO

Para la primera pantalla, encontraremos un menú con distintas opciones este será nuestro caso de uso principal. El usuario podrá empezar a jugar, ver la puntuación más alta obtenida (que será cero si no se ha jugado anteriormente) por él o cerrar la aplicación.

Cabe destacar que una vez se incluya en la página web, se podrá acceder también a las instrucciones del juego, pero eso no lo tratamos en este apartado dado que consideramos la inclusión del juego en las distintas plataformas otro proyecto aparte.

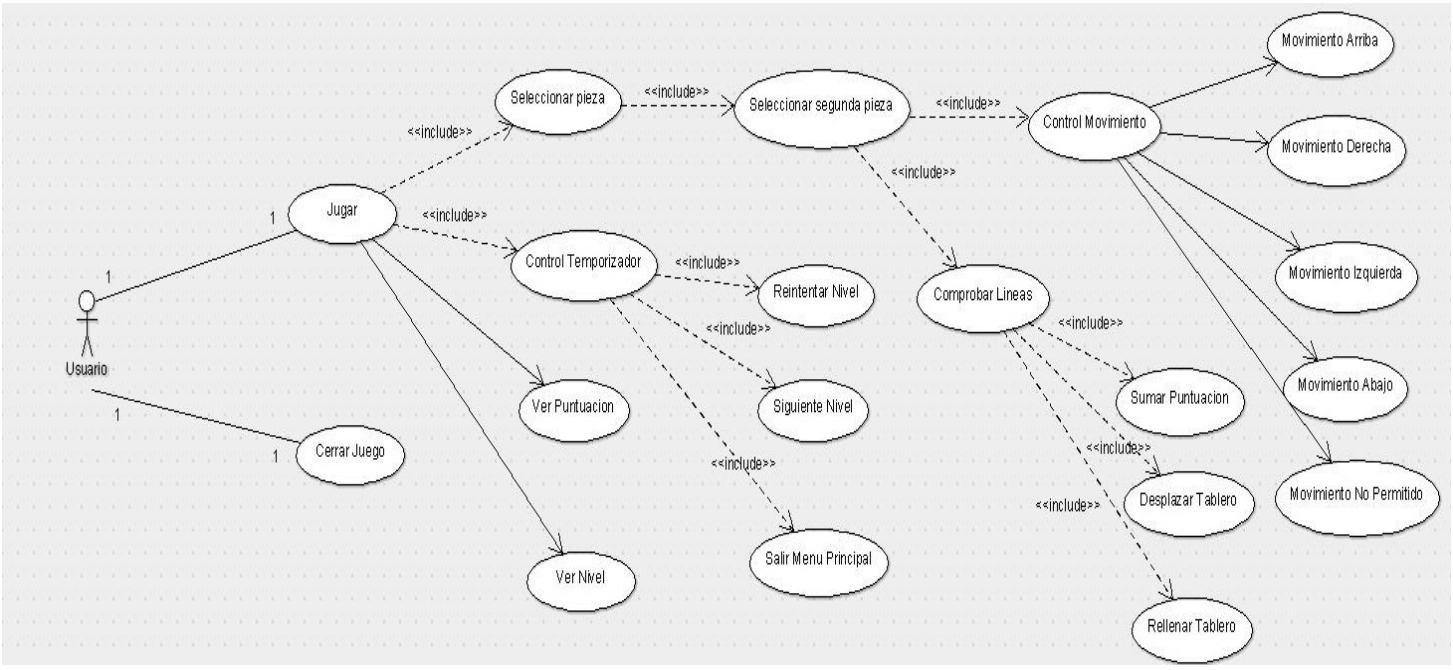


CASO DE USO: INICIO	
Descripción	**El usuario accede a la aplicación y puede ver su puntuación mas alta obtenida anteriormente o empezar a jugar.**

Precondición	**Tener instalada la aplicación o haber accedido a la versión online.**
Camino Básico	**1. El usuario abre la aplicación.** **2. Se presenta el menú principal** **3. Se ve la puntuación.** **4. Pulsa sobre la opción de jugar.** **5. Se pasa a la siguiente pantalla elegida.**
Camino Alternativo	**6. El usuario pulsa atrás y sale de la aplicación o la cierra.**
Postcondición	**El usuario pasa a la siguiente pantalla de juego.**

CASO DE USO: JUEGO

Tras acceder a la pantalla de juego, nos encontramos con la situación en la que el jugador debe elegir una pieza del tablero y acto seguido una segunda con la intención de intercambiar su posición y así generar líneas de elementos iguales para poder sumar puntos.

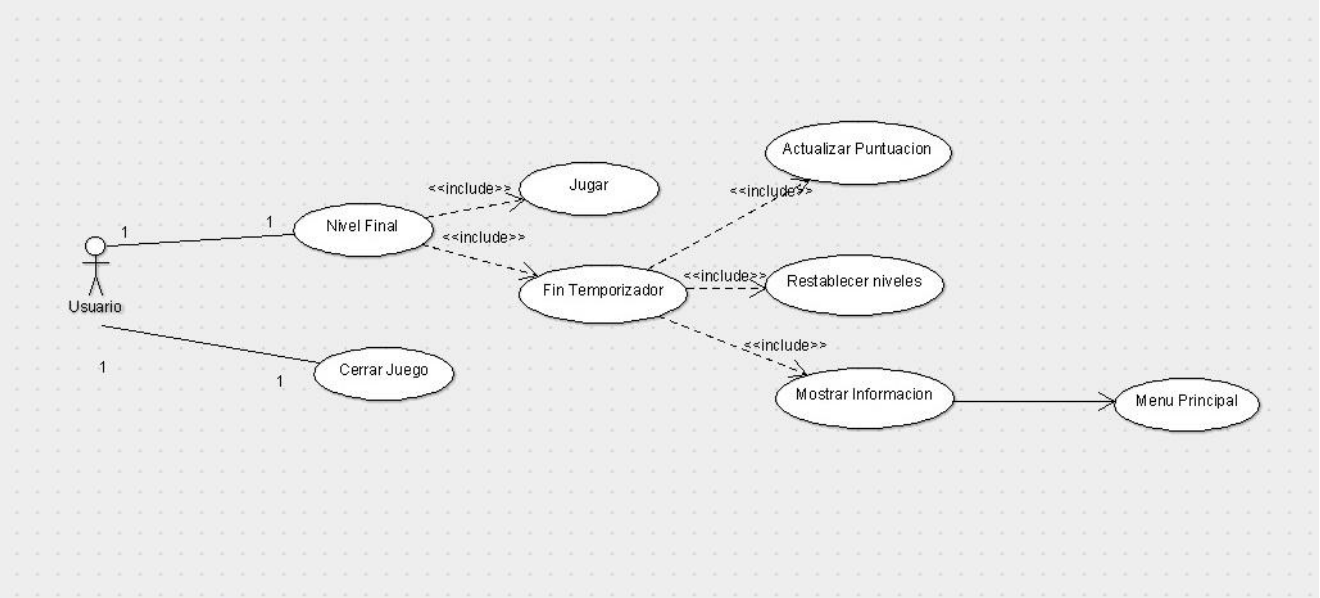


CASO DE USO: JUEGO	
Descripción	**Se presenta al usuario el tablero con las piezas repartidas para que éste interactué con él, de forma que intente lograr la generación de filas o columnas de elementos iguales. Todo esto bajo una cantidad de tiempo fijada y con la finalidad de obtener la máxima puntuación posible.**
Precondición	**Haber pulsado en el botón de jugar del menú anterior.**
Camino Básico	**1. Pulsar sobre una pieza del tablero.** **2. Pulsar sobre la segunda pieza a intercambiar con la primera.** **3. Comprobaciones.** **3.1 Controlar movimiento.** **3.2 Comprobar líneas.** **3.2.1 Sumar puntuación.** **3.2.2 Desplazar piezas del tablero.** **3.3.3 Rellenar tablero.** **4. Ver puntuación.** **5. Control de tiempo.** **5.1 Fin Tiempo.** **5.1.1 Reintentar.** **5.1.2 Siguiente nivel.** **5.1.3 Ir al menú principal.**

Camino Alternativo	**6. Pulsar atrás y salir de la aplicación.**
Postcondición	**El usuario juega y acumula puntuación.**

CASO DE USO: FINAL

El usuario tras agotar el tiempo del último nivel adquiere su puntuación final y es informado en una pantalla final de una reflexión acerca del uso de preservativos. Se restablecen los niveles y se vuelve al menú principal.



CASO DE USO: FINAL	
Descripción	**Tras agotarse el tiempo en el ultimo nivel, se presenta al usuario una pantalla final con un consejo para prevenir las enfermedades venéreas con el fin de concienciar mediante el juego. Se actualiza y se obtiene la máxima puntuación acumulada a lo largo de todos los niveles y podemos acceder al menú principal.**

Precondición	**Estar en el ultimo nivel del juego y agotarse el tiempo.**
Camino Básico	**1 Actualizar puntuación.** **2 Mostrar información.** **3. Acceder al menú principal.**
Camino Alternativo	**4. Pulsar atrás y salir del juego.**
Postcondición	**El usuario queda informado y se entretiene jugando.**

2.1.3.2.2 DIAGRAMAS DE SECUENCIA

A continuación se describen los diagramas de secuencia los cuales describen la interacción del usuario con la aplicación a través del tiempo, explicando en todo momento el intercambio de mensajes entre el usuario (actor) y los objetos de la aplicación y la comunicación entre ellos.

DIAGRAMA DE SECUENCIA: INICIAL

En este diagrama se muestra la primera interacción del usuario con el juego.

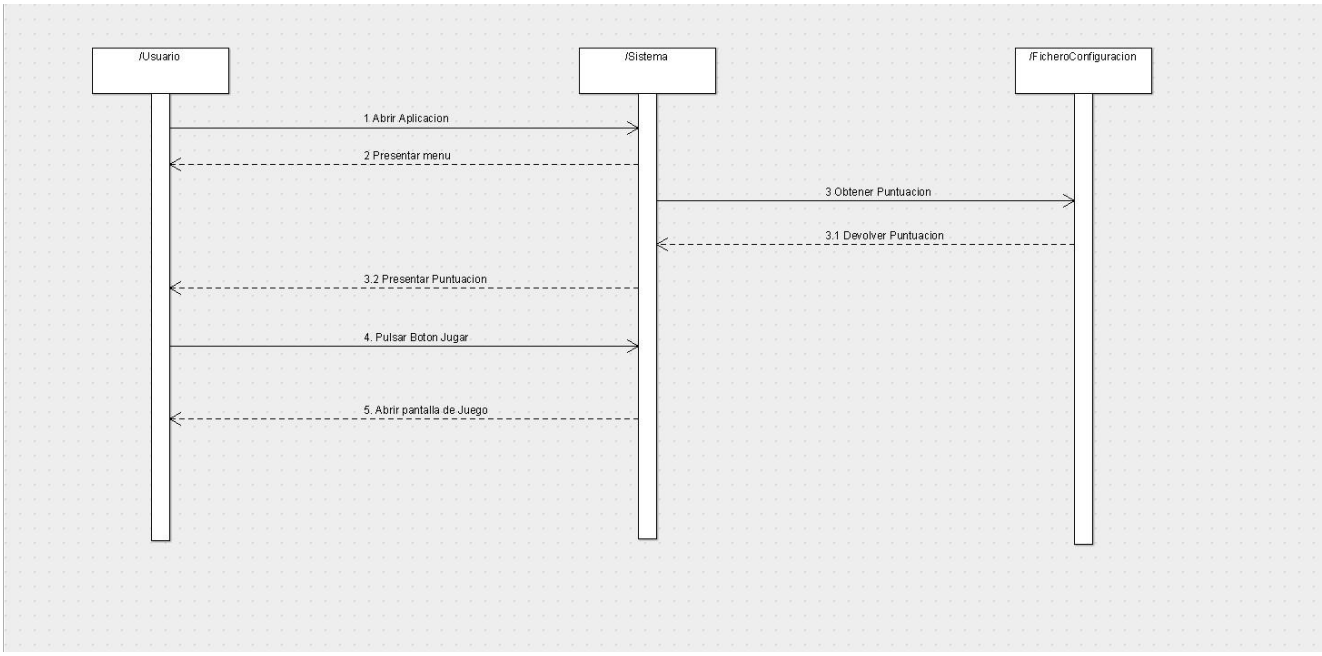


DIAGRAMA DE SECUENCIA: JUEGO

En este grafico vemos el intercambio de mensajes entre el usuario y la aplicación mientras se está jugando.

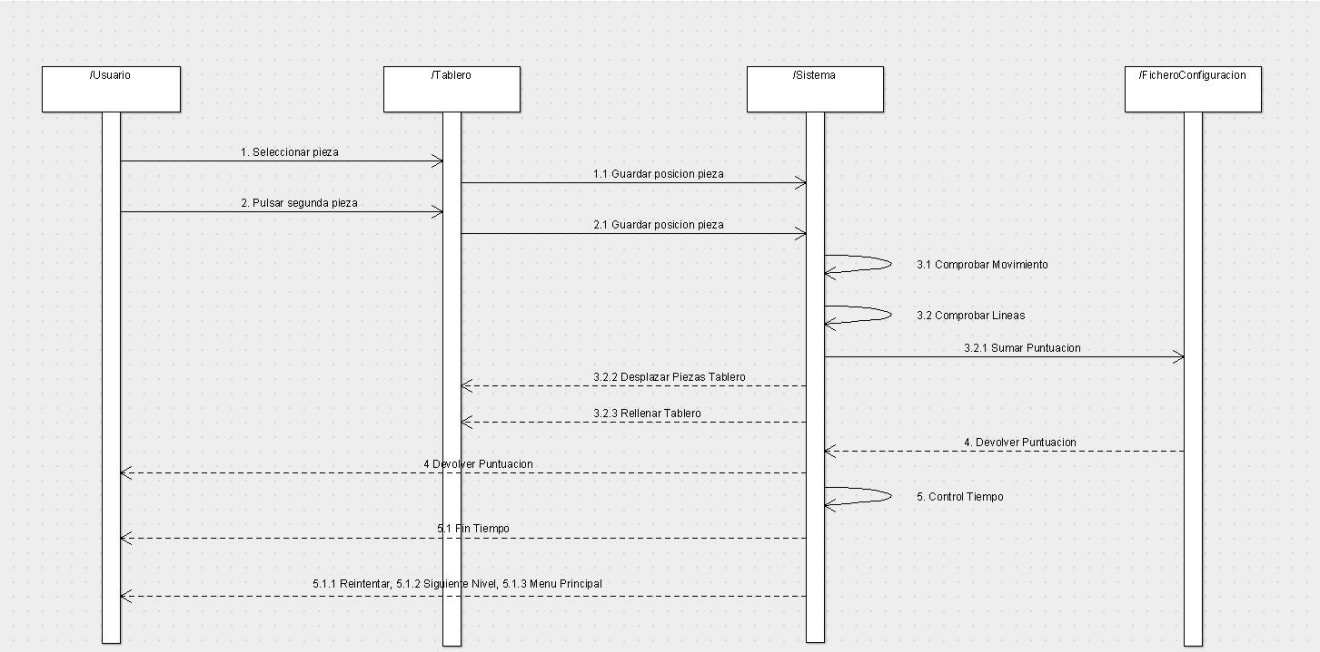
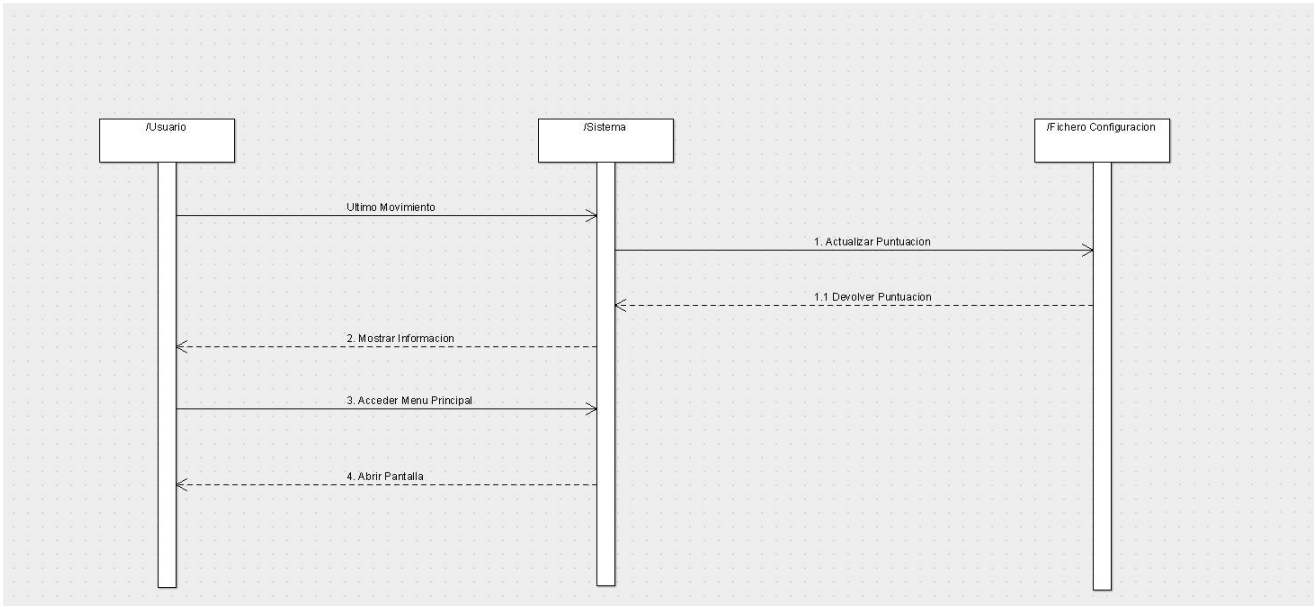


DIAGRAMA DE SECUENCIA: FINAL

En este ultimo diagrama se muestra la comunicación del usuario con el juego tras haber superado todos los niveles.



2.1.3.2.3 DIAGRAMA DE CLASES

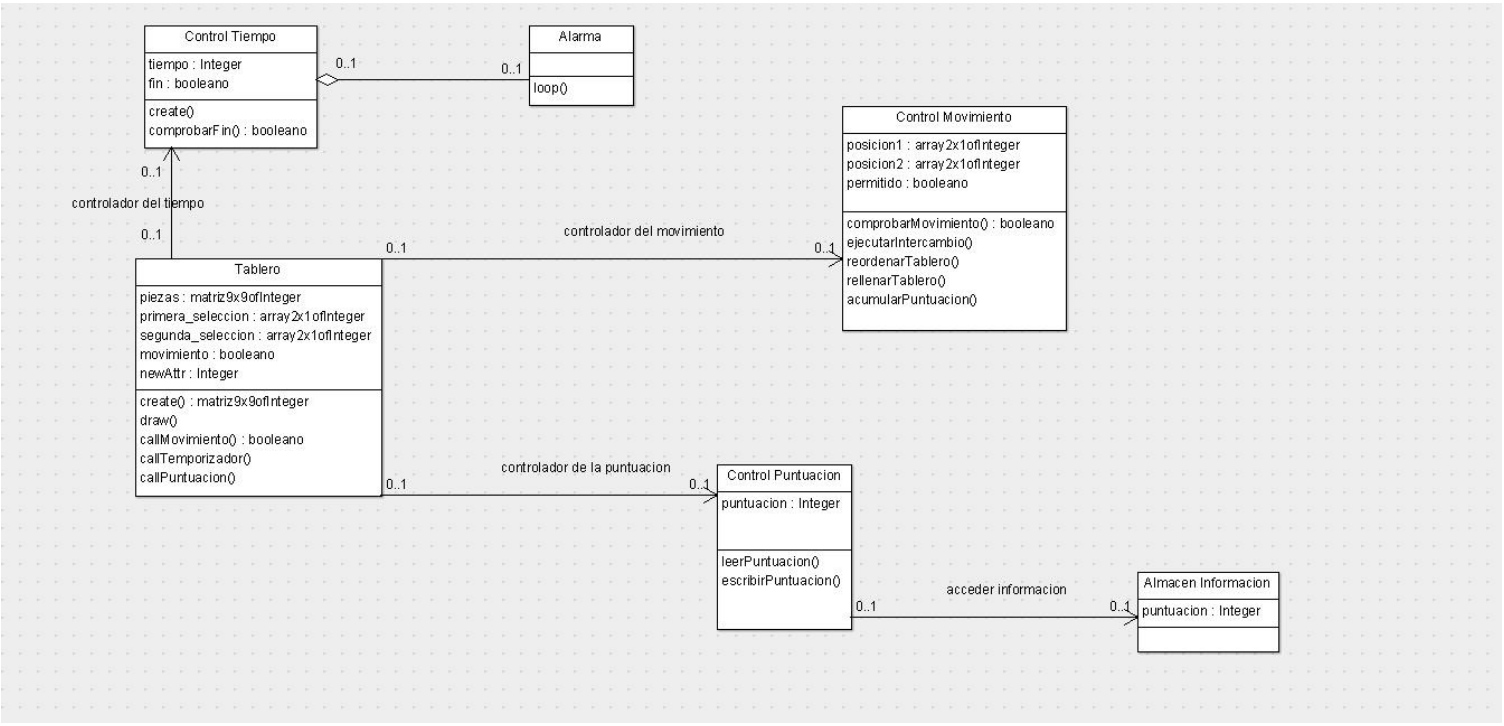
Nos encontramos con que tras los diseños anteriores, surgen varias clases triviales necesarias para el desarrollo de este proyecto.

La clase principal Tablero que es la encargada de generar un escenario aleatorio de piezas, pintarlas y escuchar los eventos que se producen tras la interacción con el usuario.

Esta clase anterior, se comunica directamente en cada instante del juego con lo que llamamos controladores, en primer lugar existe el controlador del tiempo de cada nivel que generará alarmas hasta alcanzar un tiempo deseado.

Por otro lado tenemos el evento que controla el movimiento tras la selección de dos piezas del tablero, esta se encargara de comprobar si es un movimiento permitido o no, es decir si el intercambio de esas dos piezas sirve para juntar tres o más preservativos del mismo tipo en una línea horizontal o vertical. Además es el encargado de ejecutar el cambio si este es válido, reordenar el tablero para eliminar huecos entre las piezas, rellenar los que se han quedado vacios y acumular la puntuación obtenida por haber conseguido formar líneas. A su vez refresca el tablero mediante transiciones visuales.

Por último tenemos la clase encargada de gestionar la puntuación y almacenarla para futuros usos. Esta a su vez también puede ser leída por la misma clase para notificarla al resto de clases que la necesiten.



3 HERRAMIENTAS Y TECNOLOGIA

3.1 GAME MAKER STUDIO

Game Maker Studio, antes llamado Game Maker es una herramienta de desarrollo rápido de aplicaciones basada en un lenguaje de programación interpretado y un kit de desarrollo de software (SDK) para desarrollar videojuegos. Fue creado por Mark Overmans en lenguaje especifico Delphi, y en su principio orientado a usuarios novatos o con pocas nociones de programación. El programa es gratuito aunque existen otras versiones comerciales ampliadas con características adicionales. La ultima versión disponibles es la 1.4 y es la que hemos utilizado para desarrollar el proyecto.

Se creó en 1999 cuando el profesor Overmans intentaba crear una herramienta de animación para ayudar a sus estudiantes. Con el tiempo su proyecto se convirtió en una herramienta muy utilizada en el desarrollo de videojuegos ya que finalmente decidió liberar su software.

3.1.1 INTRODUCCION

Tal y como dicen los mismos de www.yoyogames.com (la web detrás de este programa): todos sabemos que es divertido jugar a videojuegos, pero es más divertido diseñar tus propios juegos y que otras personas lo jueguen.

Pero crear un juego no es una tarea sencilla, nos podemos imaginar que implica crear cualquiera de los juegos que jugamos: un guión, sonido y música para ambientarnos, un diseño coherente con lo que queremos mostrar, horas de programación para que todo funcione perfectamente, más horas para probar que todo lo que hemos comentado funcione.

El programa está diseñado para permitir a sus usuarios desarrollar fácilmente videojuegos sin tener que aprender un lenguaje de programación como C++ o Java. Para los usuarios experimentados Game Maker contiene un lenguaje de programación de scripts llamado Game Maker Language (GML), que permite a los usuarios personalizar aún más sus videojuegos y extender sus características. Los videojuegos pueden ser distribuidos bajo cualquier licencia sujeta a los términos del EULA de Game Maker, en archivos ejecutables no editables ".exe", paquetes Android ".apk", y conjuntos de script HTML5.

