
TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Ángel Luis González Martín

Departamento de Informática

# Manual Técnico del proyecto

# SARSCOVKILL

# 1 INTRODUCCIÓN

Como alumno de segundo curso de Diseño de Aplicaciones Multiplataforma y para el proyecto de fin de grado se realiza el diseño e implementación de una aplicación multiplataforma.

Esta aplicación es un videojuego con el nombre de SarsCovKill.

La elección del título viene determinada por aprovechar la actual “popularidad” del Covid y los beneficios que tiene en lo que a distribuir la aplicación se refiere.

## 1.1 Género, estilo y ambientación

El género del videojuego puede catalogarse como un shooter de scroll horizontal o como un (Shoot 'em up).

Su estilo es de tipo arcade clásico en dos dimensiones y su ambientación viene determinada por un futuro oscuro y apocalíptico en el que las pandemias asolan a una sociedad altamente tecnológica.

## 1.2 Concepto del juego

El concepto de SarsCovKill nace de combinar la pasión por los videojuegos clásicos tipo arcade con un tema de actualidad como es el Covid19 y la pandemia que estamos sufriendo desde Enero del 2020.

## 1.3 Objetivo

El objetivo principal cuando juegas a SarsCovKill es controlar un nanobot el cual tendrá que destruir o evitar ser destruido por hordas de virus los cuales tratarán de quedar sin energía al Nanobot con sus impactos, para eliminarlos debemos alcanzarlos lanzado cadenas de adn mediante el generador que incorpora el Nanobot, al eliminar los objetivos ganaremos puntos, los cuales ampliarán la puntuación de nuestro marcador y en consecuencia aumentarán nuestro nivel en el videojuego.La dificultad irá aumentando a la par del nivel en el que estemos en el videojuego y los enemigos cada vez serán más numerosos.

## 1.4 Público al que se dirige

SarsCovKill es un videojuego el cual puede ser interesante para amantes de los videojuegos clásicos como los de las máquinas arcade de los 80s/90s y los nuevos jugadores que quieran poner a prueba sus reflejos con un juego medianamente simple.

Además para muchos puede ser muy reconfortante el eliminar virus por la situación que atravesamos actualmente.

## 1.5 Plataforma

Su plataforma principal es desktop (windows x86-64) además de Android y en futuras versiones la idea es poder compilarlo también para ios.

## 1.6 Logotipo


## 1.7 Autor, Ciclo Formativo, Tutor y Centro Educativo

Autor: Ángel Luis González Martín

Ciclo Formativo: Desarrollo de aplicaciones multiplataforma

Tutor: Eloy García Álvarez

Centro Educativo: I.E.S. Albarregas

# 2 OBJETIVOS

Los principales objetivos que me he propuesto para el diseño de SarsCovKill pueden enumerarse en los siguientes apartados:

- Personales
- De desarrollo
- Adaptable (Responsive)
- Multiplataforma
- Sistema de puntuación y niveles
- Bonificadores en la partida
- Sistemas de control de calidad y controladores alternativos
- Posibilidad de pausar y guardar partida (persistencia de información)
- Enemigos con movimientos, propiedades y spawn (aparecer) aleatorios y deferentes.
- Sonidos y músicas
- Diseño

## 2.1 Personales

Respeto a los objetivos personales que he tenido en la creación de este proyecto de fin de ciclo, el principal es reforzar los conocimientos adquiridos en el módulo de grado superior Desarrollo de Aplicaciones Multiplataformas.

La elección de crear un videojuego multiplataforma viene determinada tras evaluar otras opciones (quizás más asequibles), pero considero importante explorar la programación de videojuegos debido principalmente a que en el ciclo formativo no se ha tocado nada referente al diseño de videojuegos.

## 2.2 De desarrollo

Respecto al desarrollo de la aplicación y teniendo como hándicap el que desarrollar un videojuego en tan poco tiempo es una tarea dificil, se intentarán complir unos

objetivos minimos de funcionalidades y calidad en el diseño que enumero a continuación.

## 2.3 Adaptable (Responsive)

El videojuego será diseñado para poder ejecutarse de forma responsive para que pueda adaptarse a cualquier tamaño de pantalla, con esto se tratará de hacerlo compatible con la mallor cantidad de dispositivos y casos de uso.

## 2.4 Multiplataforma

Aunque la plataforma principal a la que va enfocado es Desktop (Windows (windows x86-64). Además tengo como objetivo poder compilar para otras plataformas como Android, debido al auge de los juegos en plataformas móviles que usan este sistema operativo.

## 2.5 Sistema de puntuación y niveles

En un videojuego el objetivo principal es el entretenimiento, se buscará que no resulte frustrante ni aburrido, por este motivo tendrá un sistema de niveles para que la dificultad se incremente a la par de los mismos, además el conseguir una puntuación o nivel siempre es un aliciente para seguir jugando.

