
Trabajo de Final de Título en Ingeniería Informática

# "SweetRunner": juego de plataforma estilo runner para Android.

**Autor:** Cynthia Paola Ramos Martínez.

**Tutor:** Cayetano Guerra Artal.

**Tutor:** José Miguel Santos Espino.

**Grado en Ingeniería Informática**

**Curso** 2014-2015 **Convocatoria Extraordinaria**

**Universidad de Las Palmas de Gran Canaria**

# AGRADECIMIENTOS

*Al redactar esta sección es inevitable no ponernos nostálgicos, empezamos a recordar todo lo que hemos pasado para llegar hasta este punto e incontables imágenes invaden nuestra cabeza: tantas noches de estudio con compañeros o acompañados simplemente de un buen café, risas y momentos llenos de alegría, los rostros profesores que en muchas ocasiones fueron verdaderos mentores, y por supuesto, los cálidos rostros de nuestros familiares incitándonos a seguir.*

*A esas dos personas, que con paciencia y comprensión me supieron guiar, aconsejar y apoyar en la realización del presente trabajo de final de título. Mis dos estimados tutores, Cayetano Guerra Artal y José Miguel Santos Espino, sin ustedes nada de esto habría sido posible.*

*Amigos y compañeros entrañables, con los que compartimos diversos momentos de estrés por exámenes, agobio para llegar a tiempo para entregar prácticas, alegrías al aprobar, diversión, entre otros. Chinito, eres uno de los pilares fundamentales de mi vida, fuiste mi soporte cuando lo necesité, me brindaste tu cariño y amistad, por todo esto y más te doy las gracias.*

*A Mi querida familia, que son el pilar más importante de mi existencia, a ustedes que me enseñaron todo lo que se, que me mantuvieron a flote en los momentos duros, que me han dado su apoyo de forma incondicional y a quienes tanto quiero. Mami, Moché, Abi Javi, Cris y Fridel, sencillamente, gracias por ser ustedes.*

*A esa persona, que casi desde el momento en que nos conocimos ha estado alentándome a seguir, me ha otorgado su afecto y amistad. Sin ti no lo hubiese logrado.*

***A todos, muchas gracias.***

# RESUMEN

El primordial objetivo del presente Trabajo de Final de Título (TFT) es realizar un juego de plataforma estilo runner para dispositivos móviles con sistema Android, y por medio de este, realizar un análisis de las diferentes herramientas que se pueden emplear para el desarrollo juegos para móvil estilo runner.

El resultado que se obtuvo fue SweetRuner, una aplicación móvil que permite jugar en dispositivos móviles que posean sistema Android en una versión igual o superior a Honeycomb.

Las características que posee SweetRunner son: creación aleatoria de los elementos del juego; obstáculos y bonificaciones; aumento de dificultad conforme aumenta la puntuación; cálculo de puntuación y puntuación máxima; finalmente, comunicación con la red social Facebook.

El desarrollo de SweetRunner se realizó en el motor de videojuegos Unity3D, ya que es considerado la mejor opción para este tipo de juegos, por las facilidades que brinda y por su sencillez y amigabilidad.

La mayoría de las herramientas empleadas en el desarrollo de la app son de software libre, pero también se ha trabajado con software privativo. Es por esta razón que el presente trabajo se distribuye bajo la licencia GNU LGPL versión3, ya que con esta licencia se pueden enlazar a un programa no GPL, que puede ser software libre o no.

El presente trabajo tiene el potencial necesario para convertirse en una guía para desarrollar juegos para móviles estilo runner. Esta guía facilitará el proceso de desarrollo y optimización, además de ayudar a entender el funcionamiento y la potencialidad de las diferentes herramientas y componentes que se usaron.

# ABSTRACT

The primary objective of this Final Working Title (TFT) is making a runner style platform game for mobile devices with Android system, and through this, an analysis of the different tools that can be used for developing games mobile, runner style.

The result obtained was SweetRuner, a mobile application that is optimized for play on mobile devices that have a version equal to or greater than Honeycomb Android system.

SweetRunner possessing characteristics are: random creation of game elements; obstacles and bonuses; Difficulty increases with increasing score; calculation of score and maximum score; finally, communication with the social network Facebook.

The development of SweetRunner was performed in professional Unity3D game engine, as it is considered the best choice for this type of game, for the facilities provided, for its simplicity and friendliness.

Most of the tools used in the development of the app is free software, but has also worked with proprietary software. It is for this reason that the present work is distributed under the GNU LGPL Version3, because with this license can be linked to a non- GPL program, which can be free or not software.

The present work has the potential to become a guide for developing mobile games runner style. This guide will facilitate the process of development and optimization, and help to understand the functioning and potential of the different tools and components that are used.

# TABLA DE CONTENIDOS

# 1 INTRODUCCIÓN

La historia de los videojuegos tiene su origen en los finales de la década de los 40, concretamente tras la segunda guerra mundial. Las potencias vencedoras iniciaron carreras tecnológicas para construir las primeras supercomputadoras programables como ENIAC, una computadora enorme que ocupaba una superficie total de 167 $m^2$ y alcanzaba un peso de 27 toneladas.

Los primeros videojuegos estaban concebidos mayormente como experimentos y pruebas académicas por parte de científicos y físicos, no fue hasta la década de los 70 que los videojuegos empezaron a adquirir un carácter más comercial, marcando así el inicio de una nueva era. Una era en la que los videojuegos se acabarían convirtiendo en un fenómeno culturar de masas, llegando incluso a lograr el estatus de medio artístico en ciertos países.

El inicio se suscitó en el año de 1994, con el juego del teléfono pionero en la instalación de juegos, se trató de una versión de Tetris en la Hagenuk MT-2000. Sin embargo, no fue sino hasta tres años después de que los teléfonos móviles tuvieron su primer verdadero éxito. Podría haber sido muy básico, pero la aparición de la serpiente (Snake) en el Nokia 6610 fue un momento decisivo para la industria, fue el momento en el que una nueva era vio la luz. Algunas actualizaciones más adelante, 400 millones de juegos instalados y la sencillez de este juego, clave fundamental para atraer una gran audiencia de jugadores, fueron los elementos que cambiaron el la concepción que se tenía de los teléfonos móviles. De repente, la gente empezó a usar de forma regular sus teléfonos para algo más que para hacer llamadas.

Al igual que la industria del cine, los primeros años de juegos móviles eran exclusivamente en blanco y negro. No fue hasta cuatro años más tarde, cuando Nokia introdujo pantallas a color, generando con esto que lo juegos móviles dieran su próximo gran salto tecnológico

La llegada de tecnología del Protocolo de Aplicaciones Inalámbricas (WAP) permite a los dispositivos móviles conectarse a Internet, causando otro importante adelanto para los juegos. Otro hecho clave fue la propagación del entorno de

programación Java, con teléfonos como el popular Nokia 3410 que apoyan juegos Java.

La aparición de la pantalla táctil ofrece nuevas y accesibles formas de jugar. Sin embargo, no fue hasta el lanzamiento de la App Store el año siguiente que se dio un nuevo gran salto en el mundo de los juegos móviles. Los usuarios ahora podían comprar y descargar juegos produciéndose así una revolución en las ventas de juegos. Pero no sólo fue una revolución en ventas, sino que pequeños desarrolladores de pronto encontraron una plataforma para llegar a un público amplio.

En la siguiente imagen se pude apreciar algunos de los juegos más relevantes de la historia de los juegos para móviles:


*Imagen 1: Juegos móviles más relevantes a través de la historia.*

## 1.1 ESTADO ACTUAL

En la actualidad, existen diversos tipos de videojuegos para dispositivos móviles, algunos de los juegos más populares son: Angry Birds, Plantas Vs Zombies, Minecraft, Word Crack, Candy Crush Saga, Real Racing, Despicable Me, entre otros.

Los juegos que están más de moda son los que solo requieren un solo dedo (tap) para jugarlos. Así tenemos juegos de disparo, estrategia, runner, entre otros, que solo requieren la acción de un solo tap para disparar, ubicar tropas, atacar, saltar, moverse con un solo dedo.

Sin duda una incorporación relevante que han añadido a los videojuegos móviles es la comunicación con las redes sociales, consiguiendo con ello aumentar la competitividad entre jugadores y creando mayor adicción a ellos.

## 1.2 OBJETIVOS

Como se mencionó anteriormente el crecimiento de los videojuegos ha sido exponencial, lo cual incita a pensar que existen numerosas técnicas y herramientas que proporcionan diversos métodos de desarrollo para videojuegos para dispositivos móviles.

La experiencia ha demostrado que muchas metodologías empeladas no han logrado llegar a niveles aceptables en lo que respecta a videojuegos para dispositivos móviles.

Por estas razones, los objetivos planteados para el desarrollo del presente Trabajo de final de Titulo son:

- Desarrollar una app plataforma estilo runner, para dispositivos móviles basados en Android.
- Analizar tanto dificultades como facilidades que nos brinda Unity3D para el desarrollo de juegos para dispositivos móviles. En concreto nos centraremos en juegos plataforma estilo runner.
- Estudiar la compatibilidad de Unity3D con diferentes entornos de programación.

- Determinar la mejor manera de desarrollo, tanto gráfico como codificado, para encontrar la optimización del rendimiento de la aplicación.
- Establecer diferencias entre el desarrollo de un juego para PC y para dispositivos móviles.

# 2 COMPETENCIAS

Durante el desarrollo del presente trabajo de final de título se cubrieron satisfactoriamente una serie de competencias, que a continuación se especificarán detalladamente.

## 2.1 CII08

*Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.*

La satisfacción de esta competencia salta a la vista en los capítulos Metodología y Plan de Trabajo y Desarrollo. Donde se muestra las decisiones de análisis y diseño tomadas para la realización de este TFT.

## 2.2 CII09

*Capacidad de conocer, comprender y evaluar la estructura y arquitectura de los computadores, así como los componentes básicos que los conforman.*

Esta competencia se cumple en el capítulo de Desarrollo, en el apartado de diseño e implementación.

## 2.3 CII18

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

El cumplimiento de esta competencia se ve reflejado en el apartado Normativa y Legislación.

## 2.4 ISO2

*Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las*

*limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.*

Esta competencia se cumple en el capítulo de Desarrollo. A lo largo de este capítulo se llegan a consensos entre los requisitos de usuario y el rendimiento que se espera de SweetRunner.

## 2.5 ISO5

*Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse.*

Esta capacidad se satisface en el capítulo de Desarrollo en el apartado de Diseño e Implementación, al realizar pruebas y tomar decisiones para solventar los problemas que se presentaban o los que se pudiesen presentar.

# 3 PLIEGO DE CONDICIONES

El pliego de condiciones especifica las condiciones generales, técnicas, económicas, materiales y los equipos sujetos al desarrollo del presente proyecto, con el objeto de establecer cómo y con qué se debe llevar la ejecución del mismo.

## 3.1 PLIEGO DE CONDICIONES GENERALES

Con el presente trabajo se pretende desarrollar un juego de plataforma estilo runner para Android. A continuación se describen sus características generales:

- Sistema aleatorio, creación aleatoria del nivel o fase a jugar, entorno y enemigos.
- Almacenamiento de puntuación máxima alcanzada durante las partidas jugadas.
- Implementación de efectos con los sistemas de partículas que proporciona Unity3d.
- Capacidad para la transmisión, en tiempo real, de datos a las redes sociales, en específico Facebook.

Si el uso de la aplicación desarrollada conlleva el tratamiento de datos personales se ha de aplicar la legislación vigente en España. En la actualidad la ley Orgánica de Protección de Datos de Carácter Personas (LOPD), es la encargada de este tipo de regulaciones. Así mismo, es necesario cumplir con ley de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSI). Y no se debe olvidar la Normativa de Cookies.

## 3.2 PLIEGO DE ESPECIFICACIONES TÉCNICAS

Dentro del pliego de especificaciones técnicas podemos encontrar los siguientes dos apartados:

### 3.2.1 ESPECIFICACIONES DE MATERIALES Y EQUIPOS

Es preciso hacer una diferenciación entre los materiales software y hardware que han sido empleados para el proyecto.

#### 3.2.1.1 Hardware

- Ordenador Portátil Personal, con los ordenadores actuales no es requerido que el equipo posea características especiales.
- Dispositivo móvil Smartphone para testeo de la aplicación. Es necesario que posea sistema Android igual o superior a la versión 3.0 (Honeycomb).
- Dispositivo móvil Tablet para testeo de la aplicación. Al igual que el dispositivo móvil, debe poseer sistema Android igual o superior a la versión Honeycomb.

#### 3.2.1.2 Software

- Unity3d Pro, versión 4.5.1.
- MonoDevelop, entorno de programación propio de Unity3D Pro.
- Eclipse, entorno de desarrollo multiplataforma.
- SDK de Facebook, se empleará la versión para Android.
- Inkscape, para la creación de los elementos gráficos, como vectores, empleados en la app.
- GIMP, para añadir detalles y efectos a los elementos gráficos.
- Adacity, usado en la grabación y edición de audio.

### 3.2.2 ESPECIFICACIONES DE EJECUCIÓN

El proceso que se ha empleado para la elaboración del trabajo está estructurado en diferentes etapas de un desarrollo incremental:

- Familiarización con el software Unity3d a través de la realización de distintas pruebas con el fin de evaluar el entorno, los distintos elementos que posee, potencialidad y facilidades que brinda.
- Familiarización con la herramienta Inkscape para la creación de elementos gráficos vectorizados.
- Definición de las historias de usuario que se emplearán para el desarrollo.

