UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

AGE OF BUSINESS

Ingeniería informática

Memoria presentada por:

Daniel Pellicer Selfa

Tutor del alumno:

Jordi Joan Linares Pellicer

***Convocatoria**: julio 2017*

Resumen:

El proyecto desarrollado muestra cómo diseñar un juego en base al mercado actual y cómo aprovechar pautas psicológicas para crear una posible adicción en el jugador en base a acciones que liberan dopamina en el cerebro. En este documento se recogen aspectos clave para dicha acción
Respecto al desarrollo del juego, se ha utilizado el motor de videojuegos Unity3D del cual se tenía un conocimiento base junto al “plugin” de Google Play Services.

Palabras clave: Unity3D, gamificación, juego, minijuego, android, diseño, análisis, google, play, services, dopamina.

Abstract:

The project developed shows how to design a game based on the real market and how to take advantage of psychological guidelines to create a possible addiction in the player based on the actions that release dopamine in the brain. This document contains the key to action Regarding the development of the game, it has been used the video game engine Unity3D which had basic knowledge and also Google Play Services unity plugin.

Keywords: Unity3D, gamification, game, minigame, android, design, analysis, google, play, services, dopamine.

ÍNDICE

1- INTRODUCCIÓN

- Descripción del proyecto

Este proyecto trata sobre el desarrollo de varios minijuegos(juego de minijuegos) estilo 2 dimensiones utilizando el motor de desarrollo Unity3D pero centrándose en la retención del jugador mediante elementos de alto nivel de frustración.

El juego está dirigido a aquellas personas que posean un Smartphone con Android como sistema operativo y que deseen pasar muy poco tiempo en un juego el cual les proporciona un gran reto.

- Motivación

Mi motivación principal para este juego es obtener una gran capacidad de retención y captación de los usuarios, no solo se ha investigado sobre el lenguaje de programación C#, empleado en el motor Unity3D, sino también en el elemento químico del cerebro encargado de la adicción (dopamina) como por ejemplo el efecto que tiene la nicotina en una persona. El gran reto ha sido tratar de plasmar esta idea en el proyecto y obtener un resultado semejante en la experiencia de juego.

El mundo de la creación de videojuegos realizados por una persona es muy competitivo y hay muy pocos casos de éxito. Mediante este proyecto intento demostrar que es más importante trabajar en manejar los sentimientos de las personas para proporcionarles un gran nivel de satisfacción, que viene dado por un gran esfuerzo en un intervalo de tiempo muy reducido.

- Originalidad de la idea

La idea de este proyecto surgia por un juego de hace muchos años, de cuando no existían los Smartphones, el cual se trataba de pulsar la tecla 0 en diez segundos el mayor número de veces posible. Fue un juego extremadamente sencillo pero me creo una gran adicción. La idea ha sido replicar la esencia de este juego dando mi valor añadido en base al elemento químico del cerebro, la dopamina..

- Estado del arte

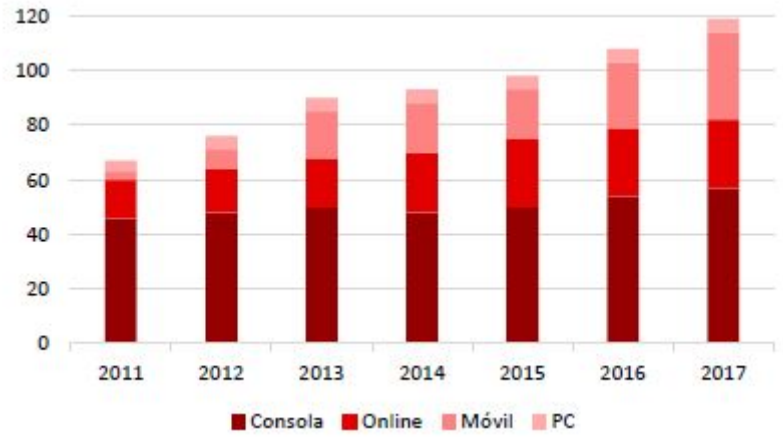
En la actualidad, dedicarse profesionalmente a la creación de videojuegos se está volviendo más popular y las compañías que desarrollan los motores de desarrollo de videojuegos hacen inciso en la fácil accesibilidad para toda persona que quiera formar parte de esta comunidad, cada vez los motores de desarrollo se vuelven más intuitivos y a la vez reducen su precio de adquisición hasta el punto de darlo gratuito como el caso de unity3D que ha pasado de ser de pago a ser gratuito. Además estas empresas te proporcionan informacion, documentación, videos realizando proyectos como ejemplo, etc para que tu aprendizaje sea más llevadero, y no solo las empresas, también hay usuarios que se dedican a compartir el conocimiento adquirido mediante el uso del motor.

El punto malo de una fácil accesibilidad es que conlleva a tener una mayor competencia en el mercado con lo cual también conlleva un porcentaje de éxito cada vez más reducido.

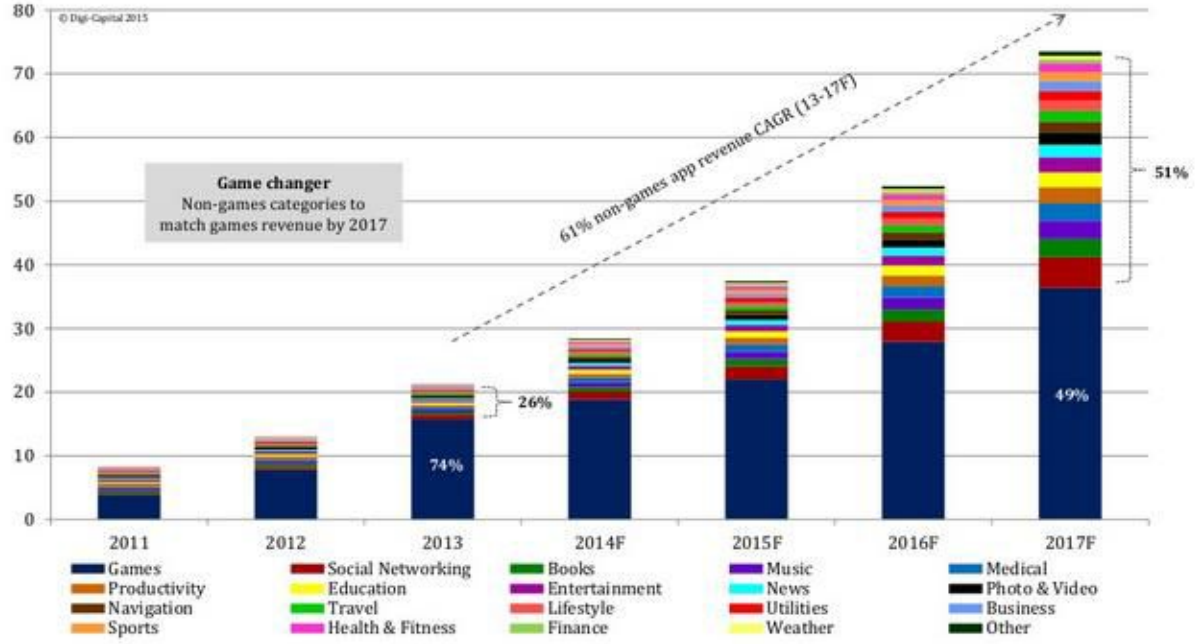
Obviamente si eres un desarrollador que realiza los proyectos en solitario o con un equipo muy reducido (freelance), es muy complicado competir con grandes empresas como Microsoft Studios, Nintendo,...

La industria del videojuego está cada vez más en auge, sobre todo en el ámbito de los dispositivos móviles, así es que tanto en la AppStore como en GooglePlay, el 20% de aplicaciones corresponde a juegos.

Según un estudio realizado por la firma Digi-Capital2, la industria de los videojuegos alcanzará un volumen de negocio de 100.000 millones de dólares en 2017, lo que muestra el gran potencial que ofrece.

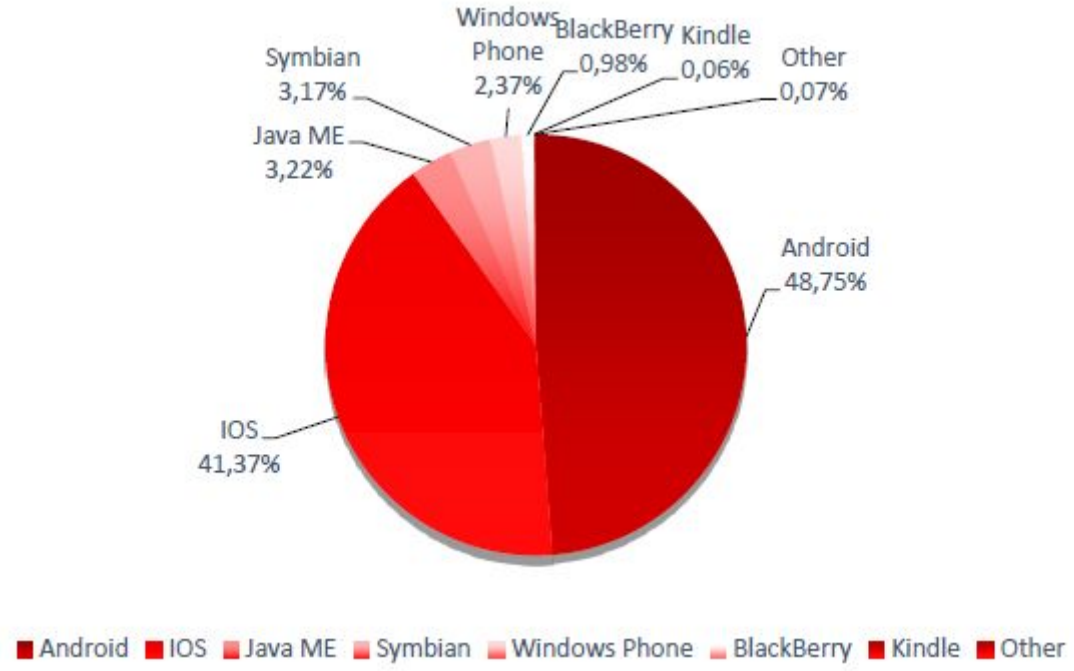


Los juegos son y seguirán siendo el principal tipo de aplicación de descarga en dispositivos móviles, como muestra la siguiente imagen:



Comparando entre entornos móviles, IOS, Android, WindowsPhone, como podemos observar en la siguiente imagen, a diferencia de hasta ahora, Android supera a IOS en la

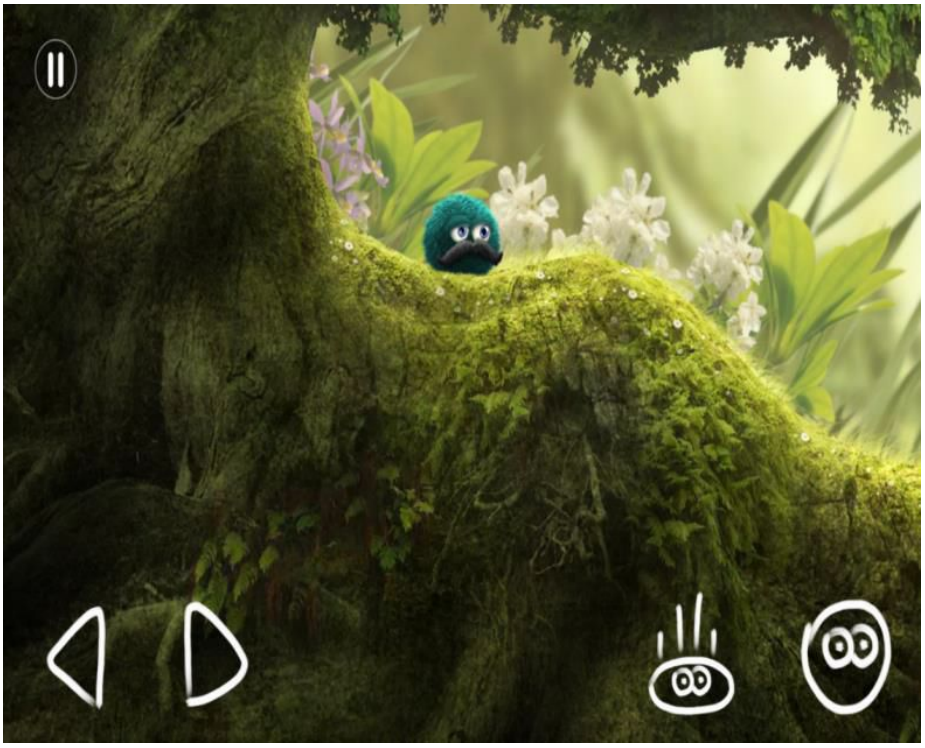
cabeza de la lista, aumentando de un 35% a un 48%, mientras que IOS baja de un 54% a 41%. El problema que tenía Android era la fragmentación de sus versiones, pero debido a las medidas tomadas por Google para solucionar este tema, se ve cómo le gana la primera posición a IOS. Symbian y Windows Phone parecen no ser una gran amenaza por ahora.