## 2.6 Bonificadores en la partida

Para evitar la monotonía se implementan ítems bonificadores en la partida, estos aparecerán de forma random y pueden recargar la energía (vida) del nanobot, bonificadores de puntuación (subirás un nivel), bonificación de velocidad que se pueden acumular, etc.

## 2.7 Sistemas de control de calidad y controladores alternativos

Para una buena experiencia jugando con un videojuego es muy importante que los controles estén correctamente calibrados para un movimiento de calidad y puedan ofrecer diferentes alternativas de control, el videojuego se podrá controlar de forma táctil, con solo el mouse y con teclado. Además que siempre que puedas usar teclado se podrán conectar dispositivos de juego que emulen la pulsación de las teclas requeridas mediante software externo.

El juego dispone de un touchpad “analógico” para el movimiento que puede utilizarse de forma táctil o mediante el movimiento de mouse y brinda la posibilidad de mover al nanobot a diferentes velocidades.

## 2.8 Posibilidad de pausar y guardar partida

En un videojuego de calidad mínima es importante el poder pausar la partida en cualquier momento y poder conservar el avance del jugador en el videojuego cuando salga o pierda la partida.

Para ello se implementará un sistema de guardado para las partidas (persistencia), el jugador tiene un sistema de slots de guardado (cinco en total) por lo que solo tendrá que seleccionar uno, elegir si quiere continuar con la partida que tiene almacenada o bien resetearlo para empezar de nuevo, ademas que da la posibilidad que varios usuarios puedan realizar guardados en diferentes slots.

El sistema de guardado se realiza de forma automática al salir de la partida (muy cómodo) y al conservar su avance en el juego, cuando quiera volver a jugar aparecerá en el nivel donde lo dejó, manteniendo la puntuación el nivel y la cantidad de enemigos abatidos.

## 2.9 Enemigos diferentes

Los enemigos tendrán diferentes movimientos, aparecerán en diferentes posiciones, tendrán diferente velocidad, provocarán diferente daño incluso habrá algunos que serán inmunes a las cadenas de adn que lanze nuestro nanobot porque serán de origen no biológico (nanovirus tecnológicos).

## 2.10 Sonidos y músicas

Se implementaran sonidos y músicas para las diferentes escenas del juego y acciones del mismo.

## 2.11 Diseño

En un videojuego se busca tanto su jugabilidad como el atractivo sensorial. Para ello se prestará especial atención al arte gráfico y sonoro para conseguir una atmósfera de juego óptima.

# 3 TECNOLOGÍAS INVOLUCRADAS

Para el desarrollo de SarsCovKill se han utilizado diferentes tecnologías entre las que se pueden destacar:

- Sistema operativo
- Lenguaje de programación
- Framework
- Herramientas de Diseño
- Sistemas de construcción de proyecto y compilación
- Herramienta para el control de versiones y repositorios online GitHub
- Herramienta para generar diagramas UML de las clases del proyecto

## 3.1 Sistema operativo

El sistema operativo utilizado ha sido Windows10 Home version 10.0.19041.867

## 3.2 Lenguaje de programación

El proyecto se ha realizado enteramente con Java SE1.7 (7). La decisión de Java frente a otros lenguajes viene determinada por los siguientes puntos:

- **Es el lenguaje que mejor conozco**, es de los más usados y se encuentra mucha información del mismo.
- **Su código es robusto y no excesivamente complicado**. Maneja la memoria de forma automática, tiene ventajas como el que la máquina virtual realice comprobaciones de integridad para evitar errores además de que gestiona la liberación de memoria de forma autónoma que es ideal para la eficiencia de la misma.
- **Es multiplataforma**. Puede funcionar en casi cualquier dispositivo que tenga implementada su máquina virtual (JVM).
- **Es orientada a objetos**

## 3.3 Framework

Como framework principal o librería para el desarrollo de videojuegos se hace uso de LibGDX.


LibGDX es un framework libre y gratuito. Está escrito en Java principalmente, además de utilizarlo como lenguaje de desarrollo.

Link de la página oficial

Link de descarga gdx-setup

El decantarme por LibGDX frente a otros frameworks viene determinado por que es un motor de videojuegos que engloba características muy interesantes como las que expongo a continuación.

- Tiene APIs de alto nivel tanto para trabajar en 2D como en 3D incorporando sus respectivos motores de físicas (Box2D (2D) y Bullet (3D)).
- Es multiplataforma. Por lo que está especialmente indicado para el desarrollo de un proyecto como SarsCovKill. Ofrece soporte para realizar juegos en PC, Android, IOS, Html5, etc.
- Tiene un API para manejar el almacenamiento para la gestión de recursos del juego así como el acceso a los mismos.
- Se puede trabajar con LibGDX en los principales IDEs como Eclipse, NetBeans, Android Studio, IntelliJ, etc.
- Tiene API para gestionar los diferentes inputs necesarios a la hora de desarrollar una aplicación multiplataforma (Teclado, Ratón, Táctiles, Acelerómetro, etc.).
- Provee un fácil acceso a APIs gráficas para el desarrollo en 2D y ·3D como OpenGL 2.0 y OpenGL 3.0.
- Al ser libre concentra a un alto número de desarrolladores con la ventaja de haber gran cantidad de soporte. El desarrollo como framework de LibGDX es continuo al igual que sus actualizaciones.