La capacidad normal de Game Maker es la de crear juegos en 2 dimensiones, pero usando su propio lenguaje es posible realizar juegos en 3 dimensiones. También cuenta con herramientas simples que permiten la realización de casi cualquier juego sin tener la necesidad de conocer el lenguaje, en nuestro caso hemos utilizado ambas dos opciones ya que uno de nuestros intereses era el aprendizaje de un nuevo lenguaje de programación.

En realidad no hay límite en el tipo de juegos que se pueden crear, ya que podemos desarrollar juegos de tipo puzle, arcade, plataformas, shooter, RPG, etc.

Desde el 12 de mayo de 2012 fue publicada la versión actual de Game Maker la cual nos permite exportar a múltiples plataformas, desde Windows, MAC y Linux, hasta HTML5,

iOS, Android, Windows Phone, Windows 8 y últimamente incluso a videoconsolas como PS3, PS4, PS Vita y Xbox. Y todo esto sin cambiar de programa.

En cuanto a las versiones, actualmente, tenemos una versión gratuita que nos permite acceder a todas las posibilidades del programa, con la posibilidad de exportar a juegos para Windows solamente. La única pega es que aparece una pantalla inicial con el logo de Game Maker. Esta es la versión que ellos llaman FREE o Standard.

Si queremos optar a más opciones avanzadas, como modificar esa pantalla inicial, usar texturas en las imágenes, tener múltiples configuraciones (por ejemplo, crear un juego gratuito y otro de pago), testeo en Android, tenemos la versión *Professional*, que cuesta actualmente **149,99$**. Con esta versión también tenemos la posibilidad de comprar los diferentes módulos de exportación que nos pueda interesar, aunque viene por defecto las opciones de exportación de módulos para ordenadores de escritorio, es decir, tenemos también las exportaciones de Mac OSX yUbuntu Linux.

Si queremos la exportación para HTML5 el precio es de **99,99$**, y el de Tizen (una nueva plataforma parecida a Android que ha sacado Samsung) es de **199,99$**.

Si además queremos otras exportaciones interesantes, como pueden ser iOS, Windows Phone 8 o Android, el precio de cada exportación es **de 199,99$.** Las exportaciones a videoconsolas son gratuitas, y tenemos para Xbox One, PS4, PS Vita y PS3.

También existe una versión llamada Master Collection, que con un precio de **799,99$,** permite la exportación de todos los módulos que hemos comentado anteriormente, y además nos ofrecen todos los módulos futuros que vayan a existir.

Para este proyecto se ha utilizado la versión Proffesional en su última versión, junto con el modulo de exportación de Android y HTML5 lo que costaría 149.99 + 199.99 + 99.99, un total de 449.97 dólares. Viendo esto podemos concluir que por una diferencia de otros 350 dólares podemos obtener todas las posibles exportaciones.

3.1.2 CARACTERISTICAS

La interfaz principal para el desarrollo de videojuegos de Game Maker usa un sistema de "arrastrar y soltar", que permite a los usuarios que no están familiarizados con la programación tradicional crear videojuegos intuitivamente organizando íconos en la pantalla. Game Maker viene con un conjunto de bibliotecas de acciones estándar, que cubren cosas como movimiento, dibujo básico, y control simple de estructuras.

Game Maker usa su propio lenguaje de programación, Game Maker Language (GML), con el que se pueden conseguir impresionantes videojuegos, pues aunque Game Maker está diseñado para la creación de videojuegos en 2 dimensiones, usando GML se pueden conseguir videojuegos 3D avanzados tales como un videojuego de disparos en primera persona (*shooter*).

El desarrollo de un videojuego es realmente simple, al basarse en el manejo de recursos (gráficos, sonidos, fondos, etc.), que se asignan a objetos, eventos (presionar una tecla, mover el ratón, etc.), a través de los que se ejecutan comandos y los objetos, sobre los que se actúa en el videojuego.

Las acciones del videojuego se pueden programar de dos formas distintas:

La interfaz *drag & drop* (arrastrar y soltar): el programador arrastra "cajas" a una lista. Cada una de esas "cajas" realiza una determinada acción. Se gana en facilidad de manejo, a cambio de una menor flexibilidad y potencia.

El lenguaje GML (Game Maker Language): es el lenguaje de programación propio de Game Maker, más potente, con el que se puede acceder a todas las funciones y posibilidades de Game Maker.

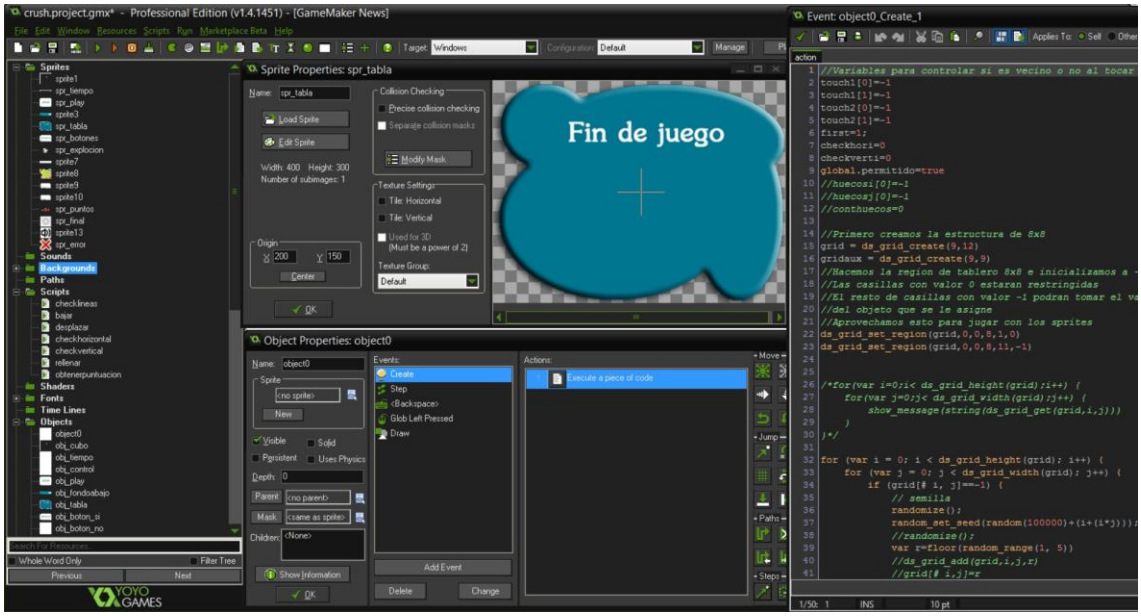
La sintaxis de la programación en GML es muy flexible, es indiferente la forma en la que se aplican algunas reglas a la sintaxis, y entre ellas están:

-Posibilidad de incluir o no el punto y coma (";") al final de cada línea.
-Delimitar bloques de código con begin y end o con corchetes ("{" y "}").
-Operador de igualdad, es indiferente si se usa un doble igual ("==") para leer el valor de una variable o para escribir un valor.

El compilador de Game Maker no compila realmente los ejecutables, sino que une el código a interpretar por su propio intérprete para formar los ejecutables de los juegos. Por ello, no resulta muy eficiente para grandes proyectos

A partir de la versión 6.1 se empezó a usar Direct3D para los gráficos, lo que hace a los videojuegos más lentos, aunque con más posibilidades que en la versión 5.3a, que utilizaba DirectDraw. También hay soporte para bibliotecas de enlace dinámico hechas en C++, Delphi y Pascal, y se pueden crear videojuegos 3D o con soporte multijugador, entre otras funcionalidades.

En la siguiente imagen podemos observar el entorno de desarrollo que incluye todas las características descritas:



En el menú lateral de la izquierda podemos ver los recursos de los que hemos hablado, sprites o gráficos, scripts, objetos, rooms etc.

Tenemos también la posibilidad de configurar y editar los sprites mediante el entorno de edición de imágenes que explicaremos más adelante.

En la parte de configuración de los objetos podemos ver como hemos creado eventos mediantes las opciones de drag&drop.

Por último vemos en la parte derecha un poco de código en lenguaje GML para dotar a los eventos de las funcionalidades que necesitamos.

3.1 OTRAS OPCIONES

Phonegap:

Es un framework para el desarrollo de aplicaciones móviles producido por Nitobi, y comprado posteriormente por Adobe Systems. Principalmente permite a los programadores desarrollar aplicaciones para dispositivos móviles utilizando herramientas genéricas tales como JavaScript, HTML5 y CSS3. Las aplicaciones resultantes son híbridas, es decir que no son realmente aplicaciones nativas al dispositivo (ya que el renderizado se realiza mediante vistas web y no con interfaces gráficas específicas de cada sistema), pero no se tratan tampoco de aplicaciones web (teniendo en cuenta que son aplicaciones que son empaquetadas para poder ser desplegadas en el dispositivo incluso trabajando con el API del sistema nativo).

PhoneGap maneja API que permiten tener acceso a elementos como el acelerómetro, la cámara, los contactos en el dispositivo, la red, el almacenamiento, las notificaciones, etc. Estas API se conectan al sistema operativo usando el código nativo del sistema huésped a través de una Interfaz de funciones foráneas en Javascript. Además permite el desarrollo ya sea ejecutando las aplicaciones en nuestro navegador web, sin tener que utilizar un simulador dedicado a esta tarea, y brinda la posibilidad de soportar funciones sobre frameworks como JQuery Mobile.

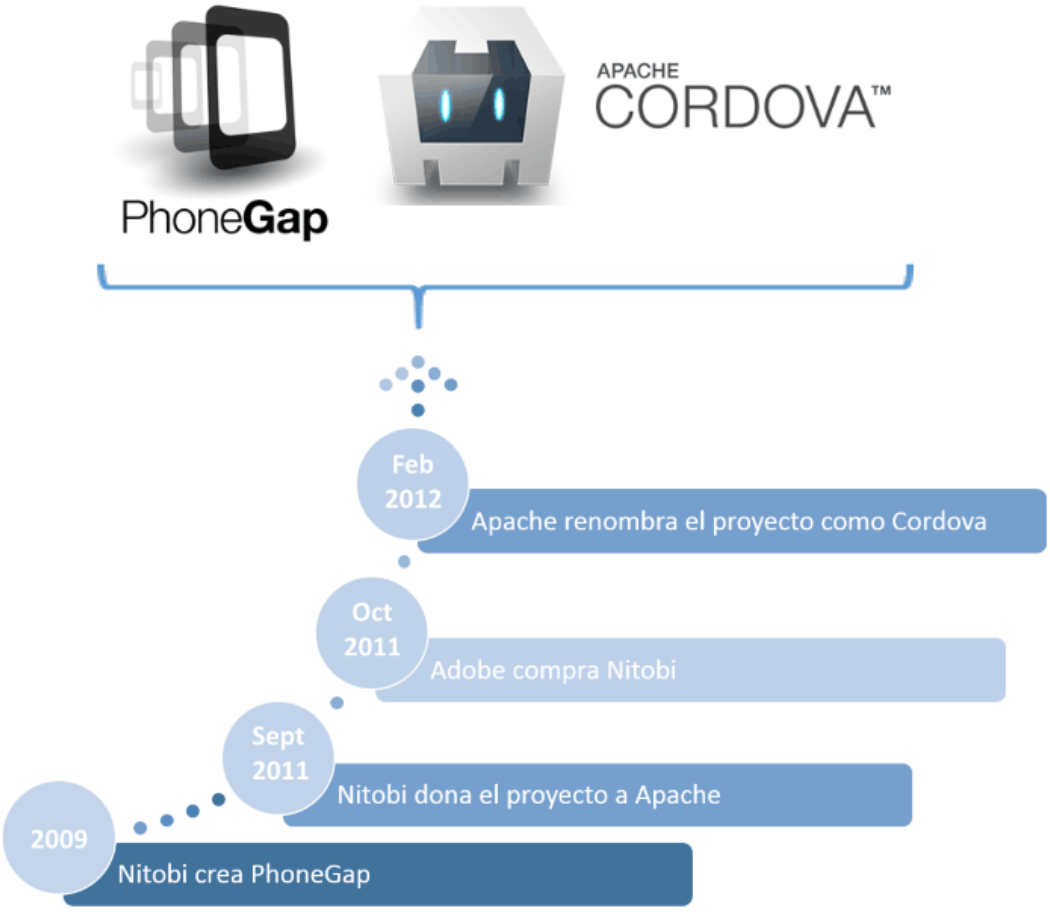
Apache Cordova:

Es un marco de desarrollo de aplicaciones móviles populares originalmente creado por Nitobi, mas tarde fue comprado por Adobe y lo rebautizaron como PhoneGap.

Apache Cordova permite a los programadores de software para construir aplicaciones para dispositivos móviles utilizando CSS3, HTML5 y JavaScript en lugar de depender de las API específicas de la plataforma como los de Android, iOS o Windows Phone. Las aplicaciones resultantes son un híbrido, lo que significa que no son verdaderamente una aplicación móvil ni código.

Parece que tanto Phonegap como Apache Cordova son el mismo producto pero esto no es así ya que aunque funcionalmente no son muy distintos y parece que uno se ha transformado en otro la única diferencia sustancial son los dueños de cada uno.

En el siguiente grafico podemos ver lo que acabamos de explicar:



En definitiva Phonegap no es más que una distribución de Apache Cordova, la diferencia entre ambos a excepción del nombre y de donde se descargan es que Adobe tiene la potestad para extender el producto y dotarlo con herramientas o características propias, por las cuales puede decidir cobrar o no. Cabe destacar también como elemento diferenciador que Phonegap posee integración con los servicios de compilación de Adobe y puede utilizar elementos de la nube. Es por todo esto que en la práctica, se habla de manera indistinta de uno o de otro como si se tratasen del mismo producto cuando en realidad no lo es.

3.1 ELECCION

Analizando en primer lugar nuestros objetivos, por una parte necesitamos una plataforma que nos brinde la oportunidad de crear una aplicación que no solo se ejecute en un PC sino que también nos permita ejecutarse en dispositivos móviles e incluso en navegadores web. Además se quiere aprender a utilizar un nuevo entorno de programación y una nueva forma de programar a la vez que queremos explorar nuevos lenguajes de programación. Por último, nuestra necesidad es la de desarrollar una aplicación de tipo juego.

Por todo esto se decide utilizar Game Maker porque consideramos que es el que más se ajusta a nuestras necesidades, ofreciéndonos así todo lo que necesitamos y además

descubriendo que cuenta con su propia comunidad de desarrolladores y sus propios foros donde se provee soporte a los programadores, siendo muy fácil así resolver cuestiones en las que se pueda necesitar ayuda.

Game Maker a su vez ha atraído un número sustancial de usuarios, entre ellos nosotros, principalmente porque permite accesibilidad a los usuarios principiantes y a los usuarios más avanzados realizar tareas más complejas. No está limitado en la creación de videojuegos como otros programas, pues admite la creación de muchos géneros de videojuegos, incluyendo videojuegos de plataformas, disparos en primera y tercera persona, videojuegos multijugador y videojuegos de simulación.

Cabe destacar que también es posible crear aplicaciones de otros tipos con Game Maker.

3.4 EDICION DE IMAGENES

En este apartado se mencionarán todos los programas utilizados para el tratamiento de las imágenes que forman parte de la aplicación.

3.4.1 FUENTES DE IMAGENES

Todas las imágenes utilizadas en la aplicación se han obtenido a partir de bancos de imágenes gratuitos con licencia Creative Commons o CopyLeft. Entre otros, se han conseguido a partir de la búsqueda de imágenes en Google Imágenes utilizando filtros por licencia, además de otras como WIKIMEDIA COMMONS u OpenPhoto. Además se han creado imágenes propias utilizando programas como Adobe Photoshop y también han colaborado distintas personas que también han participado en el proyecto global de SARE.

3.4.2 ADOBE PHOTOSHOP

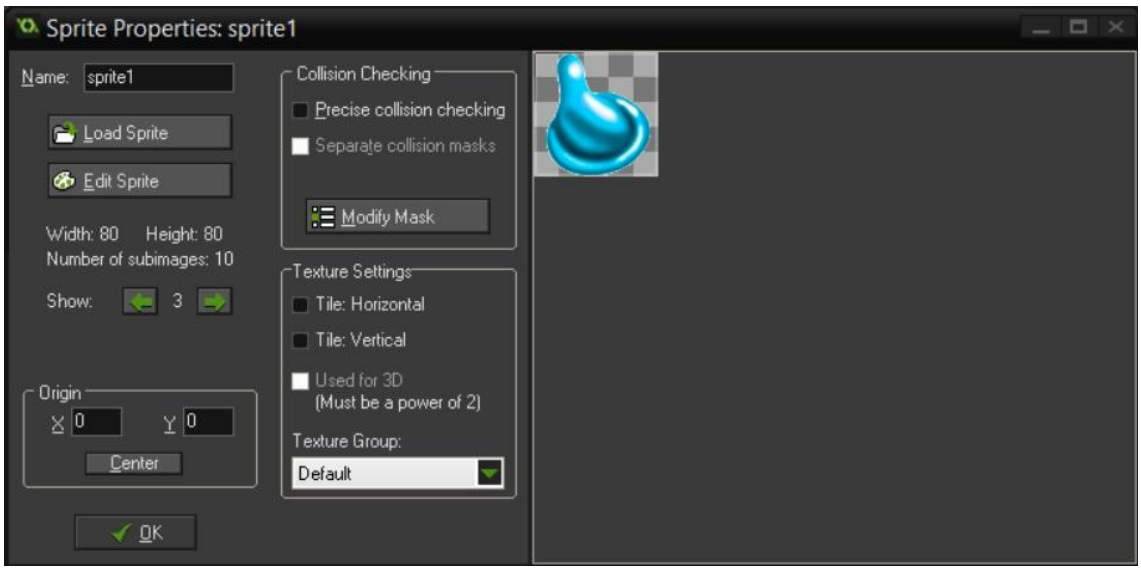
Es una aplicación destinada a la edición, retoque fotográfico y pintura a base de imágenes de mapa de bits. Se ha utilizado para retocar y manipular todas las imágenes que constituyen la aplicación en especial los sprites que forman las piezas del tablero, los backgrounds y en general para crear todas las interfaces gráficas que presenta el juego.

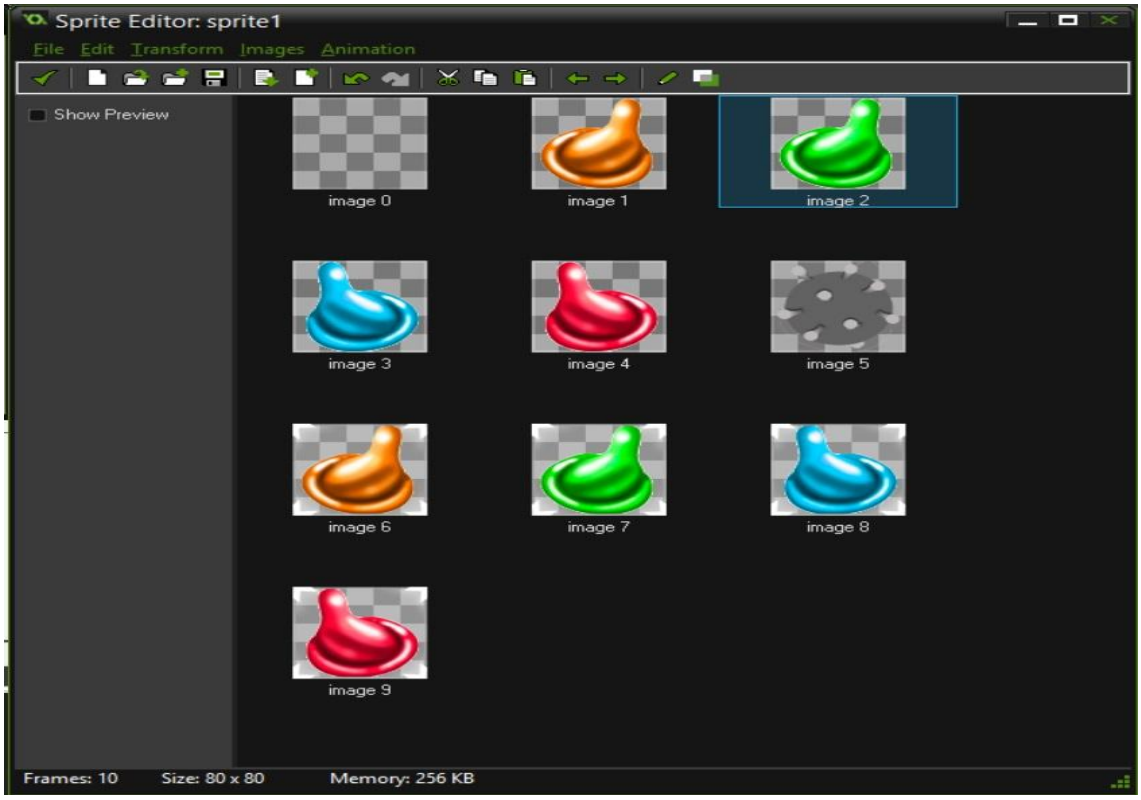
3.4.3 GAME MAKER EDITOR

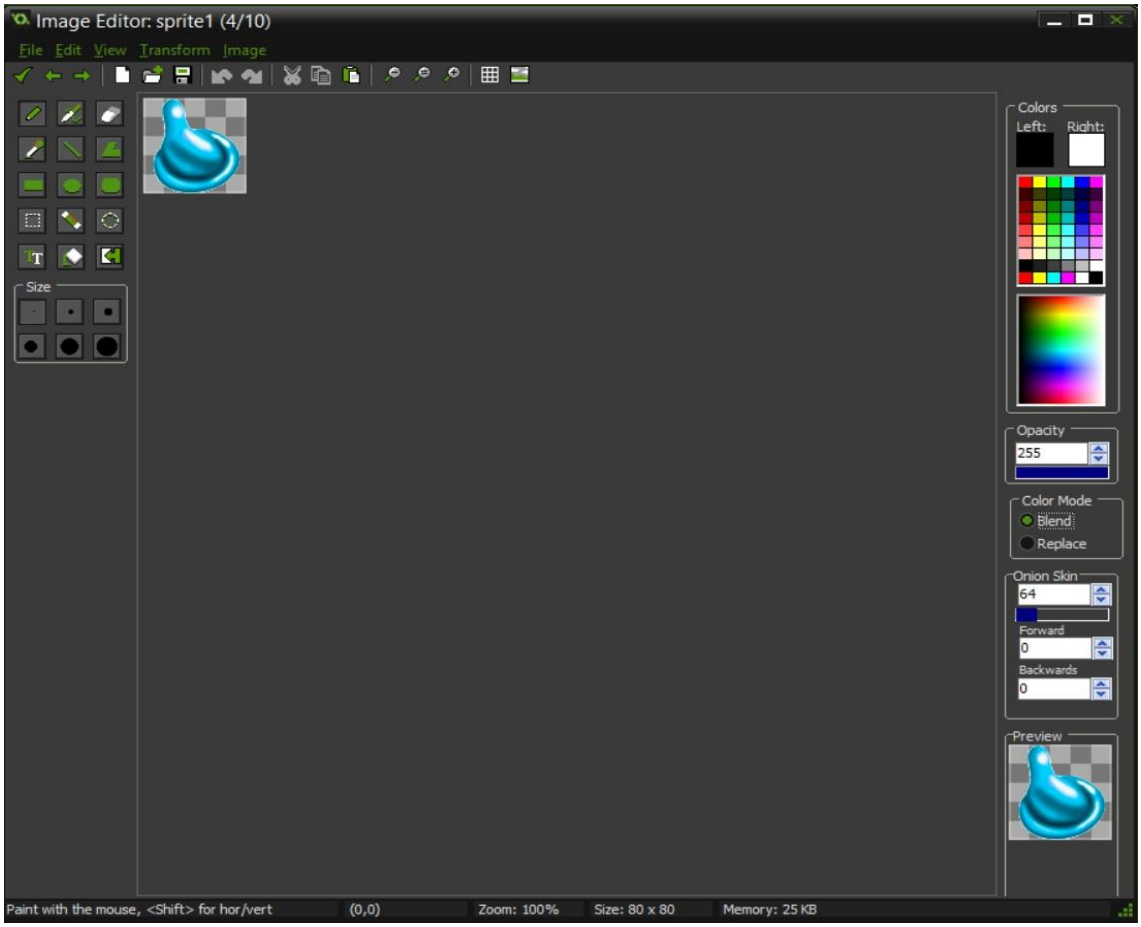
A su vez, el propio Game Maker, nos provee de herramientas para poder crear y retocar las imágenes implicadas en la creación del juego, es algo así como el famoso Paint de Windows pero bastante más completo, ya que nos ofrece la posibilidad de insertar efectos a los sprites, transformaciones, giros, sombreados, suavizado, tratamientos de bordes entre otras cosas.

