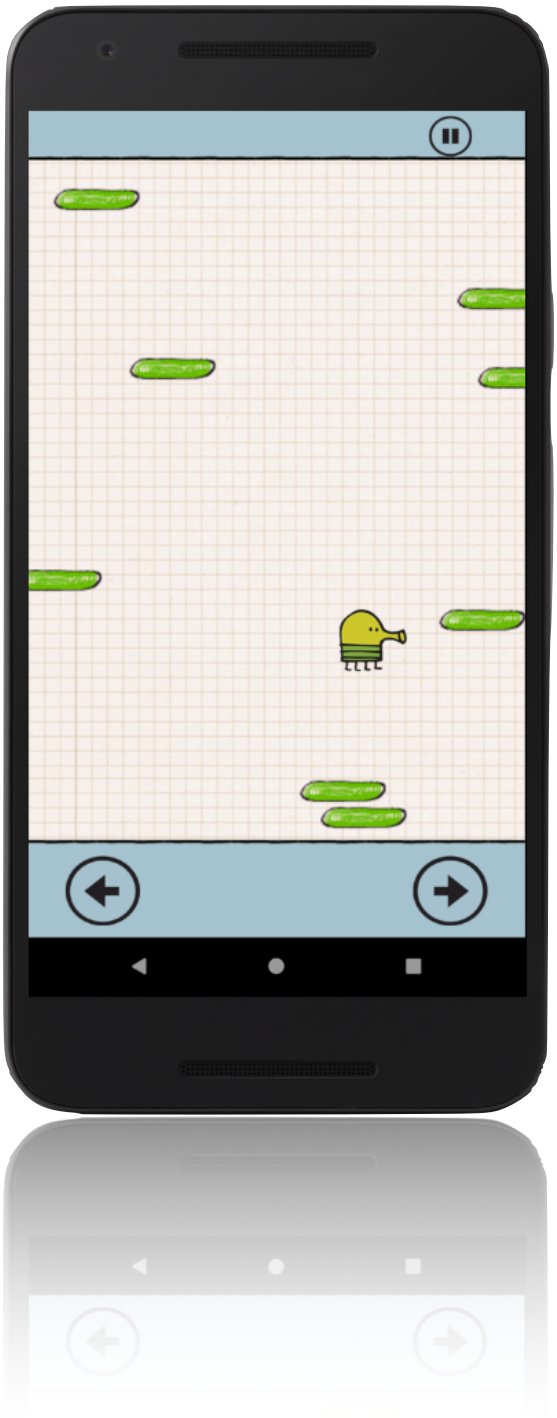
# Doodle Jump

*Desarrollo para Dispositivos Móviles*



Nicolás Tapia Sanz

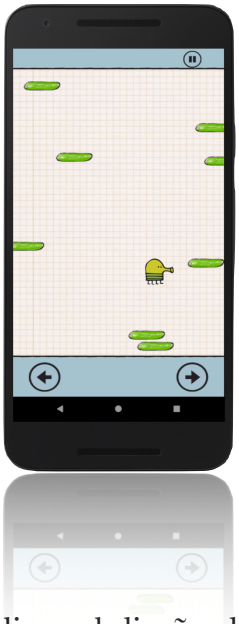
3.3. - DVJU

ESNE 2017/2018

# Doodle Jump

*Desarrollo para Dispositivos Móviles*

## Introducción



A continuación se procede a explicar el diseño del juego Doodle Jump a nivel de desarrollo, funcionamiento y planteamiento desde su pantalla de inicio hasta su Game Over.

## Objetivo

El objetivo de esta práctica es la implementación de un videojuego de temática libre usando cualquier librería de desarrollo de videojuegos para dispositivos multiplataforma basada en el lenguaje C++ e integrada con el entorno de desarrollo Android Studio.