## 3.4 Herramientas de Diseño

Para el desarrollo de la aplicación se han utilizado diferentes herramientas entre las que puedo destacar las siguientes:

## 3.5 IDE de desarrollo

El IDE utilizado como plataforma de desarrollo es Eclipse IDE for Enterprise Java and Web Developers.

Version: 2021-03 (4.19.0)

Build id: 20210312-0638

Link de descargas e información

Para preparar el entorno de desarrollo eclipse han tenido que incorporar los siguientes plugins:

- Android SDK Link de descarga SDK
- El plugin para Eclipse Android Development Tools for Eclipse
- El plugin para Eclipse Eclipse Integration Gradle
- El plugin para Eclipse ObjectAid UML Diagram

## 3.6 Herramientas para diseño y creación de texturas

Para el diseño de texturas crear texturas animadas, transformar formatos y otras acciones necesarias con imágenes he utilizado principalmente las siguientes:

- Paint 3D (El editor de imágenes de windows 10)
- Online-Convert.com: Conversor de formatos para imágenes online
- peko-stet.com: Combinador de imágenes para realizar texturas para animaciones
- Hiero: Herramienta para crear fuentes de texto en mapa de bits.
  Con esta herramienta se pueden crear fuentes de texto personalizadas para poder añadir texto al proyecto LibGDX, Estas fuentes se crean como BitMap Fonts para poder utilizarlas como si fueran una textura.

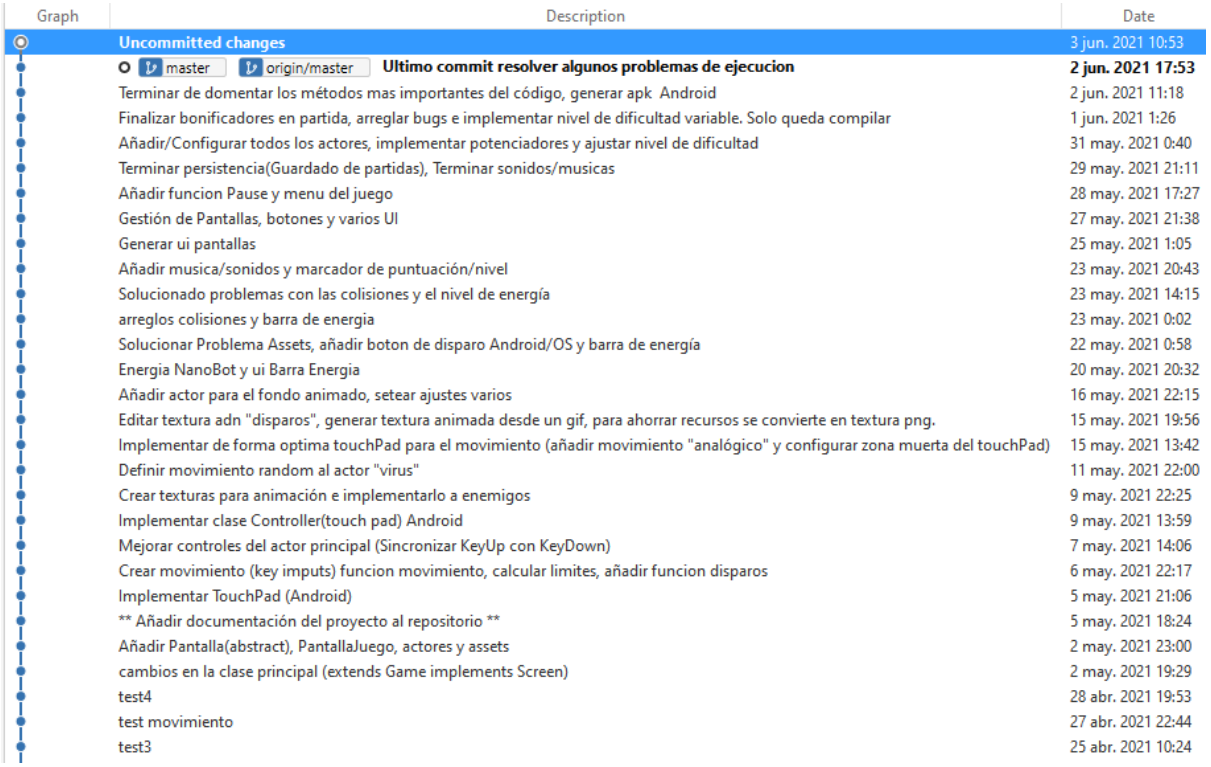
## 3.7 Sistemas de construcción de proyecto y compilación

- Para construir el proyecto con LibGDX es necesario el uso de LibGDX project setup

- Gradle incorporado tanto en el Ide como en LibGDX se ha usado para la creación del proyecto como la compilación del mismo para diferentes versiones

## 3.8 Control de versiones y repositorios online GitHub

Para el control del sistema de versiones he utilizado el programa SourceTree 3.4.4
Los diferentes commits que he realizado hasta el momento con este programa para el desarrollo de SarsCovKill son los que se indican en la siguiente imagen.