Además, es un sistema un tanto especial ya que un mismo sprite puede estar compuestos por diferentes subsprites con distintas imágenes, facilitando así la asignación de un sprite a un objeto y que este dependiendo de la situación en la que se encuentre en el juego pueda variar su apariencia sin necesidad de reasignar otra imagen simplemente cambiando su propio sprite por uno de la lista de subimagenes que posea.

En las siguientes imágenes mostramos un claro ejemplo de todo lo descrito anteriormente:





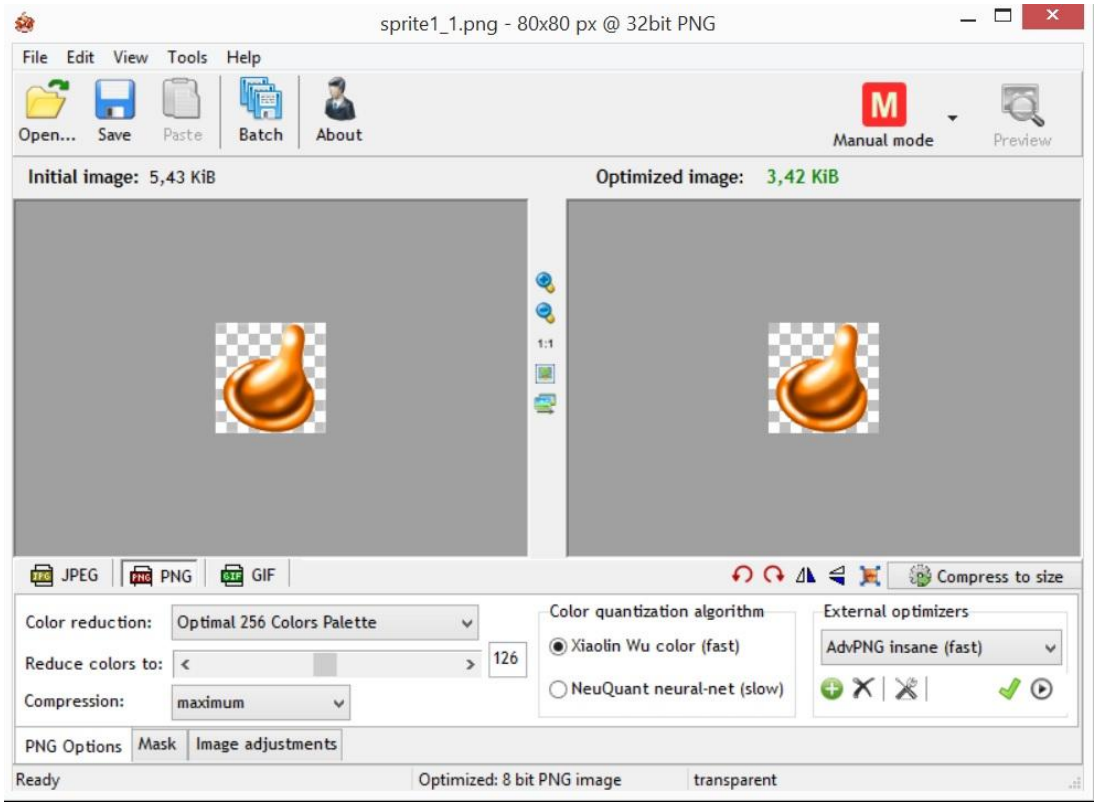


3.4.4 RIOT

Con el fin de aligerar el peso de la aplicación recurrimos al uso de programa que utilizando algoritmos, disminuyen el peso de las imágenes sin afectar a la calidad y tamaño de las mismas.

Radical Image Optimization Tool es un pequeño programa gratuito para Windows que nos permite reducir el peso de las imágenes sin perder mucha calidad en el proceso. Más adelante se verá la necesidad de que nuestra aplicación tenga poco tamaño en MB, con este programa, se ha conseguido reducir a más de la mitad el peso de las imágenes, sin que el ojo humano sea capaz de percibir dicha modificación.

Gracias a este programa se ha conseguido reducir en gran cantidad el peso total del juego ya que dependiendo de la exportación que se hacía, la diferencia entre la aplicación sin tratamiento de optimización frente a la optimizada era del doble de peso, por ejemplo en Android pasamos de 40 MB a un poco mas de 20 MB.



En la imagen anterior podemos ver los resultados de la utilización de este programa, se puede percibir que la calidad de ambas imágenes es similar en cuanto a la visión de la misma, sin embargo el tamaño de la imagen se reduce en un 63%.

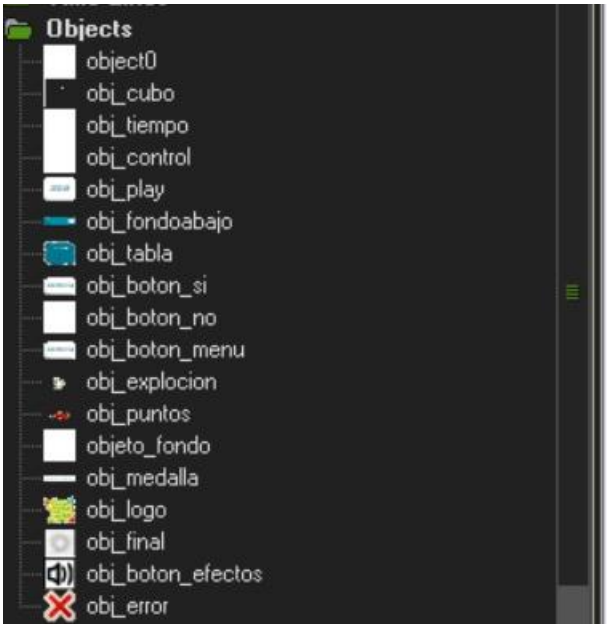
4 IMPLEMENTACION

4.1 ELEMENTOS

Un proyecto en Game Maker cuenta con una serie de elementos de obligada aparición, este es el caso de los objetos que son las piezas elementales del juego sobre los que interactúa el usuario y están dotados de su propia funcionalidad, las rooms que son las "habitaciones" o escenario donde se incluyen y se relacionan los objetos, los backgrounds o fondo sobre el que se pinta y se dibuja el juego formando así parte de la interfaz grafica y por último los sprites que son las imágenes que se asocian a los objetos para dar textura y apariencia a los mismos.

4.1.1 OBJETOS

En nuestro proyecto contamos con un total de 18 objetos, los cuales son los siguientes:



Objetc0: es el objeto principal del juego, por eso se llama así, para indicar que es el origen del juego. Este objeto es el encargado de gestionar la creación y disposición del tablero. Además es el encargado de crear los objetos o piezas que conformar dicho tablero y de controlar su interacción con el usuario en los distintos eventos en los que puede ver envuelto, de los cuales hablaremos en posteriores apartados.A partir de este objeto optamos por denotar todos los objetos como obj_ y la función que cumplan o lo que representen.

obj_cubo: este objeto es la pieza de tipo preservativo en sí. El objeto anterior genera tantos de estos como huecos del tablero, en nuestro caso 9x9 es decir 81 objetos de este tipo. En los diseños iniciales del proyecto las piezas eran cubos y otras figuras geométricas, de ahí ese nombre aunque finalmente fueron preservativos de colores.

obj_tiempo: este es el encargado de llevar a cabo la gestión y control del tiempo, ya que preestablece un tiempo definido para cada nivel, dibuja en la interfaz una barra en la que se ve como se consume este tiempo gráficamente y controla el inicio y fin de este.

obj_control: maneja el fichero de datos que guarda la puntuación de cada usuario. Es capaz de leer y obtener la información de este y sobrescribir el valor.

obj_play: es el que permite empezar a jugar, su representación es la de un botón para poder acceder a la pantalla inicial de juego y además controla pequeños efectos visuales sobre el mismo permitiendo así agrandarse y encogerse gráficamente con el fin de llamar la atención del usuario.

obj_fondoabajo: este objeto simplemente lleva a cabo la función grafica de ser la franja que separa el tablero de el resto de información (Nivel actual, tiempo y puntuación). Además se relaciona con el objeto de control para presentar la puntuación en la pantalla.

obj_tabla: se relaciona con el objeto de tiempo y se pinta cuando este detecta que el tiempo ha llegado a cero. Es la base sobre la que se dibujan los botones de reintentar, siguiente nivel y menú principal, así que verdaderamente actúa como fondo de imagen y creador de estos últimos tres mencionados.

obj_boton_si: objeto que permite reintentar el nivel en el que se encuentre el usuario dentro del juego.

obj_boton_no: objeto que permite pasar al siguiente nivel de juego.

obj_menu: objeto que permite ir al menú principal desde cualquier nivel del juego.

obj_explosion: este objeto lleva a cabo un efecto visual al generar una línea de tres o más piezas del mismo tiempo. En este caso simula una eyaculación mediante la reproducción de un gif creado expresamente para eso, incidiendo así en la temática en la que se pretende centrar el juego.

obj_puntos: este objeto pretende aportar otro efecto grafico a la hora de la creación de líneas para poder visualizar la obtención de puntos, no obstante se decidió no utilizarlo ya que interfería con el objeto anterior y no resultaba una composición agradable de ver.

obj_fondo: este objeto apoya al objeto0 y para darle la textura y presentación lograda al tablero mediante transparencias.

obj_medalla: es el encargado de dibujar la puntuación máxima obtenida en la pantalla principal. Al principio su representación iba a ser la de una medalla, por eso su nombre aunque finalmente cambio a la actual por decisiones entre desarrollador y clientes.

obj_logo: este objeto solo sirve de apoyo para dibujar la interfaz principal, se decidió aislarlo por las dudas de si dotarlo de funcionalidad o no en vez de integrarlo como background.

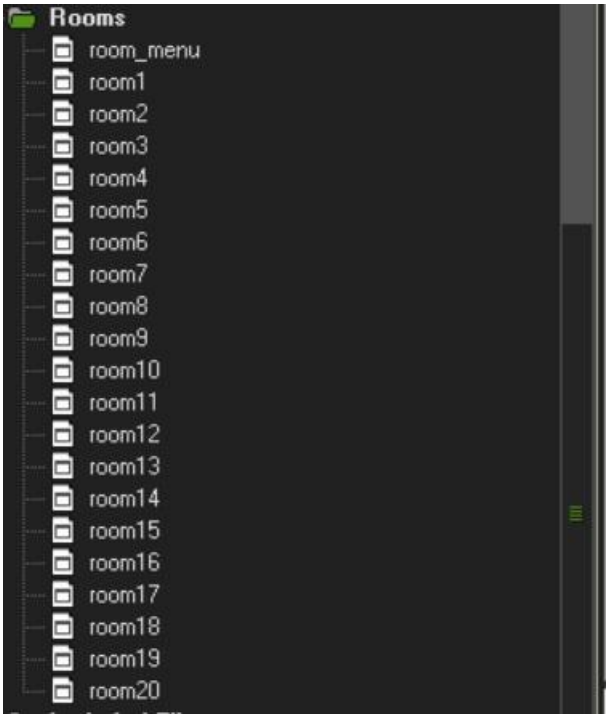
obj_final: al terminar el tiempo del último nivel, se crea este objeto que a su vez crea el objeto del menú principal para poder acceder a este. Muestra una pequeña reflexión sobre el uso de preservativos para evitar la infección de enfermedades de transmisión sexual.

obj_boton_efectos: este objeto con forma de botón controlaba el volumen de los sonidos del juego, se suprimió su uso debido a la interferencia con otras funcionalidades del juego y porque se decidió al final no utilizar audio.

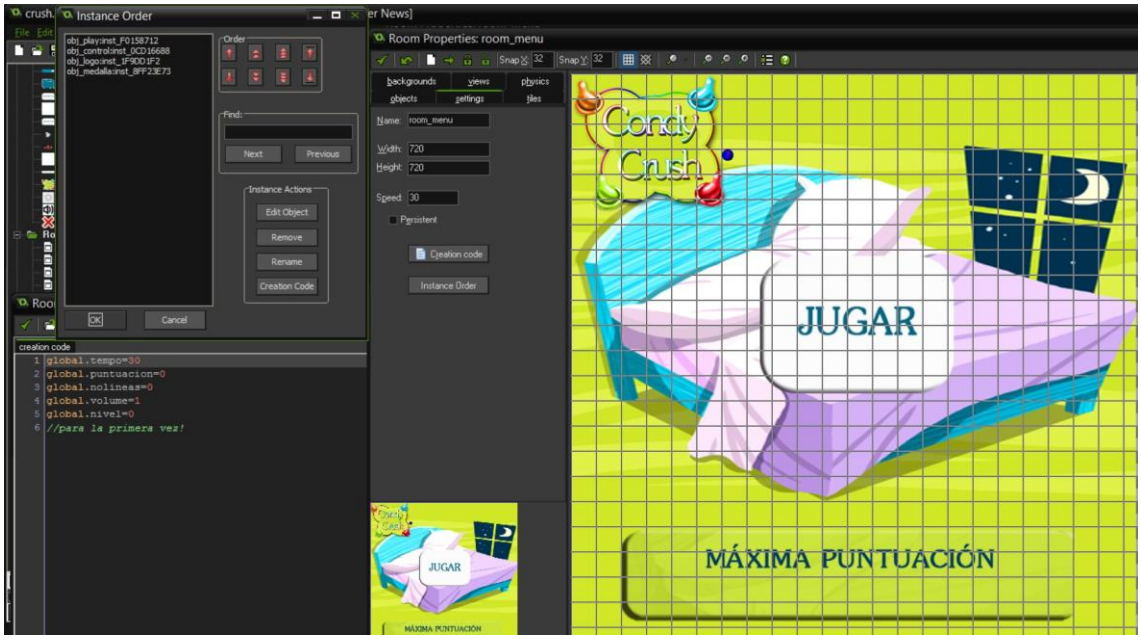
obj_error: este objeto se genera y crea un efecto óptico de movimiento no permitido tras llevar a cabo el intercambio de dos piezas del tablero sin que estas vayan a provocar líneas de elementos iguales.

4.1.2 ROOMS

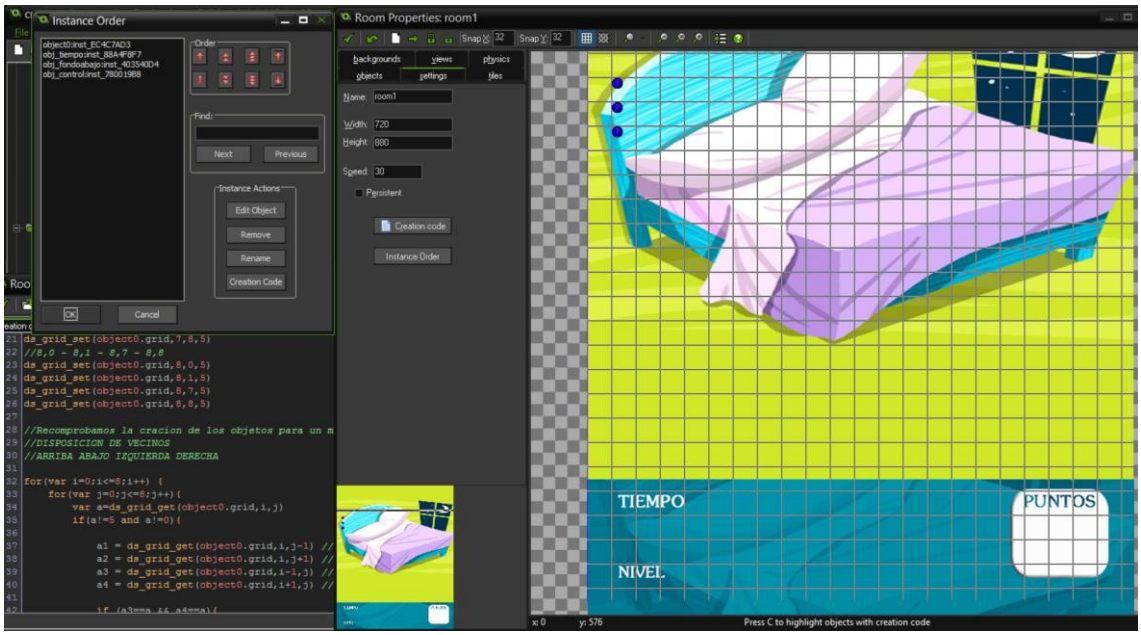
Cuando hablamos de rooms o "habitaciones", nos referimos al lugar donde cohabitan y se relacionan todos los objetos, en este proyecto se han utilizado para definir los distintos niveles del juego.



Como podemos observar en la imagen, en nuestro caso se han definido una room principal que es la que aparece en primer lugar en el juego, es como el menú principal desde el que se observa la puntuación máxima del jugador y desde el que se accede a la opción de jugar. El resto de rooms son los niveles y contamos con veinte niveles distintos, cada uno con una disposición distinta del tablero para darle más dificultad y entretenimiento al juego, haciendo así pasar al jugador por cada uno de estos veinte niveles para poder completar el juego.



En esta room podemos ver cómo queda diseñado el menú principal, iniciando así varias variables globales del juego y viendo las instancias de los objetos que se crean.



En esta otra room correspondiente al nivel 1, se pueden ver que la apariencia es prácticamente la misma que la anterior pero incorporando los elementos visuales de tiempo, puntuación actual y nivel y eliminando otros objetos como el logo y el de máxima puntuación. El código que se presenta es un fragmento en lenguaje GML de la creación y disposición de las piezas del tablero y también podemos apreciar los objetos que intervienen en el nivel en la pantalla de instancias. Además visualmente aparecen varios puntos azules con un interrogante, esto son otros objetos presentes en la room pero que no tienen un sprite definido de primeras y por eso su representación es la que es. El resto de objetos que si poseen apariencia se muestran con la suya propia como es el ejemplo del faldón inferior donde se sitúa el resto de información.

No mostraremos más room porque son idénticas, lo único que cambia es la configuración inicial del tablero cambiando así partes del código expuesto. Por tanto solo mostraremos un ejemplo completo de como se lleva a cabo dicha configuración ya que en las sucesivas solo cambian los valores de varios de estos parámetros.

creation code

```
1  //Ponemos a valor 5 las casillas no permitidas
2  //Para la primera room seran las casillas
3  global.puntuacionaux=global.puntuacion
4  global.tempo=obj_tiempo.tiempo
5  global.nivel++
6
7  //0,0 - 0,1 - 0,7 - 0,8
8  ds_grid_set(object0.grid,0,0,5)
9  ds_grid_set(object0.grid,0,1,5)
10 ds_grid_set(object0.grid,0,7,5)
11 ds_grid_set(object0.grid,0,8,5)
12 //1,0 - 1,1 - 1,7 - 1,8
13 ds_grid_set(object0.grid,1,0,5)
14 ds_grid_set(object0.grid,1,1,5)
15 ds_grid_set(object0.grid,1,7,5)
16 ds_grid_set(object0.grid,1,8,5)
17 //7,0 - 7,1 - 7,7 - 7,8
18 ds_grid_set(object0.grid,7,0,5)
19 ds_grid_set(object0.grid,7,1,5)
20 ds_grid_set(object0.grid,7,7,5)
21 ds_grid_set(object0.grid,7,8,5)
22 //8,0 - 8,1 - 8,7 - 8,8
23 ds_grid_set(object0.grid,8,0,5)
24 ds_grid_set(object0.grid,8,1,5)
25 ds_grid_set(object0.grid,8,7,5)
26 ds_grid_set(object0.grid,8,8,5)
27
28 //Recomprobamos la cracion de los objetos para un mejor reparto
29 //DISPOSICION DE VECINOS
30 //ARRIBA ABAJO IZQUIERDA DERECHA
31
32 for(var i=0;i<=8;i++) {
33     for(var j=0;j<=8;j++){
34         var a=ds_grid_get(object0.grid,i,j)
35         if(a!=5 and a!=0){
36
37             a1 = ds_grid_get(object0.grid,i,j-1) //ARRIBA
38             a2 = ds_grid_get(object0.grid,i,j+1) //ABAJO
39             a3 = ds_grid_get(object0.grid,i-1,j) //IZQUIERDA
40             a4 = ds_grid_get(object0.grid,i+1,j) //DERECHA
41
```

Room Creation Code: room1

creation code

```
37              a1 = ds_grid_get(object0.grid,i,j-1) //ARRIBA
38              a2 = ds_grid_get(object0.grid,i,j+1) //ABAJO
39              a3 = ds_grid_get(object0.grid,i-1,j) //IZQUIERDA
40              a4 = ds_grid_get(object0.grid,i+1,j) //DERECHA
41
42              if (a3==a && a4==a){
43                  a++
44                  if (a==5){
45                      a=1
46                  }
47                  //ds_grid_set(object0.grid,i,j,a)
48                  if (a1==a && a2==a){
49                      a++
50                      if (a==5){
51                          a=1
52                      }
53                  }
54              }
55              if (a1==a && a2==a){
56                  a++
57                  if (a==5){
58                      a=1
59                  }
60                  //ds_grid_set(object0.grid,i,j,a)
61                  if (a3==a && a4==a){
62                      a++
63                      if (a==5){
64                          a=1
65                      }
66                  }
67              }
68              if (a3==a || a1==a) {
69                  a++
70                  if (a==5) {
71                      a=1
72                  }
73              }
74              ds_grid_set(object0.grid,i,j,a)
75          }
76      }
77  }
```

Primero se obtiene el valor y se modifican las variables globales que se ven envueltas en la room.

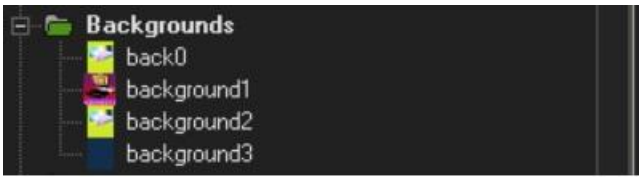
Después definimos unas casillas no permitidas dándole el valor 5 ya que nuestras piezas tienen el valor de 1 a 4 para representar un preservativo de una forma y color u otra.

Si nos fijamos bien nos referimos al object0.grid esta es la estructura de tipo matriz que forma el tablero con lo cual solo estamos modificando sus valores ya que previamente ya estaba creada esta estructura.

Así pues se recorre todo el tablero comprobando los vecinos más cercanos (arriba, abajo, izquierda, derecha) y viendo que no se generen líneas desde el principio ya eso no tendría ningún sentido y corrigiéndolo en el caso de que esto suceda.

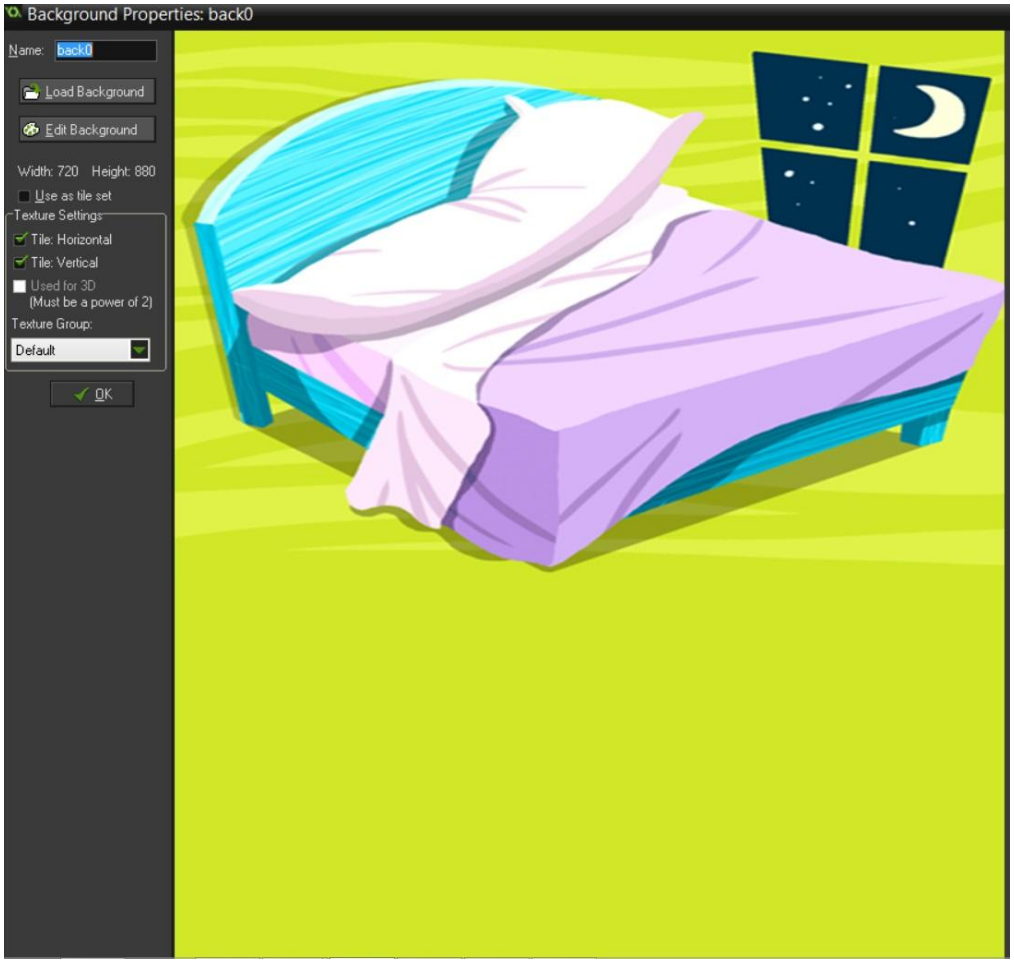
4.1.3 BACKGROUNDS

Este apartado es muy básico ya que los backgrounds no ofrecen otra cosa más que la presentación visual de la estética final del juego construyéndolo a base de imágenes.