-Juegos referencia

Juego que empezó en iOS y recibió muy buenas críticas, llegando a ganar el premio Apple Design en 2014, por lo que fue desarrollado en Android, donde también tiene muy buena nota. El protagonista es Leopold Fortunate (Leo), un ingeniero que posee una gran fortuna

de oro, al cual tendremos que ayudar a recuperar su oro robado a lo largo de 24 distintos niveles de plataformas y puzles. El fuerte de este juego es su apartado gráfico, detallado y bonito, que, junto con un control fluido y sencillo, te hace no querer parar de jugar.



✔ Pros: gráficos, controles, fluidez, chekpoints.
✘ Contras: duración. Para ser un videojuego de corta duración, el coste de $4.99 es algo excesivo, comparando con otros de características similares.

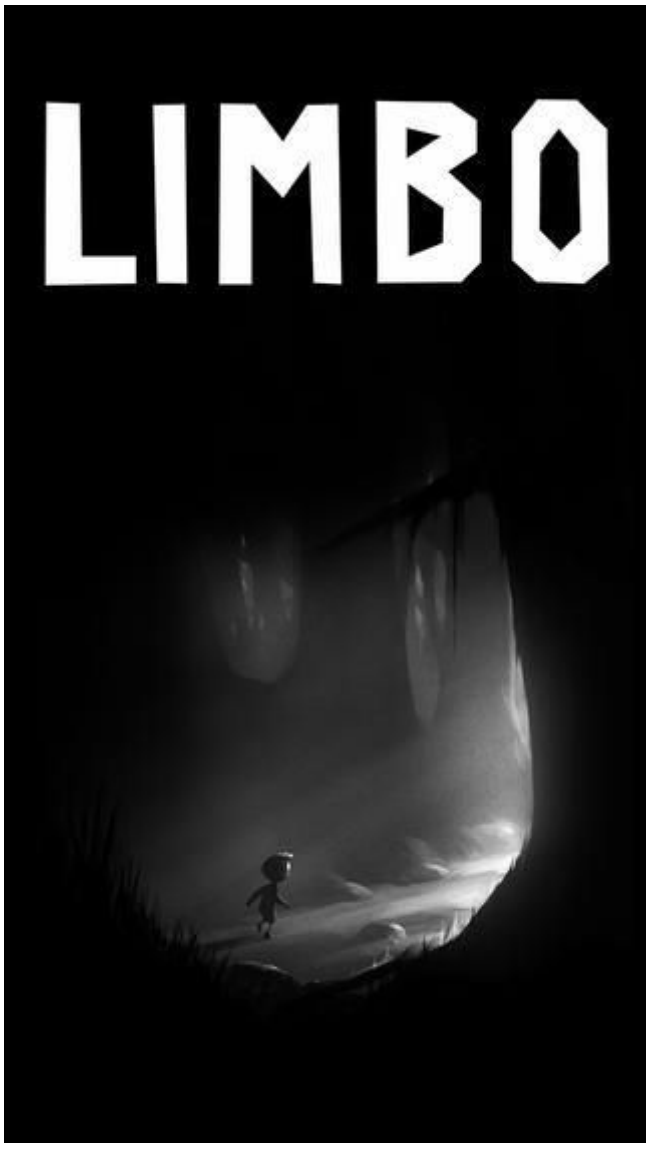
Juego que ya dispone de 3 ediciones y miles de descargas. Quiere parecerse al famoso Super Mario Bros, pero se queda en el camino. Los gráficos no son muy buenos, la interfaz algo caótica y no siempre responde correctamente al usuario. Para mi gusto es lento, sientes que el personaje se mueve con lentitud. Los niveles son muy parecidos todos, y no hay ninguna relación entre los objetos que te encuentras, tanto enemigos aleatorios como ¿piñas? que lanzas para acabar con ellos no tienen ningún tema en común. Es uno de los pocos juegos gratuitos que dispone de checkpoints, aunque escasean en los niveles.

✔ Pros: chekpoints, sencillez, duración.
✘ Contras: gráficos, interfaz, lento, publicidad.

Al igual que el Lep's World, intenta imitar al querido Super Mario Bros, copiando el nom-bre Super Mario – Super Oscar, las nubes y bloques de monedas, etc.. Pero no tiene nada que ofrecer en comparación, interfaz muy caótica, publicidad en cada paso que das, no hay check-points, y no siendo los niveles muy cortos, molesta mucho el tener que volver al inicio cada vez que mueres. Tiene bastantes bugs y, al igual que en Lep's World, pero en este caso más exagerado, los enemigos no tienen un tema en común, hay mucha variedad, pero sin ningún sentido.

✔ Pros: gráficos, sencillez, duración.

✘ Contras: interfaz, publicidad, checkpoints, bugs.



Uno de los referentes en cuando a juegos in-dies. Destacado por su apartado gráfico y sonoro mini-malistas. Sin color ni diálogos, ni historia, pero con algo que te lleva a no querer dejar de jugar. Trata sobre un niño a quien debemos guiar por un mapa en blanco y negro, sin saber a dónde ni por qué, podemos decir que estamos tan perdidos como el protagonista del juego.
Se desarrolló para Xbox, y posteriormente para PC. No hay versión para dispositivos móviles, pero he creído necesario el comentar uno de los mejores juegos indies de plataformas.
Limbo es, en sí, una obra de arte convertida en videojuego que consigue superar a algunos otros de grandes compañías.

✔ Pros: gráficos, sonido, jugabilidad, sencillez.
✘ Contras: duración. Ha recibido muchas críticas por la corta duración del juego.

Un claro ejemplo del auge de los juegos en Android y su potencia, ya que Rayman es uno de los iconos de la gran compañía Ubisoft, y han decidido a dar el salto a esta plataforma, reci-biendo una gran acogida. El que un juego cueste dinero, no siempre es malo, pues con ese dinero se mantiene y actualiza el juego constantemente, permitiendo disfrutar de un gran juego como este. En este caso no se trata de un juego de plataformas, sino uno del género conocido como endlessrun, en el que no controlamos el movimiento del personaje, sino sólo acciones como saltar, golpear. Los gráficos, música y control le hacen ser digno de ser un videojuego AAA.

✔ Pros: gráficos, sencillez, banda sonora, duración.
✘ Contras: el ser endlessrun hace repetitivos los niveles, y la música que son pocas pistas repetidas durante muchos niveles.

-Objetivos

Como se ha mencionado anteriormente el objetivo principal es desarrollar un videojuego en 2 dimensiones centrándose en la cualidad de retención y captación de jugadores. Además también se ha realizado un breve estudio sobre la estrategia de monetización empleada en el juego la cual se verá más adelante.

-- Público destinatario

El público al cual va dirigido este videojuego es toda persona que no quiera pasar más de 2 minutos aproximadamente en un juego pero quiera un gran nivel de excitación y un gran uso de su agudeza mental. El juego va a ser no solo para competir contra ti mismo sino también contra tus amigo, y aun mejor, con jugadores de todo el mundo gracias a un servicio de google que permite la implementación de una tabla de puntuación a nivel global y asi podras ver en qué posición del ranking te encuentras.

2- HERRAMIENTAS DE DESARROLLO

-Introducción a Unity3D

En este capítulo se introduce el motor gráfico Unity3D, sus características y modo de uso. El aprendizaje sobre Unity3D ha sido autónomo por medio de tutoriales, la comunidad de Unity, y mucha práctica.
Los proyectos en Unity3D se basan en la utilización de Assets (paquetes de recursos), los cuales contienen materiales, efectos, scripts, sprites, etc... Unity dispone de la Asset Store, una tienda donde pueden adquirirse recursos ya creados por otros desarrolladores, gratuitos o de pago, que permiten reducir el tiempo de desarrollo, y evitar “reinventar la rueda”. Para los iniciados en este motor gráfico, como en este caso, Unity te facilita ayuda mediante una extensa comunidad en la que puede encontrarse cualquier tipo de problema resuelto, una gran ayuda a la hora de empezar a utilizar un software desconocido y tan potente.
La Interfaz de Unity es intuitiva y sencilla, no ha sido difícil familiarizarse con ella en un periodo corto de tiempo. Esta se divide en varias secciones (Interfaz de Unity3D):

1. Project: en esta sección se muestra la estructura del proyecto en Unity, en ella se muestran los Assets que incluyen los recursos a usar en las escenas del videojuego.

2. Hierarchy: objetos de la escena actual.

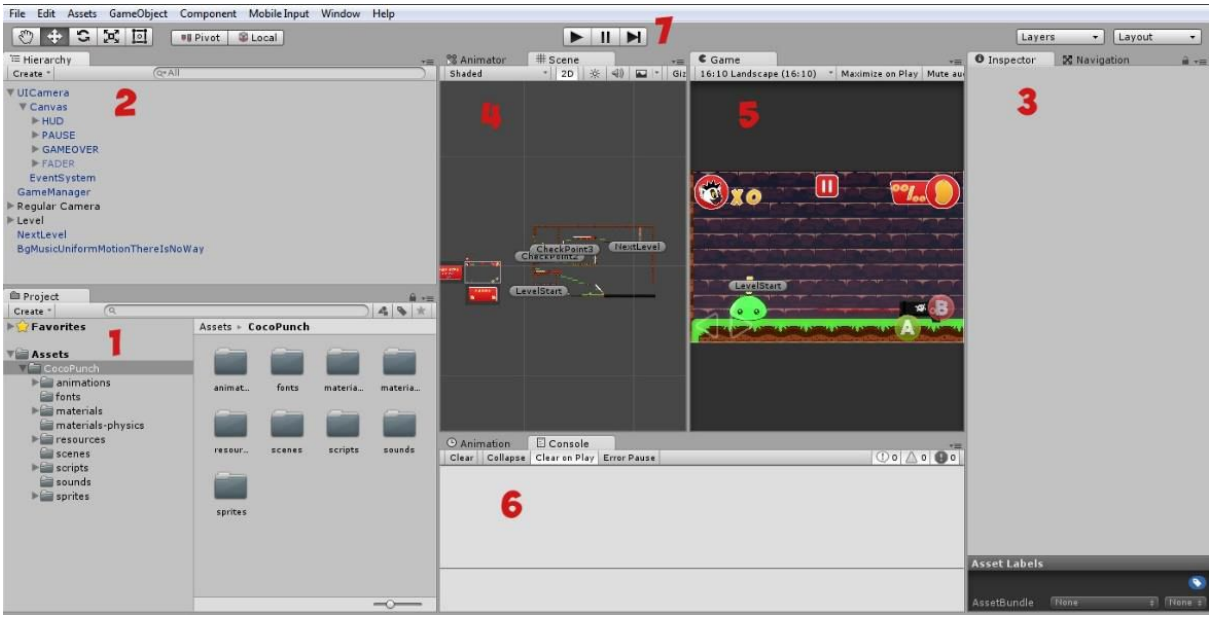
3. Inspector: una de las partes más importantes de Unity3D, aquí se muestran las características del objeto seleccionado, aquí se añaden los atributos y scripts que dan vida a los objetos.