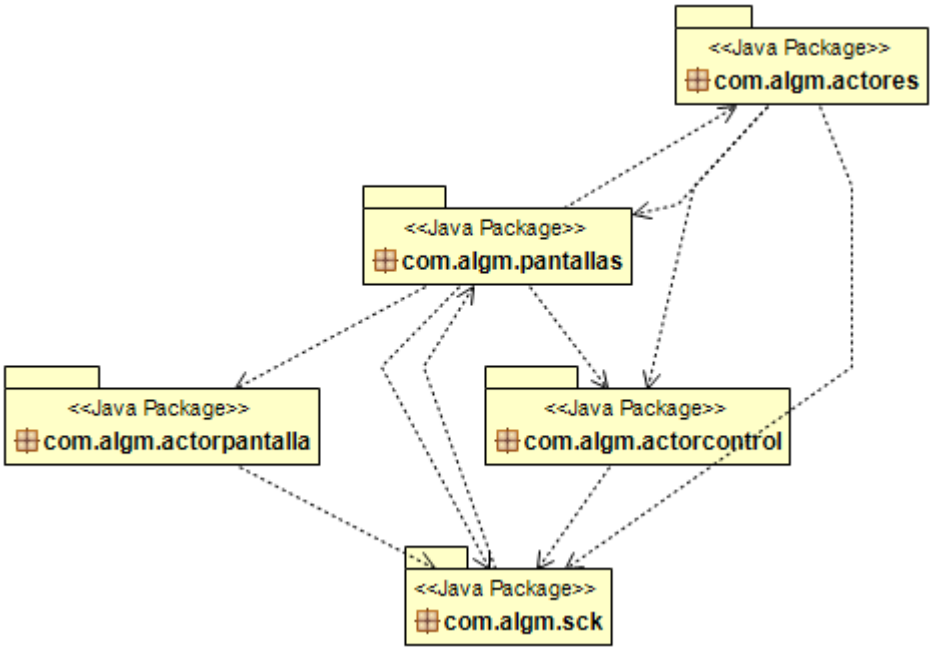


| Graph | Description | Date |
|---|---|---|
| | **Uncommitted changes** | 3 jun. 2021 10:53 |
| | master origin/master **Ultimo commit resolver algunos problemas de ejecucion** | **2 jun. 2021 17:53** |
| | Terminar de domentar los métodos mas importantes del código, generar apk Android | 2 jun. 2021 11:18 |
| | Finalizar bonificadores en partida, arreglar bugs e implementar nivel de dificultad variable. Solo queda compilar | 1 jun. 2021 1:26 |
| | Añadir/Configurar todos los actores, implementar potenciadores y ajustar nivel de dificultad | 31 may. 2021 0:40 |
| | Terminar persistencia(Guardado de partidas), Terminar sonidos/musicas | 29 may. 2021 21:11 |
| | Añadir funcion Pause y menu del juego | 28 may. 2021 17:27 |
| | Gestión de Pantallas, botones y varios UI | 27 may. 2021 21:38 |
| | Generar ui pantallas | 25 may. 2021 1:05 |
| | Añadir musica/sonidos y marcador de puntuación/nivel | 23 may. 2021 20:43 |
| | Solucionado problemas con las colisiones y el nivel de energía | 23 may. 2021 14:15 |
| | arreglos colisiones y barra de energia | 23 may. 2021 0:02 |
| | Solucionar Problema Assets, añadir boton de disparo Android/OS y barra de energía | 22 may. 2021 0:58 |
| | Energia NanoBot y ui Barra Energia | 20 may. 2021 20:32 |
| | Añadir actor para el fondo animado, setear ajustes varios | 16 may. 2021 22:15 |
| | Editar textura adn "disparos", generar textura animada desde un gif, para ahorrar recursos se convierte en textura png. | 15 may. 2021 19:56 |
| | Implementar de forma optima touchPad para el movimiento (añadir movimiento "analógico" y configurar zona muerta del touchPad) | 15 may. 2021 13:42 |
| | Definir movimiento random al actor "virus" | 11 may. 2021 22:00 |
| | Crear texturas para animación e implementarlo a enemigos | 9 may. 2021 22:25 |
| | Implementar clase Controller(touch pad) Android | 9 may. 2021 13:59 |
| | Mejorar controles del actor principal (Sincronizar KeyUp con KeyDown) | 7 may. 2021 14:06 |
| | Crear movimiento (key imputs) funcion movimiento, calcular limites, añadir funcion disparos | 6 may. 2021 22:17 |
| | Implementar TouchPad (Android) | 5 may. 2021 21:06 |
| | ** Añadir documentación del proyecto al repositorio ** | 5 may. 2021 18:24 |
| | Añadir Pantalla(abstract), PantallaJuego, actores y assets | 2 may. 2021 23:00 |
| | cambios en la clase principal (extends Game implements Screen) | 2 may. 2021 19:29 |
| | test4 | 28 abr. 2021 19:53 |
| | test movimiento | 27 abr. 2021 22:44 |
| | test3 | 25 abr. 2021 10:24 |