Como podemos ver esta carpeta contiene las distintas imágenes que formar parte del juego como fondos. En nuestro caso se ha utilizado solo la imagen back0 y background3 ya que el resto eran pruebas similares con pequeñas variaciones de colores y formas.

He aquí el resultado de la imagen principal que se mantiene a lo largo del juego:



4.1.4 SPRITES

Los sprites son las imágenes que dotan de representación grafica a los objetos y demás elementos del juego.

Los sprites más relevantes son:

sprite1: es el que representa las piezas en forma de preservativos. En apartados anteriores hemos podido ver en una imagen la representación de estos y la composición de los mismos en subimagenes del mismo sprite. Es el que se asocia al objeto object0.

spr_tiempo: conforma dos barras de tiempo que se va agotando gráficamente, una de estas barras es el contenedor de la otra de forma que se crean dos subimagenes para llevar a cabo el efecto. Se asocia al obj_tiempo.

spr_play y spr_botones: representan todos los botones del juegos.

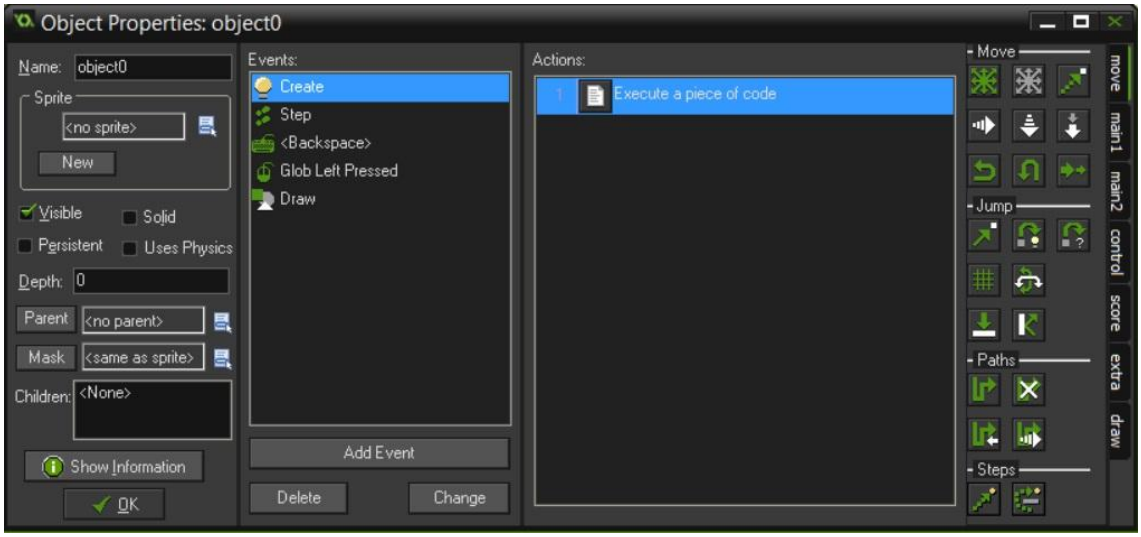
spr_explocion: efecto grafico que se lleva a cabo tras generar líneas.

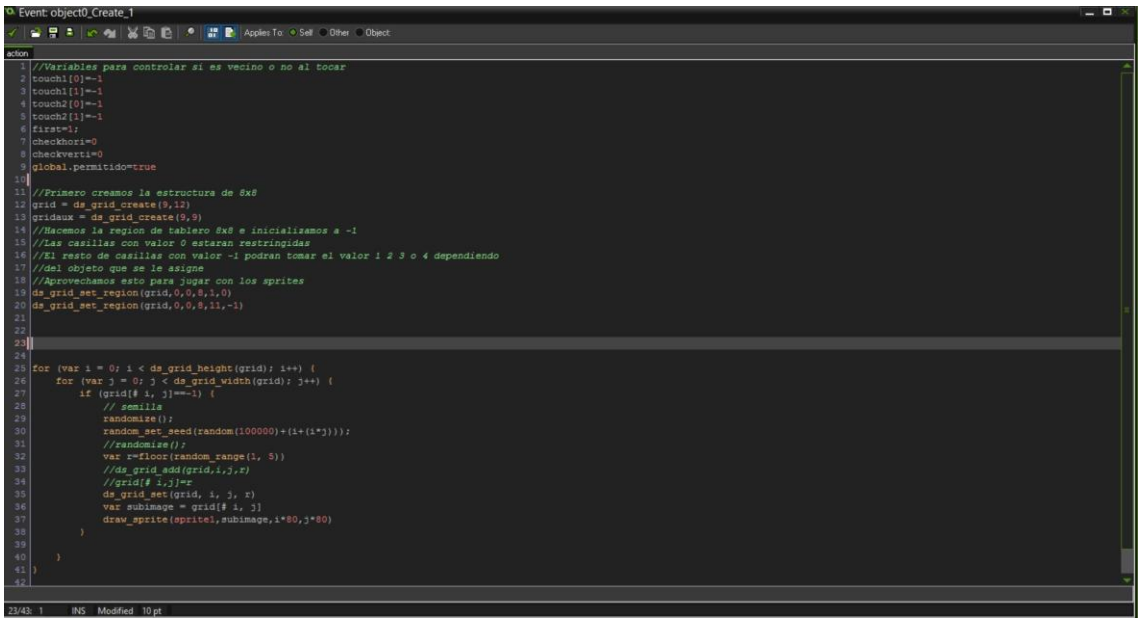
El resto de ellos podemos ver o bien por el nombre o ver por la imagen en miniatura lo que representan y deducir para que se utilizan.

4.1 EVENTOS

Game Maker posee unas herramientas o mecanismos que poseen los objetos y que recogen los sucesos que ocurren en el juego tras la interacción con estos. En esta sección mostraremos entonces los eventos que hemos necesitado para llevar a cabo la correcta funcionalidad del juego.

En la mayoría de objetos se ha utilizado el evento **Create**, este se da cuando se instancia por primera vez un objeto en la creación de una room. Se usa principalmente para inicializar variables y se ejecuta solo una vez por instancia.



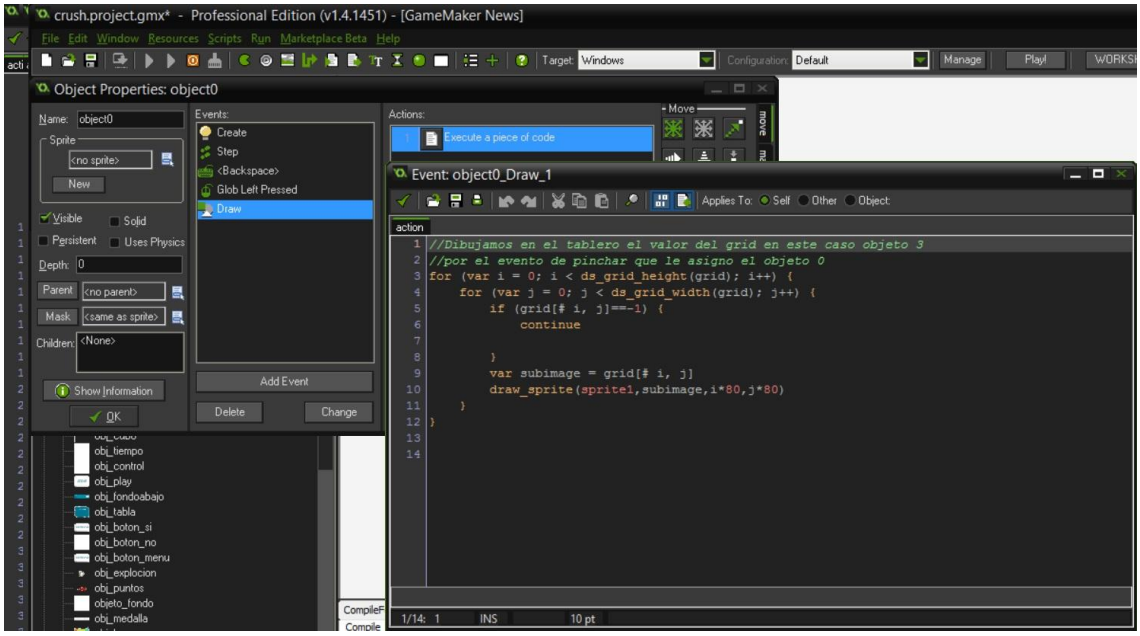


En el evento Create del object0 podemos ver como se inicializan las variables al principio del código y se lleva a cabo un algoritmo para inicializar el tablero con piezas de forma aleatoria.

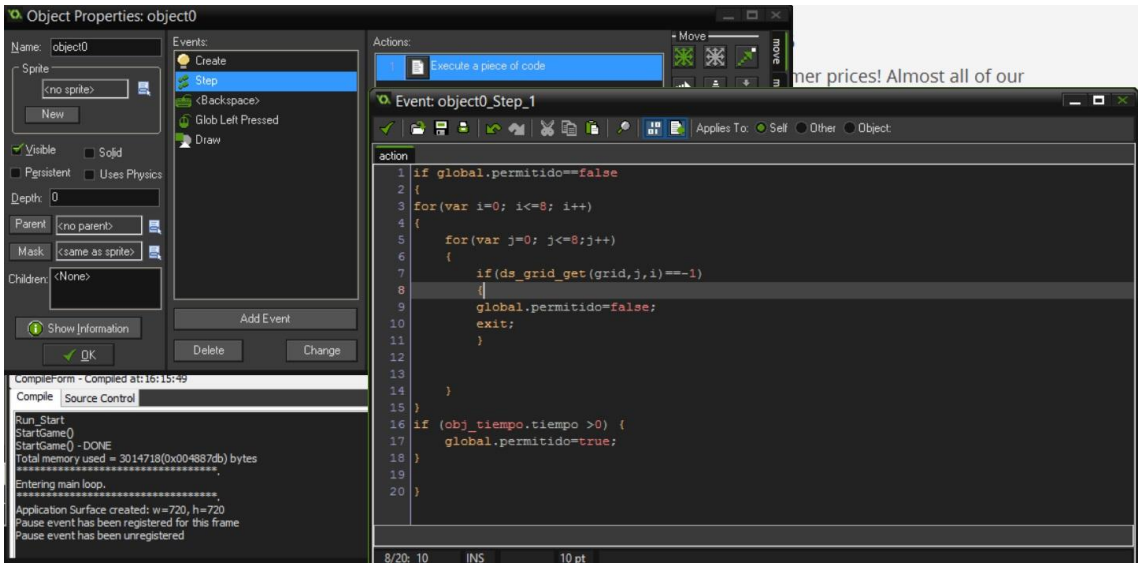
Por otro lado, nos encontramos con un evento muy importante, este es el evento **Step**, el cual lleva la cuenta del número de acciones que se realizan por segundo, en la mayoría de los casos estos están directamente ligados a los frames por segundo (FPS). El numero de estos pasos se configuran en cada room del juego por separado través del parámetro speed, en nuestra caso hemos elegido 30 pasos por segundo.

Por ejemplo, si en un objeto utilizamos este evento para sumar 1 a una variable esto producida que en un segundo se sumen 30 a esa variable.

Este evento influye directamente en el evento **Draw** ya que define el número de veces que se dibujara la pantalla por segundo. En el evento Draw podremos elegir la prioridad de dibujo es decir nos permite preparar en qué orden se dibuja todo lo que tenga cada objeto.

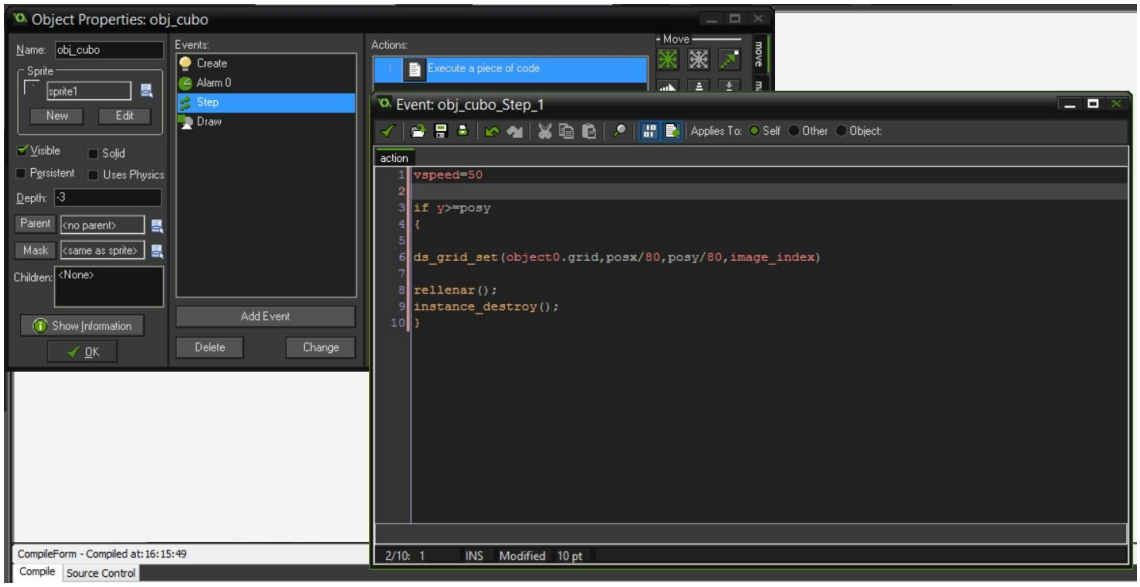


En definitiva el evento Step nos sirve para llevar a cabo comprobaciones continuamente y si fuera necesario este se descompone en dos subeventos Begin Step y End Step por si se necesitase hacer estas comprobaciones al inicio o al final de cada paso.

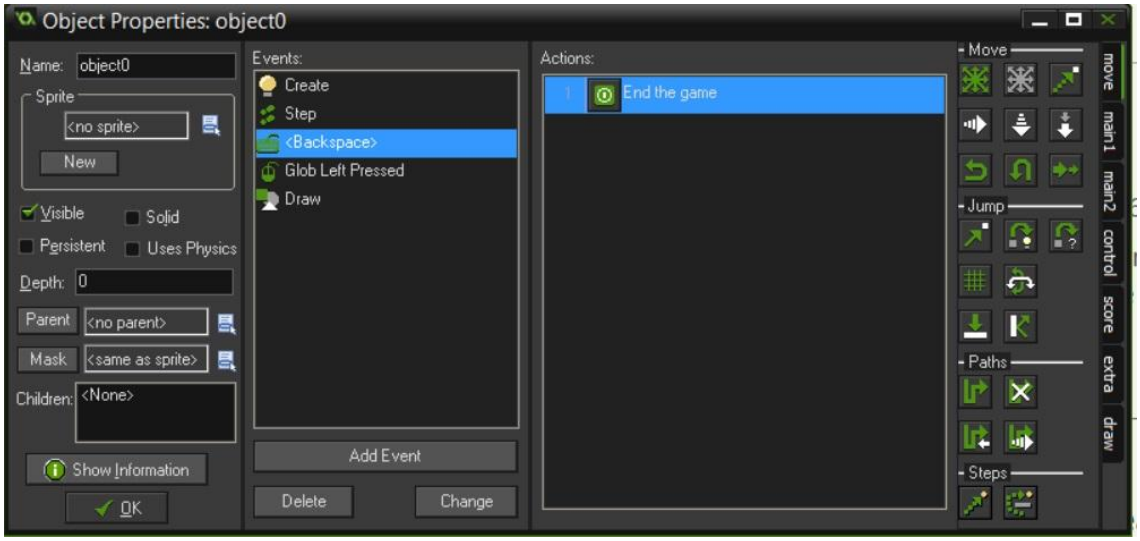


Por ejemplo en el object0 el evento step nos sirve para comprobar si se puede jugar (porque estamos dentro del tiempo de juego) o no y llevar a cabo las acciones necesarios.

Existen otros objetos que también utilizan este evento, este es el caso del obj_cubo (las piezas que forman el tablero), que nos sirve para asignar una velocidad de desplazamiento vertical a los objetos de este tipo y si se cumple una condición determinada actualizar varios valores, llamar a otras funcionalidades o scripts de los cuales hablaremos mas tarde y destruir el objeto.

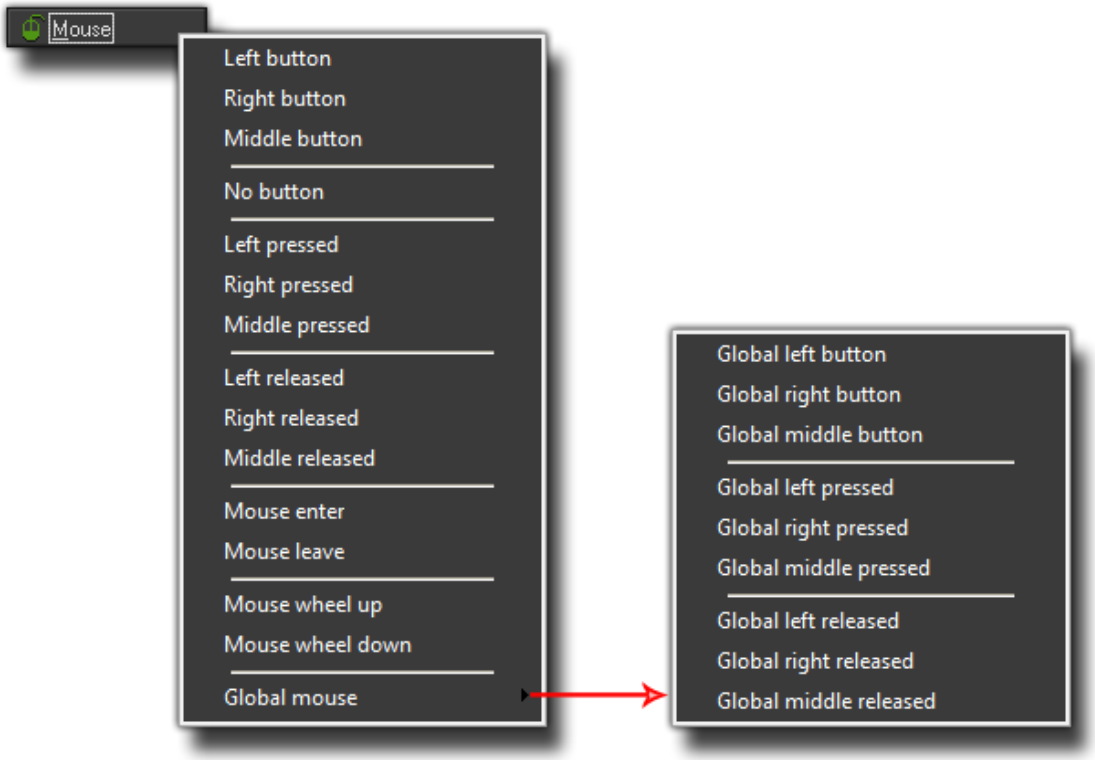


El evento **BackSpace** nos sirve para comprobar cuando un usuario ha pulsado la tecla de atrás en un dispositivo móvil o la tecla con el mismo nombre en un teclado si se ejecuta en un entorno de ordenador. La acción básica de este evento es la de terminar el juego o ir a la pantalla principal.



También nos encontramos con eventos de mouse o touch como son **Global Left Presed**, **Global Left Released**, **Mouse Leave**, etc. que sirven para controlar la interacción del usuario al hacer click o al tocar estos objetos en una pantalla.

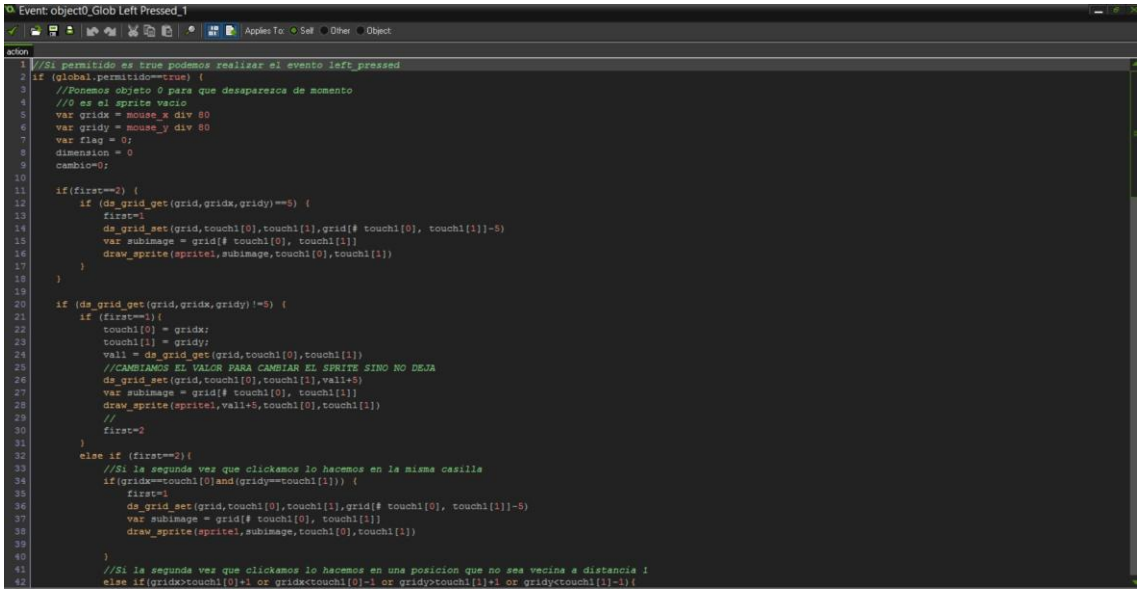
Como podemos ver existen los eventos de ratón normales y los globales. La diferencia de estos últimos con respecto al resto es que no solo toman en cuenta si la propia instancia de un objeto controla el evento de click sino que tiene en cuenta todas las instancias de todos los objetos de una room al clicar en el.



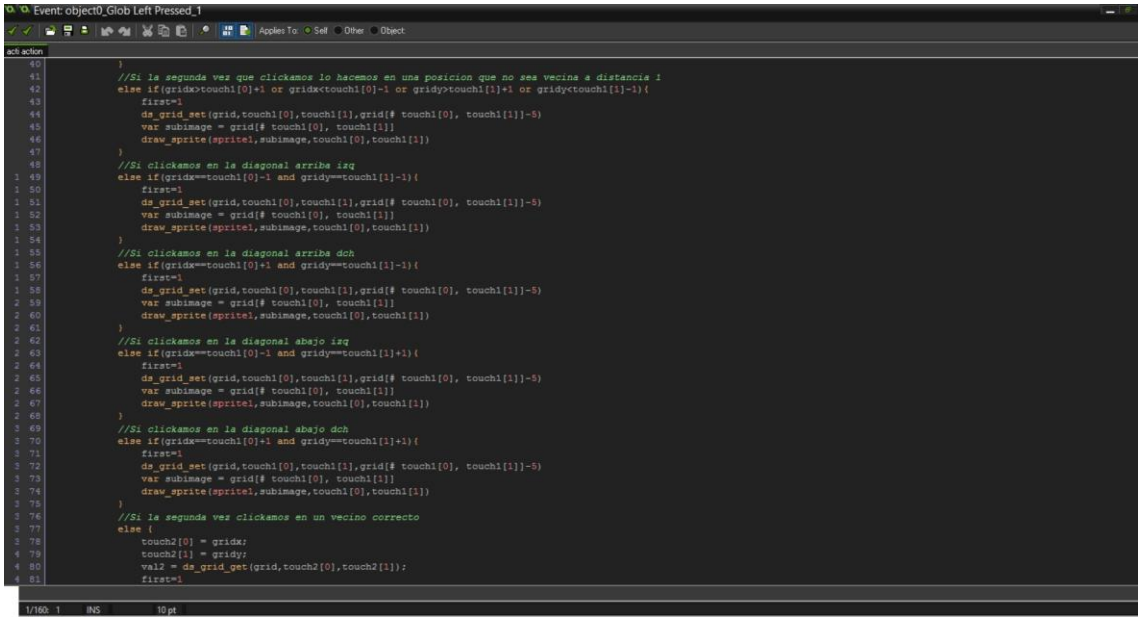
Por ejemplo el evento Global Left Pressed del objetc0 lleva a cabo las siguientes acciones:

Si podemos jugar, obtenemos las coordenadas de la pieza, comprobamos si estamos en la primera selección de la pieza o si hemos seleccionado ya la segunda pieza para hacer el cambio. Obtenemos los valores que necesitamos y decidimos si se puede llevar a cabo el intercambio.