- Desarrollo de un prototipo inicial capaz de mostrar un terreno y mover la cámara.
- Mejora del prototipo con la incorporación de un objeto capaz avanzar de forma autónoma por el terreno y de saltar a voluntad del usuario.
- Mejora del prototipo agregando enemigos y bonificaciones.
- Mejora del prototipo consistente en creación automatizada y aleatoria de entorno, terreno, enemigos y bonificaciones.
- Mejora del prototipo, añadiendo los elementos gráficos Background, GUI y efectos (partículas).
- Mejora del prototipo en base a la funcionalidad de comunicación con redes sociales y generación de puntuación (score).

## 3.3 PLIEGO DE CLAÚSULAS ADMINISTRATIVAS

El presupuesto necesario para desarrollo de SweetRuner es de euros. A continuación veremos con mayor detalle cómo se obtuvo esta cifra:

| **Recurso** | **Costo** |
|---|---|
| Programador (230 hora) | 40 euros / hora |
| Diseñador (15 horas) | 30 euros/hora |
| Ordenador Portátil Lenovo B590 | 346 euros |
| SmartPhone Primux TECH alpha 3x | 89 euros |
| Tablet Samsung GT-P5110 | 124, 99 euros |
| Licencia Unity3D Pro (3 meses) | 75 euros / mes |
| **Total** | **10434,99 euros (IGIC incl.)** |

*Tabla 1: Relación de costes del proyecto.*

## 3.4 LICENCIAS

El presente trabajo se distribuye bajo la Licencia Pública General Reducida (GNU LGPL) versión 3, que en realidad es un conjunto de permisos añadidos a la GNU GPL (Licencia Pública General Reducida). Dado que Unity3d no es software

libre con esta licencia se pueden enlazar a un programa no-GPL, que puede ser software libre o no. Al igual que otras licencias de este tipo, garantiza la libertad de compartir y modificar el software que está bajo ella, asegurando que el software es libre para todos sus usuarios.

# 4 APORTACIONES

Desde que salieron los juegos para Android, hace aproximadamente seis años, hasta nuestros días, su incremento ha sido de forma exponencial. Este crecimiento nos insta a pensar, que existen numerosas técnicas y herramientas usadas por desarrolladores que nos proporcionan un sin fin de métodos para el desarrollo de los citados juegos.

Sin embargo, muchas de las metodologías empleadas no han logrado llegar a un nivel óptimo de funcionalidad, y en muchos casos ni a niveles aceptables. Esto es debido a que al no mantenerse un estándar que pueda homologar, de cierta forma, estos métodos, se generan diversos tropiezos e inconvenientes sobre los equipos donde son armados. Es evidente que la mayor parte de dificultades se suscita en personas que empiezan o tienen poca experiencia.

Es por eso que a la hora de plantearnos el presente TFT, ha sido pensando en generar un análisis de las herramientas de desarrollo, realizando comparativas entre ellas, estableciendo las metodologías de trabajo más útiles. Todo esto se basará la experiencia propia, en este caso en particular, del desarrollo de un juego plataforma estilo runner.

A continuación, se listarán las diferentes aportaciones que ofrece ***SweetRunner:***

- Razones por las que se ha escogido Unity3D como principal herramienta de desarrollo.
- Análisis de facilidades y dificultades que presenta Unity3D a la hora de desarrollar.
- Estudio de compatibilidad que tiene Unity3D con diferentes entornos de programación, lenguajes de programación y herramientas de creación gráfica.
- Determinación de la mejor manera de desarrollo, tomando en cuenta factores gráficos, limitaciones de arquitectura de los dispositivos, y

codificación, para finalmente poder optimizar el rendimiento de la aplicación.

Como hemos podido observar, las aportaciones que ofrece el presente trabajo consisten en una serie de estudios y análisis basados en la experiencia, con el fin de facilitar la vida a quienes deseen desarrollar un juego de este tipo.

# 5 NORMATIVA Y LEGISLACIÓN

Este capítulo trata de la legislación vigente que afecta al presente trabajo de final de título.

## 5.1 LEY ORGÁNICA DE PROTECCIÓN DE DATOS (LOPD)

La ley Orgánica de Protección de Datos estipula en el artículo 1 "La presente Ley Orgánica tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar".

Así mismo, el artículo 6 "Consentimiento del afectado" sección 1 dice: "El tratamiento de los datos de carácter personal requerirá el consentimiento inequívoco del afectado, salvo que la ley disponga otra cosa".

Finalmente, en el artículo 11 "Comunicación de Datos" sección 4 se establece: El consentimiento para la comunicación de los datos de carácter personal tiene también un carácter de revocable.

En nuestro caso, ***SweetRunner*** no recogerá datos de ningún tipo, sin embargo va a permitir la compartición de información a través de las redes sociales, previo consentimiento del usuario. No obstante, en mejoras futuras cabe la posibilidad de pedir al usuario algunos daros, obviamente contando con su consentimiento.

## 5.2 LEY DE SERVICIOS DE LA SOCIEDAD DE LA INFORMACIÓN DE COMERCIO ELECTRÓNICO (LSSI)

El artículo 1 de la Ley de Servicios de la Sociedad de la Información de Comercio Electrónico estipula: "Es objeto de la presente Ley la regulación del régimen jurídico de los servicios de la sociedad de la información y de la contratación por vía electrónica, en lo referente a las obligaciones de los prestadores de servicios incluidos los que actúan como intermediarios en la transmisión de contenidos por las

redes de telecomunicaciones, las comunicaciones comerciales por vía electrónica, la información previa y posterior a la celebración de contratos electrónicos, las condiciones relativas a su validez y eficacia y el régimen sancionador aplicable a los prestadores de servicios de la sociedad de la información."

El artículo 22 "Derechos de los destinatarios de servicio" sección 2 señala: "Los prestadores de servicios podrán utilizar dispositivos de almacenamiento y recuperación de datos en equipos terminales de los destinatarios, a condición de que los mismos hayan dado su consentimiento después de que se les haya facilitado información clara y completa sobre su utilización, en particular, sobre los fines del tratamiento de los datos, con arreglo a lo dispuesto en la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal.

Cuando sea técnicamente posible y eficaz, el consentimiento del destinatario para aceptar el tratamiento de los datos podrá facilitarse mediante el uso de los parámetros adecuados del navegador o de otras aplicaciones.

Lo anterior no impedirá el posible almacenamiento o acceso de índole técnica al solo fin de efectuar la transmisión de una comunicación por una red de comunicaciones electrónicas o, en la medida que resulte estrictamente necesario, para la prestación de un servicio de la sociedad de la información expresamente solicitado por el destinatario. "

## 5.3 NORMATIVA DE COOKIES

La Agencia Española de Protección de datos publica la Normativa de Cookies con la finalidad hacer cumplir el artículo 22 sección 2 de la LSSI. Así, la información debe facilitarse según lo establecido en el artículo 22.2.

La información sobre cookies facilitada en el momento de solicitar el consentimiento del usuario, debe ser lo suficientemente completa para permitir que el usuario pueda entender la finalidad para las que se instalaron y conocer los usos que se les darán.

En caso de que el usuario preste su consentimiento para el uso de cookies, la información sobre cómo revocar este consentimiento y la posibilidad de eliminar las cookies deberá estar a su disposición de forma accesible y permanente.

SweetRunner no usará cookies propias pero si de terceros, Al subir la aplicación a la Google Play Store se debe asumir el uso de cookies de terceros.

## 5.4 LICENCIAS

En este apartado se va a detallar las licencias que están implicadas en el desarrollo del proyecto.

### 5.4.1 LICENCIA PÚBLICA GENERAL (GNU LPG)

La licencia Pública General de GNU o General Public License (GNU GPL en inglés) fue creada por Richard Stallman, fundador de la Software Foundation (FSF) para el proyecto GNU. Se encarga de garantizar a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software. Con el objetivo declarar que el software que esté bajo esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Están bajo esta licencia GIMP, Inkscape, Audacity y Android.

### 5.4.2 LICENCIA PÚBLICA GENERAL REDUCIDA (GNU LGPL)

Al igual que la GNU GPL, la Licencia Pública General Reducida de GNU o Lesser General Public License (en inglés), fue creada por la Free Software Foundation. Esta licencia pretende garantizar la libertad de compartir y modificar el software que está bajo ella, asegurando que el software es libre para todos sus usuarios.

Ahora bien, la principal diferencia con la GPL es que la LGPL puede enlazarse a (en el caso de una biblioteca, 'ser utilizada por') un programa no-GPL, que puede ser software libre o software no libre. Así pues la GNU LGPL versión 3 se presenta como un conjunto de permisos añadidos a la GNU GPL.

Bajo esta licencia se encuentran el anteriormente citado Inkscape y el presente trabajo de final de título.

### 5.4.3 APACHE 2.0

La licencia Apache (en inglés, Apache License o Apache Software License para versiones anteriores a 2.0) es una licencia de software libre creada por la Apache Software Foundation (ASF). Esta licencia requiere la conservación del aviso de copyright y el disclaimer, pero no es una licencia copyleft, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.

Apache concede al usuario del software la libertad de usarlo para cualquier propósito, distribuirlo, modificarlo, y distribuir versiones modificadas de ese software.

A diferencia de las licencias anteriores, Apache no exige que las obras derivadas del software se distribuyan usando la misma licencia, ni siquiera que se tengan que distribuir como software libre. Así Apache sólo exige que se mantenga una noticia que informe a los receptores que en la distribución se ha usado código con la Licencia Apache.

El software que se encuentra bajo esta licencia es Android.

### 5.4.4 CREATIVE COMMONS (CC)

Creative Commons es una organización sin ánimos de lucro que permite usar y compartir tanto la creatividad como el conocimiento por medio de varios instrumentos jurídicos que son de carácter gratuito. CC, se encuentra al frente del movimiento copyleft, que pretende construir una alternativa al copyright “todos los derechos reservados”.

Los citados instrumentos jurídicos no son más que un conjunto de licencias de derechos de autor, licencias Creative Commons, que permiten al autor de una obra otorgar permiso al público en general de compartir y usar su trabajo creativo bajo los términos y condiciones que él escoja.

Los términos de cada licencia dependen de cuatro condiciones: atribución; no comercial; no derivadas; compartir igual. Con estas cuatro condiciones se obtienen

las siguientes licencias: Atribución; Atribución-CompartirIgual; Atribución-NoDerivadas; Atribución-NoComercial; Atribución-NoComercial-CompartirIgual; Atribución-NoComercial-NoDerivadas.

Bajo esta licencia se encuentran los sonidos que se descargaron para la incorporación en SweetRunner.

# 6 REQUISITOS

Los requisitos empleados para el desarrollo de ***SweetRuner***, se describirán detalladamente en los apartados Hardware y Software respectivamente:

## 6.1 HARDWARE

- Ordenador portátil personal. Como se mencionó en el capítulo Pliego de Condiciones, se ha empleado un Lenovo B590 y no es necesario que el equipo tenga características especiales, basta que posea la capacidad de ejecutar Inkscape y Unity3D.
- Dispositivo móvil Smartphone para testear la aplicación. Se requiere que posea sistema Android a partir de la versión 3.0 (Honeycomb). El Smartphone que se ha utilizado es un Primux Tech alpha 3x.
- Dispositivo móvil Tablet para testeo de la aplicación. Es necesario que esté basado en sistema Android igual o superior a la versión Honeycomb. El Tablet empleado en este TFT fue un Samsung GT-P5110.

## 6.2 SOFTWARE

- Unity3D Pro. Versión 4.5.1.
- Monodevelop. Se trata de un entorno de programación propio de Unity3D que soporta varios lenguajes de programación: C#, JavaScript y Boo.
- Eclipse.
- SDK de Facebook para Android.
- Inkscape. Como se ha mencionado antes, se ha empleado en la creación de elementos gráficos vectoriales, la versión empleada fue la 0.48.5 que es una versión estable.
- GIMP. También ha sido empleado en la realización los elementos gráficos, la versión que se usó es la 2.8.14.

- Audacity. Usado en la grabación y edición de audio, la versión empleada fue la 2.0.6.
- Microsoft Word 2013. Para la redacción del presente documento.

# 7 METODOLOGÍA Y PLAN DE TRABAJO

## 7.1 METODOLOGÍA

Para el desarrollo del presente trabajo no se ha seguido una metodología exacta, se ha optado por una metodología que recoge los principales aspectos del proceso unificado y de metodologías de desarrollo ágil, Scrum y el manifiesto ágil han sido una gran fuente de inspiración.

El siguiente listado muestra las características que posee la metodología que se desarrolló:

- **Iterativo e Incremental:** consiste en la descomposición del proyecto en mini-proyectos, cada mini-proyecto es una iteración. Se incorporó también las fases propias del Proceso de desarrollo unificado: Inicio, Elaboración, Construcción y Transición.
- **Dirigido por historias de usuario:** En cada iteración se satisfacen un conjunto de historias de usuario.
- **Flexibilidad ante el cambio:** Respuesta ante el cambio sobre seguir un plan. Es decir, se trata de proporcionar una gestión de cambios ágil.
- **Enfocado en el riesgo:** Se procura identificar potenciales riesgos críticos en etapas tempranas, pese a tener una gestión ágil al cambio.
- **Desarrollo orientado a pruebas:** consiste en implementar el código necesario para superar las pruebas.
- **Refactorización para mejorar la productividad:** consiste en refactorizar una vez superadas las pruebas, con el afán de mantener el código.