## 3.9 Plugin para generar diagramas UML

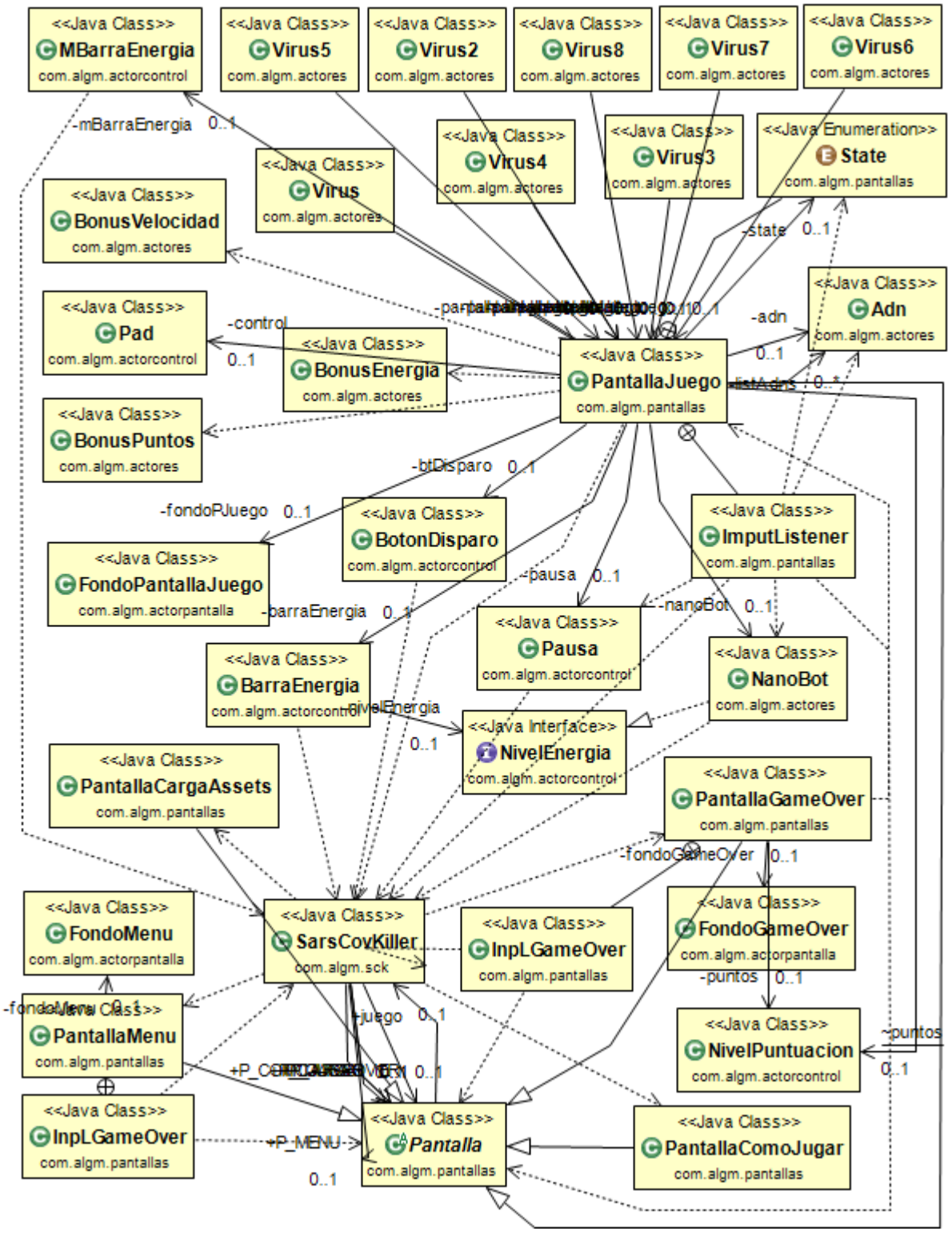
He utilizado el anteriormente mencionado plugin ObjectAid UML Diagram y el resultado en la generación de UML son los siguientes:

### 3.9.1 Diagrama UML de los paquetes del proyecto




### 3.9.2 Diagrama UML de sin relación en atributos

- El diagrama completo con las relaciones se adjunta como anexo debido al gran tamaño de la imagen se aprecian mejor las relaciones que en la siguiente captura.