4. Scene: al igual que en Android se tienen Activities, o pantallas que el usuario visualizará, Unity tiene las Scenes, donde se pueden añadir los recursos para construir los niveles o pantallas del videojuego.

5. Game: sección que muestra cómo se verá el juego al compilarse, hay que tener cuidado con el AspectRatio o resolución de la pantalla elegida, dependiendo de la plataforma sobre la que se desarrolle el proyecto, se mostrarán unas u otras. En el proyecto se utilizará una resolución para dispositivos Android en posición horizontal (16:10 Landscape).

6. Console: muestra los mensajes de error/debug.

7. Game controller:
Play: ejecuta/detiene el juego.
Pause: pausa el juego, permitiendo observar la escena sin necesidad de detener el juego por completo.



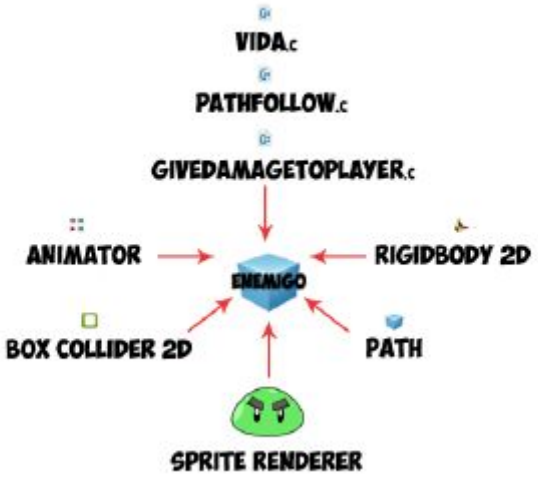
Al tratarse de un videojuego en 2D, no van a usarse materiales ni físicas en 3D, sino en 2D. Para materiales, en Unity se utilizan los denominados Sprites, mapa de bits 2D, y los SpriteSheet, que son conjuntos de sprites. El empaquetar sprites en spritesheets reduce la memoria utilizada; Unity permite dividir los sprites de un conjunto.
Un videojuego en Unity se basa en GameObjects (objetos), que siempre tienen una característica en común:

Transform indica la posición, rotación y tamaño del objeto en las 3 dimensiones (X,Y,Z), al trabajar en 2, solo se hace caso a X e Y.
A estos GameObjects se les puede añadir todo tipo de Componentes (efectos, audio, físicas, render, scripts, UI, etc..) y es con lo que compone los niveles del juego.
Unity facilita las cosas permitiendo crear Prefabs, que son objetos que predefinidos con unos componentes, y puedes añadirlos las veces que quieras con solo arrastrarlos hacia tu Scene.

Componentes necesarios para conocer son, entre otros:
· Sprite Renderer: convierte un objeto en un sprite 2D.
· Animator: permite a un objeto tener distintos estados de animación.
· Physics 2D: permite a un objeto colisionar con otros.
· RigidBody: permite a un sprite interactuar con las físicas del juego.
· Camera: convierte un objeto en una cámara.
· Particle System: convierte un objeto en generador de partículas.
· Script: permite añadir funcionalidades y comportamientos a los objetos.

Unity se compone de GameObjects que tienen distintos componentes, pero ¿cómo hacer que un objeto tenga vida, o que un objeto te ataque? De eso se encargan los scripts, donde se programan los comportamientos de los objetos, y el funcionamiento del videojuego. Por

ejemplo, si se tiene un objeto Enemigo, y se quiere que tenga vida, que patrulle un camino y que pueda atacarnos al entrar en su visión, se le deberán añadir distintos componentes:



En cada script deberá programarse el código que realice la función requerida. Los scripts en Unity deben heredar de la clase MonoBehaviour, que es la clase principal de Unity, disponiendo de distintas funciones que siguen un ciclo de vida como Awake() – inicializa el script, Start() – se ejecuta una sola vez al activarse el script o Update() – se ejecuta una vez cada frame.

-Plataformas objetivo de unity:
Web:
- WebGL

PC:
- Windows
- Windows Store Apps
- SteamOS
- OS X
- GNU/Linux

Dispositivos móviles:
- iOS
- Android
- Windows Phone
- Tizen

Smart TV:

- tvOS
- Samsung Smart TV
- Android TV

Consolas:

- PlayStation Vit
- PlayStation 4
- Xbox 360
- Xbox One
- Wii U
- Nintendo 3DS
- Nintendo Switch

Dispositivos de realidad virtual:

- Oculus Rift
- Google Cardboard
- HTC Vive
- PlayStation VR
- Samsung Gear VR
- Microsoft Hololens

-Licencias

Antes existían dos licencias principales para desarrolladores: Unity Personal y Unity Professional, a partir de mediados del 2016 Unity Technologies anunció que cambiaba su modelo de licencias a la siguiente: Unity Personal (todas las prestaciones del motor con únicamente la restricción de compilar con un splash screen con el logo y la leyenda "Made with Unity", tiene un tope de ingresos de $100 mil dólares, al llegar a dicho tope será necesario subscribirse a la Plus o a la licencia Pro), Unity Plus(enfocado a desarrolladores móviles, topado en $200 mil y servicios limitados de Unity)con un precio de suscripción de US $35 con un periodo determinado de compromiso(usualmente 1 año), Unity Pro(sin tope de ingresos, acceso a todos los servicios de Unity y hasta 200 usuarios simultáneos con Unity Multiplayer) y un precio de US $125 al mes, adicionalmente a éstas licencias existen: Enterprise (se tiene que contactar con la empresa directamente para ver si se califica para ella y convenir un precio, adicionalmente los servicios se pueden personalizar), Education (que ofrece acceso al Unity Educator Toolkit y certificaciones), la AEC (para arquitectura, ingeniería y construcción que se puede obtener únicamente a través de VIM su distribuidor oficial con éstos fines) y la Gambling(que por el tipo de negocio se requiere un contrato de uso distinto).

Todas las versiones dan acceso a la documentación del motor y a tutoriales o vídeos de entrenamiento. La versión Pro y Plus ofrece soporte a una versión, ejemplo si ha comprado Unity 5 esta licencia le da acceso a todas las actualizaciones y soporte de las siguientes mejoras de la versión (Unity 5.x), al igual que le da acceso a las versiones beta.

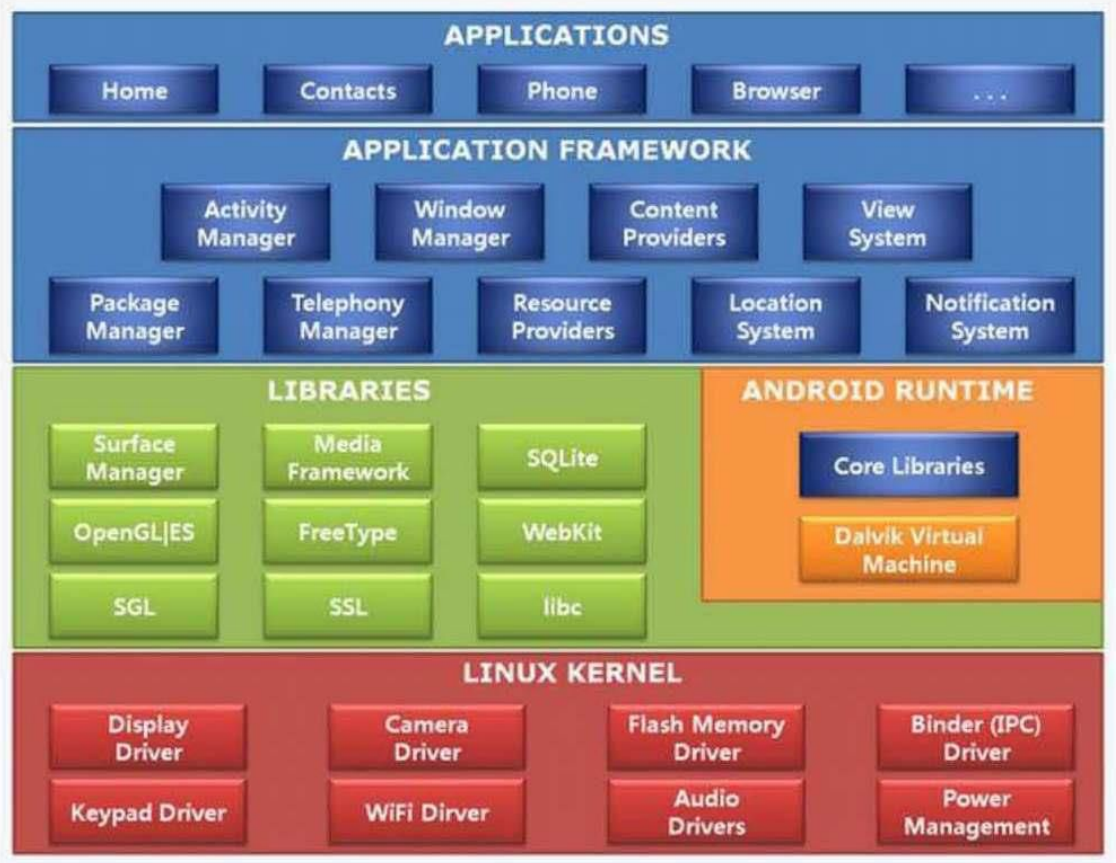
¿Por qué Unity3D?

Se ha decidido hacer uso de la plataforma de desarrollo Unity3D, debido a su gran potencial para la creación de videojuegos y su integración multiplataforma, tanto para PC como para consolas (Xbox, PlayStation) como para dispositivos móviles (IOS, Android), permitiendo desarrollar y exportar los juegos a distintas plataformas a gran velocidad sin necesidad de demasiados cambios, excepto los específicos para cada una.
Inicialmente se trataba de un entorno de desarrollo de juegos 3D, por lo que los desarrolladores de juegos en 2D optaban por otras plataformas como Cocos2D, pero Unity introduce el modo 2D en su versión 4.6, que simplifica el desarrollo en 2D ofreciendo físicas 2D, cámaras 2D, etc..
Otra de sus ventajas es el ser gratuito, tiene sus limitaciones como la aparición obligatoria del logo de Unity en el SplashScreen (pantalla previa al juego), y su versión Pro que es muy cara (1500€), pero para el desarrollo de un juego sencillo no es necesario.
La mayor de sus ventajas es su gran documentación (tanto en inglés como en español http://docs.unity3d.com/Manual/index.html), para distintos lenguajes de programación (C#, Javascript y Boo), hay gran cantidad de tutoriales y el foro de ayuda de Unity te resuelve cualquiera de tus dudas. Y los assets, que son paquetes ya desarrollados que realizan distintas funcionalidades, desde juegos completos hasta el controlador del jugador, que permiten utilizarlos sin necesidad de programarlos uno mismo, ahorrando tiempo (https://www.assetstore.unity3d.com/).
La posibilidad de programar con distintos lenguajes de programación (a la vez), permitiendo usar C#, Javascript o Boo, lenguajes más fáciles que C++, y más eficientes para el desarrollo en Unity, pues permite centrarse sólo en las funcionalidades del juego y no en la gestión de memoria, punteros, etc.. ahorrando gran cantidad de tiempo. Es una gran ventaja de este motor gráfico. Su editor gráfico es intuitivo y de fácil uso, con distintas partes que te permiten trabajar de forma eficiente.
Todas estas ventajas y características, hace a Unity3D una de las plataformas más utilizadas y potentes en la actualidad a la hora de desarrollar videojuegos, y cada vez más en el ámbito específico de Android.

¿Por qué Android?