El proceso optado sigue los siguientes pasos: lo primero que realiza es un diseño testeable en base a las historias de usuario principales. Una vez comprobado el correcto funcionamiento del prototipo creado se refactoriza el código y finalmente se pasa a añadir funcionalidad en base a las restantes historias de usuario. Estos pasos o etapas se vuelven a repetir hasta terminar el proyecto

A continuación se muestra de manera gráfica, ver imagen 2, las etapas descritas anteriormente:

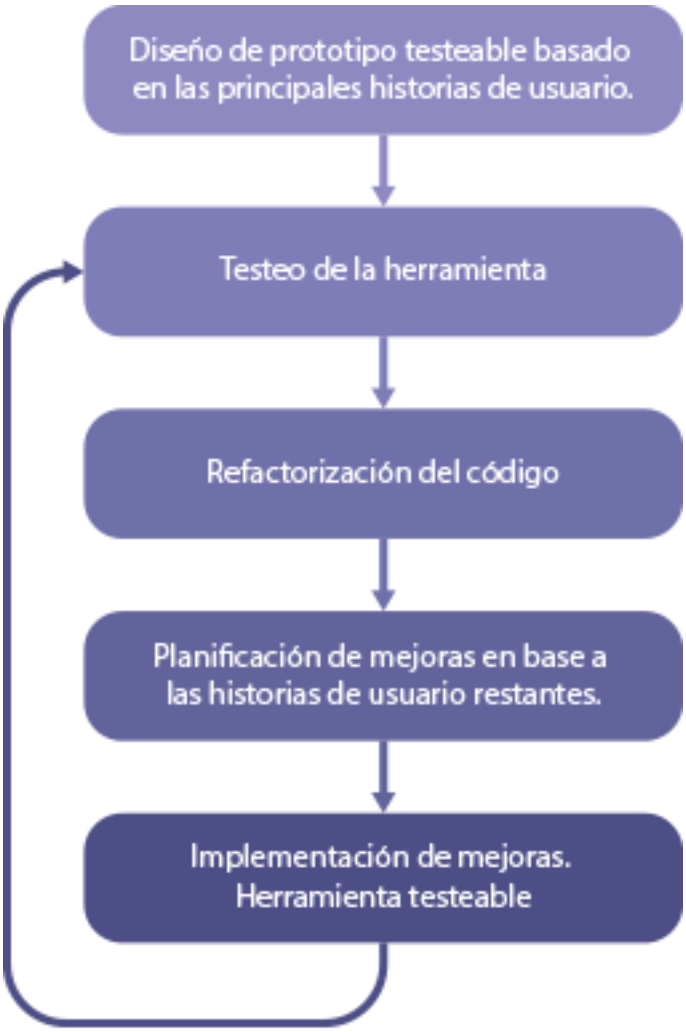


*Imagen 2: Metodología de Desarrollo.*

## 7.2 PLAN DE TRABAJO

El plan de trabajo seguido durante el desarrollo del presente trabajo, se divide en 3 etapas:

- **Análisis.** En esta etapa se escogieron las herramientas con las que se desarrolló el TFT. Se hizo un estudio de la problemática y se definieron las historias de usuario.
- **Desarrollo.** Lo primero que se realizó es un prototipo en base a las historias de usuario más importantes, posteriormente se siguieron añadiendo funcionalidades y finalmente, se añadió una interfaz de usuario. Todo esto con las pruebas y refactorización adecuadas, como se mencionó en la metodología.
- **Documentación y Defensa.** Para finalizar se realizó el presente documento, en base a la documentación generada en las etapas anteriores. Se preparó material y se preparó la defensa.

En la siguiente tabla se puede observar la estimación de horas trabajadas, las fases y las actividades principales de cada fase:

| **Fases** | **Duración total** | **Actividades principales** |
|---|---|---|
| **Análisis** | 90 horas | • Determinación de las herramientas que se emplearán.<br>• Familiarización con las diferentes herramientas.<br>• Estudio y elaboración de historias de usuario.<br>• Determinación de las Historias de Usuario para el primer prototipo. |
| **Desarrollo** | 150 horas | • Implementación del primer prototipo.<br>• Mejoras a los prototipos y testeo.<br>• Elaboración e integración de elementos gráficos.<br>• Elaboración e integración de elementos de audio. |
| **Documentación y Defensa** | 60 horas | • Elaboración de documentación.<br>• Elaboración de material para la defensa.<br>• Preparación de la defensa. |

*Tabla 2: Estimación temporal por fases del TFT*

# 8 CONCEPTOS Y FUNCIONES PROPIAS

Este capítulo ha sido creado con la finalidad de definir algunos conceptos y funciones propias de Unity3d, necesarios para la comprensión del presente Trabajo de Final de Título.

## 8.1 CONCEPTOS

### 8.1.1 ISLAS Y SPRITES

Para lo concerniente al desarrollo de este trabajo, tomaremos a las islas y sprites como lo mismo. De esta manera, diremos que son imágenes usadas para representar un ente gráficamente, o parte de él, y poder posicionarlo en el lugar que se desee de una escena.

### 8.1.2 ATLAS

Un atlas es un conjunto no ordenado de islas o sprites.

### 8.1.3 SPRITE TEXTURE

Un Sprite Texture, a diferencia de un Atlas es un conjunto ordenado de islas o sprites.

### 8.1.4 GAMEOBJECT

Los GameObject son los objetos más importantes de Unity, dado que cada objeto del juego es un GameObject. Sin embargo, los GameObject por sí mismos no hacen nada son necesarias propiedades que harán que ese GameObject sea un ambiente, un efecto, un personaje, un terreno, entre otros.

De lo mencionado anteriormente, se puede decir que Los GameObject son contenedores a los que se les puede asociar diferentes componentes, atendiendo al objeto que se desee crear se agregarán diferentes combinaciones de componentes como: colisionadores, animaciones, efectos, scripts, entre otros.

### 8.1.5 PARTÍCULAS

Uno de los componentes de Unity son los sistemas de partículas, que consisten en una fuente de emisión de planos texturizados que siempre muestran sus caras hacia la cámara. Con un texturizado apropiado permiten generar efectos como fuego, humo, lluvia, entre otros.

Estos efectos permiten generar efectos gráficos muy vistosos y en su gran mayoría animados. En la imagen siguiente se puede apreciar las opciones que tienen los sistemas de partículas:

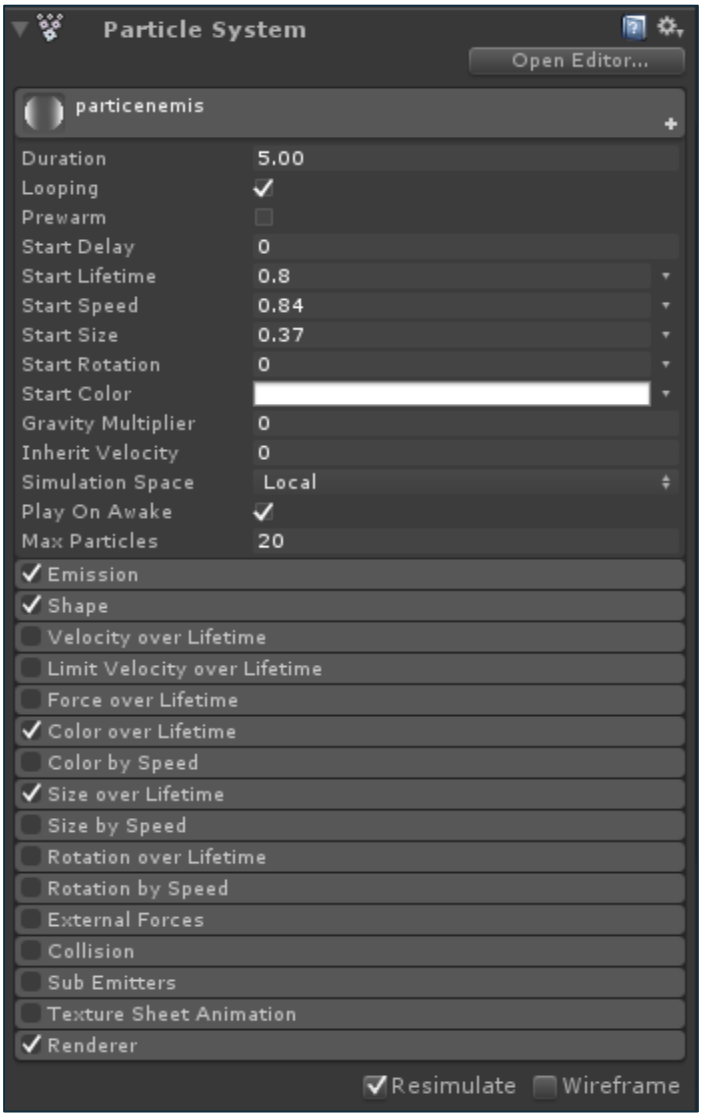


*Imagen 3: Elementos de los Sistemas de Partículas.*

### 8.1.6 CÁMARA

La cámara es la encargada de renderizar el resultado de la escena de juego en la pantalla. Al crear una nueva escena, Unity siempre incorpora una cámara en ella por defecto.

Cabe añadir que se pueden añadir varias cámaras a una escena. Además, las cámaras no sólo sirven para el renderizado de la escena, pueden usarse para simular cámaras de videovigilancia, para simular retrovisores, para crear juegos para varios jugadores (pantalla partida), entre otros.

La cámara, es un componente más que se agrega a los GameObjects. Sin embargo, puede contener una colección diferente de componentes, otorgando funciones adicionales a los GameObjects.

En la imagen 4 se pueden apreciar los elementos de la cámara principal (main camera).

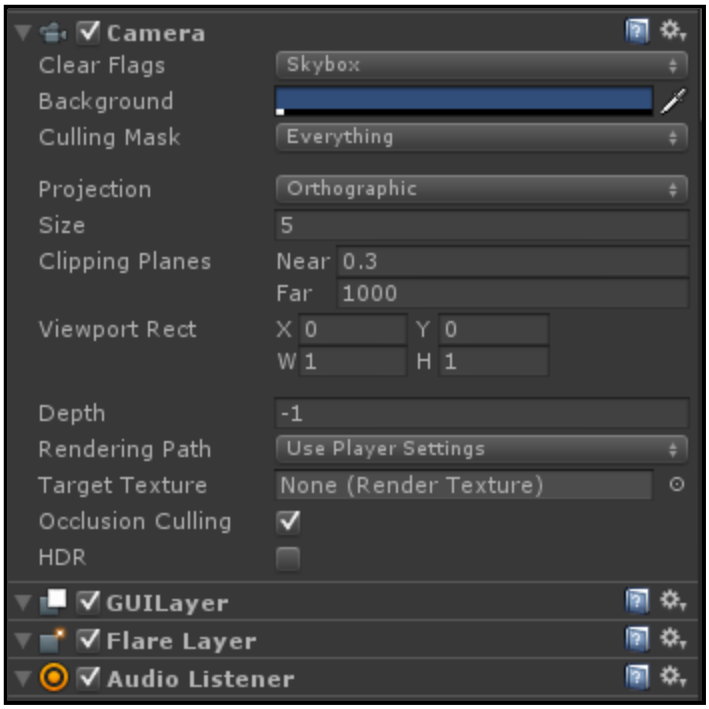


*Imagen 4: Elementos de la Cámara.*

## 8.2 FUNCIONES PROPIAS

Unity posee algunas funciones propias que son sobreescribibles, es decir permiten que el usuario defina su contenido. Respecto a estas funciones, Unity tiene un orden de ejecución, se puede decir que es el cuándo y nosotros desarrollaremos el qué y el cómo.

Para entender mejor lo mencionado anteriormente, se hablará del orden en el que Unity inicializa los distintos elementos que lo componen cada vez que se carga una escena del juego. De este modo, lo primero que se carga son los Game Object y componentes, seguidamente se cargan los scripts que van vinculados a estos Game Object, y finalmente se llaman una serie de funciones en un orden específico (Awake, Satart, Update / FixedUpdate y LateUpdate).

A continuación se dará una breve descripción de las funciones que se han empleado en el TFT.

### 8.2.1 AWAKE

Al cargarse la escena cuando se inicia el script, la función Awake es llamada. Esta función es útil para iniciar variables o estados del juego antes de que éste empiece. Hay que tener en cuenta que Awake será llamada aunque en el inspector la instancia del script esté deshabilitada. Además, no puede formar parte de una corrutina.

```
void Awake(){

}
```

### 8.2.2 START

Start es llamada justo después de Awake, se diferencia de esta última en que es llamada si la instancia del script está habilitada en el inspector.

```
void Start () {

}
```

### 8.2.3 UPDATE

Esta función es llamada cada frame, al igual que Start siempre y cuando la instancia del script esté activa. Es la función más usada en los scripts pese a que cada ordenador puede tener un framerate distinto.

```
void Update () {

}
```

### 8.2.4 FIXEDUPDATE

FixedUpdate, a diferencia de Update, se ejecuta cada cierto número fijo de frames. Es recomendable utilizarla cuando se hagan implicaciones de físicas y cuando se utilice un Rigibody.

```
void FixedUpdate () {

}
```

# 9 ANÁLISIS DEL PROBLEMA

Como se mencionó anteriormente, en el capítulo de aportaciones, el planteamiento del proyecto fue pensando en generar un análisis de las distintas herramientas que se usan para el desarrollo de un juego de plataforma estilo runner. Así pues, hablaremos del motivo por el cual se ha escogido cada una de ellas, sus ventajas, desventajas y los beneficios que ha aportado al proyecto.