El juego que se ha desarrollado en este caso es el conocido Doodle Jump, un juego arcade en el que el jugador controla a un pequeño personaje que rebota sobre unas plataformas. El jugador debe tratar de botar sobre dichas plataformas para alcanzar la mayor altura posible. Para ello, puede mover al personaje de un lado al otro de la pantalla. El objetivo secundario es evitar caerse por el borde inferior de la pantalla.

## Fechas

Comienzo: 22/01/2018
Entrega: 11/02/2018

## Gameplay

El juego tiene 6 pantallas: pantalla de Intro, el Menú Inicial, la pantalla de Ayuda, el juego en sí mismo, el menú de Pausa y el menú de Game Over.

Al arrancar el juego lo primero que aparece es la imagen de Intro a pantalla completa con un fundido de entrada y, tras un segundo, desaparece con otro fundido.

Tras esto, el juego aparece en el menú inicial. El menú se compone de un fondo, el titulo del juego y tres botones:

- Jugar: conduce a una partida.
- Ayuda: conduce a una pantalla donde se muestran los controles y los créditos del juego.
- Salir: sale del juego.

Al comenzar una nueva partida se carga una nueva escena. Mientras se cargan los Assets se mostrará una imagen de "Loading…". Una vez cargado todo, comenzará automáticamente el juego. Lo primero que verá el jugador será al personaje controlable lanzado hacia arriba.

El objetivo del jugador es rebotar sobre las distintas plataformas que aparecen en pantalla para alcanzar la mayor altura posible. Dichas plataformas aparecen por la parte superior de la pantalla de manera aleatoria y desaparecen por la parte inferior. Para moverse de izquierda a derecha el jugador podrá hacer uso de los botones que hay en la zona inferior de la pantalla.

Si el jugador cae por la parte inferior, la partida acabará. Al suceder esto, se mostrará una pantalla de Game Over que contendrá un nuevo fondo, un título de Game Over, un botón para jugar de nuevo y un botón para regresar al menú inicial.

Durante la partida, en la esquina superior derecha, se encontrará el botón de pausa. Si el jugador lo pulsara, aparecerá una pantalla de pausa muy similar a la de Game Over pero con la diferencia de que el título indica al jugador que está en Pausa y que un botón te permitirá continuar con la partida que estaba jugando.

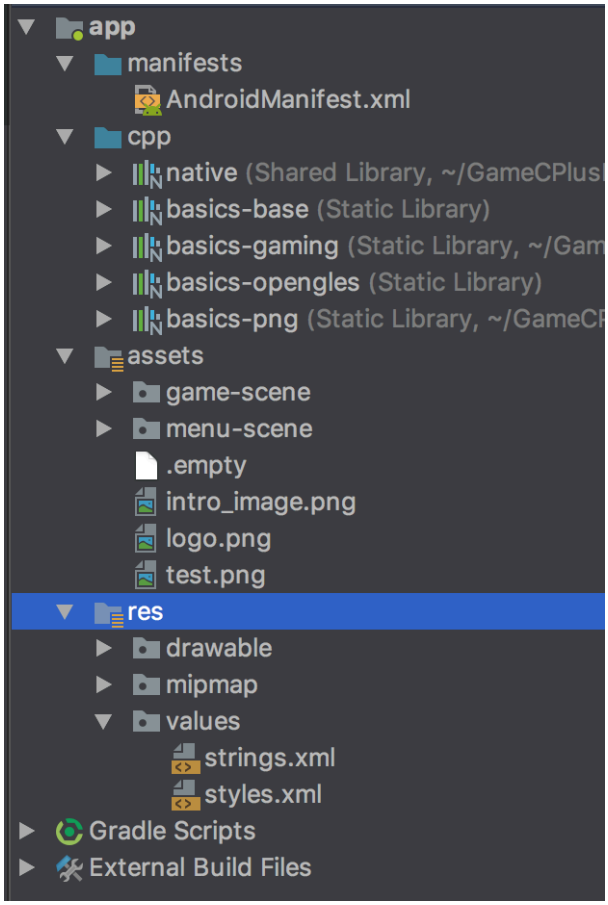
La pantalla de Ayuda indica los controles y explica muy básicamente el gameplay al jugador. También en esta pantalla se encuentran los créditos del juego. Así mismo, hay un botón para retroceder hasta el menú inicial.