Android es un sistema operativo de código abierto basado en Linux para dispositivos móviles, tanto smartphones, como tablets, PCs, etc.. El lenguaje de Android es Java y la extensión de las aplicaciones Android se denomina Android Package (.apk).
En su inicio fue desarrollado por Android Inc., pero en 2005 Google la adquirió, y en 2008 lo sacó al mercado, hasta llegar actualmente a ser el mayor sistema operativo en dispositivos móviles.
El funcionamiento de Android se basa en la máquina virtual Dalvik, que es la encargada de compilar el código Java el tiempo de ejecución. La estructura del sistema operativo se basa

en un sistema de capas ejecutado sobre un framework de Java orientado a objetos en la máquina virtual Dalvik:



Junto a las librerías, el núcleo Kernel de Linux constituyen el corazón de Android, entre las librerías se pueden encontrar OpenGL que se encarga de los gráficos o SQLite que gestiona las bases de datos.

Las ventajas de Android, aparte de ser de código libre, pudiendo modificarlo y compartirlo libremente, son:
· El más utilizado actualmente: si se quiere adquirir un dispositivo móvil, la mayor parte de las posibilidades serán Android, debido a su gran potencial y ser el más utilizado hoy en día.
· Precio: comparado con otros sistemas operativos, debido al amplio rango de compañías que disponen de móviles con dicho sistema operativo, se pueden encontrar dispositivos muy baratos con gran relación precio/calidad.
· Google: el ser hijo de Google permite disponer de cantidad de servicios de esta compañía, como GoogleDrive, Gmail, Youtube, etc.. garantizando la compatibilidad con ellas.
· Multitarea: permite ejecutar distintas aplicaciones a la vez, sin necesidad de cerrarlas, permaneciendo en background.
· GooglePlay: a diferencia de la AppleStore de Apple, las aplicaciones de la tienda de Android no tienen por qué ser de pago, al contrario, la mayor parte de las aplicaciones que encontramos se tratan de aplicaciones gratuitas. Además, los desarrolladores deberán pagar solamente una licencia a la hora de registrarse en la tienda, a diferencia de en la AppleStore que deberían pagar cada vez que deseen publicar una aplicación en la tienda.

· Exportación: Unity3D permite exportar los proyectos a Android rápidamente y sin tener que pagar, al contrario que con iOS, que debes disponer de la versión Pro del motor gráfico.
No todo son ventajas, Android también tiene varias desventajas:
· Seguridad: el poder publicar cualquier tipo de aplicación sin pagar hace que se cuelen aplicaciones ofensivas o perjudiciales. Aunque Android dispone de varios controles, siempre hay alguna forma de ignorarlos y conseguir introducirse en otros dispositivos.
· Gasto de batería: el sistema operativo Android consume mucha batería debido a su característica multitarea.
Pese a ello, sigue siendo el más utilizado en la actualidad, y un gran mercado para los iniciados en el desarrollo de aplicaciones, o videojuegos. Es por ello por lo que se ha elegido para el desarrollo del proyecto, a parte del no encontrar ningún juego que cumpla con los objetivos planteados en él, teniendo una buena visión de negocio.

--Otras herramientas utilizadas

-Gimp

Una alternativa gratuita a Photoshop para editar imágenes

-Paint

Utilizada para hacer ediciones mínimas de imágenes

-Visual Studio

Herramienta de programación que es utilizada para crear los scripts

-Monodevelop

Herramienta sustitutiva a visual studio ya que es menos pesada y trabaja con más fluidez a la hora de compilar.

-Google drive

Utilizado para realizar copias de seguridad en la nube.

Google Play Console:

Plataforma web de google donde se publican las tus aplicaciones y se puede monitorizar datos varios relacionados con la aplicación y los clientes como por ejemplo donde se están realizando las descargas, con que dispositivos,...

Admob console:

Plataforma web donde crearemos los anuncios para monetizar nuestra aplicación y monitorizar datos varios.

Draw.io

Plataforma web para desarrollar diagramas utilizado para la memoria

3-DISEÑO E IMPLEMENTACIÓN

-Diagramas y casos de uso

Diagrama de estructuración del proyecto:

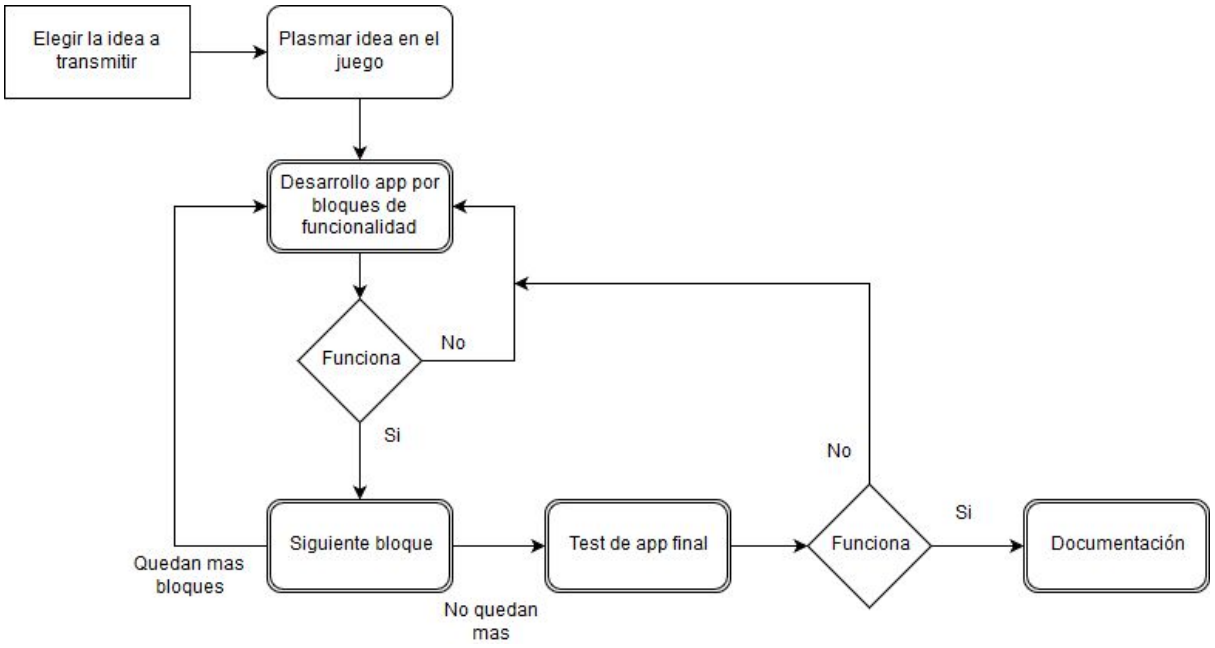
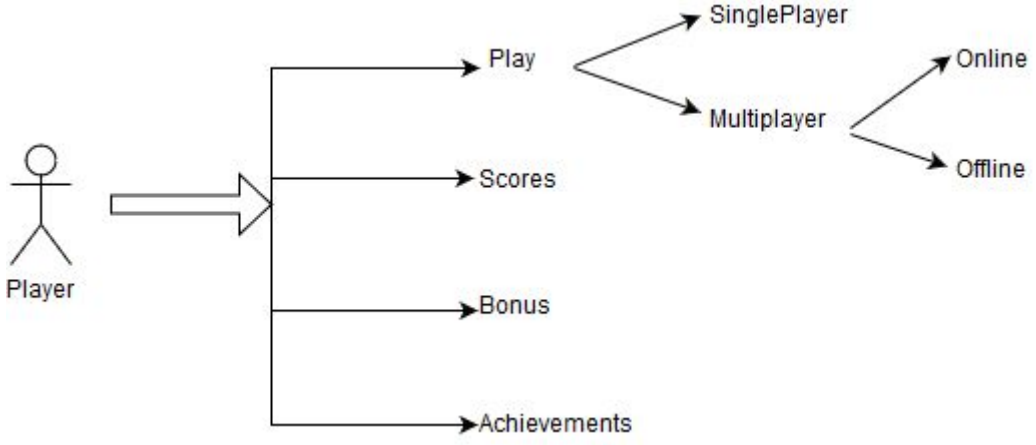
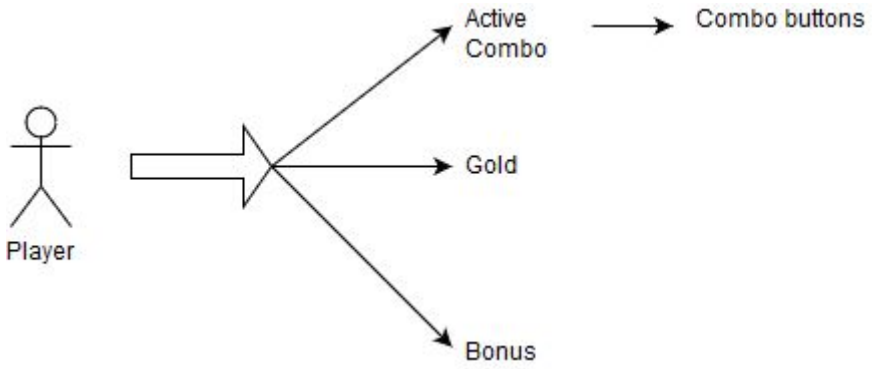


Diagrama de casos de uso:

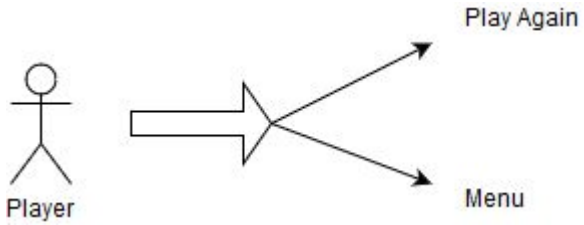
Menú principal:



Escena del minijuego principal:



Escena Game Over:



-Elementos de mayor complejidad:

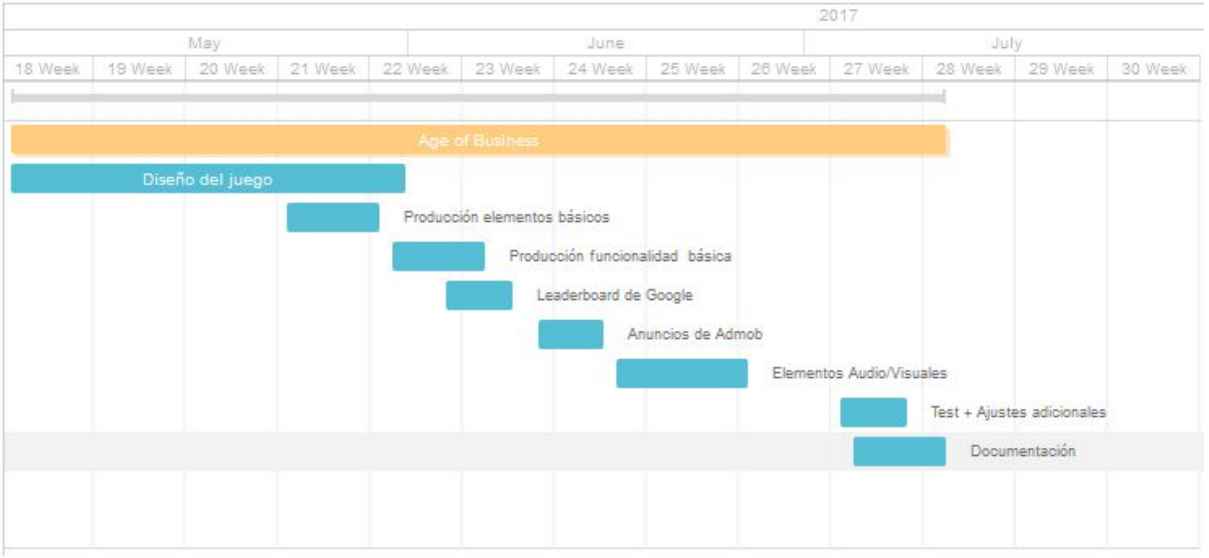
El elemento de mayor complejidad no ha sido la parte de programación, sino ajustar el rango de variables aleatorias para que el jugador experimente la experiencia de juego que se ha deseado plasmar. Para poder realizar los ajustes pertinentes se ha requerido hacer diversas pruebas con distintos tipos de usuarios para observar las distintas reacciones frente a todos los elementos y así poder alcanzar un nivel óptimo de juego.
Luego también ha resultado complicado la opción de multiplayer para realizar partidas online con otros jugadores y la sincronización de ambos juegos.