A la hora de plantearnos el desarrollo de un juego de este tipo surgen muchas incógnitas como, que motor de videojuego usar, que herramientas emplear para crear los gráficos y el sonido, que lenguaje de programación es el más adecuado, entre muchas otras. A continuación daremos respuesta a estas cuestiones.

## 9.1 UNITY3D

Unity3D es un motor de videojuegos multiplataforma muy versátil, permite publicar o buildear para PC, en los sistemas operativos: Windows, Linux e IOS; para dispositivos móviles: Android, iPad, iPhone y Windows Phone; y para consolas de juegos: Xbox 360, PlayStation 3, PlayStation Vita, Wii, Wii U, entre otras. Además permite el buildeo en Web, haciendo uso de un complemento llamado Unity Web Player que es gratuito.

Esta herramienta creada por Unity Technologies cuenta con dos tipos de licencia, una es gratuita aunque posee ciertas limitaciones y otra llamada Unity-Pro que es pagada. En la actualidad se encuentra disponible para el desarrollo en los sistemas operativos Windows, Mac IOS.

Es importante recalcar que Unity posee un editor propio, Monodevelop, que soporta los lenguajes JavaScript, C# y Boo. Además posee un editor visual muy intuitivo lo que facilita el trabajo con los elementos visuales y en donde se puede observar la ejecución del programa. La creación de los ejecutables para cada una de las distribuciones es bastante sencilla, basta con elegir el sistema operativo y la arquitectura.

Finalmente, Unity cuenta con una interfaz que puede ser modificada de acuerdo a las necesidades del usuario, esta interfaz posee 5 elementos o vistas como se puede observar en la Imagen 5:

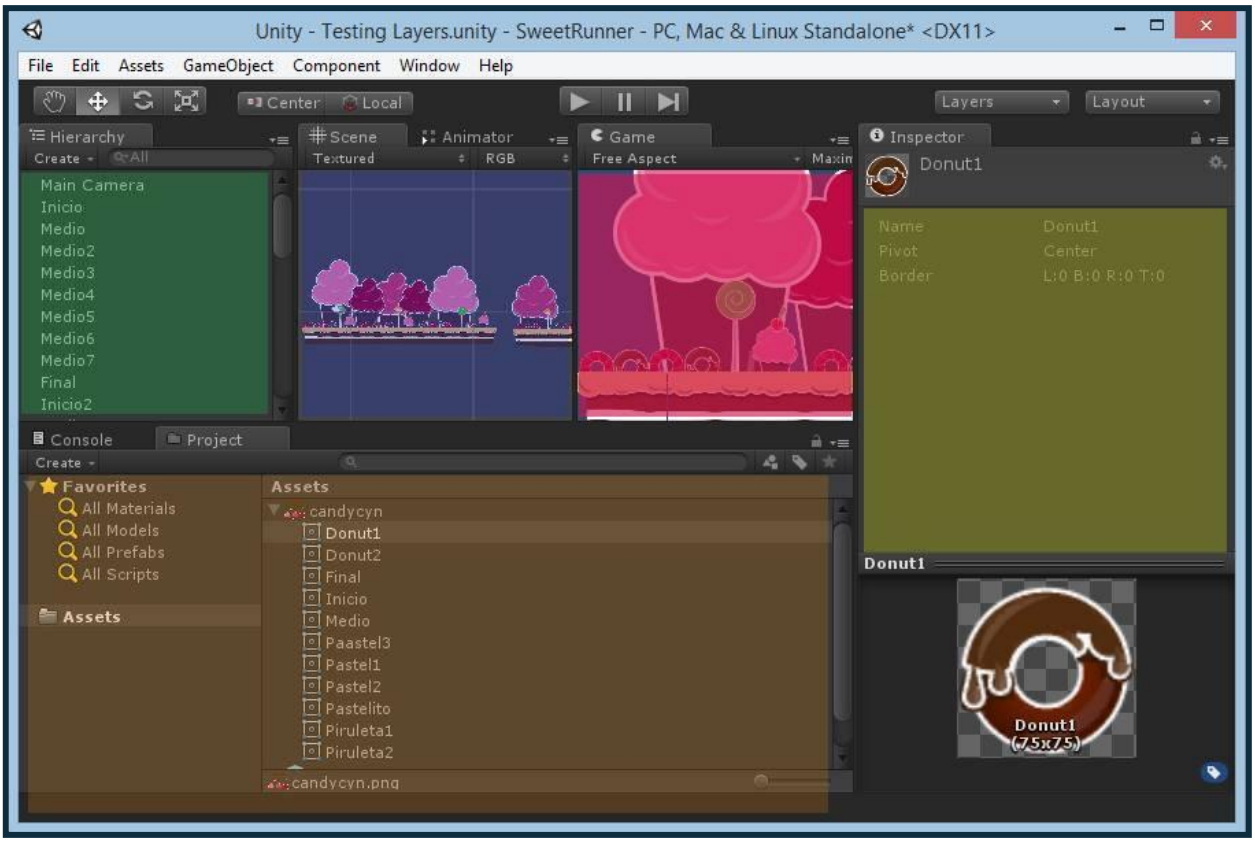


*Imagen 5: Interfaz de Unity3D.*

La vista de Escena (color azul) permite construir la parte visual de la aplicación, se pueden posicionar objetos importados, rotarlos, escalarlos y moverlos.

La vista de Juego (color rosa) posibilita previsualizar lo creado en la vista de Escena, permitiendo ejecutar la aplicación que está desarrollándose para probar su funcionamiento.

En la vista de Proyecto (color marrón) se encuentran todos los assets del proyecto, normalmente ordenados en carpetas. Es precisamente aquí donde se importan o crean los elementas de la escena, tales como objetos, texturas, scripts,

entre otros. El manejo de estos elementos es muy sencillo, basta con arrastrarlos en la escena.

La vista de Jerarquía (color verde) contiene los objetos que se encuentran en un momento dado en la escena. Gracias a esta vista resulta sencillo encontrar todos los elementos de la escena, sin la necesidad de moverse por ella.

Por último el elemento Inspector (color olivo), en él se encontrará toda la información detallada de los assets de la escena.

En resumen, las principales características que posee Unity3D son las siguientes:

- Software gratuito en su versión básica.
- Disponible para Windows y Mac OS.
- Espacio de trabajo intuitivo y muy sencillo, de fácil manejo y con opciones para ejecutar, editar y probar la aplicación que se está desarrollando.
- Alta calidad visual y de sonido, permite mezclar en tiempo real gráficos, tanto 2D como 3D, y audio.
- Poderoso sistema de animación que permite fluidez en las escenas.
- Multiplataforma, permite publicar o buildear en varios sistemas de manera sencilla.
- Soporta assets que pueden exportarse desde múltiples herramientas de modelado y se importan automáticamente.
- Manejo de un solo editor en tiempo real.
- Sistema de partículas en tiempo real.
- Manejo de inteligencia artificial.
- Iluminación dinámica en alta velocidad.
- Procesador de alta velocidad.
- Se puede encontrar gran cantidad de documentación.
- Cuenta con una amplia comunidad que brinda soporte.

En síntesis, Se ha optado por usar Unity3D por todo lo descrito anteriormente pese a que en el mercado se encuentran diversos motores de videojuegos igual de válidos. Sin embargo, Unity sigue siento el mejor a la hora de generar juegos en 2D y 3D para móviles. Otra de las razones definitivas fue su interfaz que es mucho más amigable e intuitiva, bajo mi punto de vista basado en la experiencia, que la de otros.

## 9.2 C#

Al hablar del lenguaje de programación escogido es necesario tomar en cuenta que Unity soporta, como ha mencionado en capítulos pasados, JavaScript, C# y Boo.

C#, es un lenguaje de programación orientado a objetos un poco menos flexible que JavaScript pero nos permite manipular mayor cantidad de datos.

Se trata de un lenguaje un poco más estricto que JavaScript, al ser más preciso es necesario escribir todos los tipos de variables y funciones dejando poco margen de error.

Finalmente, cabe añadir que nos da mayor acceso al System tolos, nos permite comunicarnos con cosas externas a Unity.

Estas son las razones por las que se ha optado por este lenguaje.

## 9.3 MONODEVELOP

MonoDevelop es un entorno de desarrollo gratuito y libre, se encuentra bajo la licencia GNU GPL. Está diseñado para soportar principalmente los lenguajes C#, JavaScript y Boo.

En la actualidad esta herramienta es multiplataforma, puede ser usada en GNU/Linux, Windows y Mac.

Además, es el entorno de desarrollo propio de Unity3D con lo cual se pude comprobar inmediatamente lo que vamos realizando. Al escribir un script, basta con asignarle a un Game Object para ejecutarlo desde la Vista de Juego de Unity y

observar todo lo que se va realizando. Se puede abrir Monodevelop desde la Vista de Proyecto de Unity, basta con dar doble clic sobre el script que se desea abrir.

A continuación, en la imagen 6 se puede observar el entorno de desarrollo Monodevelop.

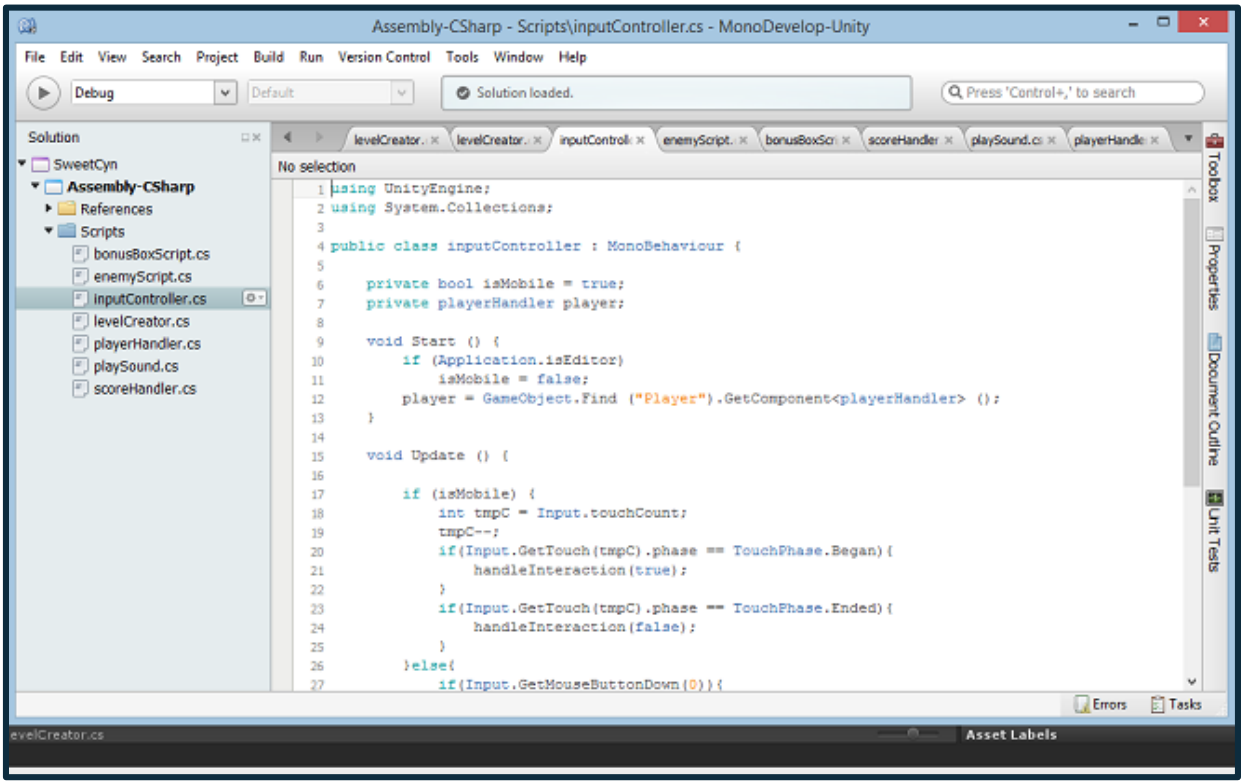


*Imagen 6: Entorno de Desarrollo Monodevelop.*

## 9.4 INKSCAPE

Inkscape es un editor de gráficos vectoriales de código abierto, se distribuye bajo la licencia GNU. Además, es un software gratuito.

Algo que vuelve a Inkscape único es que el formato nativo que usa es el Scalable Vector Graphics (SVG), que es un estándar abierto de W3C basado en XML.

Pese a que este software se encuentra desarrollado principalmente para el sistema GNU/Linux, es una herramienta multiplataforma que puede ser empleada en Windows, Mac OS X y otros sistemas derivados de Unix.

En cuanto a sus características, Inkscape es una herramienta de dibujo bastante potente y muy sencilla de usar, es totalmente compatible con los estándares XML, SVG y CSS. Además, posee una comunidad bastante amplia.

Por todo lo mencionado anteriormente es que se decidió escoger Inckscape.