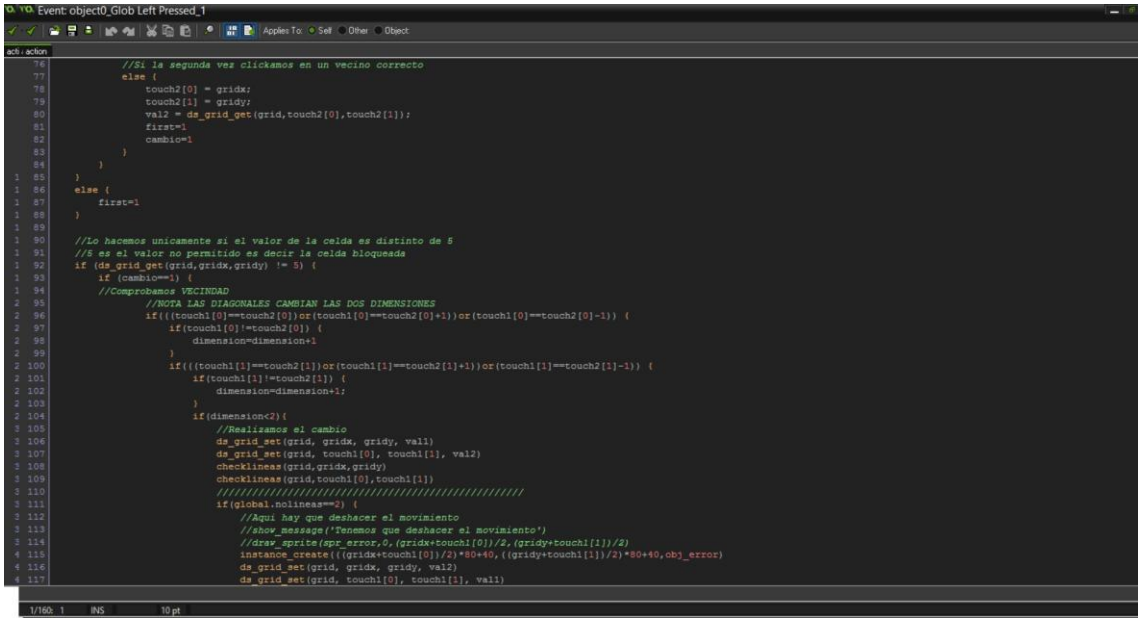
Si esto es así se refresca el tablero con los cambios actualizados y llevamos a cabo la comprobación de la generación de líneas, llamando a su vez a otras funciones que comprueban si la generación de líneas aporta así nuevas líneas las cuales también deben ser contabilizadas, destruidas y sustituidas por otros elementos.

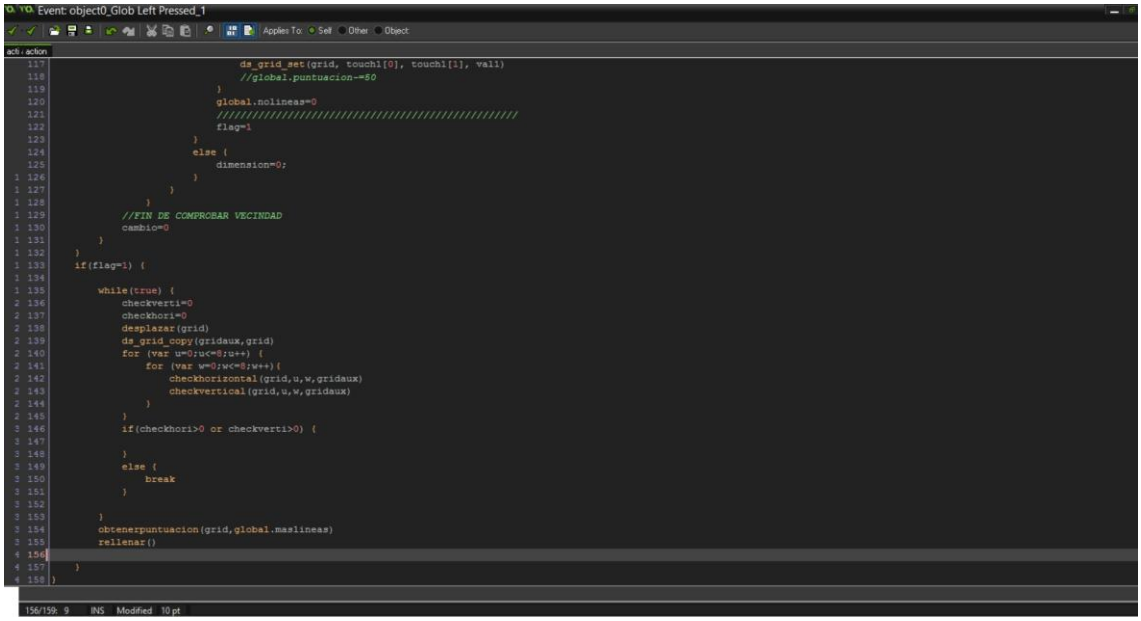



```
//Si permitido es true podemos realizar el evento left_pressed
if (global.permitido==true) {
    //Ponemos objeto 0 para que desaparezca de momento
    //0 es el sprite vacio
    var gridx = mouse_x div 80
    var gridy = mouse_y div 80
    var flag = 0;
    dimension = 0
    cambio=0;

    if(first==2) {
        if (ds_grid_get(grid,gridx,gridy)==5) {
            first=1
            ds_grid_set(grid,touch1[0],touch1[1],grid[# touch1[0], touch1[1]]-5)
            var subimage = grid[# touch1[0], touch1[1]]
            draw_sprite(sprite1,subimage,touch1[0],touch1[1])
        }
    }

    if (ds_grid_get(grid,gridx,gridy)!=5) {
        if (first==1){
            touch1[0] = gridx;
            touch1[1] = gridy;
            val1 = ds_grid_get(grid,touch1[0],touch1[1])
            //CAMBIAMOS EL VALOR PARA CAMBIAR EL SPRITE SINO NO DEJA
            ds_grid_set(grid,touch1[0],touch1[1],val1+5)
            var subimage = grid[# touch1[0], touch1[1]]
            draw_sprite(sprite1,val1+5,touch1[0],touch1[1])
            //
            first=2
        }
        else if (first==2){
            //Si la segunda vez que clickamos lo hacemos en la misma casilla
            if(gridx==touch1[0]and(gridy==touch1[1])) {
                first=1
                ds_grid_set(grid,touch1[0],touch1[1],grid[# touch1[0], touch1[1]]-5)
                var subimage = grid[# touch1[0], touch1[1]]
                draw_sprite(sprite1,subimage,touch1[0],touch1[1])

            }
            //Si la segunda vez que clickamos lo hacemos en una posicion que no sea vecina a distancia 1
            else if(gridx>touch1[0]+1 or gridx<touch1[0]-1 or gridy>touch1[1]+1 or gridy<touch1[1]-1){
```

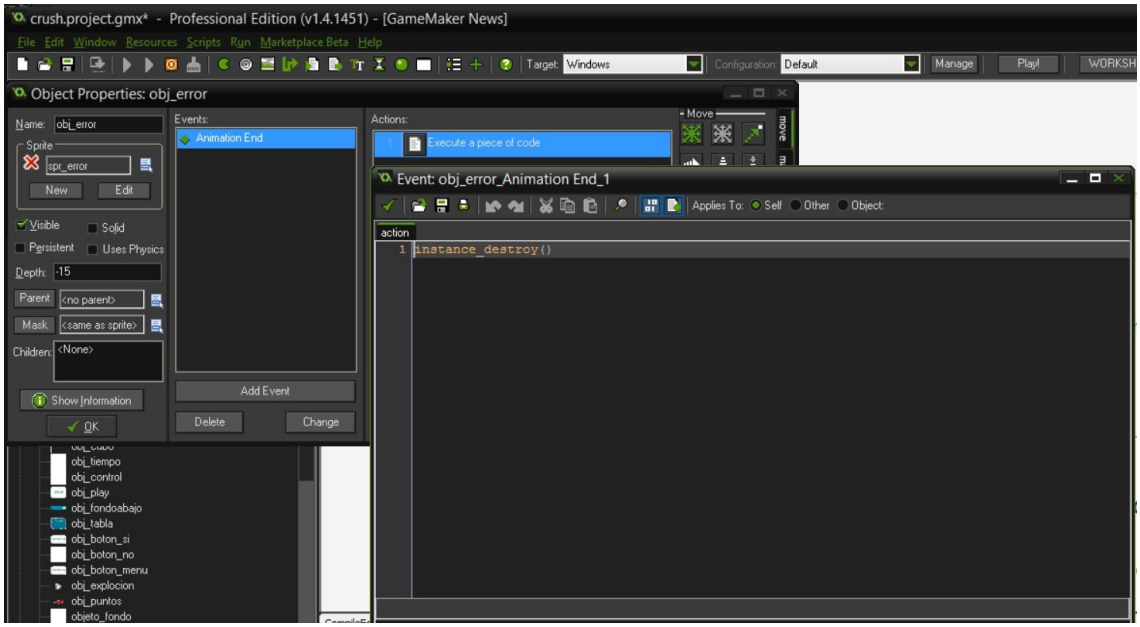



```
            }
            //Si la segunda vez que clickamos lo hacemos en una posicion que no sea vecina a distancia 1
            else if(gridx>touch1[0]+1 or gridx<touch1[0]-1 or gridy>touch1[1]+1 or gridy<touch1[1]-1){
                first=1
                ds_grid_set(grid,touch1[0],touch1[1],grid[# touch1[0], touch1[1]]-5)
                var subimage = grid[# touch1[0], touch1[1]]
                draw_sprite(sprite1,subimage,touch1[0],touch1[1])
            }
            //Si clickamos en la diagonal arriba izq
            else if(gridx==touch1[0]-1 and gridy==touch1[1]-1){
                first=1
                ds_grid_set(grid,touch1[0],touch1[1],grid[# touch1[0], touch1[1]]-5)
                var subimage = grid[# touch1[0], touch1[1]]
                draw_sprite(sprite1,subimage,touch1[0],touch1[1])
            }
            //Si clickamos en la diagonal arriba dch
            else if(gridx==touch1[0]+1 and gridy==touch1[1]-1){
                first=1
                ds_grid_set(grid,touch1[0],touch1[1],grid[# touch1[0], touch1[1]]-5)
                var subimage = grid[# touch1[0], touch1[1]]
                draw_sprite(sprite1,subimage,touch1[0],touch1[1])
            }
            //Si clickamos en la diagonal abajo izq
            else if(gridx==touch1[0]-1 and gridy==touch1[1]+1){
                first=1
                ds_grid_set(grid,touch1[0],touch1[1],grid[# touch1[0], touch1[1]]-5)
                var subimage = grid[# touch1[0], touch1[1]]
                draw_sprite(sprite1,subimage,touch1[0],touch1[1])
            }
            //Si clickamos en la diagonal abajo dch
            else if(gridx==touch1[0]+1 and gridy==touch1[1]+1){
                first=1
                ds_grid_set(grid,touch1[0],touch1[1],grid[# touch1[0], touch1[1]]-5)
                var subimage = grid[# touch1[0], touch1[1]]
                draw_sprite(sprite1,subimage,touch1[0],touch1[1])
            }
            //Si la segunda vez clickamos en un vecino correcto
            else {
                touch2[0] = gridx;
                touch2[1] = gridy;
                val2 = ds_grid_get(grid,touch2[0],touch2[1]);
                first=1
```



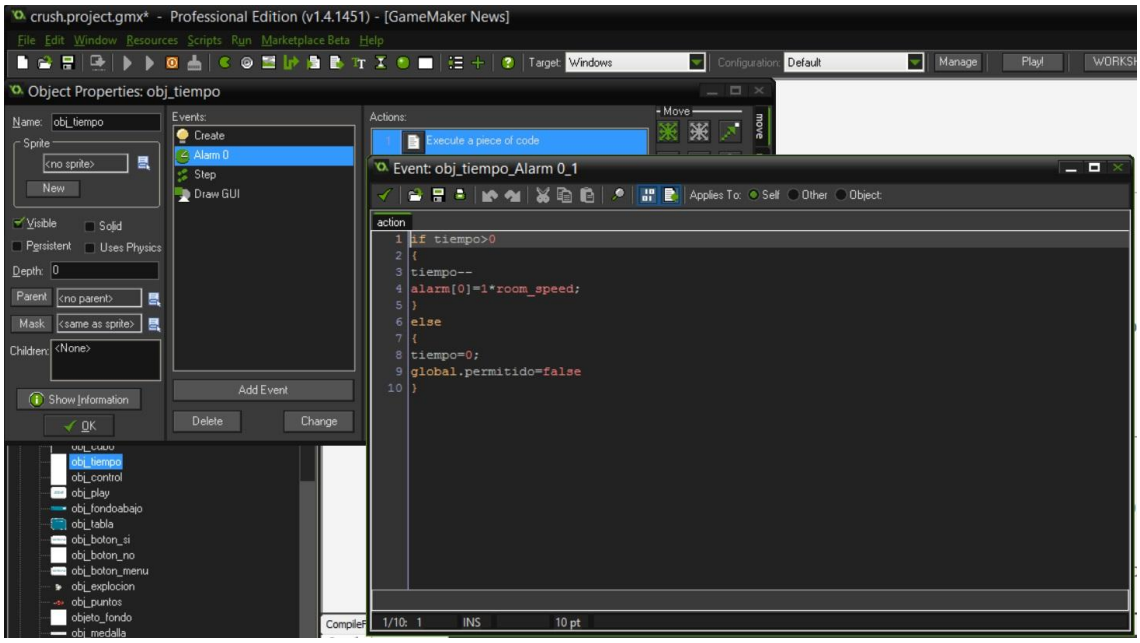


Por otro lado tenemos el evento **Animation End** que es utilizado en los objetos que llevan a cabo transiciones y efectos visuales y sirve para controlar el fin de esa animación, en nuestro proyecto esto es cuando un GIF termina su reproducción.



Otro evento muy importante que utilizan muchos de los objetos que componen el juego es el evento **Alarm**, en realidad nos encontramos con un array de 12 alarmas denotadas desde Alarm0 hasta Alarm11 y nos sirven para llevar a cabo acciones tras concluir cierto tiempo medido en steps. Hay que mencionar que una alarma se ejecuta solamente si se le ha llamado previamente con alguna acción.

Por ejemplo para controlar el tiempo de juego en cada nivel, creamos el evento Alarm0 y asignamos las acciones de comprobar si nuestra variable de tiempo es mayor que 0, si esto es así le restamos uno para decrementarlo y asignamos de nuevo un valor a la alarma para que se vuelva a lanzar y comprobar. Si nuestra alarma comprueba que nuestro tiempo es 0 significa que se ha terminado el tiempo de juego y no habilitamos seguir tocando las piezas del tablero.

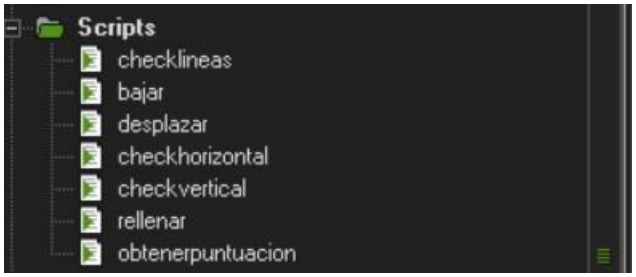


En resumen estos son los eventos más importantes que hemos utilizado a lo largo del proyecto no obstante, existen otros eventos que no hemos usado como el evento destroy, el evento colisión, otro eventos de teclado y de ratón, eventos de desplazamiento entre rooms, eventos de intersección de límites, de inicio y fin de juego, eventos que miden las "vidas" o oportunidades de juego, eventos asíncronos para cargar imágenes, realizar peticiones HTTP, eventos sociales, etc.

4.3 SCRIPTS Y ALGORITMOS

Game Maker provee de la creación y edición de scripts, esto es un trozo de programa escrito en lenguaje GML para poder aislar una funcionalidad. Los scripts aceptan valores de entrada que se almacenan en variables que puede ser de tipo numero, texto, sprites, etc. A estos valores les llamamos parámetros. También es posible devolver un valor.

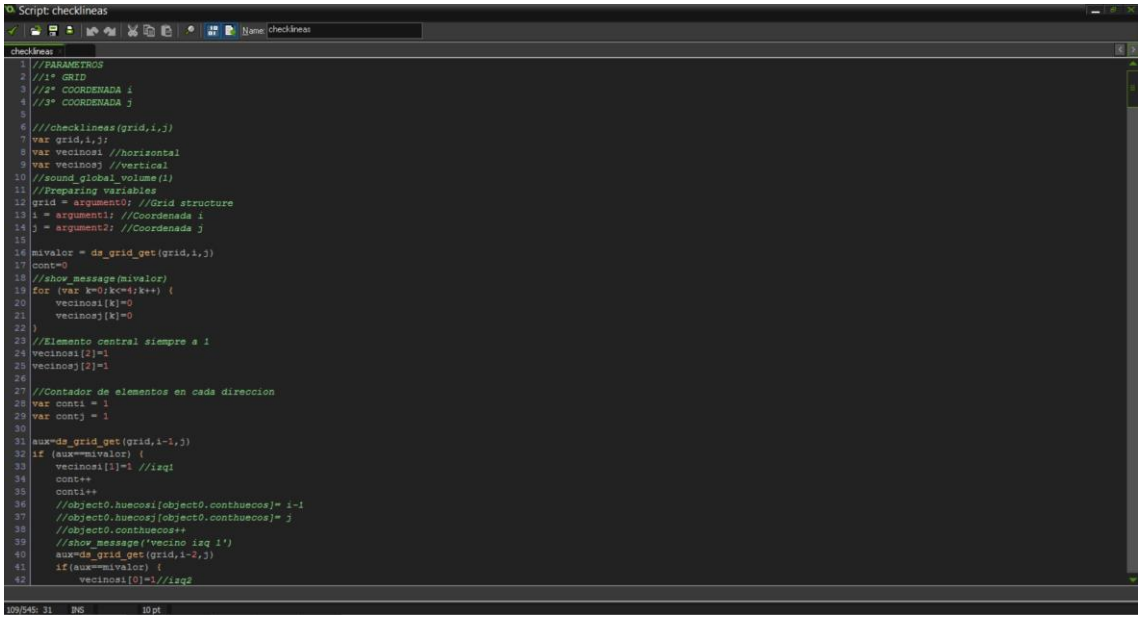
En nuestro proyecto hemos desarrollado los siguientes scripts:

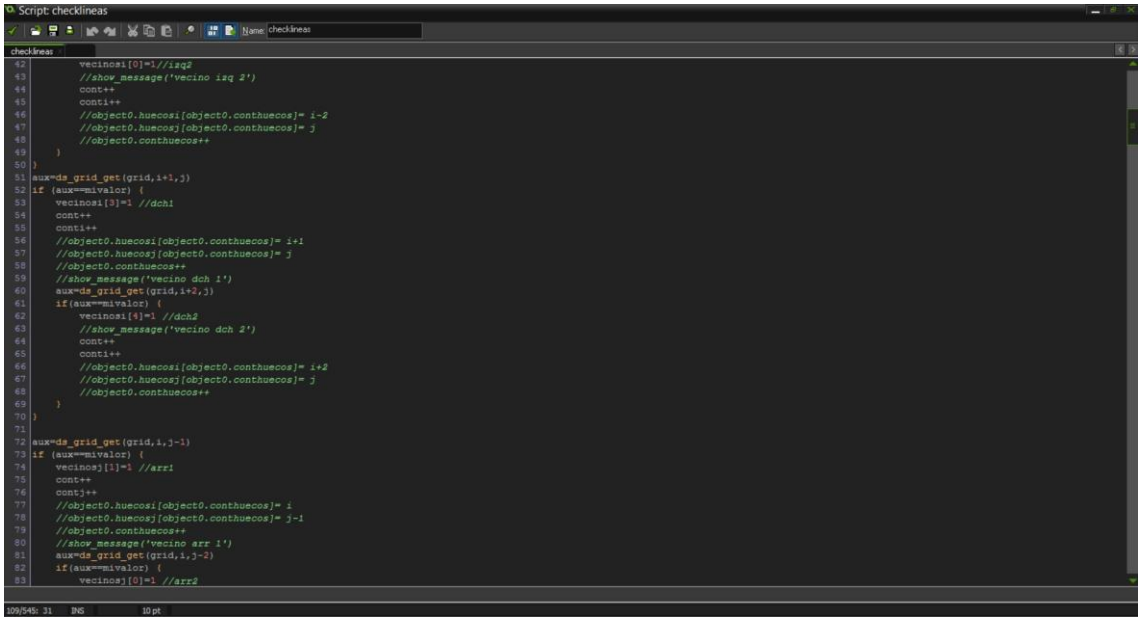


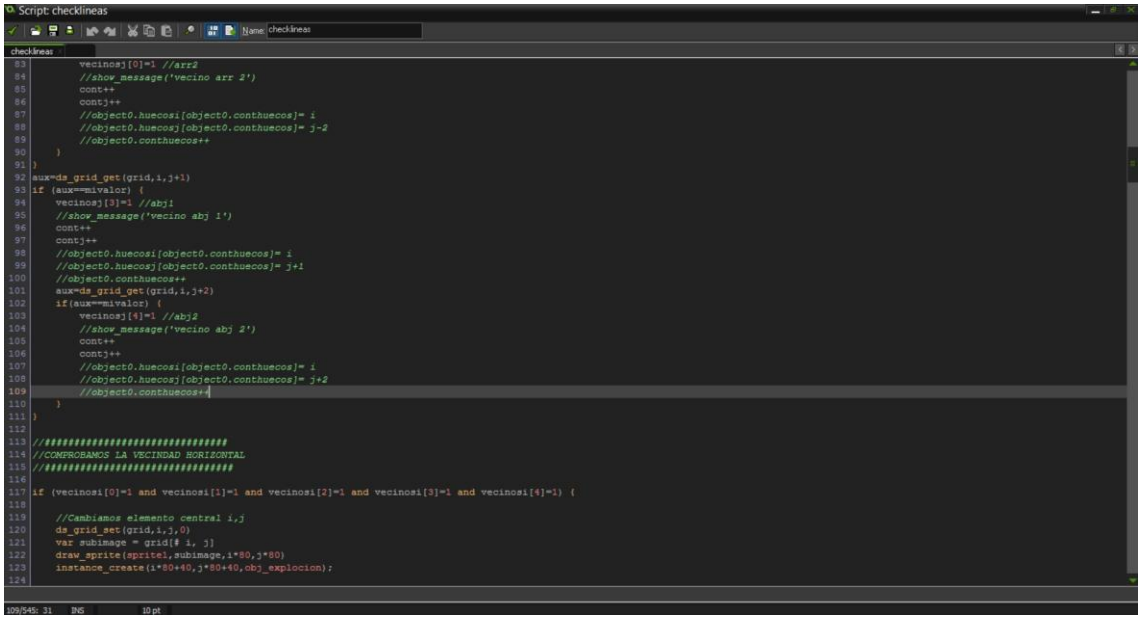
checklineas: script que comprueba la existencia de líneas de tres o más elementos del mismo tipo, en nuestro caso del mismo tipo de preservativos con la misma apariencia, tras el intercambio de dos fichas del tablero.

Consideramos este script uno de los más importantes ya que es una función básica del juego por lo tanto lo explicaremos más extensamente.