-Retos tecnológicos

El reto, tecnológicamente hablando, ha sido realizar el proyecto sin utilizar ningún simulador de smartphone por lo cual se tenía que estar seguro de lo que se estaba haciendo para que a la hora de extraer el fichero .apk tuviese el resultado esperado ya que de lo contrario habría supuesto dedicar más horas en el desarrollo estético del juego. Igualmente se debian realizar pruebas para la parte del multiplayer ya que se necesitaban varios dispositivos móviles para dicha acción.

-Workflow

Task name	Start date	Duration day
Total estimate	*01/05/2017*	*71.04*
Age of Business	01/05/2017	71.04
Diseño del juego	01/05/2017	30.00
Producción elementos básicos	22/05/2017	7.00
Producción funcionalidad básica	30/05/2017	7.00
Leaderboard de Google	03/06/2017	5.00
Anuncios de Admob	10/06/2017	5.00
Elementos Audio/Visuales	16/06/2017	10.00
Test + Ajustes adicionales	03/07/2017	5.00
Documentación	04/07/2017	7.00



En este diagrama de Gantt se puede apreciar sin ningún ápice de duda que la tarea más costosa ha sido diseñar el juego la cual conlleva distintas pruebas de ideas diferentes para plasmar la idea deseada, en ella va incluida la investigación de la dopamina del cerebro y acciones en el juego que puedan llegar a liberar dicho elemento.
A partir del diseño de juego el proceso era autoformación, mediante material formativo de acceso público, e implementación de cada tarea.
Al final se facilitó la aplicación a un grupo de usuarios cerrado y se observó el comportamiento de ellos frente a los factores críticos del juego para así poder realizar los ajustes que eran necesarios, como el rango de variables aleatorias.

4-Multiplayer en tiempo real.

Para la implementación de la versión “multiplayer” vamos a utilizar la API multijugador en tiempo real en los servicios de juegos de Google Play para conectar varios jugadores juntos en una sola sesión de juego y transferir mensajes de datos entre jugadores conectados. El uso de la API multijugador en tiempo real ayuda a simplificar el esfuerzo de desarrollo de juegos porque la API gestiona las siguientes tareas en su nombre:

- Gestiona las conexiones de red para crear y mantener una sala multijugador en tiempo real (una construcción virtual que permite la comunicación en red entre varios jugadores en la misma sesión de juego y permite a los jugadores enviar datos directamente entre sí).

- Proporciona una interfaz de usuario de selección de jugador (UI) para invitar a los jugadores a unirse a una habitación, buscar jugadores aleatorios para auto-matching, o una combinación de ambos.

- Almacena la información de estado de los participantes y de la sala en los servidores de servicios de juegos de Google Play durante el ciclo de vida del juego multijugador en tiempo real.

- Envía invitaciones a la sala y actualizaciones a los jugadores. Las notificaciones aparecen en todos los dispositivos en los que el jugador está conectado (a menos que esté deshabilitado).

Antes de diseñar e implementar el juego utilizando la API multijugador en tiempo real, se debe familiarizarse con los siguientes conceptos relacionados con el ciclo de vida típico de un juego multijugador en tiempo real.

Inicialización de la habitación

Internamente, la sala establece una red de malla peer-to-peer entre los participantes donde los clientes pueden comunicarse directamente entre sí, en lugar de a través de los servidores de servicios de juegos de Google Play.

Antes de iniciar una sesión de juego multijugador en tiempo real en un dispositivo, el usuario del dispositivo debe iniciar sesión en su juego. El jugador local (es decir, el usuario que ha iniciado sesión en el dispositivo donde se ejecuta el juego) puede iniciar una sesión de juego multijugador invitando a los amigos a unirse al juego o solicitando que se autocompare.

Si su juego se ejecuta en un dispositivo móvil, la API multijugador en tiempo real proporciona una interfaz de usuario de selección de reproductor predeterminada. La interfaz de usuario permite a los jugadores invitar a sus amigos o seleccionar una serie de oponentes de auto-match. Esto simplifica la codificación de la interfaz de usuario, pero también puede optar por implementar la IU de selección de reproductor.

En función de la selección de jugadores y de los detalles de la configuración de la habitación (introducidos por el jugador local a través de la interfaz de usuario o programados por su juego), los servicios de juegos de Google Play intentarán crear una sala para la sesión en tiempo real de juegos multijugador.

Si la sala se crea correctamente, los servicios de juegos de Google Play notifican a tu juego mediante una devolución de llamada que está registrada en tu juego. El jugador local se unirá automáticamente como participante en la sala.

Configuración de la habitación

Debe especificar el número de jugadores que desea permitir en su habitación. Actualmente, los servicios de juegos de Google Play admiten un máximo de ocho jugadores en un juego multijugador (incluido el jugador que inicia el partido).

Opcionalmente, es posible que desee asegurarse de que sólo los jugadores que estén interesados en un tipo específico de variante de juego se ajusten automáticamente a la habitación. Por ejemplo, en un juego de carreras, puedes combinar automáticamente a los jugadores que sólo quieren jugar un mapa específico de carreras o nivel de dificultad. Las variantes se pueden utilizar para combinar automáticamente a los jugadores que estén interesados en diferentes estilos de juego, como la jugabilidad de jugador contra jugador (PvP) o "capturar la bandera". Si hay diferentes versiones de la aplicación, también puede utilizar variantes para asegurarse de que sólo los jugadores que se encuentran en versiones compatibles se coinciden automáticamente.

Participantes

Cuando los jugadores inician un juego multijugador, pueden optar por invitar a personas específicas o tener los servicios de juegos de Google Play automáticamente seleccionando a otros participantes de forma aleatoria mediante la combinación automática. También pueden solicitar una mezcla de los dos (por ejemplo, un jugador específico de sus círculos, y dos jugadores auto-emparejados).
Si los jugadores eligen invitar a una persona específica a un juego, dependiendo de la configuración de su cuenta, la interfaz de usuario que utilizan y los ámbitos del juego, pueden tener diferentes personas disponibles para elegir.
En general, inviters siempre será capaz de invitar a los opositores recientes, y siempre será capaz de automatch jugadores.
En la interfaz de usuario predeterminada suministrada por Google, también verán la opción "Reproductores cercanos".
En la interfaz de usuario predeterminada suministrada por Google, los jugadores con perfiles detectables también verán amigos en círculo, independientemente de si tienen el ámbito PLUS_LOGIN.
Utilizando el método getInvititablePlayers (), los juegos que han solicitado el ámbito PLUS_LOGIN obtendrán amigos en círculo en los valores de retorno, independientemente de si tienen un perfil detectable.

Tenga en cuenta que en algunos casos, los jugadores pueden no tener amigos inviables por defecto.
Para invitar a otros jugadores, el invitador puede usar la función de búsqueda en la interfaz de usuario de selección de reproductor.

Auto-matching

Un participante de Auto-matching no tiene que ser un contacto en los círculos del jugador local o cualquier otra conexión. Cuando se hace un auto-matching, los servicios de juegos de Google Play simplemente buscan otros participantes en ese momento que también están iniciando un juego y solicitando un auto-matching. Los participantes de coincidencia automática no reciben notificaciones para unirse a un juego; Desde su perspectiva, parece que están iniciando individualmente el juego.
En los juegos multijugador en tiempo real, los participantes de coincidencia automática aparecerán como jugadores anónimos entre sí (incluso si se conocen entre sí).

“Connected set”

A medida que los jugadores se unen o salen de la sala, los servicios de juegos de Google Play intentan activamente crear una malla de conexiones peer-to-peer entre todos los participantes. Esto forma un grupo de participantes conectados en la sala, donde cada jugador en el conjunto conectado está completamente conectado a los otros jugadores en el conjunto. El conjunto conectado puede consistir en un subconjunto de todos los jugadores que se han unido a la sala. Si algún jugador se desconecta de otro jugador en el conjunto conectado, el conjunto se reduce a los jugadores restantes que todavía están totalmente conectados. Depende de su juego determinar cómo proceder si esto sucede.

Cuando todos los participantes en una sala de tiempo real están completamente conectados, los servicios de juegos de Google Play lo notifican mediante una devolución de llamada. Su juego puede enviar mensajes a los participantes que están conectados a su habitación. Esto se describe más adelante en Enviar datos de juego.

Envío de datos de juego

Puede utilizar los servicios de juegos de Google Play para transmitir datos a los participantes en una sala o permitir a los participantes intercambiar mensajes entre sí. Los mensajes de datos se pueden enviar mediante un protocolo de mensajería confiable o poco confiable proporcionado por los servicios de juegos de Google Play.

Mensajes confiables(Reliable messaging). Con una mensajería confiable, la entrega de datos, la integridad y el pedido están garantizados. Puede elegir que se le notifique el estado de entrega mediante una devolución de llamada. La mensajería confiable es adecuada para enviar datos no sensibles al tiempo. También puede utilizar mensajería confiable para enviar grandes conjuntos de datos donde los datos se pueden dividir en segmentos más pequeños, enviados a través de la red y, a continuación, reensamblados por el cliente receptor. La mensajería confiable puede tener latencia alta. El tamaño máximo

de un mensaje confiable que puede enviar es de 1400 bytes.

Mensajes poco fiables(Unreliable messaging). El cliente del juego envía los datos una sola vez ('fire-and-forget') sin garantía de entrega de datos o datos que lleguen en orden. Sin embargo, la integridad está garantizada, por lo que no es necesario agregar una suma de comprobación. La mensajería no confiable tiene baja latencia y es adecuada para enviar datos sensibles al tiempo. Su aplicación es responsable de garantizar que el juego se comporta correctamente si los mensajes se descartan en transmisión o se reciben fuera de servicio. El tamaño máximo para un mensaje no fiable que puede enviar es 1168 bytes.

Cierre de la habitación

El juego es responsable de salir de la habitación (es decir, desconectar la habitación de los servidores de servicios de juegos de Google Play) cuando un participante se desconecta del juego o sale de la parte multiplayer. Su juego también debe manejar el escenario en el que todos los participantes excepto el jugador local han salido de la habitación. Cuando esto sucede, su juego debe desconectar al jugador local de la habitación inmediatamente.