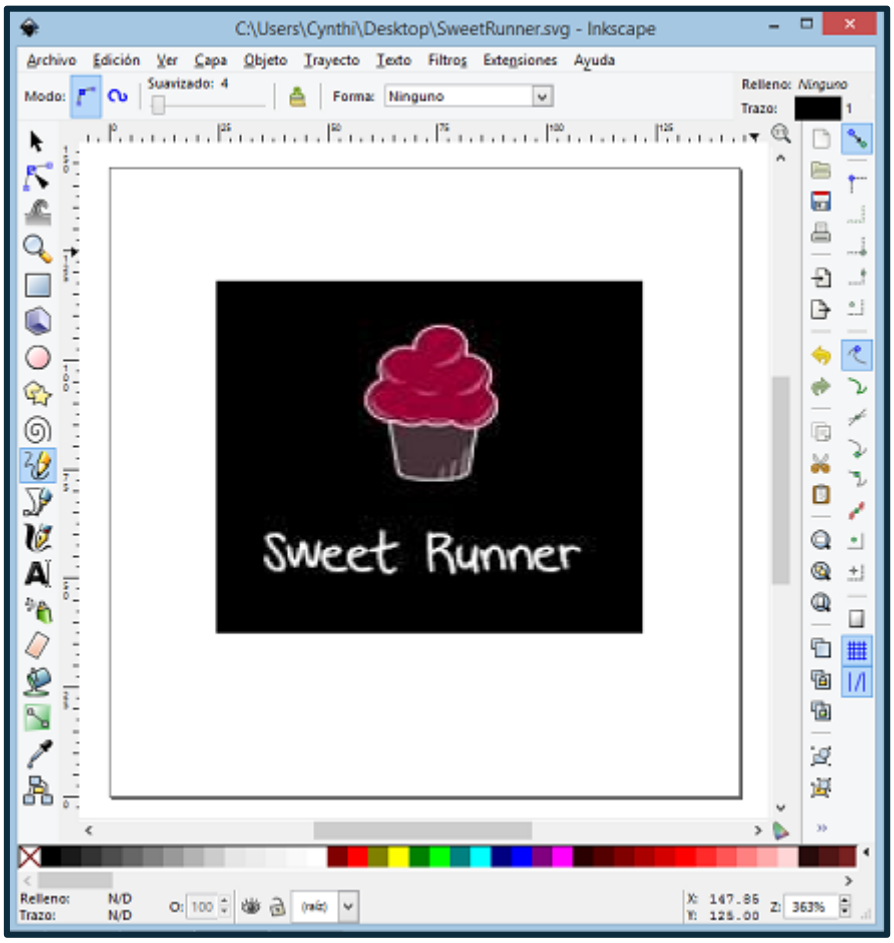


*Imagen 7: Interfaz de Inkscape.*

## 9.5 GIMP

GIMP (GNU Image Manipulation Program), es un programa de edición de imágenes digitales, tanto dibujos como fotografías, en forma de mapa de bits. Es un software gratuito y libre, es parte del proyecto GNU y se encuentra bajo las licencias GPL y LGPL.

GIMP permite el tratado de imágenes por capas, esto permite que cada objeto de la imagen pueda ser tratado de forma totalmente independiente. Soporta los formatos JPG, GIF, PNG, PCX, TIFF, la mayoría de psd y además posee su propio formato abierto, XCF. Es preciso añadir que permite importar imágenes vectoriales en formato SVG.

La interfaz de esta herramienta está disponible en varios idiomas. Además, es multiplataforma está disponible para los sistemas operativos Unix, GNU/Linux, FreeBSD, Solaris, Windows, Mac OS X, entre otros.

La razón principal por la que se decide usar GIMP para el desarrollo del TFT es por la gran compatibilidad que presenta con el resto de herramientas con las que se trabaja. Como se mencionó anteriormente, permite importar imágenes en formato SVG, que pueden ser creadas por Inkscape. Así mismo, recordemos que permite generar extensión PSD, que es compatible con Unity3D. Esta compatibilidad facilita mucho el trabajo puesto que se pueden realizar cambios desde GIMP que se cargarán automáticamente en el entorno gráfico de Unity.

En la siguiente imagen se muestra el entorno de GIMP.

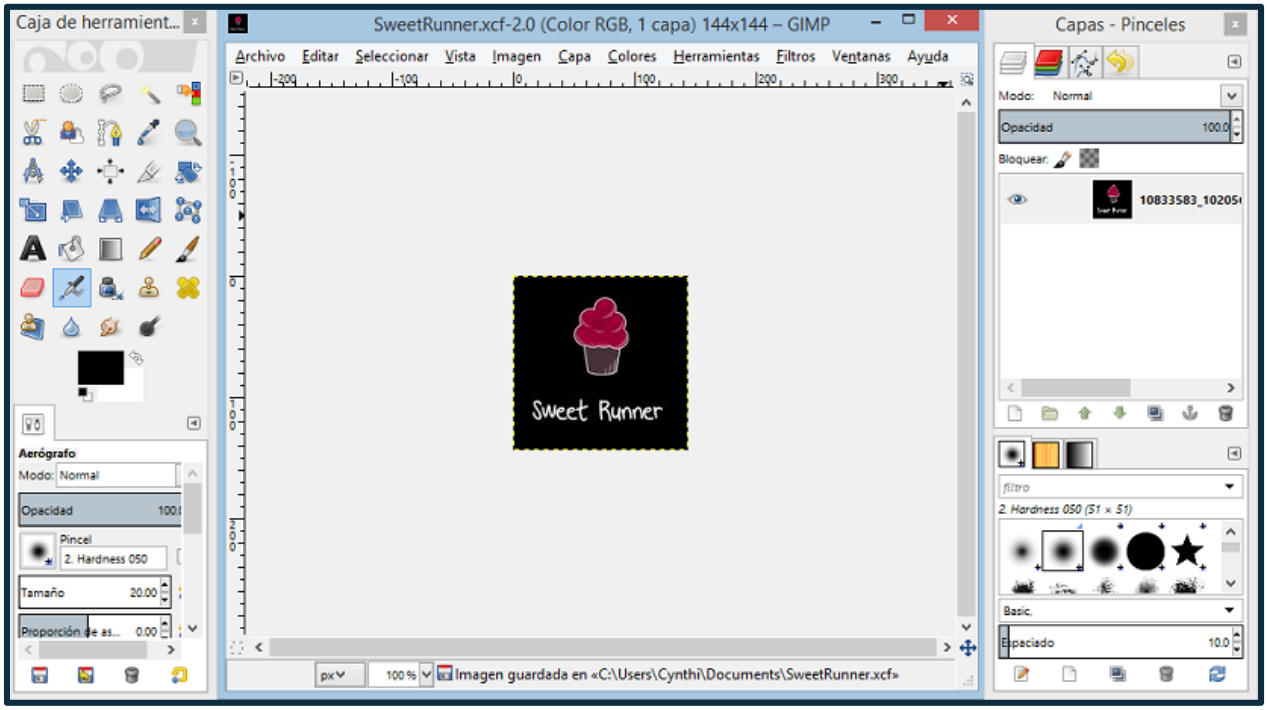


*Imagen 8: Interfaz de GIMP*

## 9.6 AUDACITY

Audacity es un software de grabación y edición de audio, se encuentra bajo la licencia GNU GPL.

Se trata de un software multiplataforma, se puede usar en Windows, Mac OS X, GNU/Linux, y otros sistemas. Su interfaz es muy amigable e intuitiva, además se encuentra traducida a varios idiomas.

Audacity permite grabar audio en vivo, grabar el sonido que se esté escuchando en el equipo, convertir cintas y grabaciones a sonido digital, cortar, copiar, unir o mezclar sonidos, entre otras cosas.

Estas fueron las razones principales por las que se decidió usar este software en SweetRuner. A continuación se puede observar una imagen en la que se muestra la interface de Audacity.

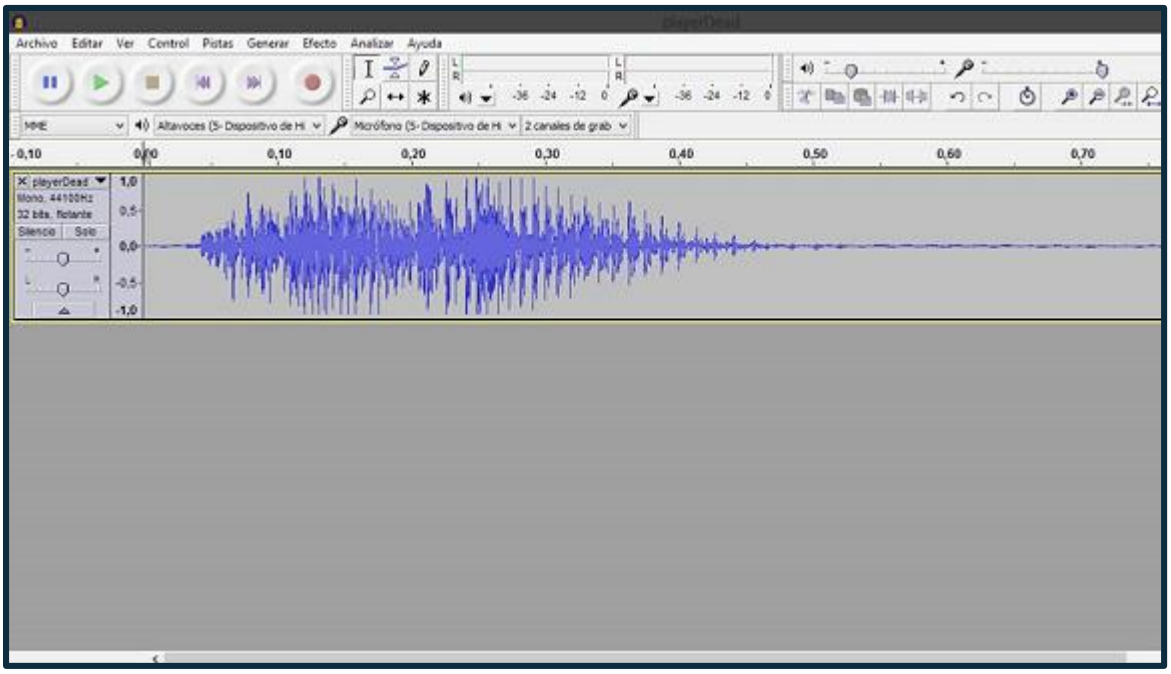


*Imagen 9: Interfaz de Audacity.*

# 10 DESARROLLO

Mientras nos adentramos en este capítulo hablaremos acerca del diseño, la implementación, pruebas y resultados del software desarrollado.

## 10.1DISEÑO E IMPLEMENTACIÓN

A lo largo de este apartado se tratará el proceso que se siguió mientras se realizaba el desarrollo del TFT. Cada uno de los apartados se corresponderá con cada una de las fases o iteraciones que se realizaron durante el desarrollo

### 10.1.1 PRIMERA APROXIMACIÓN

Esta fase se trata de una primera aproximación a lo que ahora es SweetRunner, se consigue introducir las primeras piezas gráficas al juego y otorgarles movimiento. A continuación se describe con mayor detalle cómo se realizó esta tarea:

#### 10.1.1.1 Apartado visual

En esta etapa inicial se consiguió crear el piso o terreno del juego y se agregó el primer objeto que recepta órdenes.

Es preciso añadir de forma muy breve el proceso llevado a cabo al incorporar los elementos gráficos a Unity3D:

- Para la creación de los citados elementos gráficos, se empleó una herramienta muy básica, Paint, dado que era para realizar una primera prueba. Es oportuno añadir que se terminó usando GIMP para cambiar la imagen creada a formato png.
- Una vez importada la imagen a Unity, es necesario cambiarla a formato ***Sprite*** (2d/uGUI) ya que por defecto se importa como una textura y las propiedades que tiene un sprite como Pixels to unit, sprite editor, Sprite Mode son imprescindibles para nuestro caso.
- Es preciso dividir el *Sprite Atlas* en las islas que se emplearán. Una vez realizada esta tarea se procede a crear, en la parte de Jerarquía, GameObjects según la cantidad de islas a usar. A cada GameObject se le asignarán 2 componentes: Renderizado, que es aquí donde se

agregará una de las islas previamente divididas del Atlas; y físicas y colisiones, que en este caso son del tipo 2D.

A continuación, se puede observar en la imagen 10 un GameObject señalado (Player), en la vista de Escena se puede apreciar gráficamente el GameObject Player (recuadro verde) y finalmente, en el Inspector se ven los componentes agregados.