# Descripción Técnica

A continuación se explican los componentes de la app y su estructura:

## Estructura

Esta es la estructura del juego. Dado que es un juego para Android respeta la estructura clásica de las apps nativas para este sistema operativo. La carpeta "manifests" contiene el manifiesto del juego llamado AndroidManifest.xml.




La carpeta cpp contiene las distintas librerías de la app. Me voy a centrar en explicar el contenido de "native", debido a que el resto son de OpenGL, lectura de Sprites, etc. "Native" almacena los códigos que yo he creado, a partir de lo dado por el profesor, que contienen la lógica de funcionamiento del juego.

La carpeta "assets" contiene todas las imágenes, sprites, que utilizará el juego. La carpeta "res>values" contiene el archivo strings.xml donde se encuentra el título del juego.

## Intro_Scene.cpp/.hpp

Esta es la escena que mostrará la imagen de Intro al iniciarse el juego. Arranca con un fundido inicial, se muestra la imagen durante un segundo y desaparece con otro fundido. Este código como tal no lo he creado yo, dado que el profesor proporcionó el código ya creado. Lo único que realicé realmente fue la inclusión de una imagen propia para mostrar durante la Intro.

## Menu_Scene.cpp/.hpp

Esta escena se encarga de mostrar el Menú, como su propio nombre indica, y también de mostrar la pantalla de ayuda/créditos. El funcionamiento es simple. Tiene 4 estados: Loading, para cuando carga los sprites a mostrar; Running, para cuando se está ejecutando el menú normal; Helping, para cuando se encuentra en la pantalla de ayuda/créditos; y Error, para cuando se ha producido un error.

Dependiendo del estado en el que se encuentre, realizará unas acciones de Update u otras, así como renderizará unos sprites u otros. El método render y el método update tienen cada uno un switch para decidir a que otros métodos llamar dependiendo del estado. En el caso de estar en el modo Running, se llamará a los métodos update_menu() y render_menu(). En otro caso, si estuviera en el estado de Helping se llamaría a update_helping() y a render_helping(). Los métodos load_textures() y create_sprites() se encargan de

cargar las texturas y de crear los sprites y asignarlos a sus variables respectivamente.

## Game_Scene.cpp/.hpp

Esta escena es el juego en sí mismo. Contiene toda la carga de sprites, gameplay, pantalla de Pausa y pantalla de Game Over. Tiene cinco estados: Loading, cuando está cargando los sprites; Running, cuando está ejecutando el juego como tal; Paused, cuando el juego está pausado; Over, cuando la partida ha acabado; y Error, cuando se ha producido un error interno del juego.

Al igual que el código anterior, dependiendo del estado de juego en que se encuentre, el update y el render llamarán a métodos distintos para mostrar y/o comprobar unas cosas u otras:

•Loading: render_loading() y load_textures().
•Running: update_user() y render_playfield().
•Paused: update_menu() y render_pause().
•Over: update_menu() y render_over().

Al comienzo aparece una pantalla de Loading. Esta, probablemente, no se podrá a penas ver ya que la carga de sprites se hace prácticamente al instante.

Una vez que se hayan cargado todos los sprites del Atlas y se hayan asignado correctamente, comenzarán a generar las plataformas. Dichas plataformas son 10 en total, sin embargo puede que dentro de la pantalla se vean menos. Esto es debido a que algunas se generan por encima del borde superior, donde se encuentra el botón de pausa. Según las plataformas pasen el borde inferior, donde están los botones de dirección, se recolocan en una parte superior no visible todavía para el usuario para dar la sensación de generación procedural. Lo realicé de esta manera ya que consideraba más optimo conservar las plataformas en vez de destruirlas y crearlas constantemente.