La habitación se considera "cerrada" cuando todos los participantes han salido de la habitación. En este punto, su juego debe apagar cualquier juego en curso, y asegúrese de guardar los datos del juego de manera adecuada. Para obtener más información sobre cómo guardar datos de juegos en Google Play, consulta Juegos guardados.
Vamos a ver ahora un poco de código como ejemplo de “sign-in” para google play games services.

```
using System.Collections;
using GooglePlayGames;
using GooglePlayGames.BasicApi;
using UnityEngine;

public class MainMenuEvents : MonoBehaviour {

    // ... other code here...
    public void Start() {
       GameObject startButton = GameObject.Find("startButton");
       EventSystem.current.firstSelectedGameObject = startButton;


        //  ADD THIS CODE BETWEEN THESE COMMENTS

        // Create client configuration
        PlayGamesClientConfiguration config = new
            PlayGamesClientConfiguration.Builder()
            .Build();

        // Enable debugging output (recommended)
        PlayGamesPlatform.DebugLogEnabled = true;

        // Initialize and activate the platform
        PlayGamesPlatform.InitializeInstance(config);
        PlayGamesPlatform.Activate();
       // END THE CODE TO PASTE INTO START
    }

    // ...
}
```

Aquí tendríamos un trocito de código muy sencillo para conectarse como cliente y para hacer uso de las funciones específicas de la API tendremos que declarar estas nuevas clases mediante este código.

```
using GooglePlayGames.BasicApi;
using GooglePlayGames;
```

Podemos crear una función llamada SignInCallback, la cual es llamada una vez el proceso de autentificación ha finalizado:

```
public void SignInCallback(bool success) {
        if (success) {
            Debug.Log("(Lollygagger) Signed in!");

            // Change sign-in button text
            signInButtonText.text = "Sign out";

            // Show the user's name
            authStatus.text = "Signed in as: " + Social.localUser.userName;
        } else {
            Debug.Log("(Lollygagger) Sign-in failed...");

            // Show failure message
            signInButtonText.text = "Sign in";
            authStatus.text = "Sign-in failed";
        }
    }
```

5-Competencia y pautas a seguir(Problemática del mercado)

-Seleccionar competencia

Debemos saber a qué sector nos vamos a enfrentar como desarrolladores de juegos. Siendo un desarrollador "freelance" se ha de saber que nuestro poder económico no es el mismo que el de una compañía ya desarrollada, por lo tanto no podemos pagar por una buena campaña de marketing, así que se ha de trabajar en el factor de captación de jugadores. Dado que al ser un programador en solitario tampoco disponemos del tiempo ni conocimientos que un grupo con personal especializado en distintas áreas. Todo esto nos

lleva a optar por la opción de los juegos que son tremendamente simples pero altamente adictivos.

Para esto vamos a basarnos en una serie de pautas y variables y así podremos optar a crear un juego que pueda tener un breve pero gran impacto en la sociedad "Gamer", pero esto no quiere decir que siguiendo estos consejo se vaya a lograr el éxito, sino que tenemos más posibilidades de ello.

-Dopamina en el juego

La dopamina es un elemento químico que libera el cerebro y nos proporciona un sentimiento de satisfacción. El alcohol, la nicotina, etc, son sustancias que producen dopamina y nos crean una adicción. Pero no se libera solo mediante sustancias externas sino también por acciones que realizamos nosotros como ser humano. Vamos a ver un ejemplo el cual se ha extrapolado al proyecto:
Imagínese una situación donde usted tiene una reunión o una cita importante y va justo de tiempo, empieza a recoger los últimos detalles que le faltan y de repente se da cuenta de que no encuentra las llaves. Se empieza a alterar buscandolas por todos los rincones de la casa, el tiempo pasa y usted se estresa. A medida que el tiempo pasa usted realiza cosas más ilógicas como buscar dichas llaves en lugares donde jamas estarian. Llega un momento en el que usted, lleno de frustración y estrés, encuentra las llaves y en ese momento tiene una sensación de descanso donde se le va todo el estrés y frustración, se ha liberado dopamina en el cerebro.

Se ha extrapolado este ejemplo a uno de los juegos, plasmando todos los estados anímicos de dicho ejemplo. La idea es causar una gran sensación de estrés y presión a medida que va pasando el tiempo, por eso se han añadido elementos que aparecen según un rango de variables aleatorias, así el jugador no podrá predecir en partidas futuras cuando podrá realizar sus jugadas.
El jugador empieza pulsando el botón para ganar oro y empieza su fase de estrés, una vez puede activar el combo(la posibilidad de activación es en un momento aleatorio) empieza una segunda fase de estrés y la primera de presión, el jugador ha de estar concentrado en pulsar la combinación correcta lo mas rapido posible para asi conseguir su dinero extra, si el jugador se equivoca en un botón, no le dará la puntuación extra hasta no completar la combinación siguiente. En los últimos segundos de la partida aparece el último elemento, el cual está delimitado por un marco de tiempo de 3 segundos. Al quedar tan poco tiempo, el jugador entra en una fase de estrés y presión máxima, en el momento en el que el jugador toma una decisión respecto al último elemento puede causar 2 efectos ya que el juego termina en ese momento: frustración o satisfacción, y todo este proceso delimitado por un marco de tiempo de 20 segundos donde el jugador tendrá una gran experiencia de juego a pesar de su sencillez.

Un videojuego no triunfa ni hace que nuestro cerebro quiera repetir solo por una cosa, sino que hay varios factores a tener en cuenta, y todos ellos son los que se sincronizarán para

que nuestro cerebro siga segregando dopamina continuamente. He ahí el secreto de la gamificación.

- Sensaciones:

Los videojuegos, igual que las películas pero en forma de gráficos cada vez más realistas, combinan las sensaciones visuales, táctiles y auditivas provocando así la activación simultánea de varias regiones cerebrales. Evidentemente, a más activación, más estimulación, más “adicción”, y en última instancia, mayor aprendizaje.

- Seguridad:

Como he mencionado anteriormente, los videojuegos combinan el riesgo de ganar o perder con la seguridad de encontrarnos en un mundo virtual. Cualquier error puede rectificarse y cualquier prueba repetirse sin sufrir daños, lo que nos da vía libre a tomar más y mayores riesgos (con su consecuente aumento de dopamina cerebral).

- Ambiente amigable

Los juegos hacen gala de poner en práctica un ambiente sustentado por cinco factores clave en el comportamiento humano:

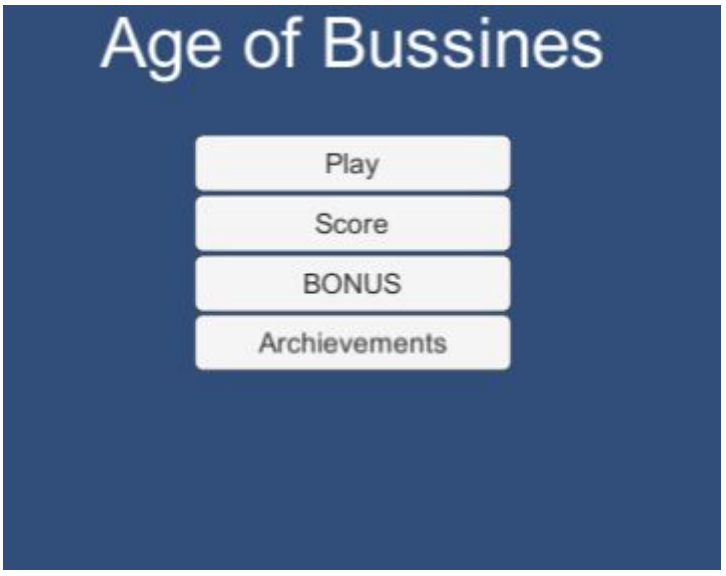
1. *Colaboración:* Los seres humanos somos sociables por naturaleza.
2. *Estado:* Necesitamos competir y nos gusta que nos reconozcan nuestros logros.
3. *Novedad:* Continuamente nos hacen centrar nuestra atención en el videojuego.
4. *Certeza:* La información se da en tiempo real y de forma continua.
5. *Equidad:* Se intenta evitar la percepción de injusticia o parcialidad.

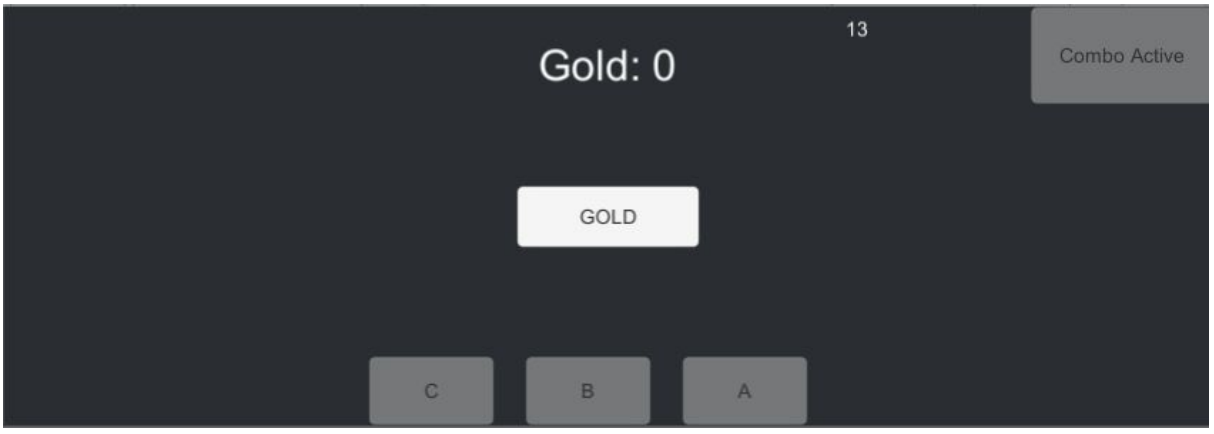
Como veis, existen no solo uno, sino muchos factores a tener en cuenta para la relación entre cerebro y videojuegos. La típica frase de “quien prueba, repite” no es casualidad en este ámbito, porque, de hecho, el objetivo de los videojuegos es justamente que nuestro cerebro y sus picos de dopamina quieran repetir.

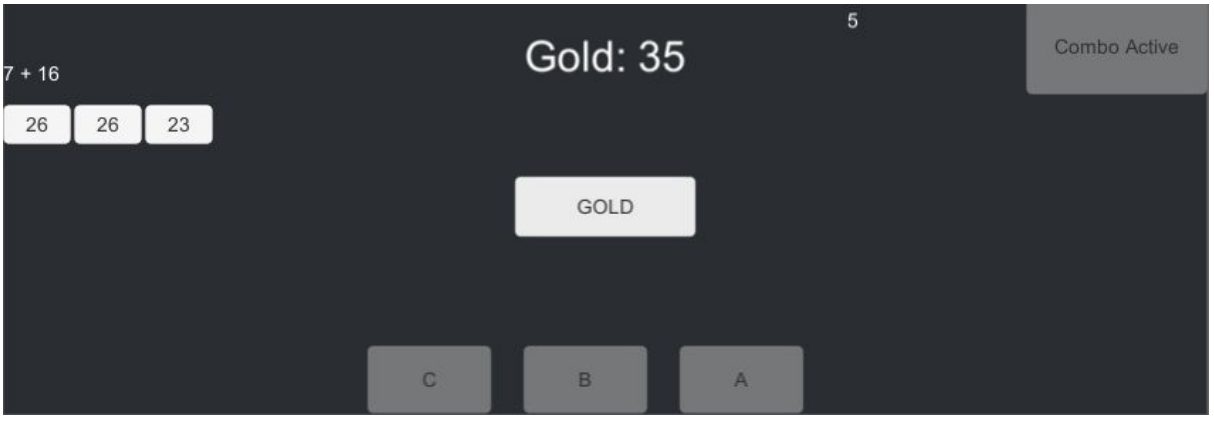
6-Conclusiones

-Capturas de aplicación:

Aquí tenemos unas imágenes del minijuego principal.