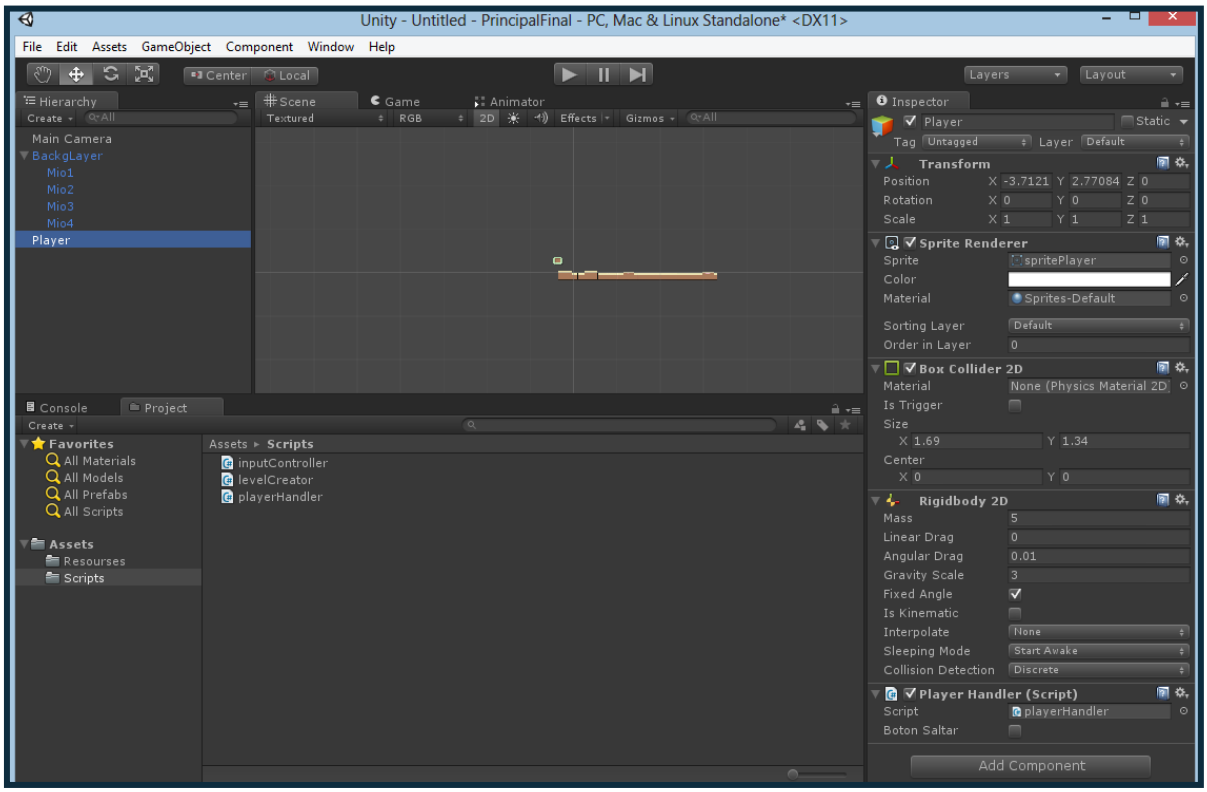


*Imagen 10: Entorno gráfico de Unity, tras crear los primeros elementos gráficos.*

### 10.1.1.2 Movilidad

Como se menciona en capítulos pasados, el entorno de programación empleado es Monodevelop. Se consiguió que el GameObject Player se moviera por el terreno del juego y que receptara órdenes del usuario. Consiguiendo así que el objeto realizara saltos por comando del usuario, mientras que el objeto avanzaba automatizado por un script.

Esta funcionalidad se realizó en tres scripts: levelCreator, imputController y playerHandler.

#### 10.1.1.3 Pruebas Realizadas

Al realizar las pruebas de este primer prototipo, nos encontramos con algunos problemas, en cuanto a funcionalidad, que fueron solventados antes de pasar a las siguientes etapas:

- **Salto hasta el infinito.** Al realizar saltos el objeto se elevaba demasiado, y se perdía. Para solucionar esto se tuvo que arreglar la física del elemento móvil, agregando mayor fuerza en la coordenada Y para que la distancia del elemento en esa coordenada, al saltar, sea menor. El fallo fue controlado en la función saltar con la siguiente línea de código:

```
this.rigidbody2D.AddForce (Vector2.up * 3000);
```

- **Saltos consecutivos infinitos.** Nos dimos cuenta que el objeto podía repetir consecutivamente infinitas veces la orden de salto. Se solventó el problema con una variable booleana que permita el control de saltos consecutivos.

A continuación se mostrará una imagen donde se puede apreciar el resultado obtenido.

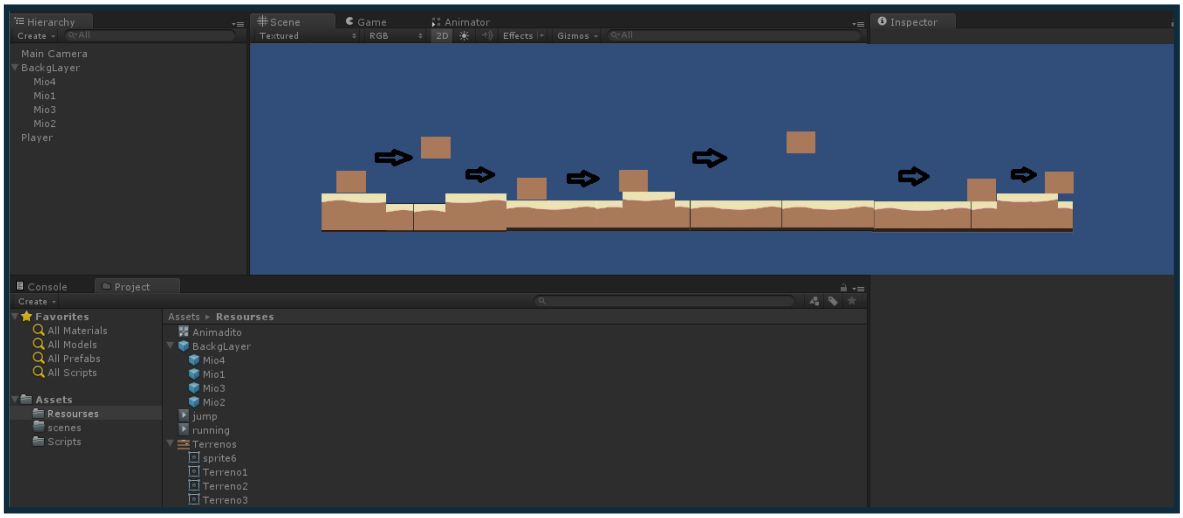


*Imagen 11: Ejecución del primer prototipo.*

### 10.1.2 Obstáculos y Bonificaciones

La segunda mejora que se incorpora al prototipo creado son los Obstáculos y Bonificaciones. Esta mejora se hizo en base a las siguientes historias de usuario:

- **Obstáculos.** Como jugador de SweetRunner quiero enfrentarme a obstáculos en el juego, para desarrollar mi agilidad psicomotriz y por extensión mi desarrollo mental.
- **Bonificaciones.** Como jugador de SweetRunner quiero tener bonificaciones para optar a tomarlas o no, pese a que las bonificaciones generalmente son una ayuda en este caso podrían contribuir a la pérdida de la partida. Debemos tomar la decisión correcta a la hora de coger o no la bonificación y de esta forma desarrollar nuestra velocidad al momento de afrontar problemas de decisiones.

Para cumplir con estás historias de usuario se realizaron las siguientes actividades:

- Se empleó un sistema de colisiones que permitirá detectar el momento en el que el jugador colisiona con un objeto o cuando obtiene una bonificación.
- La bonificación poseerá tres posibilidades (color azul, rosa, y verde con su respectiva iconografía) que van cambiando con el paso del tiempo.

Atendiendo al color que el jugador escoja podrá disminuir la velocidad del jugador, generar un salto muy alto (fuerza x 6000) o aumentar su velocidad, respectivamente. Adicionalmente el "***bonusBox***" se moverá de arriba abajo con el objetivo de realizar más compleja la tarea de recogerlo.

- Las historias de usuario fueron desarrolladas en los ficheros bonusBoxScript y enemyScript, respectivamente.

#### 10.1.2.1 Pruebas Realizadas

Para testear las mejoras añadidas al prototipo desarrollado se superaron las siguientes pruebas:

- Correcto desempeño al colisionar con un objeto: perdida de la partida.
- Correcto movimiento y cambio de color de la "bonusBox" con el paso del tiempo.
- Correcta acción al recoger cualquier color de una bonificación (bonusBox).
- Correcto funcionamiento de lo mencionado anteriormente tanto en los dispositivos móviles Smartphone y Tablet.
- Rendimiento aceptable del juego en ejecución.

A continuación se puede observar el resultado obtenido tras añadir las historias de usuario antes señaladas. La línea discontinua muestra el final de una partida, al colisionar el objeto con el obstáculo la partida termina. También se puede observar como la bonusBox va cambiando de color según el paso del tiempo y finalmente se puede apreciar la acción que realiza el objeto al coger la bonusBox. En este caso un salgo muy elevado.

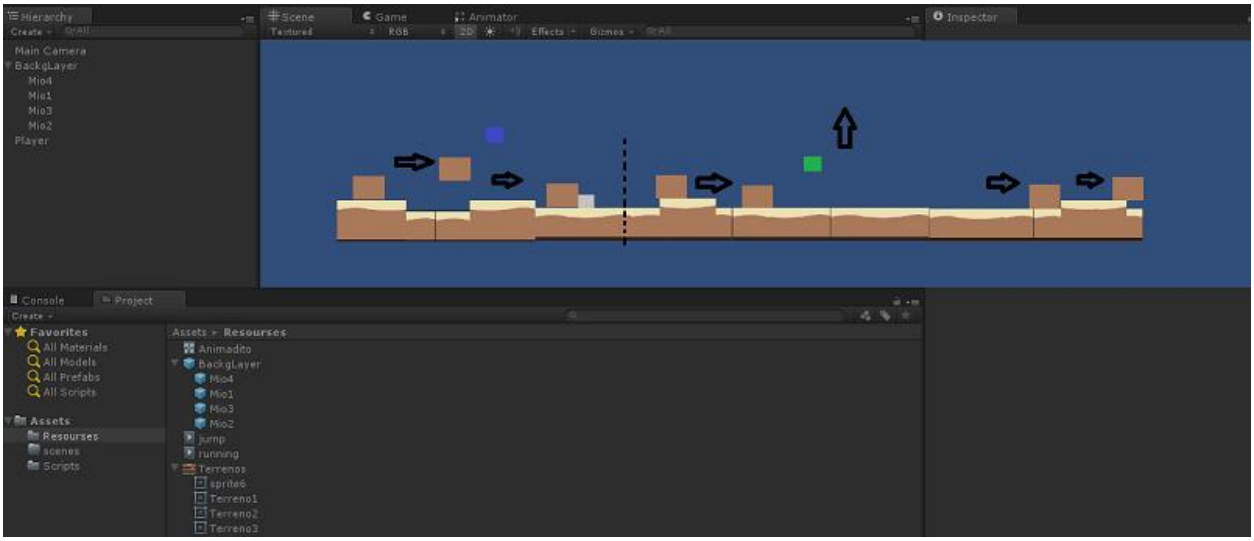


*Imagen 12: Ejecución del juego con obstáculos y bonificaciones.*

### 10.1.3 Generación Aleatoria

Un juego de este estilo debe tener gran cantidad de niveles y muy extensos para que no se vuelva monótono, en vez de ello se optó por crear de forma aleatoria e infinita el entorno, el terreno, los objetos y las bonificaciones. Con esto se satisfizo la siguiente historia de usuario:

- Como usuario de la aplicación quiero un juego que no sea monótono para que no se convierta en algo aburrido.

Para lograr esta mejora hizo falta desarrollar un sistema de reutilización de elementos. Para ello, se crean una cantidad predefinida de elementos en pantalla con los componentes de terreno y vacíos, estos elementos se crearán y destruirán sin sobrepasar ni ser inferiores al número que se definió con antelación. Este proceso se repetirá tantas veces sea necesario mientras el juego esté en funcionamiento.

El desarrollo de la generación aleatoria de elementos se realizó en el script levelCreator.

#### 10.1.3.1 Nivel extenso Vs Aleatoriedad infinita

A continuación, se explicarán las razones por la que se escogió el sistema descrito anteriormente para satisfacer la historia de usuario correspondiente a esta mejora.

#### 10.1.3.1.1 Siempre nuevo

Al crear un nivel muy extenso puede que las primeras veces no se monótono. Pero sin duda, tras jugar muchas veces llegará un punto en el que ya sabremos exactamente dónde van los obstáculos, cuando aparecen las bonificaciones y como es el terreno o las plataformas, convirtiéndolo inevitablemente, tarde o temprano, en un juego aburrido.

Al desarrollar un juego aleatorio e infinito se rompe con las problemáticas citadas anteriormente, se tendrá al jugador en expectativa para sortear los obstáculos, coger las bonificaciones y moverse por las diferentes plataformas o terrenos que se van creando.

#### 10.1.3.1.2 Rendimiento y Ralentización

Es inevitable hablar de rendimiento y ralentización a la hora de debatir entre nivel extenso y aleatoriedad infinita.

Desarrollar un juego con niveles extensos implica que las plataformas o terrenos, enemigos, bonus y entorno deben ser creados en su totalidad y almacenados para poder jugar.

Al realizar la tarea de este modo, todo se almacena en memoria y deberá ser llamado en su totalidad. Así por ejemplo, si tenemos un nivel con unas 70 plataformas de diferentes tamaños, 120 obstáculos, 20 bonusBox, 200 pastelitos de diferentes tamaños y el cielo. Todos estos elementos se almacenan en memoria y se realizan 411 llamadas para lograr recrear las imágenes.

En nuestro caso en particular, se creó un nivel no tan extenso como el descrito anteriormente, y al realizar las pruebas en los dispositivos móviles nos encontramos con que había ralentización en la creación de las imágenes y el objeto móvil principal tenía lag.

Si bien este tipo de juegos presenta los problemas que se describieron antes, también hay algunas maneras de solucionarlos. En este caso en particular, y dado que se debe tener en cuenta la jugabilidad requerida por el usuario, se optó por la solución de la Aleatoriedad.

Al escoger esta opción, los problemas de rendimiento que se presentaban anteriormente se solventan. Al generar un número predefinido de elementos en pantalla, que son construidos y destruidos tantas veces sea necesario, lo único que se llamará serán esos elementos.

#### 10.1.3.2 Pruebas y Problemas Solventados

Para satisfacer la historia de usuario concerniente a esta mejora, se comprobó el correcto funcionamiento de lo descrito anteriormente en los dispositivos móviles.

Como se mencionó en el apartado anterior, al realizar las primeras pruebas se encontró algunos problemas que se solventaron haciendo un mejor uso de los recursos.