Durante el pausado o el game over no se mueve ningún tipo de físicas, aunque los sprites conservan sus posiciones pero no se renderizan.

Las colisiones con las plataformas pueden notarse algo distintas a las del juego original. Esto es debido a que, debido al posicionamiento aleatorio de las plataformas en la escena, puede darse el caso de no alcanzar las plataformas en algunos instantes. La colisión se calcula a partir de la mitad inferior de cada plataforma. De esta manera, es más fácil que se alcancen y al rebotar sobre cada una la sensación visual que da es que rebota sobre la plataforma y no en el borde de esta.

El menú de Pausa y el de Game Over son básicamente el mismo pero cambiando los sprites de Resume por el de Play Again y los títulos. De esta manera puedo re-aprovechar código sin complicarme demasiado. Dependiendo del estado en el que se encuentre el juego, al pulsar dicho botón se vuelve a la partida o se recarga una nueva Game_Scene.

## Problemas Encontrados

Me he encontrado con numerosos problemas a la hora de desarrollar el juego. Nombraré algunos de los que más quebraderos de cabeza me han dado y algunas de las soluciones que he realizado para resolverlos.

Comencé programando el juego y con ello la pantalla de carga. No pude solucionar el hecho de que no se mostrara el Loading, ya que los ejemplos que teníamos se realizaban con Texturas o Sprites y no con Atlas, que es lo que yo utilizo. Esto dio lugar a numerosos crasheos y a que no cargara la escena de juego. Finalmente, decidí que se mostrara como pudiera aunque el jugador no fuera capaz de verlo. En caso de que tuviera que cargar muchos más sprites probablemente haya más posibilidades de que pueda verlo.

La carga de Sprites mediante Atlas también fue algo lioso, ya que no tenía muchos conocimientos sobre como se usaba la herramienta para crearlos. Sin embargo, y tras varios intentos, conseguí crear correctamente los Spritesheets y cargarlos. También me pasó que no entendía bien desde donde se pintaban los Sprites y tardé en darme cuenta de que era desde su punto central, al igual que en el momento de renderizarlos.

Una vez conseguí crear todos los sprites, el hecho de hacer botones también me dio algún quebradero de cabeza, ya que no detectaba demasiado bien donde tocaba y donde se mostraba el Sprite, lo que me llevó bastante tiempo descubrir bien el error: era que estaba pintando mal los sprites.

Una vez solucionado, ya pude mover al personaje de un lado a otro. Cuando tuve esto, llego el momento de aplicarle gravedad y velocidad vertical. Para solucionar esto recurrí a mis apuntes de física del año pasado, cosa que me vino fantástica, ya que nunca había programado algo con gravedad fuera de Unity y sin usar un Rigidbody.

La colisión con las plataformas fue medianamente sencilla, ya que era un código que ya teníamos hecho con el pong. La generación de estas fue más o menos sencilla, aunque siguen apareciendo realmente un poco como quieren. La recolección de estas tras desaparecer de la pantalla fue fácil.

Una vez hecho el gameplay, el siguiente problema con el que me encontré fue el de crear las pantallas de pausa y game over, lo que me obligó a re-estructurar bastante parte de mi código.

Cuando tuve hecho esto, la parte de realizar el menú fue bastante fácil, aunque me encontré con un problema que a día de hoy sigo sin poder resolver y es que, al darle al botón de Back en la pantalla de ayuda, se intercambia entre esta y la pantalla de menú continuamente en cada frame. Aun revisando todo el código, no he conseguido dar con una solución en el momento de escribir estas palabras.

Tampoco he podido dar una funcionalidad al botón de Exit del menú. Así mismo, probé a compilar el juego en mi móvil y la generación de plataformas era totalmente distinta a como se mostraba en el emulador. Tampoco he podido dar con una solución a este problema.