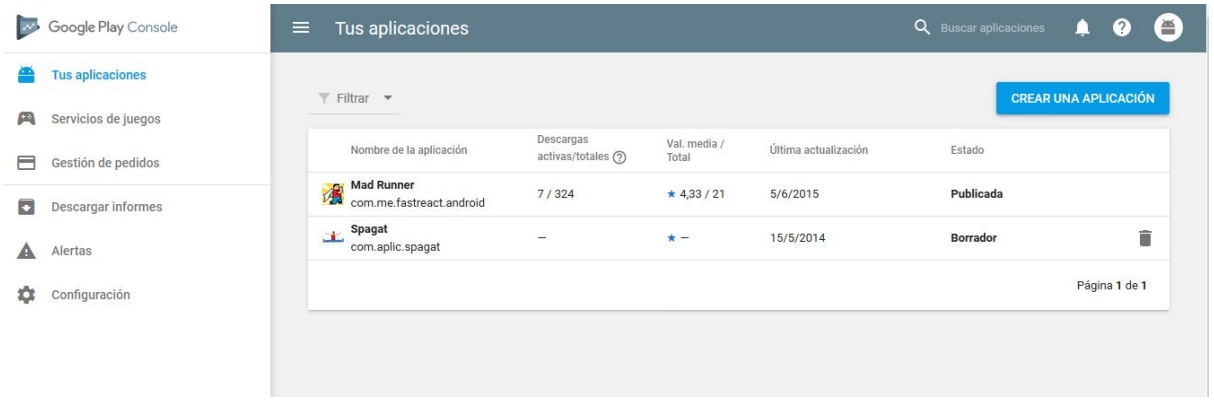

Age of Bussines
Play
Score
BONUS
Archievements


13
Gold: 0
Combo Active
GOLD
C
B
A

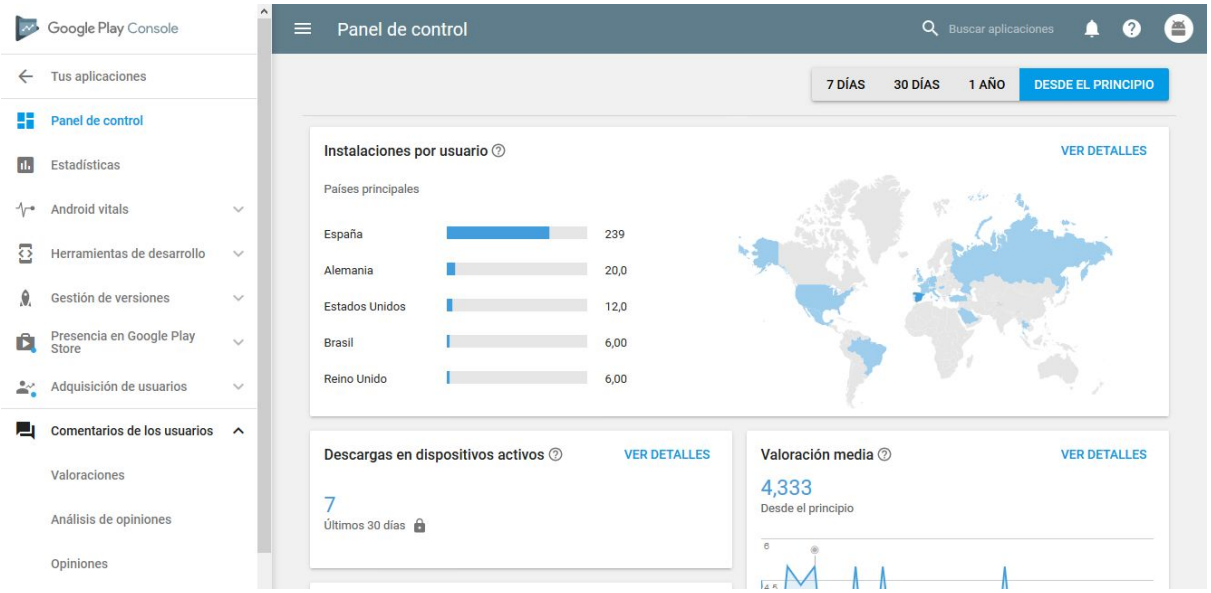

5
Gold: 35
Combo Active
7 + 16
26
26
23
GOLD
C
B
A

-Publicación en store

La aplicación será publicada en unos días en la Play Store, correspondiente a los dispositivos Android, donde el creador podrá hacer un seguimiento de diversas características como por ejemplo en qué país se están realizando las descargas, cuantas descargas están activas, etc, y los jugadores podrán descargar el videojuego.



Aquí se puede ver ya una aplicación publicada y otra en estado de borrador. Un punto curioso es que podemos poner la aplicación en estado alfa y dar acceso solo a las personas que nosotros queramos para que ellas mismas sean las primeras en probar la aplicación y aportar su opinión sobre el proyecto.



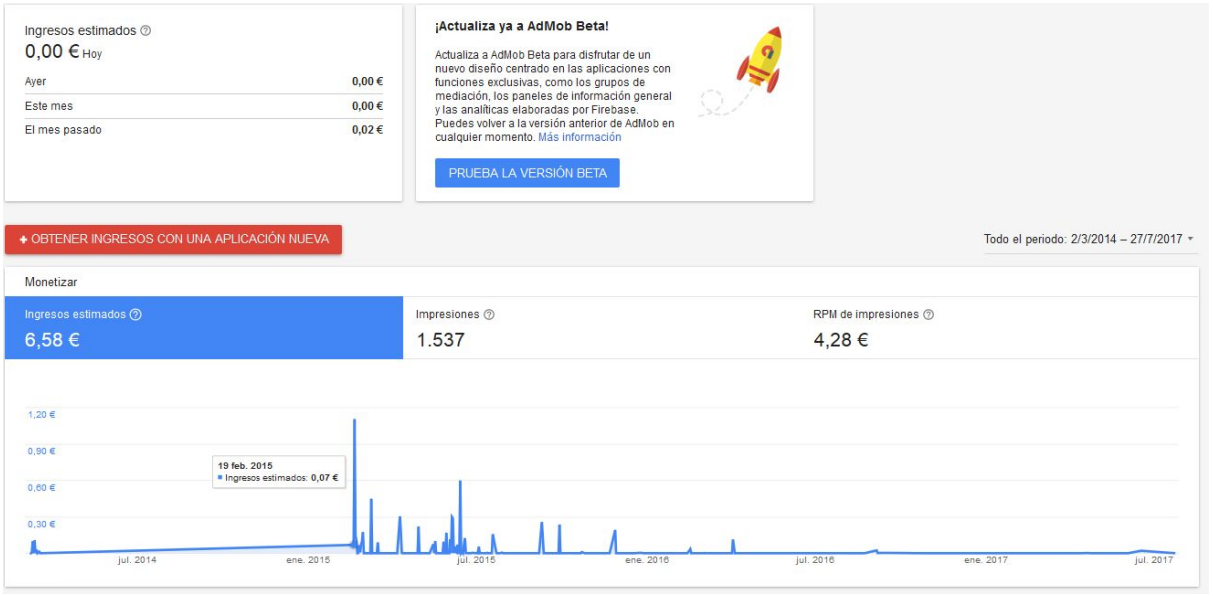
Aquí tenemos otra imagen donde nos muestra datos interesantes y útiles sobre la aplicación que tengamos publicada en el marco de tiempo deseado.

-Monetización

Existen diversos métodos para monetizar los juegos en dispositivos móviles, ya sea mediante objetos de pago dentro del juego, publicidad dentro del juego, descarga de pago, etc.
El método utilizado en este juego ha sido por publicidad y para ello hemos utilizado la plataforma web Admob, que pertenece a google, y esta nos permite crear todo tipo de anuncios para posteriormente implementarlos en nuestra aplicación.
Vamos a crear anuncios de tipo “video reward” lo que significa que cada vez que se haga una visualización de este, se recibirá una recompensa por ello.

Mediante el uso de Admob podemos realizar diversos análisis:

Aquí podemos ver la pantalla principal de la plataforma admob.



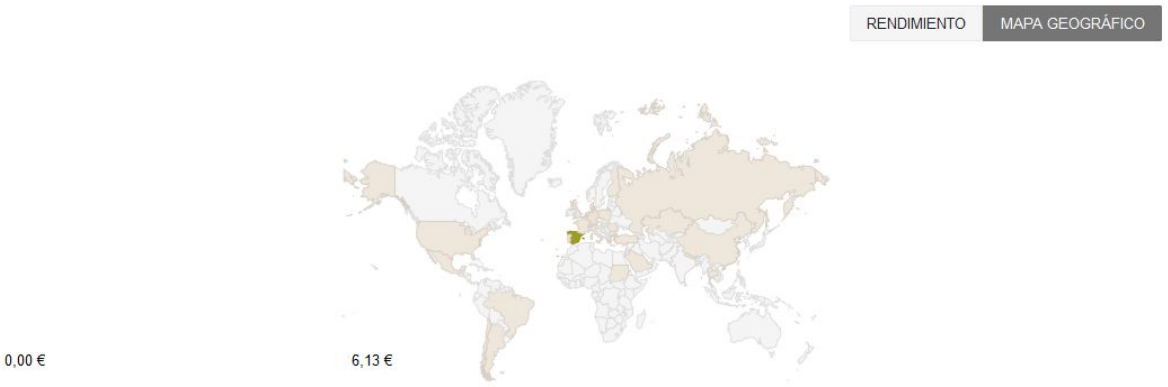
Con unas aplicaciones reales que nos sirven como ejemplo.

Aplicación	Ingresos estimados	Impresiones	RPM de impresiones
Age of Business Android | Enlazar su aplicación	–	–	–
Fast React Android | Enlazar su aplicación	–	–	–
Mad Runner GRATIS | Android	6,32 €	1.393	4,54 €
Spagat Android | Enlazar su aplicación	0,26 €	144	1,81 €

Mostrar filas: 15 1-4 de 4

País	Solicitudes de la Red de AdMob	Solicitudes de anuncios respondidas	Porcentaje de coincidencias	Impresiones	Clics	CTR de impresiones	RPM de impresiones	Ingresos estimados
Alemania	132	132	100,00%	71	1	1,41%	3,38 €	0,24 €
Arabia Saudí	9	9	100,00%	2	0	0,00%	0,00 €	0,00 €
Argentina	4	4	100,00%	1	0	0,00%	0,00 €	0,00 €
Brasil	32	32	100,00%	24	0	0,00%	0,02 €	0,00 €
Bulgaria	3	3	100,00%	1	0	0,00%	0,00 €	0,00 €
Bélgica	47	47	100,00%	23	0	0,00%	1,23 €	0,03 €
Chile	5	5	100,00%	3	0	0,00%	0,00 €	0,00 €
China	16	16	100,00%	8	0	0,00%	0,00 €	0,00 €
Dinamarca	7	7	100,00%	0	0	–	–	0,00 €
España	2.897	2.896	99,97%	1.280	59	4,61%	4,79 €	6,13 €
Estados Unidos	61	61	100,00%	24	0	0,00%	0,45 €	0,01 €
Finlandia	3	3	100,00%	0	0	–	–	0,00 €
Francia	31	31	100,00%	11	2	18,18%	0,13 €	0,00 €

Como se puede apreciar podemos obtener bastante informacion solo por los anuncios de la plataforma de google.



Además de la localización geográfica categorizado por países a nivel mundial de donde se está accediendo al apartado de los anuncios del juego.

Tipo de anuncio	Solicitudes de la Red de AdMob	Solicitudes de anuncios respondidas	Porcentaje de coincidencias	Impresiones	Clics	CTR de impresiones	RPM de impresiones	Ingresos estimados
(Solicitudes de anuncios no coincidentes)	1	0	0,00%	0	0	–	–	0,00 €
Animación	564	564	100,00%	234	6	2,56%	2,92 €	0,68 €
Imagen	1.312	1.312	100,00%	583	26	4,46%	2,46 €	1,44 €
Rich media	244	244	100,00%	100	5	5,00%	1,48 €	0,15 €
Texto	910	910	100,00%	452	21	4,65%	5,89 €	2,66 €
Video	406	406	100,00%	168	6	3,57%	9,82 €	1,65 €
Total	3.437	3.436	99,97%	1.537	64	4,16%	4,28 €	6,58 €

Mostrar filas: 15 **1-6** de **6**

Esta es la información que nos muestra esta plataforma de google y gracias a ella podemos tomar futuras decisiones para el diseño de nuestro juego.

-Trabajo de futuro

Después de ver las reacciones de las personas que han realizado los testeos se ha valorado la opción de añadir más minijuegos a este proyecto para así expandir la experiencia de diversión de los jugadores, además de su publicación en multiplataforma.