En la imagen 13 se puede apreciar la mejora implementada:

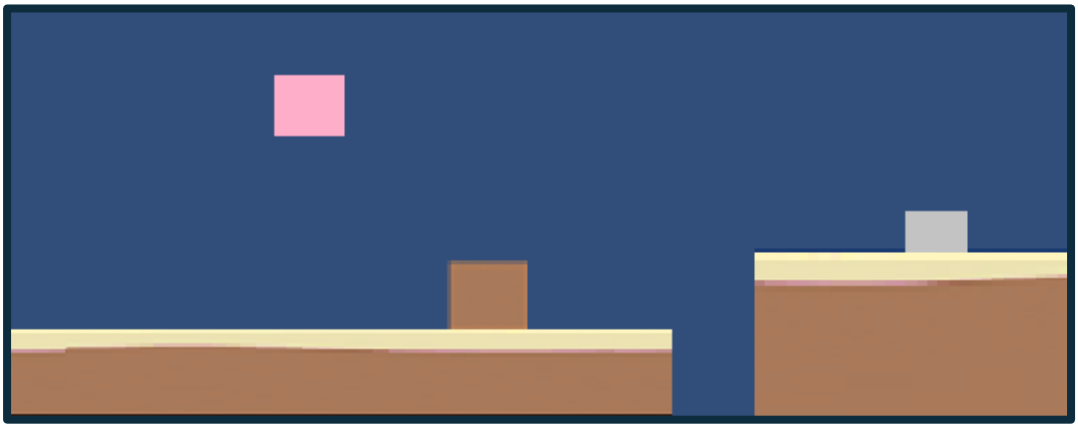

*Imagen 13: Generación aleatoria de elementos.*

.

### 10.1.4 Elementos Gráficos

En esta mejora se incorporan los elementos gráficos que serán parte de SweetRunner. En la Imagen 14 se muestran los elementos gráficos que fueron introducidos:

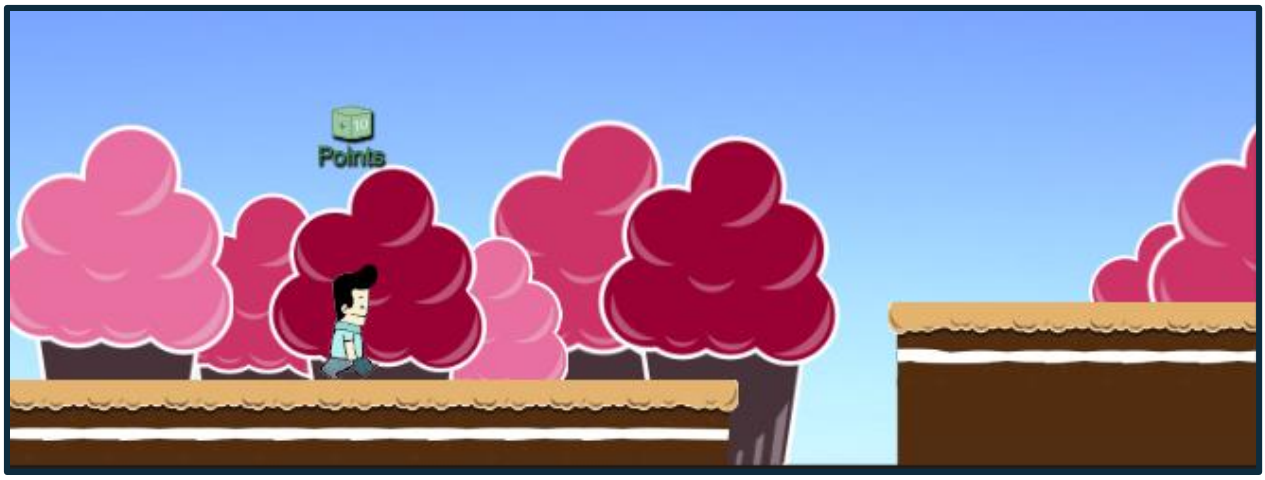


*Imagen 14: Gráficas del Juego.*

Lo primero que se realizó fue ir cargando uno a uno los gráficos y se les dio el valor de Sprite Texture, tal y como se realizó la primera vez al desarrollar el prototipo inicial.

#### 10.1.4.1 Rendimiento

Al realizar la integración de imágenes como se describe anteriormente, se pudo notar que el rendimiento de la CPU resultó seriamente penalizado. Este fenómeno se produce porque se llaman una a una cada isla o sprite. La GPU no presentó perdidas en rendimiento.

Para evitar la penalización en rendimiento, se optó por juntar todas las imágenes en 2 atlas:

- **Atlas 1:** Terreno o plataforma, background y personaje.
- **Atlas 2:** bonusBox y obstáculos

Se emplearon dos atlas atendiendo a las propiedades de sus islas, la bonusBox y los obstáculos poseerán efectos posteriormente.

Una vez conformados los atlas, se notó que el rendimiento de la CPU mejoró notablemente, el rendimiento fue superior al que tenía cuando se trabajaba con las imágenes del prototipo inicial.

Esta mejora tan significativa se debe a que al tener solo dos atlas, se aligeró el coste de trabajo producido por la CPU ya que ahora tan solo deberá realizar dos llamadas.

#### 10.1.4.2 Luces

Es imprescindible la incorporación de luces en el escenario dado que éstas también generan efectos como sombras, y contorneados.

En este caso se utilizó una luz para iluminar, y con ello hacer visible, el plano posterior azul.

#### 10.1.4.3 Pruebas

Al realizar pruebas en los dispositivos móviles se notó que había ralentización del proceso de llamadas gráficas. El problema se solventó como se describe en el apartado anterior, rendimiento.

Las pruebas finales se realizaron en los dispositivos móviles y esta vez se realizó un número elevado de pruebas, con el fin de detectar cualquier problema de elementos no eliminados. Una vez más las pruebas se superaron satisfactoriamente.

#### 10.1.4.4 Programación Vs Facilidades de Unity

Empezaremos hablando de algunas de las facilidades que nos proporciona Unity.

Con el fin de ahorrar tiempo Unity brinda algunas facilidades, con sus diversos componentes. Como por ejemplo, los componentes de físicas, que aparte de emular bastante bien la realidad son totalmente manipulables y realmente flexibles. También permite integrar colisiones con los componentes de colisiones, al igual que las físicas también se pueden manipular.

Además de esas facilidades, Unity también permite la programación de estos componentes. Es decir, permite que un desarrollador que programe sus propias físicas, colisiones, luces, entre otros.

En algunos casos es mejor programar diversos elementos y en otros emplear los componentes proporcionados, no hay un estándar que especifique nada de esto,

todo va depender del proyecto que se esté desarrollando, su tamaño, las necesidades que se presenten, entre otros.

En el caso del presente TFT se emplearon unos pocos componentes de los que ofrece Unity. Se empleó la luz, los componentes de sonido, y el sistema de partículas y colisionadores. Sin embargo se desarrollaron otros componentes como físicas, movilidad del personaje, avance de cámara, entre otros.

### 10.1.5 Integración De Puntuación

Esta mejora tiene el fin de satisfacer las siguientes historias de usuario:

- Como jugador de SweetRunner quiero saber mis logros para así saber mi evolución en el juego.
- Como jugador de SweetRunner quiero saber cuáles han sido mis mayores logros saber mi record.

Para solventar estás historias de usuario se creó un sistema de puntuación que se incrementa a medida que avanzamos en el juego. El sistema de puntuación se desarrolló en el fichero “scoreHandler” además recibe información de “levelCrator” y bonusBox. El score o puntuación ha sido cifrado con MD5 para asegurarnos que no es susceptible de ser modificado manualmente.

En pantalla se muestran dos puntuaciones, en el lado Izquierdo superior, la puntuación que vamos acumulando a medida que avanza el juego, y en la parte superior derecha la puntuación más alta o High Score.

#### 10.1.5.1 Puntuación BonusBox

Una mejora añadida, o cambio realizado, fue en la bonusBox. Se decidió cambiar la bonificación recibida al tomar el color verde, ahora en lugar de aumentar la velocidad incrementará la puntuación en 10 puntos.

Esta mejora se realizó con el fin de volver más tentadora la idea de coger la bonusBox, ya que al juagar SweetRunner nos percatamos que muchas veces se

dejaban de lado las bonificaciones pues no había un incentivo lo suficientemente fuerte como para asumir el riesgo de cogerlas.

En la imagen 15 se observa la mejora incorporada.

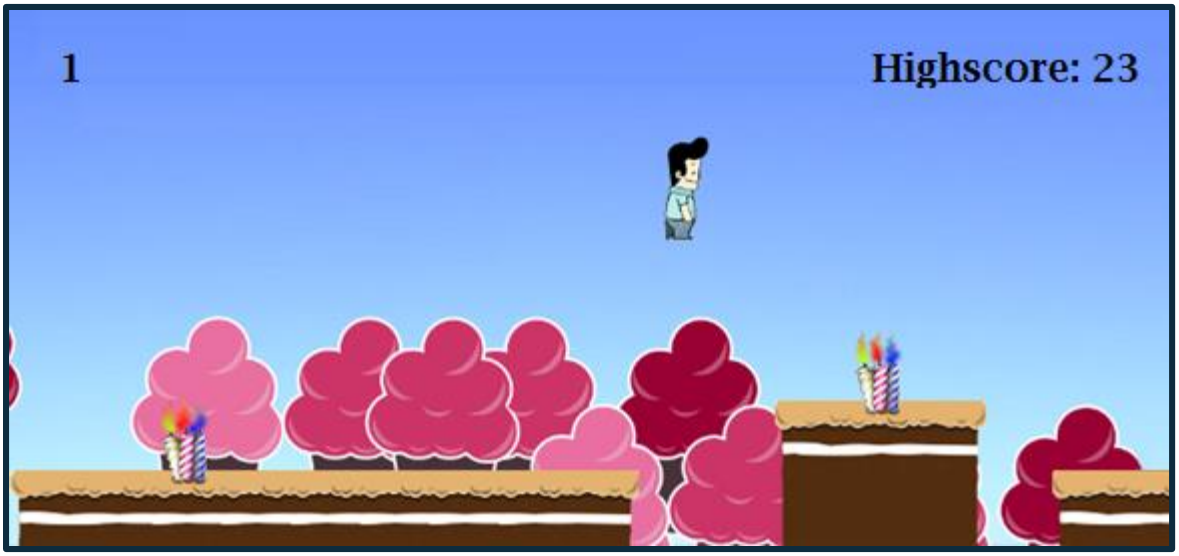


*Imagen 15: Incorporación de la Puntuación.*

### 10.1.6 GUI

Esta mejora se corresponde con la Interfaz Gráfica de Usuario. Se ha desarrollado una interfaz gráfica que nos permite mostrar de manera más vistosa la puntuación máxima que el usuario ha acumulado.

Al realizar las pruebas se presentó una problemática. Al ejecutarse en el dispositivo móvil tablet no hubo ningún problema. Sin embargo, al probarlo en un Smartphone el texto e imágenes de la GUI perdieron la posición en la que debían estar.

El trabajo con las GUI por lo general representa un reto y es que el diseño se debe adaptar a diferentes resoluciones. Es decir, se debe lograr que la GUI sea responsive o adaptativa.

Para conseguir la adaptabilidad se debe hacer uso de valores dinámicos, se trabaja en base al tamaño de la pantalla en la que se esté ejecutando el juego:

```
void OnGUI(){
    style.alignment = TextAnchor.UpperLeft;
    GUI.Label(new Rect (20,20,200,200), score.ToString(),style);
    style.alignment = TextAnchor.UpperRight;
    GUI.Label(new Rect (Screen.width - 220,20,200,200), "Highscore: "+
              bestscore.ToString(),style);
}
```

En la siguiente imagen se podrá apreciar las mejoras Integración de GUI de puntuación máxima.


*Imagen 16: GUI de Puntuación.*

### 10.1.7 EFECTOS

Esta mejora se trata de introducir ciertos efectos gráficos en el juego para volverlo más vistoso. La mayor parte de juegos emplean estos efectos para otorgarle un realce gráfico pues realmente se gana mucho, se puede trabajar en base a detalles y la mejor forma es con sistemas de partículas.

#### 10.1.7.1 Sistemas de Partículas

Los efectos se crean, como se mencionó anteriormente, con sistemas de partículas. Unity ofrece dos sistemas de partículas, Legacy y Shuriken, con los que se puede crear una gran variedad de efectos.

Estos sistemas vienen predefinidos en cuanto a la cantidad de variaciones que se pueden realizar, lo que quiere decir que no se puede realizar cualquier tipo de efecto. Posee limitaciones.

En el caso del presente proyecto, se decidió incorporar el sistema de partículas Shuriken para los obstáculos empleados, concretamente para hacer las flamas de las velas, que son los obstáculos del juego.

#### 10.1.7.2 Rendimiento y Ralentización

Tras realizar pruebas sobre los nuevos componentes agregados, se apreció un nuevo cambio significativo en cuanto a rendimiento. Las aparición de imágenes se ralentizó y los movimientos dejaron de ser fluidos..

Al empezar a trabajar con partículas, se hizo uso de gran cantidad de efectos sin tomar en cuenta la penalización que estos podrían generar. En realidad, nunca se midió el desgaste de memoria que se estaba generando.

Con el fin de solventar el problema de rendimiento descrito, se tuvo que llegar a un consenso entre rendimiento y funcionalidad.