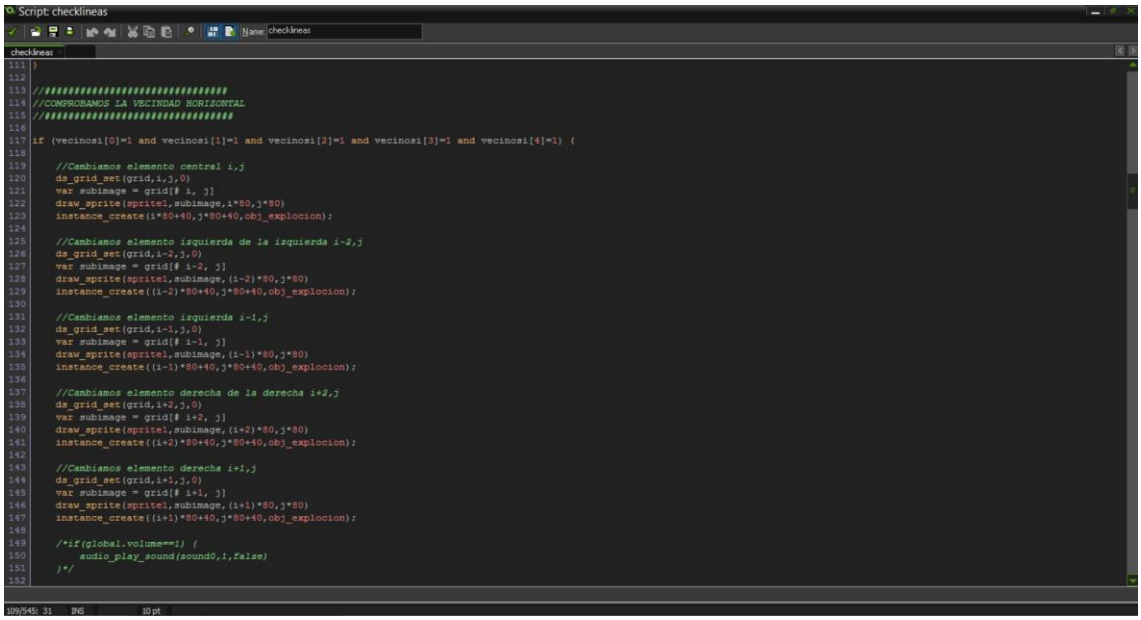
Primero se declaran las variables y se reciben los parámetros. Después inicializamos dos arrays de vecinos a 0, estos son los vecinos horizontales y verticales mas próximos a las coordenadas de la pieza seleccionada, los dos vecinos de arriba, los dos de abajo, los dos de la izquierda y los dos de la derecha.

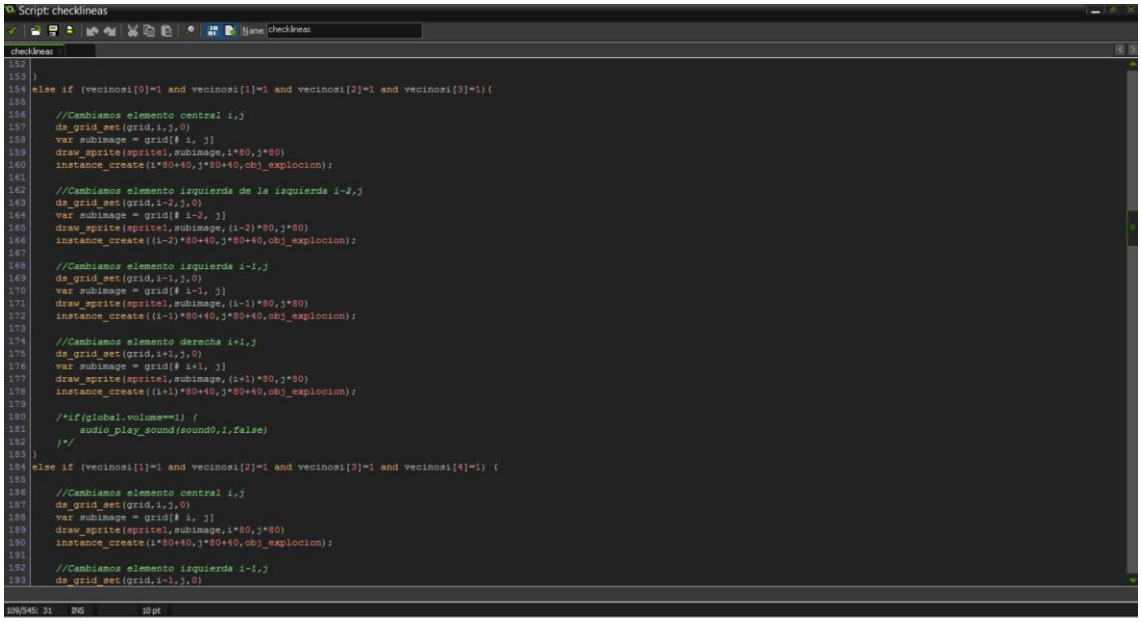




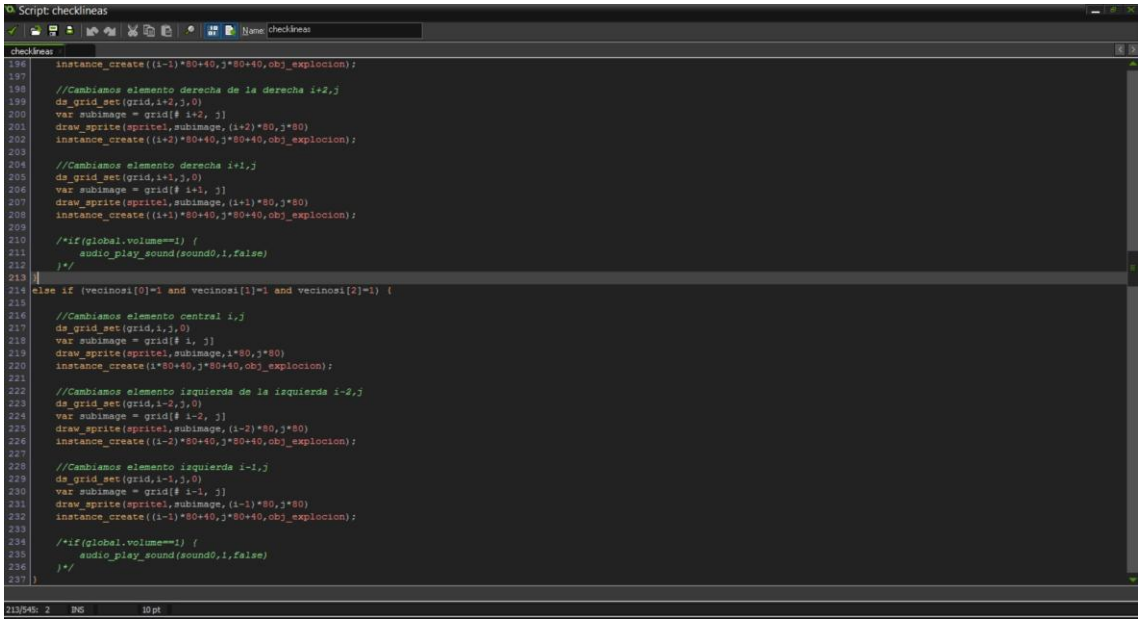


Una vez tenemos los arrays de vecinos indicando si estos poseen el mismo valor que la pieza seleccionada se comprueban las distintas opciones de vecindad posibles, pudiéndose formar tanto líneas horizontales como líneas verticales de mínimo 3 elementos hasta un máximo de 5. Si esto sucede se reemplazan todos los elementos iguales destruyéndose mediante un efecto visual.

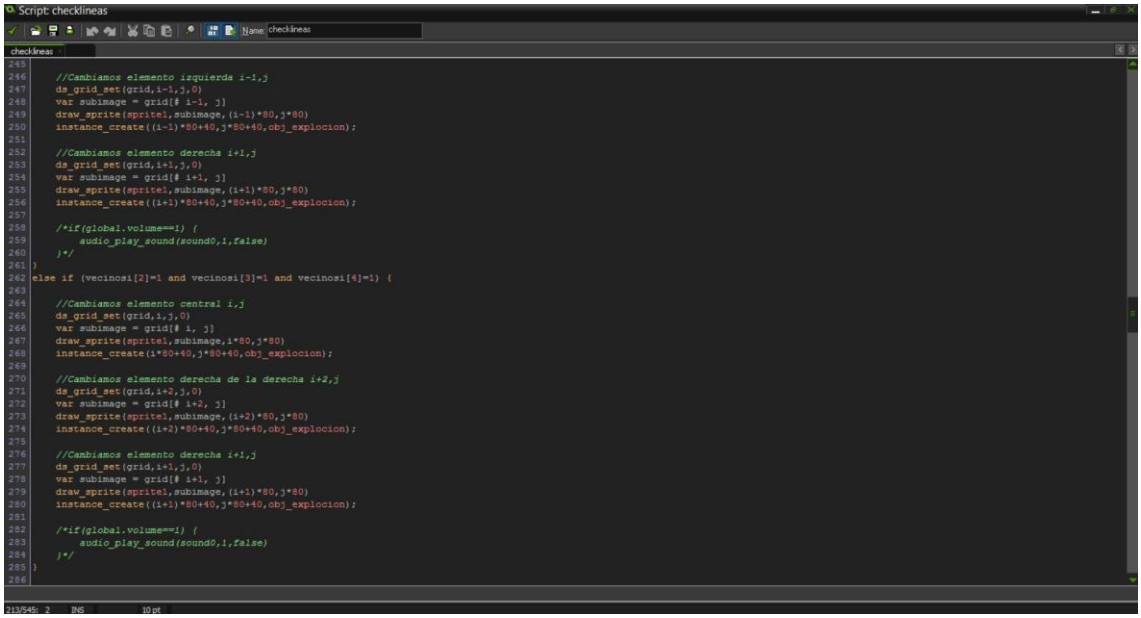




```
}
else if (vecinosi[0]=1 and vecinosi[1]=1 and vecinosi[2]=1 and vecinosi[3]=1){

    //Cambiamos elemento central i,j
    ds_grid_set(grid,i,j,0)
    var subimage = grid[# i, j]
    draw_sprite(sprite1,subimage,i*80,j*80)
    instance_create(i*80+40,j*80+40,obj_explocion);

    //Cambiamos elemento izquierda de la izquierda i-2,j
    ds_grid_set(grid,i-2,j,0)
    var subimage = grid[# i-2, j]
    draw_sprite(sprite1,subimage,(i-2)*80,j*80)
    instance_create((i-2)*80+40,j*80+40,obj_explocion);

    //Cambiamos elemento izquierda i-1,j
    ds_grid_set(grid,i-1,j,0)
    var subimage = grid[# i-1, j]
    draw_sprite(sprite1,subimage,(i-1)*80,j*80)
    instance_create((i-1)*80+40,j*80+40,obj_explocion);

    //Cambiamos elemento derecha i+1,j
    ds_grid_set(grid,i+1,j,0)
    var subimage = grid[# i+1, j]
    draw_sprite(sprite1,subimage,(i+1)*80,j*80)
    instance_create((i+1)*80+40,j*80+40,obj_explocion);

    /*if(global.volume==1) {
        audio_play_sound(sound0,1,false)
    }*/
}
else if (vecinosi[1]=1 and vecinosi[2]=1 and vecinosi[3]=1 and vecinosi[4]=1) {

    //Cambiamos elemento central i,j
    ds_grid_set(grid,i,j,0)
    var subimage = grid[# i, j]
    draw_sprite(sprite1,subimage,i*80,j*80)
    instance_create(i*80+40,j*80+40,obj_explocion);

    //Cambiamos elemento izquierda i-1,j
    ds_grid_set(grid,i-1,j,0)
```

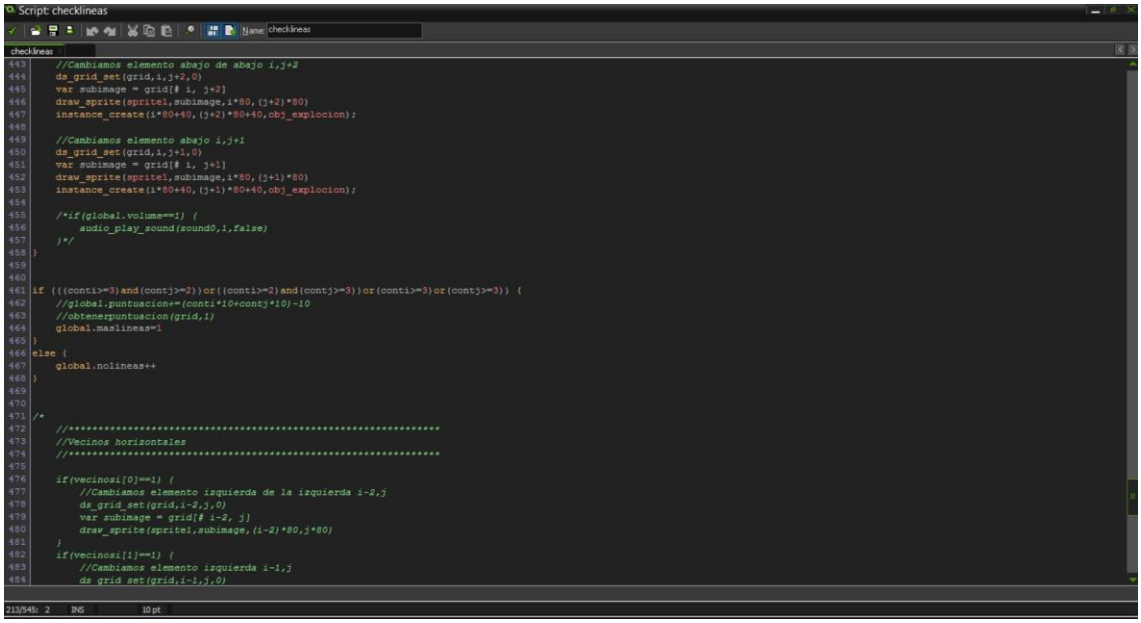


```
    instance_create((i-1)*80+40,j*80+40,obj_explocion);

    //Cambiamos elemento derecha de la derecha i+2,j
    ds_grid_set(grid,i+2,j,0)
    var subimage = grid[# i+2, j]
    draw_sprite(sprite1,subimage,(i+2)*80,j*80)
    instance_create((i+2)*80+40,j*80+40,obj_explocion);

    //Cambiamos elemento derecha i+1,j
    ds_grid_set(grid,i+1,j,0)
    var subimage = grid[# i+1, j]
    draw_sprite(sprite1,subimage,(i+1)*80,j*80)
    instance_create((i+1)*80+40,j*80+40,obj_explocion);

    /*if(global.volume==1) {
        audio_play_sound(sound0,1,false)
    }*/
}
else if (vecinosi[0]=1 and vecinosi[1]=1 and vecinosi[2]=1) {

    //Cambiamos elemento central i,j
    ds_grid_set(grid,i,j,0)
    var subimage = grid[# i, j]
    draw_sprite(sprite1,subimage,i*80,j*80)
    instance_create(i*80+40,j*80+40,obj_explocion);

    //Cambiamos elemento izquierda de la izquierda i-2,j
    ds_grid_set(grid,i-2,j,0)
    var subimage = grid[# i-2, j]
    draw_sprite(sprite1,subimage,(i-2)*80,j*80)
    instance_create((i-2)*80+40,j*80+40,obj_explocion);

    //Cambiamos elemento izquierda i-1,j
    ds_grid_set(grid,i-1,j,0)
    var subimage = grid[# i-1, j]
    draw_sprite(sprite1,subimage,(i-1)*80,j*80)
    instance_create((i-1)*80+40,j*80+40,obj_explocion);

    /*if(global.volume==1) {
        audio_play_sound(sound0,1,false)
    }*/
}
```

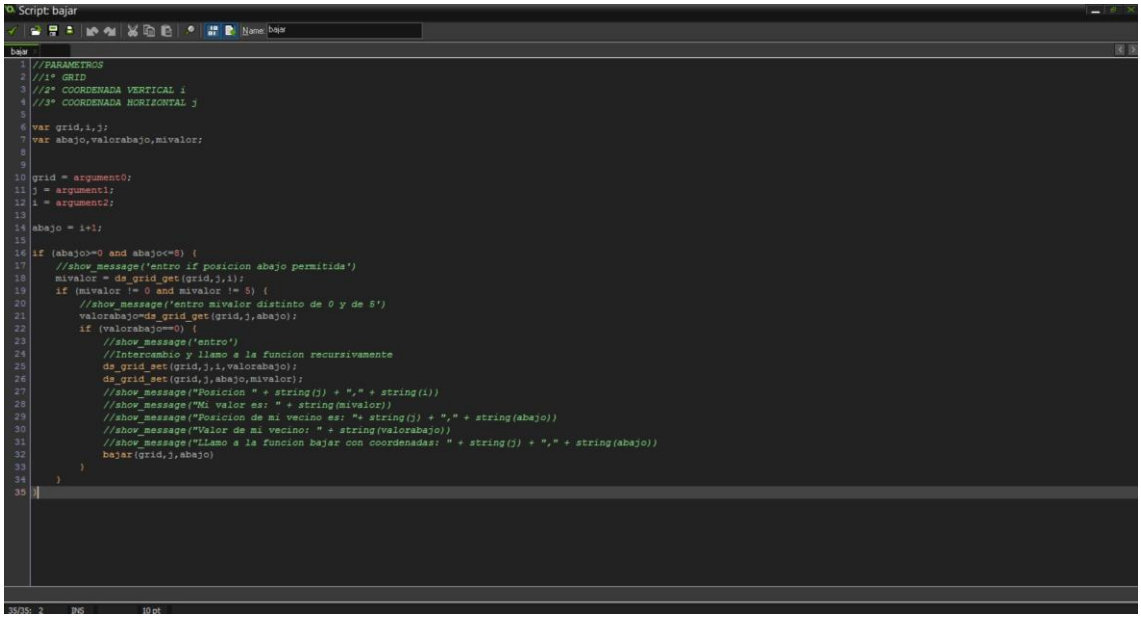


En las imágenes anteriores se han mostrado la programación de la vecindad horizontal, no mostramos la parte de la vecindad vertical debido a que son las mismas operaciones sobre el otro array de vecinos.

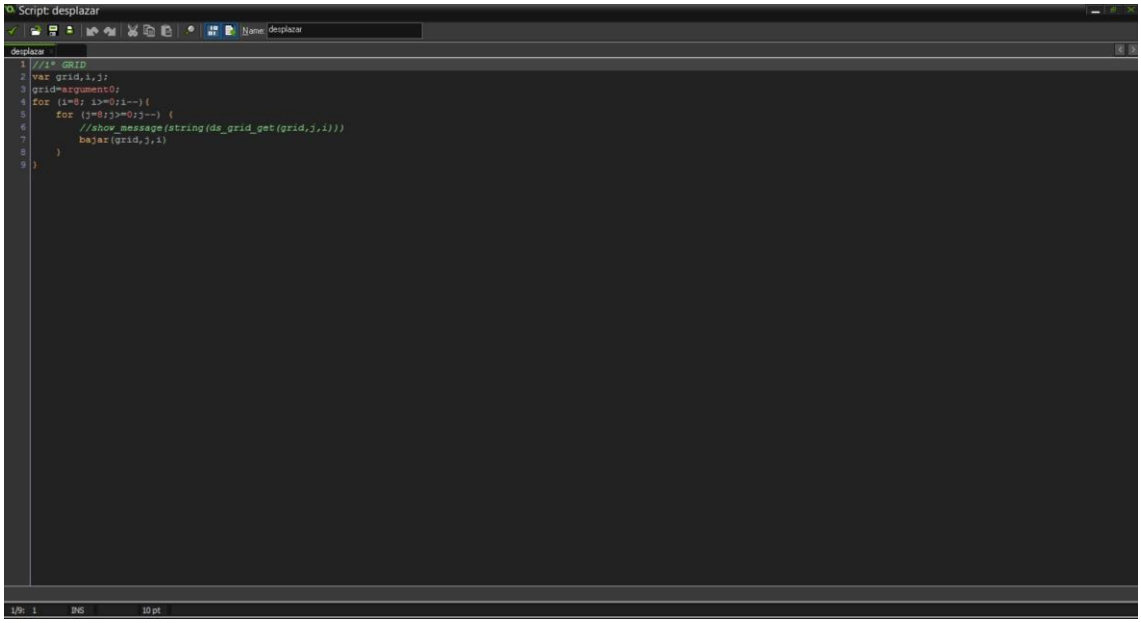
Por último se comprueba si la formación de estas líneas, genera a su vez otras líneas que también deben ser eliminadas.



bajar: esta es una función recursiva que sirve para recorrer el tablero y comprobar si se han creado huecos en el tablero a consecuencia de la formación y destrucción de líneas. Si esto sucede las piezas que quedan por arriba se desplazan verticalmente hacia abajo en el tablero hasta que no existan más huecos por debajo de ella.



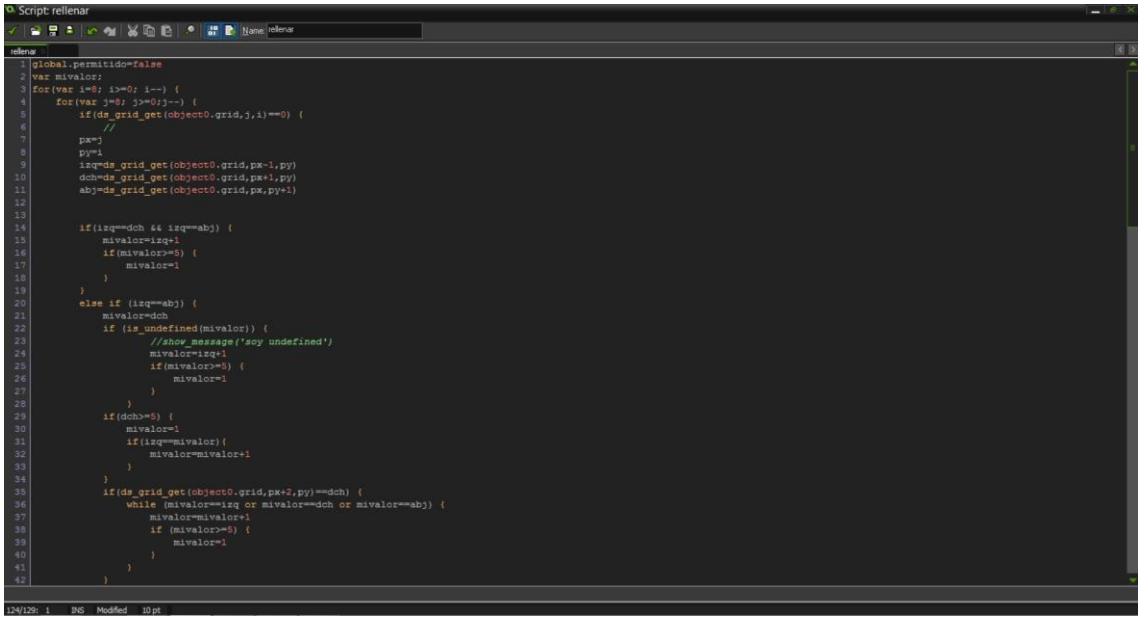
desplazar: esta función se encarga de recorrer el tablero desde abajo a la derecha hasta arriba a la izquierda llamando a la función mencionada anteriormente.