BIBLIOGRAFÍA

Aquí tenemos la plataforma web de google play donde nos da toda la información acerca de la publicación de aplicaciones
https://play.google.com/apps/publish/

Aquí el link de la plataforma web para la monetización del juego
https://www.google.es/admob/

El enlace del plugin para añadir google play games services a unity
https://github.com/playgameservices/play-games-plugin-for-unity

Pequeño ejemplo implementando google play services
https://codelabs.developers.google.com/codelabs/playservices_unity/index.html?index=..%2F..%2Findex#0

Para el estudio de la dopamina se ha investigado las conferencias de Simon Sinek
https://www.ted.com/speakers/simon_sinek

Otro enlace sobre información de la gamificación
http://omicrono.elespanol.com/2015/01/por-que-tu-cerebro-le-encantan-los-videojuegos/

Una de las mayores fuentes de información sobre el desarrollo de juegos
https://unity3d.com/es/learn/

Otro fuente para el desarrollo
https://forum.unity3d.com/

Canal de youtube oficial de unity, útil para estar al dia
https://www.youtube.com/user/Unity3D

La biblia de unity
https://docs.unity3d.com/ScriptReference/

Tutoriales sobre scripting en C#
https://unity3d.com/es/learn/tutorials/s/scripting

ANEXO

GDD (Game Design Document)

-Descripción general

Este es un juego de minijuegos donde tenemos varias opciones de juego: multiplayer(online y offline) y singleplayer. Se basa en batirte contra amigos o contra ti mismo para obtener la mejor puntuación en un rango de tiempo muy corto.
Tenemos como primer juego offline y online uno basado en pulsar un botón lo más rápido posible para ir consiguiendo oro, con este oro podremos activar la posibilidad de realizar un combo el cual nos dará una puntuación extra.
Como segundo juego tenemos un panel lleno de botones donde se iluminara un solo botón y una vez pulsado este nos otorga oro y se apagará, posteriormente se encenderá otro botón y así sucesivamente. En el caso de que pulsemos un botón incorrecto este nos restará puntuación. Todo esto dentro de un rango de tiempo de 20 segundos siendo el juego de tipo multiplayer y podremos batirnos contra otro jugador en el mismo dispositivo(pantalla dividida) o en su versión online donde podremos retar a algún amigo nuestro.

-Plataforma principal

El juego está publicado enfocado a dispositivos móviles, concretamente aquellos con Android como sistema operativo de base ya que el juego está publicado en la Play Store.

-Monetización

El juego se monetiza mediante anuncios, el cual nos dara un bono en el juego. Gracias al servicio de implementación de anuncios podremos monitorizar el comportamiento de los usuarios frente a la publicidad.

-Estilo audiovisual

La estética del juego es básicamente botones, principalmente se utilizan recursos audiovisuales minimalistas provenientes de terceros.

-Menú principal

Una vez inicializamos el juego se cargará como primera imagen el logo de unity3d, ya que hemos utilizado la versión “free” es algo que no podemos modificar.

Posteriormente se nos mostrará el menú principal del juego con las siguientes opciones:

1. Jugar. El cual nos dará 2 opciones más:
    a. SinglePlayer. Donde jugaremos contra nosotros mismos.
    b. Multiplayer. Aquí volveremos a tener dos opciones más, si queremos jugar 2 personas en un dispositivo(Offline) o si queremos batirnos contra otros jugadores que esten en linea (Online).
2. Logros. Aquí nos aparecerá todo lo relacionado con los logros, lo cual está implementado por google play games services.
3. Scores. En esta sección nos aparecerá una tabla con todos los mejores resultados de los jugadores online en el respectivo minijuego.
4. Bonus. Nos aparecerá un anuncio de tipo “video reward” lo que quiere decir que cuando lo veamos obtendremos una recompensa.

-Jugando

Una vez seleccionemos nuestro modo de juego, singleplayer o multiplayer, elegiremos el juego deseado y en el caso de ser multiplayer-online, podremos elegir a nuestro contrincante. Una vez el juego llega a su fin nos llevará a una pantalla donde nos mostrará los resultados y cual de los dos jugadores ha sido el ganador.

En el primer juego empezamos pulsando el botón de oro rápidamente, según un rango de variable aleatoria dependiente del oro, se habilita el botón de activar combo, con su pulsación se activarán 3 botones los cuales habrá que pulsar en una secuencia específica (A-A-B-B-C) la cual nos dará una puntuación extra pero si nos equivocamos no nos penalizará. En los últimos segundos de la partida aparecerá una opción bonus, la cual consta de una operación matemática sencilla y tendremos 3 posibles resultados, en el caso de acertar nos dará 70 puntos de oro y si nos equivocamos nos restará 50 puntos, pero en el caso de que no pulsamos ninguna opción no nos afectara en nuestra puntuación lo más mínimo.

En el segundo juego aparece un panel con diversos botones, algunos de los cuales se iluminarán y se obtendrá oro mediante la pulsación de dichos botones, la elección de un botón erróneo restará puntuación al jugador. El juego tiene una duración de 20 segundos, marco de tiempo en el cual gana el jugador con mejor puntuación. Al finalizar el juego saldrá en la pantalla la puntuación de cada jugador y cuál ha sido el vencedor ya sea en el caso de online o offline.

Como tercer juego tenemos un simple minijuego de carrera donde podemos ver cómo se sincronizan las variables online mediante el uso de google play services. No es un juego centrado como diversión, sino como comprensión.

-UI ”User interface”
En los dos juegos principales se presentan los mismos elementos a parte de los interactuables por la mecánica del juego, tenemos elementos que nos muestran la puntuación, el tiempo y el nombre de nuestro contrincante para el caso de los juegos multiplayer.

-Logros

Se han implementado una serie de logros sencillos

- Juega 100 partidas
- Juega por primera vez solo
- Juega por primera vez contra un amigo
- Bate por primera vez a tu rival en multiplayer.

CÓDIGO

Vamos a añadir unos trozos de código para tener una idea de cómo se programa con el lenguaje de programación C#

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Click : MonoBehaviour {
    /*
    // Use this for initialization
    void Start () {
    }
    // Update is called once per frame
    void Update () {
    }
    */

    public float myCoolTimer;
    public UnityEngine.UI.Text count;
    public UnityEngine.UI.Button button;

    /// ////////////////////////////////////

    public UnityEngine.UI.Text gpc;
    public UnityEngine.UI.Text goldDisplay;
    public static float gold = 0.00f;
    public int goldperclick = 1;
    public int gameOverScene = 1;

    private Bonus bono;
    private int tiempo = 0;

    void Start(){
        gold = 0.0f;
    }

    void FixedUpdate () { // para que el tiempo sea el mismo siempre para todos
        goldDisplay.text = "Gold: " + gold;
        gpc.text = goldperclick + "gold/click";

        myCoolTimer -= Time.deltaTime;
        count.text = myCoolTimer.ToString ("f0");
```

```
40          count.text = myCoolTimer.ToString ("f0");
41          tiempo = (int) myCoolTimer;
42          dataH.tiempo = tiempo;
43          //print (dataH.tiempo);
44
45          if (myCoolTimer <= 0.0f) {
46              //Call function when time runs out
47              dataH.contadorOro = gold;
48              count.text = gold.ToString ();
49              print (" " + dataH.contadorOro);
50              changeScene (2);
51
52
53          }
54          /*
55          if (myCoolTimer <= 0.0f && myCoolTimer >= -0.1f) {
56              button.enabled = !button.enabled;
57          }
58          */
59
60
61      }
62
63      public void Clicked(){
64          gold = gold + goldperclick;
65
66      }
67
68      public static void cPlus(){
69          gold = gold + 5.0f;
70      }
71
72      public void changeScene(int scene){
73          SceneManager.LoadScene (scene);
74      }
75
76      public static void bonusGood(){
77          gold = gold + 70.0f;
78      }
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class cCombo : MonoBehaviour {


    public static string combo = "";
    public UnityEngine.UI.Button button;

    // Use this for initialization
    void Start () {
        //Para poner los 3 botones en false
        //button.interactable = false;
    }

    void FixedUpdate () {

    }
    /*
    public void DisableButton(){
        button.interactable = false;
    }
    */

    public void setLetra(string letra){
        if (activeCombo.activado == true) {

            combo = combo + letra;
            //oro.Clicked ();
            print (combo);
            if (letra.Equals ("C")) {
                checkCombo ();
            }
        }
    }

    // CHECK THE COMBO AND WIPE THE STRING FOR THE NEW COMBO
    public void checkCombo(){
        if (combo.Equals ("AABBC")) {
            Click.cPlus ();
            combo = "";
            print ("YESSSS");
        } else {
            combo = "";
            print ("NIEEEEET");
        }
    }

}

```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Bonus : MonoBehaviour {
6
7     public UnityEngine.UI.Text master;
8
9
10    public UnityEngine.UI.Button button1;
11    public UnityEngine.UI.Text texto1;
12
13    public UnityEngine.UI.Button button2;
14    public UnityEngine.UI.Text texto2;
15
16    public UnityEngine.UI.Button button3;
17    public UnityEngine.UI.Text texto3;
18
19    int tiempo = 0, opA, opB, opS, aleatorio;
20
21
22    // Use this for initialization
23    void Start () {
24        master.text = "";
25
26        button1.image.enabled = false;
27        button2.image.enabled = false;
28        button3.image.enabled = false;
29
30        button1.interactable = false;
31        button2.interactable = false;
32        button3.interactable = false;
33
34        texto1.text = "";
35        texto2.text = "";
36        texto3.text = "";
37
38        opA = Random.Range (5, 21);
39        opB = Random.Range (5, 21);
40
```

```
41          opS = opA + opB;
42
43          aleatorio = Random.Range (1, 4);
44
45
46
47
48      }
49
50      int filtro = 1;
51      // Update is called once per frame
52      void FixedUpdate () {
53          //print (tiempo);
54          tiempo = dataH.tiempo;
55          if (tiempo == 5 && filtro == 1) {
56              activarB ();
57              filtro++;
58          }
59
60          if (tiempo == 2) {
61              desactivarB ();
62          }
63
64      }
65
66      public void activarB(){
67          master.text = opA + " + " + opB;
68
69          button1.image.enabled = true;
70          button2.image.enabled = true;
71          button3.image.enabled = true;
72
73          button1.interactable = true;
74          button2.interactable = true;
75          button3.interactable = true;
76
77
78          switch (aleatorio) {
79          case 1:
```

```
        case 1:
            print ("caso 1");
            texto1.text = "" + opS;
            texto2.text = "" + (opS - 3);
            texto3.text = "" + (opS + 3);
            break;

        case 2:
            print ("case 2");
            texto1.text = ""+ (opS + 3);
            texto2.text = "" + opS;
            texto3.text = "" + (opS - 3);

            break;

        case 3:
            print ("case 3");
            texto1.text = "" + (opS + 3);
            texto2.text = "" + (opS + 3);
            texto3.text = "" + opS;

            break;

        }



    }

    public void desactivarB(){
        master.text = "";

        texto1.text = "";
        texto2.text = "";
        texto3.text = "";
```

```
116
117         button1.image.enabled = false;
118         button2.image.enabled = false;
119         button3.image.enabled = false;
120
121         button1.interactable = false;
122         button2.interactable = false;
123         button3.interactable = false;
124
125     }
126
127     public void checkA(){
128         int A = int.Parse (texto1.text);
129         if (A == opS) {
130             print ("Solucion correcta");
131             Click.bonusGood ();
132         } else {
133             print ("Solucion erronea");
134             Click.bonusBad ();
135         }
136         desactivarB ();
137
138     }
139
140     public void checkB(){
141         int B = int.Parse (texto2.text);
142         if (B == opS) {
143             print ("Solucion correcta");
144             Click.bonusGood ();
145         } else {
146             print ("Solucion erronea");
147             Click.bonusBad ();
148         }
149         desactivarB ();
150
151     }
```

```
153     public void checkC(){
154         int C = int.Parse (texto3.text);
155         if (C == opS) {
156             print ("Solucion correcta");
157             Click.bonusGood ();
158         } else {
159             print ("Solucion erronea");
160             Click.bonusBad ();
161         }
162         desactivarB ();
163     }
164
165
166 }
```