De esta forma, se manipulo la herramienta de partículas tomando en cuenta que cada partícula emitida se puede considerar un sprite. Se ajustó el número de partículas que se deben emitir y se habilitaron funciones que ayudaron al efecto como cambios de colores de las flamas por tiempo de vida, con esta función se evitó el uso de 2 sistemas de partículas (uno de color rojo para la parte baja de la flama y uno de color amarillo para la parte alta).

El efecto se replicó para las otras dos velas, empleándose color amarillo y verde para la vela amarilla, rojo y amarillo para la vela rosa, y azul y amarillo para le vela azul. Como se puede observar en la imagen 17.

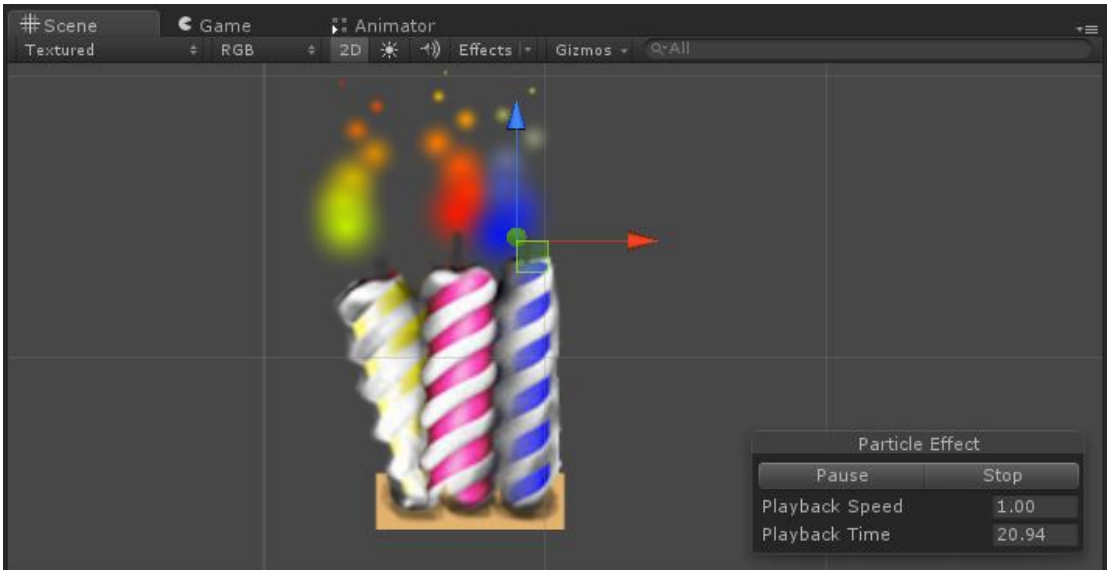


*Imagen 17: Sistema de Partículas Shuriken con sus valores en tiempo real.*

Gracias a lo descrito anteriormente, se pasó de tener una emisión de 1000 partículas a tener una emisión de 10. Se les agregó un ciclo de vida, con el propósito de poder lograr que desaparezcan, con el fin de evitar que las partículas estén siempre cargadas en memoria.

Por último, se empleó la creación elementos prefabricados de este estilo.

### 10.1.8 REDES SOCIALES

La incorporación de redes sociales a SweetRuner se hizo para satisfacer las historias de usuario siguientes:

- Como jugador de SweetRunner, quiero poder mostrar a mis amigos la puntuación máxima que he obtenido para que vean mis logros.
- Como jugador de SweetRunner, quiero poder mostrar la puntuación que obtengo en Facebook para poder etiquetar y realizar comentarios con la gente.

#### 10.1.8.1 SDK Facebook

Para la incorporación de esta última funcionalidad se trabajó con la API que se facilita en la página de desarrolladores de Facebook. La API proporciona SDKs para iOS, Android, JavaScript, PHP y Unity.

Los pasos que seguimos para la incorporación y empleo de la API fueron los siguientes:

- Al descargar el SDK de Facebook para Unity se presenta en forma de Unitypackage.
- Una vez descargado, se debe anexar el Unitypackage al proyecto y con ello se puede empezar el desarrollo.
- Se debe establecer comunicación entre la app y Facebook, para ello se deben agregar algunos parámetros desde Unity3D a la página de desarrolladores de Facebook.
- El código proporcionado posee parámetros de uso para Android, iOS y web. Es necesario seleccionar lo que se va a emplear.
- En el apartado de Android se creó un nuevo script y por medio de él se pidieron las propiedades de Android necesarias.
- Finalmente se debe unir el score de SweetRunner con los mensajes de salida para Facebook.

#### 10.1.8.2 Pruebas

Al realizar las pruebas se pudo observar que los botones incorporados, redirección a Facebbok, login, logout, funcionaban correctamente.

Cuando tratamos de acceder a login, éste solicitó un token. A la hora de trabajar en modo desarrollador el token viene a ser el nombre de usuario en un móvil, un identificador de cuenta al momento de buildear. Este proceso será necesario solo la primera vez.

En la imagen 18 se puede observar la publicación de la puntuación en Facebook.


*Imagen 18: Publicación en Facebook de la puntuación.*

### 10.1.9 Limitación De Frames

Se consideró oportuno realizar una limitación del número máximo de Frames que se pueden usar. Esto lo hicimos con la función Awake:

```
void Awake(){
    Application.targetFrameRate = 60;
}
```

Limitar el número de frames es considerado una buena práctica, debido a que en ciertos casos los dispositivos pueden generar una mejor cuenta de fps (Frames por segundo) pero esto solo terminará por consumir recursos, pudiendo acabar con la batería del dispositivo rápidamente debido al trabajo de la tarjeta de vídeo.

### 10.1.10 Incorporación de audio

Para SweetRunner se emplean 5 sonidos: de fondo, loop; de salto, jump; cuando muere el jugador, die; cuando el jugador se colisiona con un obstáculo, playerDead; y cuando el jugador coge una bonusBox, powerup.

La aparición de sonidos en el juego se realiza en los scripts: playSound, playerHandler, enemyScript, bonusBox; y levelCreator.

# 11 PRUEBAS FINALES Y RESULTADOS

Como se mencionó en capítulos anteriores, las pruebas y la correcta solventación de los errores que surgieron en cada prueba se realizó en cada iteración del desarrollo.

La vista de juego de Unity3D facilitó mucho la tarea de testear o probar la herramienta que se iba desarrollando. Una vez superadas todas las pruebas en el mencionado visor de juego, el proyecto era correctamente exportado para Android y se procedía a la realización de las pruebas en los distintos dispositivos móviles.

Es de suma importancia hacer hincapié en que no es lo mismo desarrollar para PC que para móvil. Es evidente que el nivel de procesamiento de un Pc es más elevado que el de un móvil. Sin embargo la resolución de pantalla de un dispositivo móvil es mejor que la de un PC. Es por eso que se debe tener en cuenta, cuando se desarrolla para móviles, el rendimiento, las Draw Calls, el número de scripts que se tienen, las llamadas entre scripts, entre otros.

Como se puede apreciar durante el capítulo de desarrollo, todos los cambios y anexiones que se realizaron resolvieron en un juego óptimo para móviles. Consiguiéndose los siguientes valores:

- Framerate de 71.1 fps max y 60 fps min.
- Menos de 10 drawcalls
- El uso de la Vram es mínimo con rango de 3 a 10.7 MBs.
- Los triángulos no sobrepasan los 450.
- Los vértices no sobrepasan los 310.

Todo lo mencionado anteriormente se podrá observar en la siguiente imagen

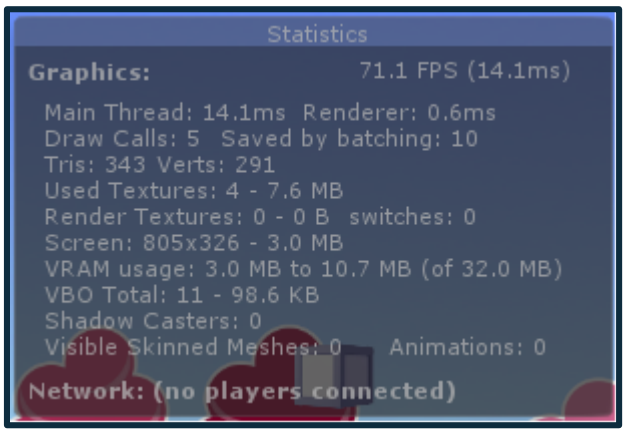


*Imagen 19: Estadísticas del Juego.*

Finalmente, cabe mencionar que SweetRunner presenta un funcionamiento adecuado tanto en PC, Smartphone y tablets. Podemos observar la aplicación en estos 3 soportes en la siguiente imagen.

*Imagen 20: SweetRunner en dispositivos móviles.*

# 12 CONCLUSIONES Y TRABAJOS FUTUROS

## 12.1 CONCLUSIONES

Las conclusiones que se obtienen de la realización del presente trabajo de final de grado, son las siguientes:

- El desarrollo de la aplicación plataforma estilo runner, para dispositivos móviles basados en Android, ha sido todo un éxito. Como resultado de este TFT se ha obtenido un juego que funciona perfectamente en dispositivos móviles Smartphone y Tablet, que además incluye un diseño adaptativo (responsive).
- Tras haber desarrollado SweetRunner en Unity3D, se pude decir que Unity nos brinda algunas facilidades como sistemas de partículas para efectos gráficos, sistemas de colisiones, luces, un entorno de desarrollo que cuenta con una vista, Game, que permite observar la ejecución del juego, un inspector, para poder observar los valores y componentes de cada elemento, entre otras. Es justo mencionar que nos encontramos con algunas dificultades a la hora de usar Unity, como la función de autoguardado, en ocasiones se estaban realizando ejercicios de ensayo y error y al tratar de volver a la versión que habíamos guardado, no podíamos pues Unity había guardado los cambios de forma automática.
- El entorno de programación compatible con Unity es Monodevelopment. Unity no acepta ningún otro entorno de desarrollo.
- Según la experiencia que se ha obtenido tras el desarrollo de SweetRunner, la mejor manera de desarrollo para dispositivos móviles es siguiendo una metodología incremental y llevando acabo optimizaciones constantes.
- Determinar la mejor manera de desarrollo, tanto grafico como codificado, para encontrar la optimización del rendimiento de la aplicación.

- Un dispositivo móvil posee menor capacidad de procesamiento que una PC, por esta razón es necesario tomar en cuenta el rendimiento y optimizar al máximo cada componente.

## 12.2 TRABAJOS FUTUROS

Pese a que se considera que le material desarrollado y el estudio realizado como satisfactorio, siempre es posible realizar mejoras sobre estos. Así se proponen las siguientes mejoras.

- Permitir que la aplicación se comunique con otras redes sociales, no solo con Facebook.
- Establecer un ranking a nivel mundial de las puntuaciones más altas, mostrando nombre del jugador o seudónimo, país y puntuación. Todo esto, siguiendo la Normativa y Legislación vigente en esa fecha.
- Controlar no solo la velocidad para aumentar la dificultad, se podría añadir mayor número de obstáculos a medida que se obtiene puntuación. También se podría disminuir el tamaño de las plataformas o terreno, al igual que los obstáculos, en función de la puntuación que se va acumulando.
- Se puede realizar un estudio empleando la nueva versión, 5.0, que sacó Unity el pasado 18 de Marzo.
- Se puede realizar un estudio comparativo entre el uso de las distintas versiones de Unity3D
- Finalmente, se puede realizar el mismo estudio empleando otro motor de videojuegos.

# 13 FUENTES DE INFORMACIÓN


[1] Juegos plataforma.

http://es.wikipedia.org/wiki/Videojuego_de_plataformas

[2] Historia de los videojuegos para móviles.

http://en.wikipedia.org/wiki/Mobile_game

[3] Página oficial de Android.

http://www.android.com/

[4] Historia y versiones de Android.

http://androidos.readthedocs.org/en/latest/

[5] Página de desarrolladores de Andorid.

http://android-developers.blogspot.in/

[6] Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.

http://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750

[7] Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico.

https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758&b=34&tn=1&p=20140510#a22

[8] Normativa de Cookies.

https://www.agpd.es/portalwebAGPD/canaldocumentacion/publicaciones/common/Guias/Guia_Cookies.pdf

[9] Licencia Pública General (GNU LPG).

http://www.gnu.org/copyleft/gpl.html

[10] Licencia Pública General Reducida (GNU LGPL).

https://www.gnu.org/licenses/lgpl.html

[11] Licencia Apache 2.0.

http://es.wikipedia.org/wiki/Apache_License

[12] Condiciones del servicio de Google Play.

https://play.google.com/intl/es-419_us/about/play-terms.html

[13] Proceso Unificado.

http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational

[14] Manifiesto ágil.

http://agilemanifesto.org/iso/es/

[15] Metodología de desarrollo ágil Scrum.

https://www.scrum.org/

[16] Manual oficial de Unity.

http://docs.unity3d.com/Manual/index.html

[17] Página oficial de Unity.

https://unity3d.com/

[18] Guía de programación de C#.

http://msdn.microsoft.com/es-es/library/67ef8sbd.aspx

[19] Página oficial de Monodevelop.

http://www.monodevelop.com/

[20] Página oficial de Inkscape.

https://inkscape.org/es/

[21] Página oficial de GIMP.

http://www.gimp.org/

[22] Página oficial de Audacity.

http://audacity.sourceforge.net/?lang=en

[23] Página de Descarga de Sonido Gratuito.

https://www.freesound.org//

[24] API Facebook.

https://developers.facebook.com/docs/android?locale=es_ES

[25] Página oficial de Eclipse.

https://eclipse.org/