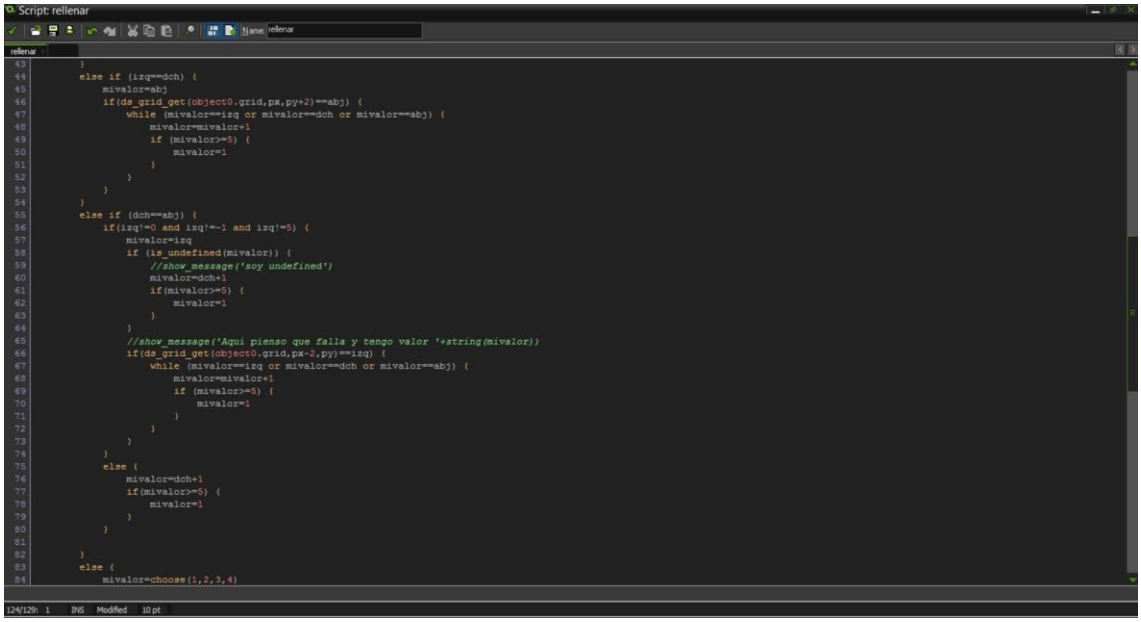


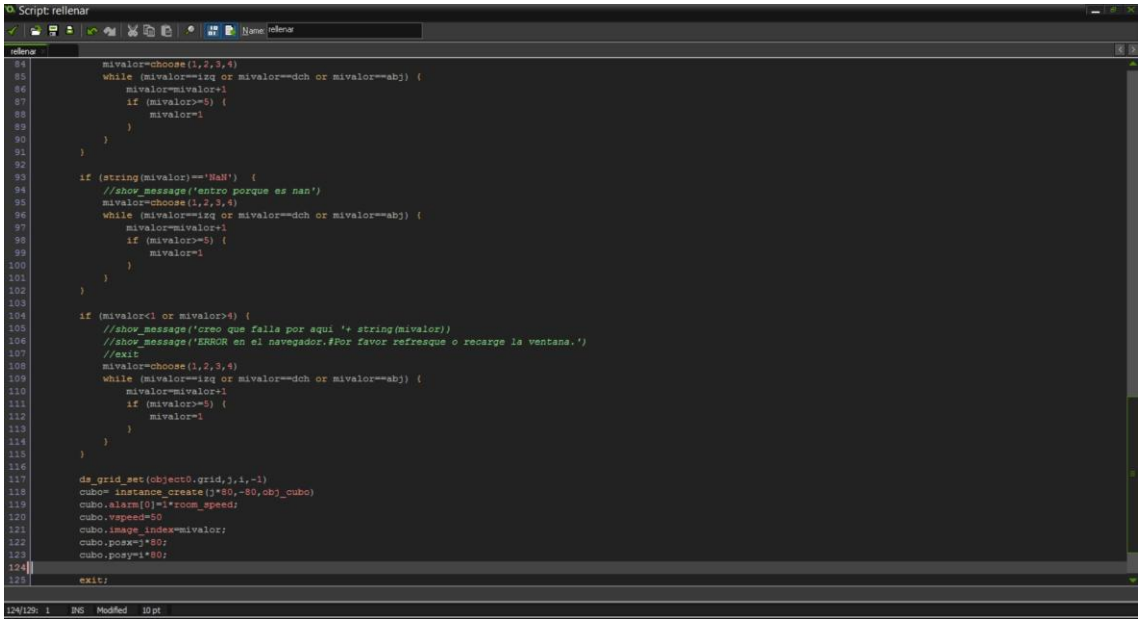
checkhorizontal: lleva a cabo la misma función que la función checklineas pero únicamente en sentido horizontal, esta se usa para la comprobación de las sublineas generadas por otras líneas.

checkvertical: es muy similar a la anterior pero realizando las operaciones en sentido vertical.

rellenar: esta función es la que se encarga de rellenar los huecos que se han formado en el tablero después de eliminar líneas y bajar el resto de piezas. Genera así nuevos elementos teniendo en cuenta los valores de sus vecinos y evitando a su vez la formación de nuevas líneas.







obtenerpuntuacion: esta última función es la suma la puntuación obtenida tras todo el proceso anterior y lo hace en función de los huecos que se han quedado en el tablero antes de rellenar los huecos finales formados.

4.4 ALMACENAMIENTO DE DATOS

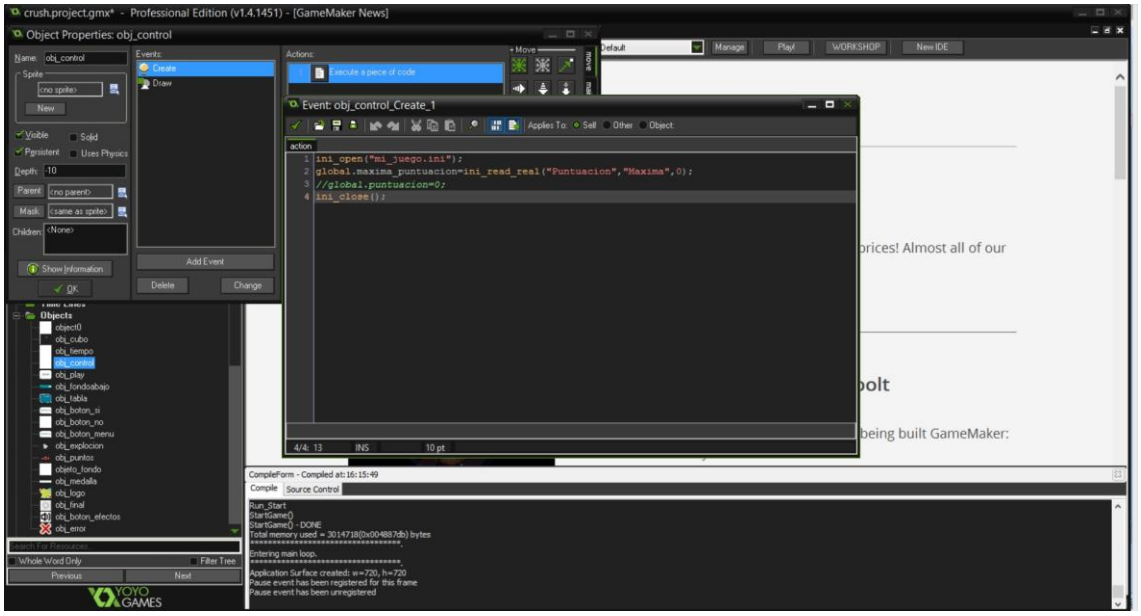
En Game Maker, se nos provee de la posibilidad de utilizar ficheros de configuración a los que llamamos ficheros ini. Estos son ficheros de texto que incluso se podrían editar con el Bloc de Notas y contiene la información de configuración del juego. No es recomendable almacenar grandes cantidades de datos pero como nuestro proyecto solo guarda una variable de puntuación máxima nos basta con él.

La estructura de estos ficheros es parecida a la de un archivo xml teniendo un tag o etiqueta principal y de ahí los nodos con su información y valores correspondientes. Es algo así como una clave/valor, en nuestro caso tiene la siguiente apariencia:

[settings]
[puntuación]
Maxima = 7650

Siempre que queramos trabajar con un fichero tenemos que abrir el fichero, escribir o leer, realizar las comprobaciones que se requieran y cerrar el fichero. Cabe destacar que tanto la lectura como la escritura permite trabajan con números reales o strings.

En nuestro proyecto tenemos el objeto obj_control que utiliza algunas de las funciones anteriormente descritas.



4.5 EXPORTACION

Como ya hemos comentado en apartados anteriores, dentro de todas las posibilidades de exportación, las que se han elegido han sido principalmente WEB, Windows DESKTOP y Android, y hemos dejado de lado y como pendiente el proceso de exportarlo para Windows Mobile e IOS. Esto es debido al alto coste monetario de los módulos de exportación y a que no se disponían tampoco de dispositivos para llevar a cabo las pruebas en estos sistemas.

5 PRUEBAS

En este apartado se contemplarán las pruebas realizadas a la aplicación para comprobar su correcto funcionamiento y asegurar que no da problemas.

Tras la programación del juego, lo primero fue confirmar la inexistencia de errores. Para ello, se presento la aplicación a un conjunto seleccionado de usuarios para confirmar la ausencia de estos.

Tras un exhaustivo análisis y estudio de la aplicación, se fue corrigiendo pequeños problemas que presentaba, en temas relacionados con compatibilidades entre sistemas, por ejemplo en la versión de Android algún objeto de imagen de fondo que se superponía a otros elementos en la pantalla.

Para la parte funcional, hemos estudiado una serie de casos de prueba para comprobar el correcto comportamiento del juego. Se estudiaron los caminos lógicos que seguía el juego según la interacción del usuario, viendo así el flujo de datos y comprobando, debugeando y corrigiendo la correcta ejecución.

Por otro lado se llevo a cabo lo que llamamos pruebas de caja negra, comprobando cada modulo, en nuestro caso los scripts, que tras unas entradas predefinidas se devolvieran unos valores esperados correctos y por tanto comportándose el juego como se estableció en un principio.

Además, como el proyecto se implemento finalmente en la página web de la asociación SARE, hemos llevado a cabo una serie de pruebas de carga, comprobando el juego desde el acceso en distintos sistemas y con distintos navegadores. Por ejemplo en el sistema operativo Android, se nos ha dado el caso que el navegador propio que incorporan ciertas versiones de este sistema y en cualquier celular que lo posea el juego no es capaz de ejecutarse. Esto es así porque ese navegador en concreto no ejecuta flash y no es capaz de manejarlo.

Cabe mencionar también que se llevaron a cabo pruebas de integración en la página web de la asociación en la versión exportada de HTML5.

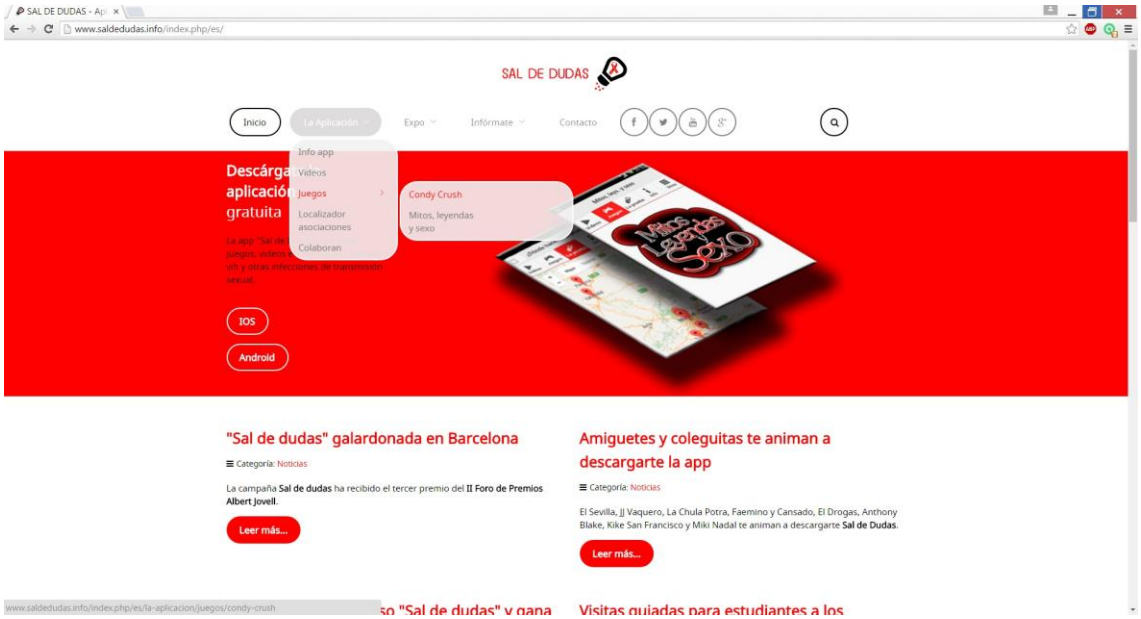
En definitiva hemos podido comprobar que, tras dejar a los usuarios enfrentarse al juego y practicar durante un tiempo prolongado el juego se ha comportado como se esperaba, se ha probado tanto con usuario de la calle, como los miembros de la asociación e incluso otros colegas informáticos, además se han corregido errores sugeridos por los mismos usuarios, se han estudiado todos los casos necesarios y caminos de ejecución distintos y se ha verificado su rendimiento.

6 MANUAL DE USUARIO

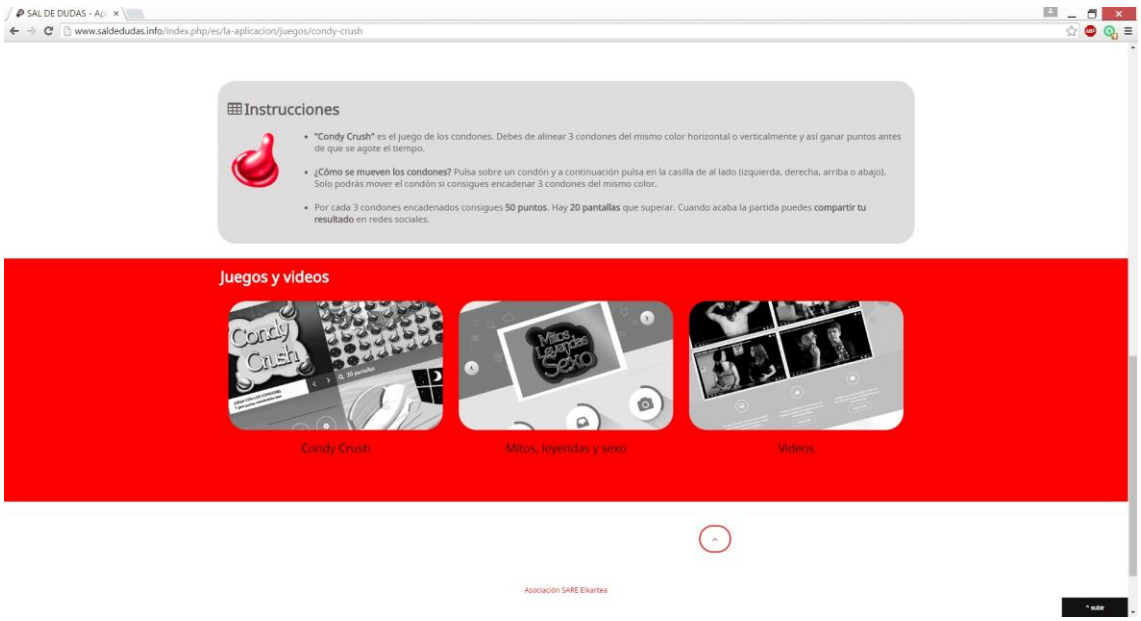
Llegados a este punto, vamos a mostrar el conjunto de pantallas de los que dispone el juego y vamos a e0xplicar como un usuario cualquiera debe interactuar con el mismo o mejor dicho como se espera que lo haga.

Lo primero que se debe hacer es ejecutar el juego o bien desde su versión de escritorio, desde la aplicación Android o desde la pagina web http://www.saldedudas.info

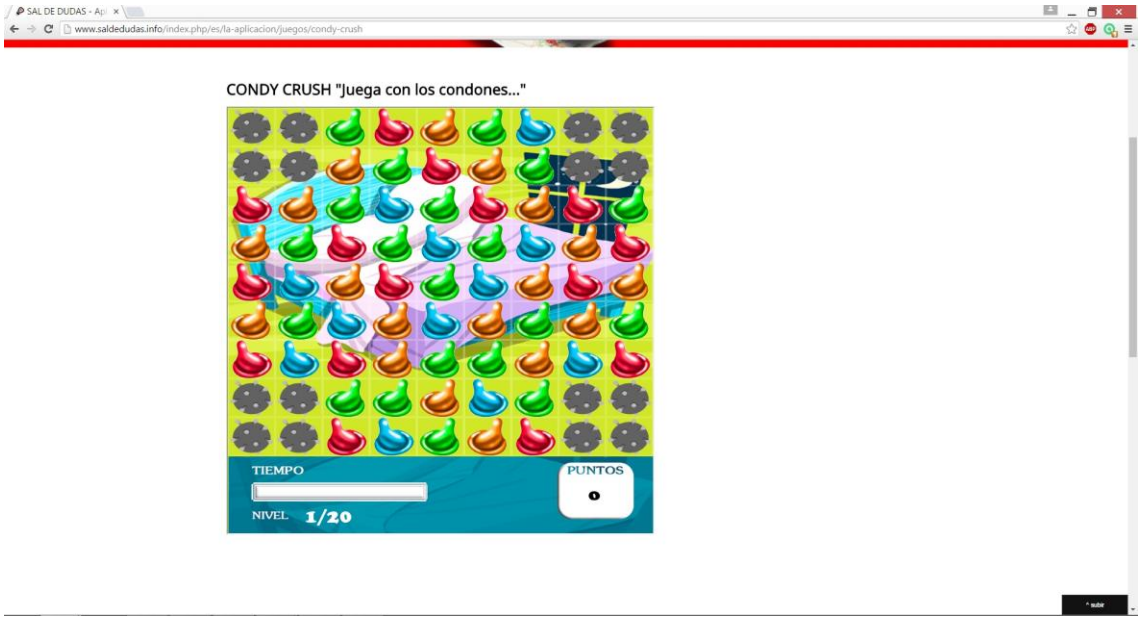
Accedemos por ejemplo a la página web:



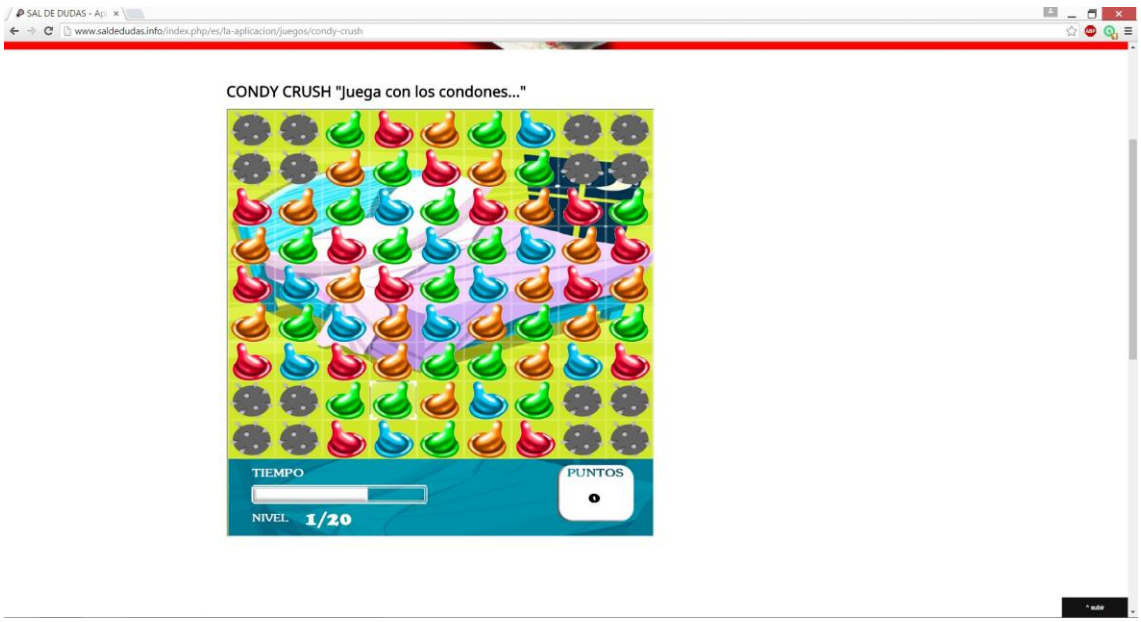
Tras acceder al enlace propio del juego, éste se carga y se muestra en el navegador, además podemos ver unas pequeñas instrucciones acerca de cómo jugar.



Viendo así la máxima puntuación obtenida, que al principio será cero, empezamos a jugar pulsando el botón JUGAR. Tras llevar a cabo esta acción nos encontramos con la siguiente pantalla:

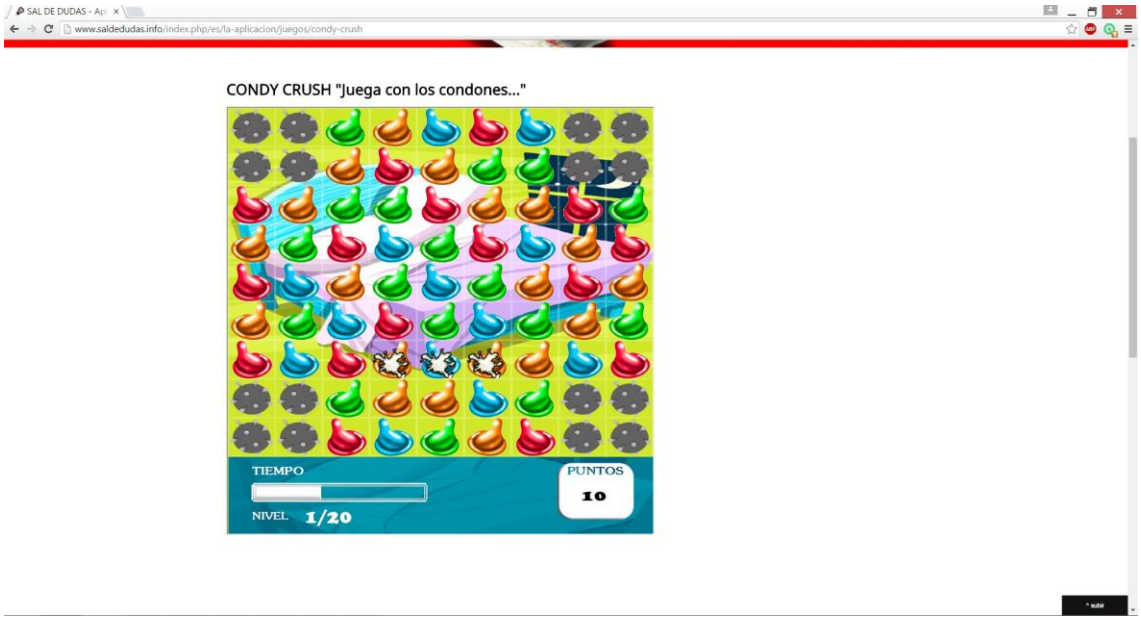


Empezamos a interactuar con el pulsando primeramente sobre uno de los preservativos. Como podemos observar también existen casillas no permitidas en las que aparece representado el virus del VIH. Al pulsar sobre nuestra pieza esta se seleccionara y seguidamente deberemos pulsar sobre otra que sea su vecina, es decir la pieza de arriba, la de abajo la de la derecha o la de la izquierda.



El juego comprueba si se puede llevar a cabo el movimiento de cambio entre las dos piezas seleccionadas. En la imagen anterior se puede ver que la segunda fina empezando por abajo en la cuarta columna hemos seleccionado un preservativo verde el cual vamos a intercambiar con el de la misma columna y fila anterior de color naranja, para formar así un línea horizontal de condones verdes.

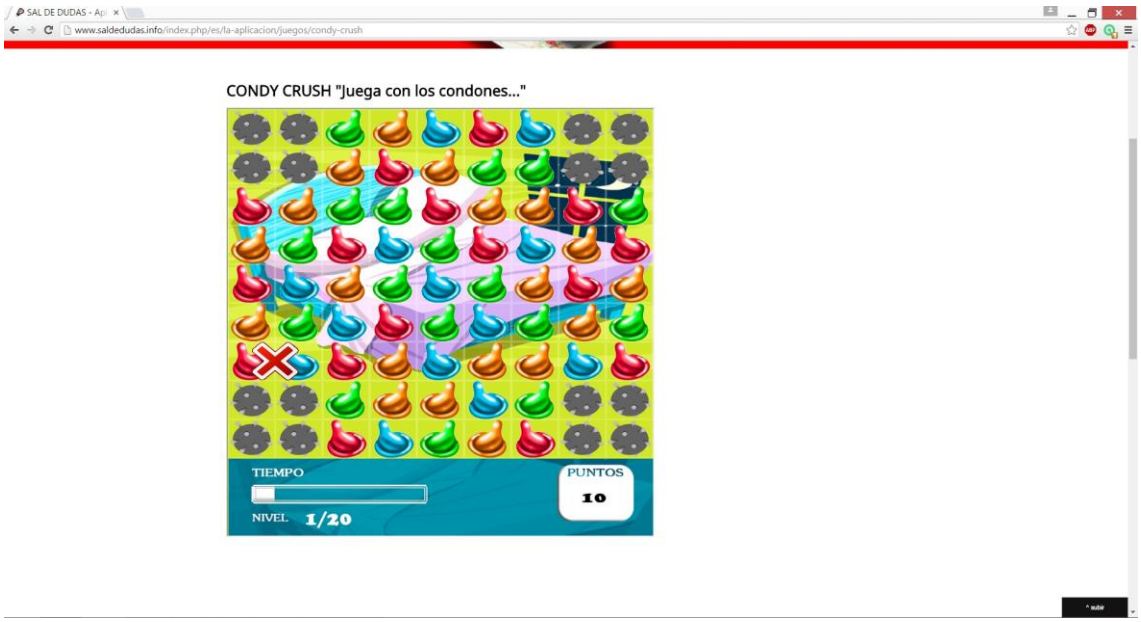
El resultado es obviamente que si que se puede y lleva a cabo una animación que evoca una eyaculación.