<<Java Class>> MBarraEnergia com.algm.actorcontrol
<<Java Class>> Virus5 com.algm.actores
<<Java Class>> Virus2 com.algm.actores
<<Java Class>> Virus8 com.algm.actores
<<Java Class>> Virus7 com.algm.actores
<<Java Class>> Virus6 com.algm.actores
-mBarraEnergia 0..1
<<Java Class>> Virus4 com.algm.actores
<<Java Class>> Virus3 com.algm.actores
<<Java Enumeration>> State com.algm.pantallas
<<Java Class>> Virus com.algm.actores
<<Java Class>> BonusVelocidad com.algm.actores
-state 0..1
<<Java Class>> Pad com.algm.actorcontrol
-control 0..1
<<Java Class>> BonusEnergia com.algm.actores
<<Java Class>> Adn com.algm.actores
-adn 0..1
<<Java Class>> PantallaJuego com.algm.pantallas
<<Java Class>> BonusPuntos com.algm.actores
-btDisparo 0..1
-fondoPJuego 0..1
<<Java Class>> BotonDisparo com.algm.actorcontrol
<<Java Class>> ImputListener com.algm.pantallas
<<Java Class>> FondoPantallaJuego com.algm.actorpantalla
-pausa 0..1
-barraEnergia 0..1
-nanoBot 0..1
<<Java Class>> Pausa com.algm.actorcontrol
<<Java Class>> BarraEnergia com.algm.actorcontrol
<<Java Class>> NanoBot com.algm.actores
-nivelEnergia
<<Java Interface>> NivelEnergia com.algm.actorcontrol
0..1
<<Java Class>> PantallaCargaAssets com.algm.pantallas
<<Java Class>> PantallaGameOver com.algm.pantallas
-fondoGameOver 0..1
<<Java Class>> FondoMenu com.algm.actorpantalla
<<Java Class>> SarsCovKiller com.algm.sck
<<Java Class>> InpLGameOver com.algm.pantallas
<<Java Class>> FondoGameOver com.algm.actorpantalla
-puntos 0..1
-juego 0..1
<<Java Class>> PantallaMenu com.algm.pantallas
<<Java Class>> NivelPuntuacion com.algm.actorcontrol
-puntos
0..1
<<Java Class>> InpLGameOver com.algm.pantallas
+P_MENU->
<<Java Class>> Pantalla com.algm.pantallas
<<Java Class>> PantallaComoJugar com.algm.pantallas
0..1

# 4 PROCESO DE DESARROLLO

El proceso de desarrollo de SarsCovKill ha sido largo y no exento de muchos problemas, en los siguientes apartados voy a explicar de manera resumida el desarrollo de las funcionalidades más notables que tiene el videojuego para no alargar mucho el documento, además de que en el código están los principales métodos comentados.

## 4.1 Crear e importar proyecto

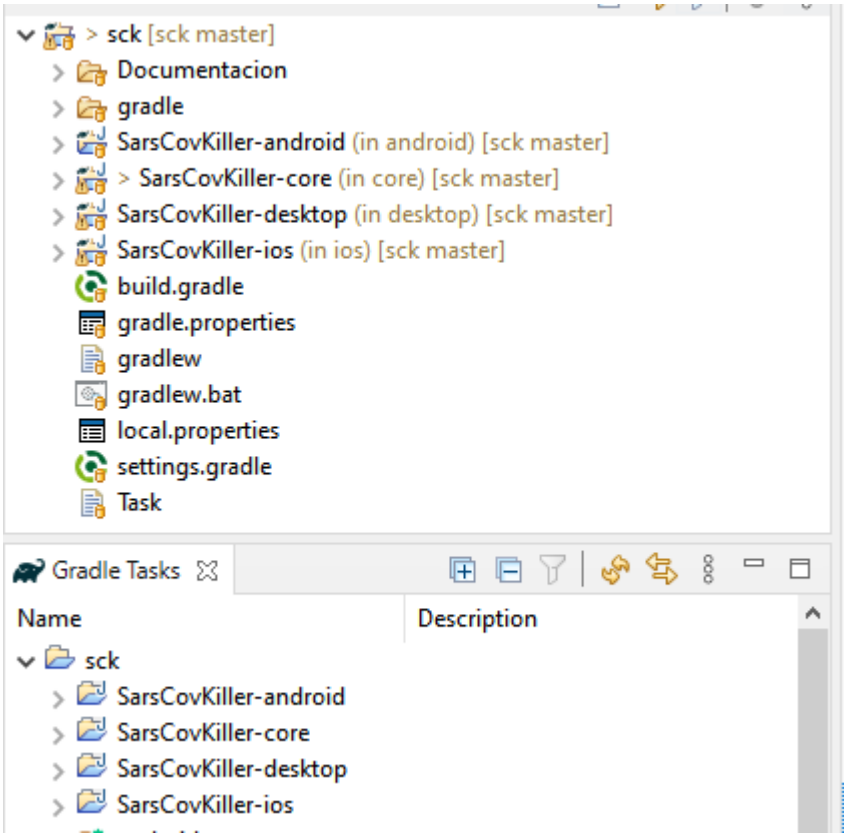
La creación del proyecto después de configurar adecuadamente el IDE (eclipse) consiste en generar mediante la herramienta [LibGDX project setup] el proyecto con las dependencias necesarias para las diferentes distribuciones que seleccionaremos en su panel de configuración. Para este proyecto se crean para Desktop, Android e IOS.


Después de generarlo se importa en el IDE Eclipse como proyecto Gradle

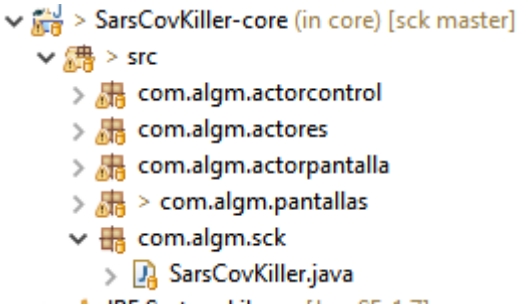
Una vez importado y corrigiendo algunos pequeños errores el proyecto queda distribuido en diferentes paquetes que pertenecen a cada plataforma y un paquete

Core donde se realiza la mayor parte de la codificación, la idea es codificar en Core para luego poder exportar los otros proyectos a partir de este.

Quedaría distribuido como se muestra en la siguiente imagen:




A su vez en el proyecto se crean diferentes paquetes que contendrán las clases necesarias lo he distribuido en los siguientes paquetes:




## 4.2 Clase Principal del videojuego

Para la creación de múltiples pantallas primeramente hay que extender a Game en la clase principal “Main” SarsConKiller.java.

Al extender a Game se podrán manejar múltiples pantallas gracias a algunos métodos auxiliares que incorpora para este propósito.

### 4.2.1 Métodos

Entre los métodos más representativos que extienden de la clase Game los mas importantes serian:

- render()
  Llamado cuando el juego tiene que renderizarse..
- resize()
  Se le llama cuando hay cambios de tamaño en la pantalla.
- pause() y resume()
  Se llama cuando el videojuego está en pausa, normalmente cuando no está la pantalla activa y resume cuando esta regresa.
- dispose()
  Se le llama cuando se cierra el videojuego.

### 4.2.2 Gestión de recursos en el juego

Para la gestión de recursos tanto imágenes como sonidos texturas o cualquier otro tipo se crea una variable estática en la clase principal de AssetsManager().

## 4.3 Pantallas

Para la creación de pantallas se crea como modelo una clase abstracta Pantalla.java que extienda de **Screen** para que las siguientes pantallas puedan implementar sus métodos, esta clase debe contener en su constructor como parámetro un objeto de de tipo SarsConKiller.java para poder acceder a los datos del juego desde cualquier pantalla.

## 4.3.1 Métodos

Los métodos sobrescritos más importantes de la clase screen que realizan la lógica del juego pueden resumirse de la siguiente manera:

- show()
  En este método se cargan todos los componentes que serán mostrados en la escena, funcionaria de manera similar a un constructor.

- resize()

  En este método se ejecutan las órdenes referentes al escalado de pantalla.
- hide()

  Se le llama cuando la pantalla ya no es la actual que se está mostrando.
- pause y resume()

  Como indican sus nombres realizan las funciones de dar el estado a la pantalla.
- render()

  Es quizás el método más importante ya que se le llama cuando la pantalla tiene que renderizarse a sí misma, esto ocurre como si estuviera en un bucle infinito y las iteraciones del mismo vienen determinadas por deltaTime() que es el tiempo que tarda en renderizar lo que la pantalla contiene y varía de la carga gráfica de la misma para un deltaTime no alterado por demasiada carga gráfica se ejecutaría el contenido de render unas sesenta veces por segundo.

## 4.4 Escenas

La clase Stage es la encargada de contener a los diferentes actores que tenga el videojuego en las diferentes pantallas, por lo que habrá que instanciar un objeto de stage en cada pantalla.

## 4.5 Actores

Cada actor en la escena se crea como una clase independiente que extiende de Actor y gracias a sus métodos heredados tendrán sus propios comportamientos en el videojuego de forma independiente.

### 4.5.1 Métodos

Los métodos sobrescritos más importantes de la clase Actor que realizan la lógica de los mismos puede resumirse de la siguiente manera:

- draw()

  En este método de la clase Actor se define las propiedades de dibujo del actor.
- act()

Cada vez que se actualice un actor se le pueden definir diferentes acciones en función de el movimiento que le adjudicamos al mismo, si le hemos definido que se mueva en el eje X a una velocidad de por ejemplo 600*DeltaTime, cada vez que se actualice aparecerá más a la derecha de la pantalla.

Se le pueden asignar muchas otras funciones desde este método como por ejemplo hacer que su rectangle de colisión se sitúe encima de sus coordenadas cada vez que se llame al act() (actualizar).

- draw()

  En el constructor de cada actor se invoca a Assets Manager para cargar los recursos en este caso de imagen (previamente cargadas en memoria desde la clase principal), en un objeto de tipo textura.