Mientras se produce la transición, las piezas caen rellenando los huecos que ha formado la fila y se rellena el tablero con elementos nuevos mediante mas efectos.

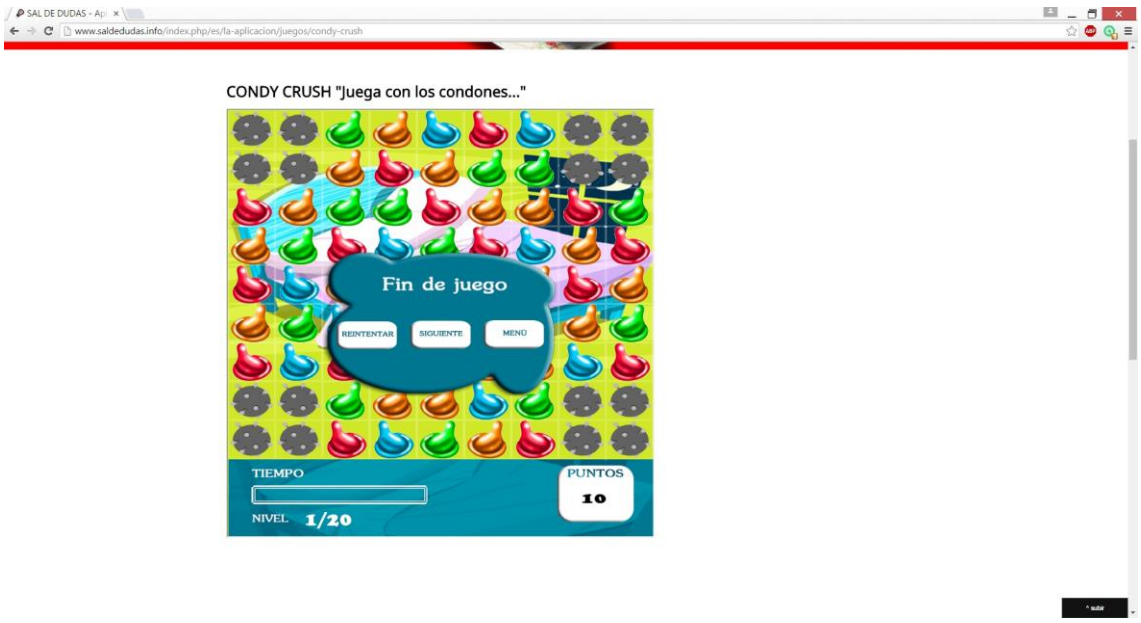
Tras realizar todo esto podemos ver a su vez como se va consumiendo el tiempo y se va vaciando la barra y también podemos ver los puntos que hemos obtenido, en este caso 10 puntos por generar la línea. Así mismo también aparece el nivel en el que estamos.

Por otro lado si realizamos un movimiento que no va a producir ningún cambio, el juego también nos lo notifica con otro efecto.

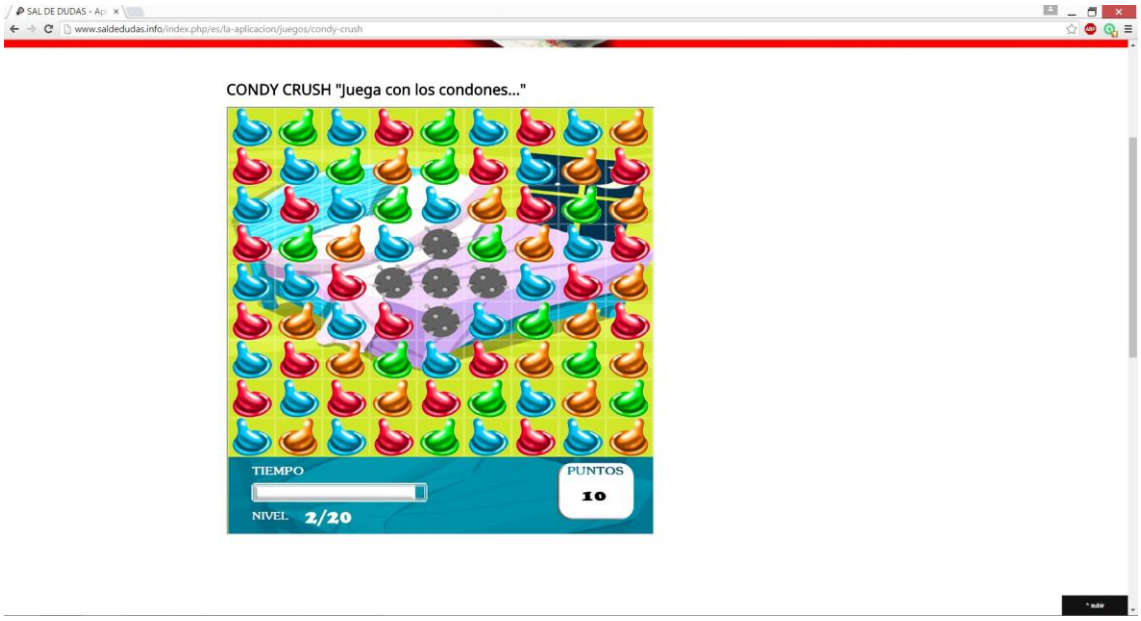


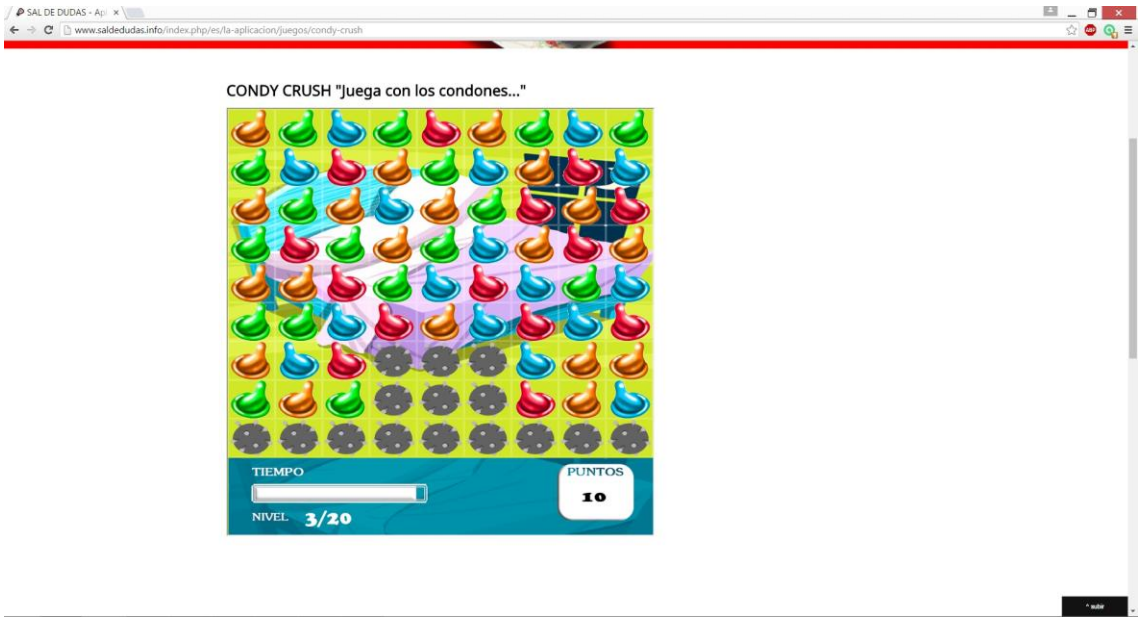
Podemos observar el aspa que indica que los elementos intervinientes entre ella no se pueden intercambiar.

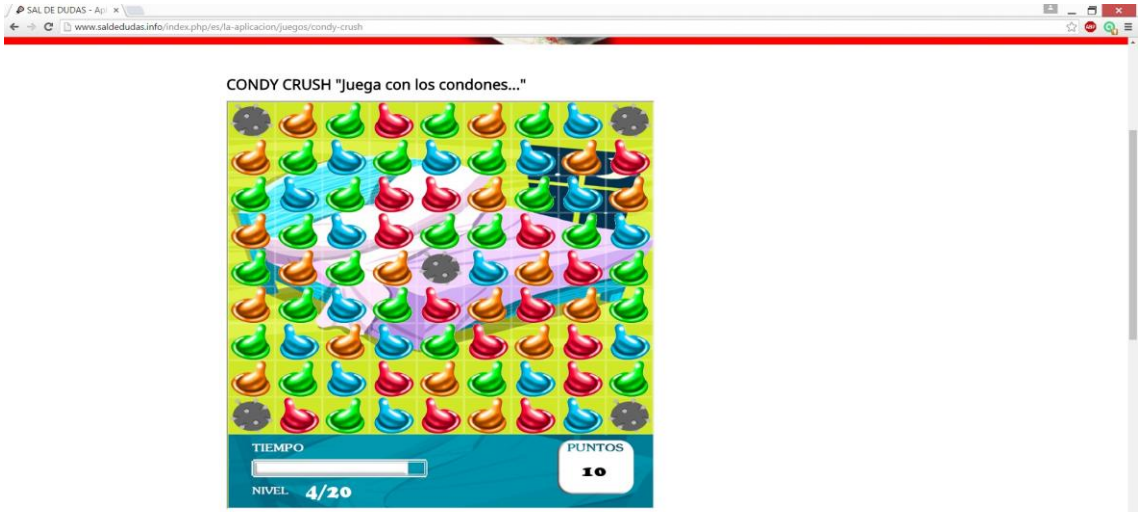
Al concluir el tiempo nos aparece un menú para reintentar el nivel y poder conseguir más puntuación, pasar al siguiente nivel o volver al menú principal si nos hemos cansado de jugar.

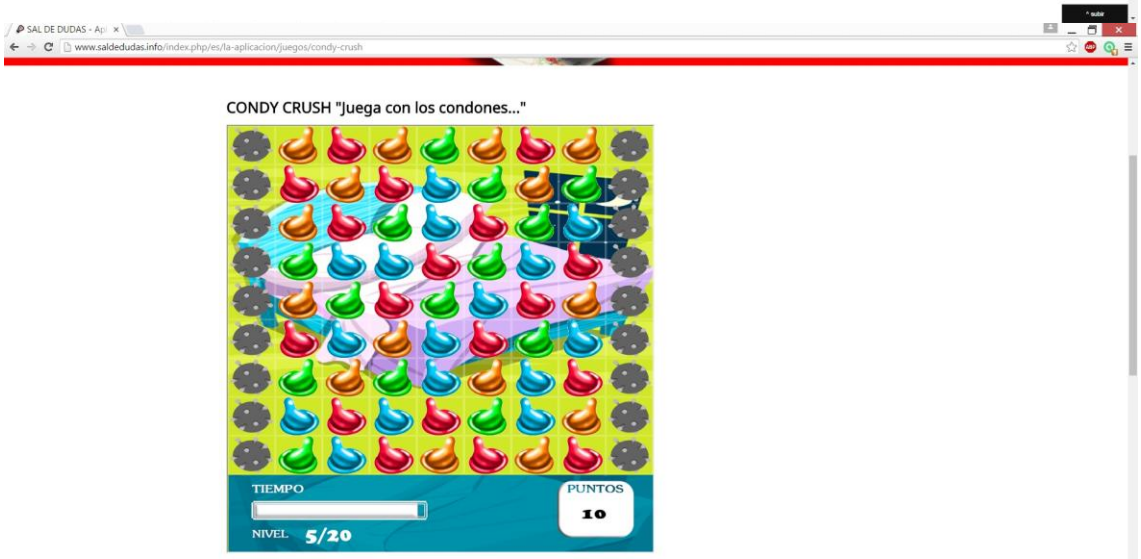


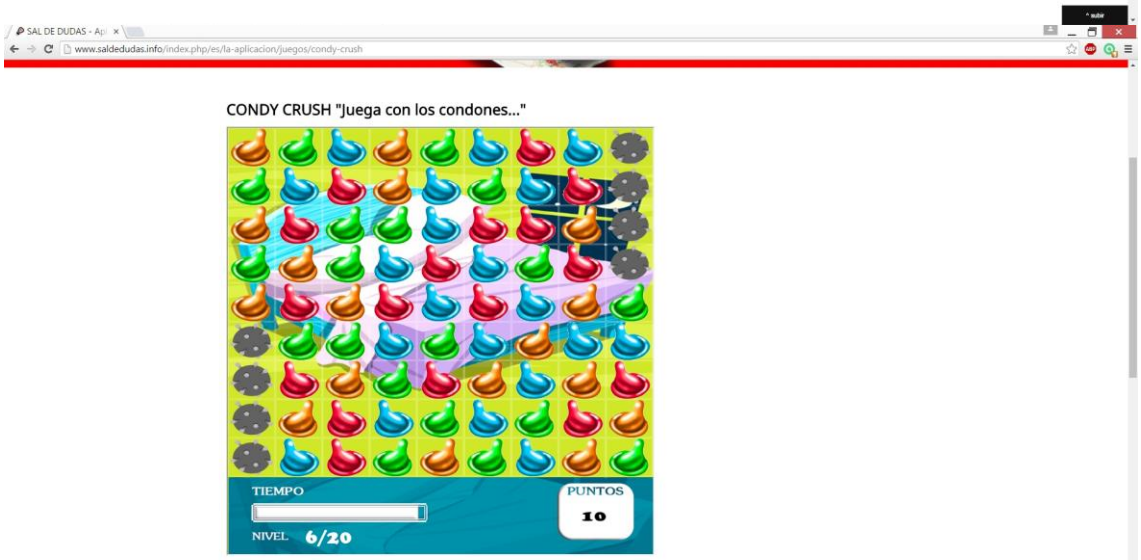
Aquí podemos ver unos cuantos niveles distintos cada vez un poco más complicados por la posición de las casillas no permitidas en el tablero y para los que se aumenta cada vez un poco el tiempo para poder pensar más.

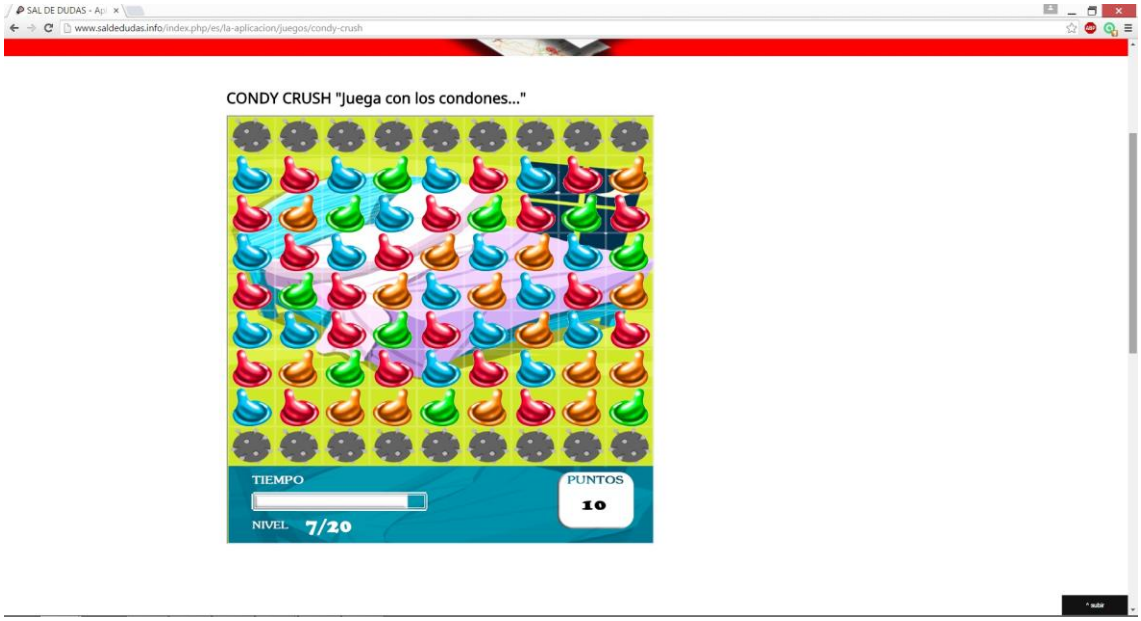






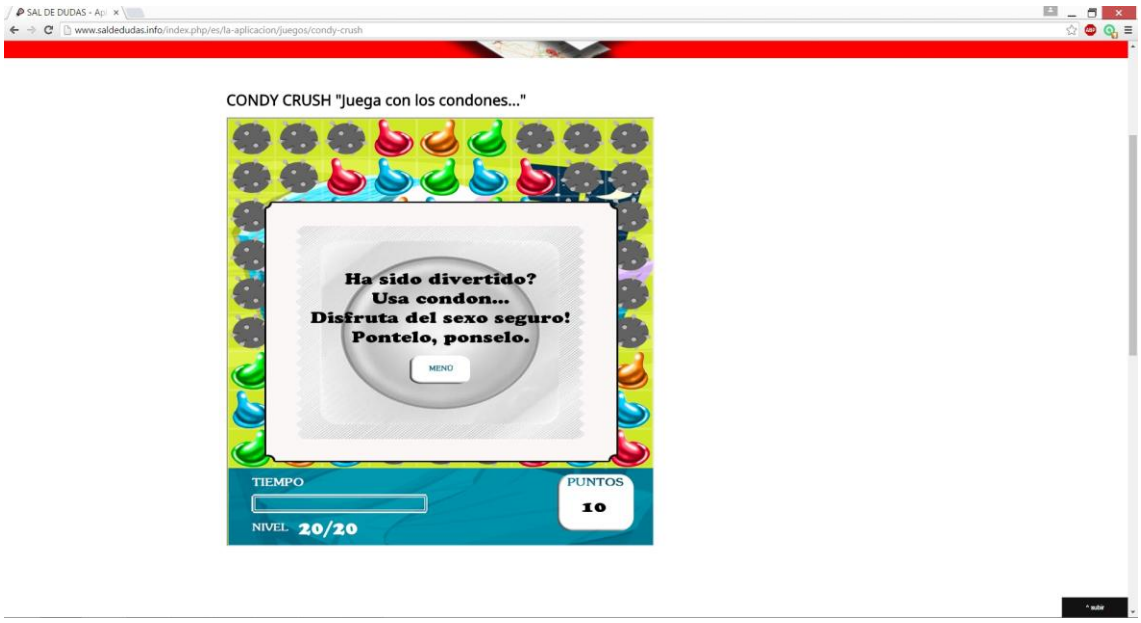






Y así sucesivamente hasta el nivel numero 20 y con distribuciones del tablero distintas.

Por último nos encontramos con el nivel final en el que tras agotarse el tiempo se nos aparece la siguiente pantalla ofreciéndonos un pequeño y claro consejo para concienciarnos de las enfermedades venéreas, su prevención e información.



7 CONCLUSIONES Y TRABAJOS FUTUROS

Con el desarrollo de este proyecto, se ha conseguido un juego que pretende evocar el uso de métodos anticonceptivos con el fin de evitar contagios por enfermedades venéreas entre ellas el contagio por el virus de VIH de una manera asociativa, que permite relacionar todos los elementos que entran en juego en esta temática.

Tras el visto bueno y la aceptación por parte de nuestros clientes podemos considerar que el juego ha tenido una buena acogida entre todos los usuarios ya que no solo la asociación ha podido jugar sino que incluimos el juego como parte de su página web y esta está disponible en versión online teniendo gran variedad de usuarios que comparten el interés del tema especificado.

Por otra parte, al hacer esta aplicación, considero que me ha ayudado a saber enfrentarme a este tipo de proyectos, no solo por haber aprendido el lenguaje GML y haber utilizado entornos de desarrollos previamente desconocidos, sino también, de cómo llevar a cabo una buena planificación de cómo, poco a poco ir estructurando, diseñando e implementando cada parte del proyecto. Además gracias a la tecnología empleada, he conseguido potenciar mi nivel de programación y descubrir cómo es la creación de juegos ya que antes nunca habíamos creado nada del estilo. Aquí entran en juego también el resto de juegos que nos sirvieron de aprendizaje.

Una de las cosas más importantes que he asimilado gracias a este proyecto es la de imaginar un producto y llevarlo a cabo, enfrentándonos a todos los problemas que van surgiendo y pensando en cómo resolverlos.

También, he aprendido que hay que adaptarse a las necesidades de los usuarios, ya que ellos son los usuarios finales, se ha de escuchar lo que piden y seguir sus consejos. En mi caso, de no haber probado el juego con ellos, seguramente, la aplicación tendría otro enfoque totalmente distinto a lo que es actualmente.

Por último, el hecho de poder participar y colaborar en un proyecto más grande que engloba el nuestro propio tiene un efecto muy positivo porque así aprendemos lo que es trabajar en equipo, aprender distintas disciplinas involucradas, se nos brinda la oportunidad de conocer gente interesante que nos pueden aportar muchas cosas e influir en nosotros.

Pensando ahora en cómo mejorar el proyecto se me ocurren las siguientes ideas:

- Ampliar el número de niveles del juego
- Crear nuevos tipos de piezas, por ejemplo jeringuillas, pastillas y demás elementos relacionados con las enfermedades de este tipo.
- Hacer el juego mas dinámico de forma que no solo se generen tableros de 9x9 piezas sino que pueda variar el numero de las mismas.

- Añadir efectos de sonido tales como una pequeña música de fondo y efectos de audio tras ciertas interacciones del usuario.
- Ofrecer nuevos métodos de interacción por ejemplo el de seleccionar las piezas deslizando el dedo por la pantalla sin levantarlo de ella, en lugar de hacer click con el ratos o con el dedo.
- Añadir más reflexiones y consejos acerca de la prevención de las enfermedades.
- Generar más efectos visuales y mejorar las interfaces graficas.
- Conseguir exportar el juego a otros sistemas que en este momento no hemos podido por el tema monetario como iOS o Windows 8 o Windows Phone.
- Crear distintos tipos de juego por ejemplo contrarreloj, salva la pieza, alcanzar una puntuación predefinida, etc.
- Utilizar sistemas de logros propios de plataformas móviles para poder competir y ver resultados online de otros jugadores.

En resumen se podría ampliar el proyecto de muchas maneras distintas, no obstante estoy muy contento con los resultados obtenidos y la aceptación que ha tenido en el entorno en el que se ha distribuido.

8 PROYECTO GLOBAL SARE

Como hemos comentado en apartados anteriores, la asociación dedicada a la prevención de enfermedades de transmisión sexual y ayuda a personas infectadas SARE, situada en el barrio de la Rotxapea en Pamplona, nos contacta con la idea de desarrollar un juego tras la mediación previa de un colega informático. Estos nos cuentan que están envueltos en un proyecto a escala comarcal mediante el cual quieren concienciar a la gente acerca de este tema, proporcionar información y promover el uso de métodos anticonceptivos que favorecen las relaciones sexuales saludables.

A su vez, este proyecto incluye distintos subproyectos como el desarrollo de su página web, la creación de la aplicación web para todas las plataformas, el desarrollo de un par de juegos entre ellos Condy Crush, y otros que no tienen tanto que ver con la informática sino en la disciplina del arte como la composición y modificación de la estética de contenedores de vidrio proporcionados por la Mancomunidad de Pamplona que luego fueron repartidos por toda la ciudad. Todo ello para conseguir promover, informar y ayudar a las personas e involucrarlas.

El viernes 2 de octubre de 2015, se celebró en el Centro Cultural de Huarte una exposición de este proyecto con todas las personas involucradas en el, ofreciendo así varias charlas informativas y la contemplación de los contenedores creados convertidos en obras de arte. Este suceso fue cubierto por el Diario de Noticias y se puede encontrar la noticia en el siguiente enlace http://www.noticiasdenavarra.com/2015/10/03/sociedad/navarra/rebrota-el-vih-en-navarra-que-llega-ya-a-1107-personas y además podemos encontrar una foto en la que aparecemos como colaborador.

Finalmente podemos encontrar toda la información y distintas aportaciones en la página web de la asociación http://www.sare-vih.org/ y a través del encale a http://www.saldedudas.info/ la web creada en este proyecto y en el que podemos encontrar nuestra aportación bajo el enlace http://www.saldedudas.info/index.php/es/la-aplicacion/juegos/condy-crush .

9 BIBLIOGRAFIA

http://docs.yoyogames.com/

http://www.comunidadgm.org/

http://www.yoyogames.com/gamemaker

https://es.wikipedia.org/wiki/GameMaker:_Studio

http://www.aprendegamemaker.com/listado-eventos/

http://www.gamedev.es/game-maker-studio-primeros-pasos/

https://www.youtube.com/playlist?list=PLwTkRS4ESjYzLPBZWE7KQzLRY856mtrI9

http://docs.yoyogames.com/source/dadiospice/002_reference/data%20structures/ds%20grids/index.html

"GameMaker Game Programming with GML" , de Matthew Delucas