### 4.5.2 Animaciones

A esta textura se le dan propiedades en cuanto al tamaño. En el caso de que se quiera realizar una animación la textura contendrá una representación de cada fotograma y estos serán almacenados en un Texture Región (array bidimensional, en el caso que los fotogramas tengan varias filas en la textura). Lo que conseguiremos con esto y mediante el TextureRegion.Split (dándole el tamaño del fotograma) mediante un bucle anidado iremos cortando los fotogramas y almacenarlos en un TextureRegion (Array de una dimensión) para posteriormente crear un objeto Animation pasandole el array Texture región y una velocidad para el cambio entre texturas.

Se puede crear un Animation con texturas sueltas, pero el rendimiento es inferior que haciéndolo con una textura más grande con todos los fotogramas ya dentro.

### 4.5.3 Colisiones

Para las colisiones se pueden usar métodos que generan un círculo de colisión o bien un rectángulo, dependiendo de la forma de la textura es más recomendable usar uno u otro, pero para cuestiones de rendimiento es más eficiente el rectángulo.

Al instanciar un objeto de tipo Rectangle con las dimensiones de nuestra textura conseguiremos controlar las colisiones entre diferentes rectangles.

En la siguiente captura gracias a activar el modo debug gráfico se aprecia cómo se generan los rectangles sobre los actores.

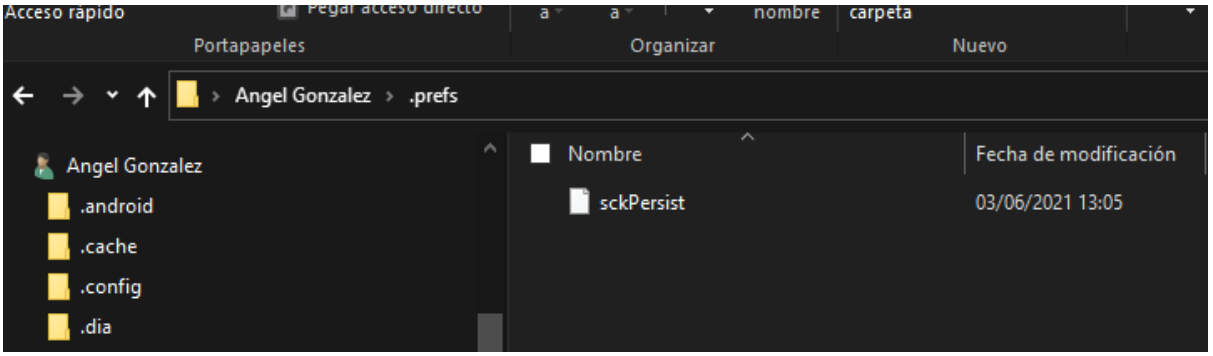
Desde la clase de juego (PantallaJuego.java) se guardan en arrays cada actor que hay en la pantalla y mediante bucles anidados se comprueban la colisiones de los mismos con comandos similares a: *virus.getRectangle() .overlaps (nanoBot.getRectangle()).* Realizando las acciones pertinentes dependiendo qué actor sea el que ha colisionado con otro.


## 4.6 Controladores de juego

Para los controladores se deben implementar métodos input Listener los cuales pueden ser variados. Pueden ser para inputs de teclado, inputs táctiles, ratón, etc.
Mediante estos inputs podremos controlar comportamientos en el videojuego como disparar, pulsar botones, etc.
Cabe mencionar que para el movimiento del nanoBot he usado input listeners con detección de pulsar tecla además de soltar tecla, esto se ha hecho para que al tener varias teclas pulsadas prevalezca la última y el movimiento de desplazamiento sea de mejor calidad, en caso de no hacer se producirían movimientos no deseados al tener varias teclas pulsadas.

## 4.7 Sistema de persistencia

El sistema de persistencia en el videojuego se utiliza para el guardado de partidas.
Después de mucho investigar sobre qué sistema sería mejor me he decantado por cuestiones de eficiencia al realizar la persistencia mediante un archivo XML, en él se pueden almacenar los datos referentes a hasta cinco espacios de guardado independientes. El fichero se crea automáticamente al ejecutar la aplicación y se basa en un archivo de tipo **preferences** como el que se puede generar con aplicaciones comerciales. El guardado es automático al salir de la partida o bien salir del juego.
El lugar de guardado definido es en la carpeta .pref que se localiza en la carpeta del usuario.




## 4.8 Resolución

Sobre el desarrollo de los diferentes métodos y las funciones que realizan están casi todos documentados en el código, ya que si los expongo en el documento y debido a la gran cantidad de métodos y funciones de las que dispone el videojuego se haría sumamente largo el documento.
En los anteriores puntos he definido lo que para mi es lo más importante en cuanto al proceso de desarrollo.

# 5 PROCESO DE DESPLIEGUE

## 5.1 Despliegue de la aplicación para escritorio (Windows)

Para el despliegue de la aplicación de escritorio simplemente es necesario tener instalada una máquina virtual de java (JVM) en el sistema operativo y ejecutar la aplicación, que viene empaquetada en un archivo .jar.

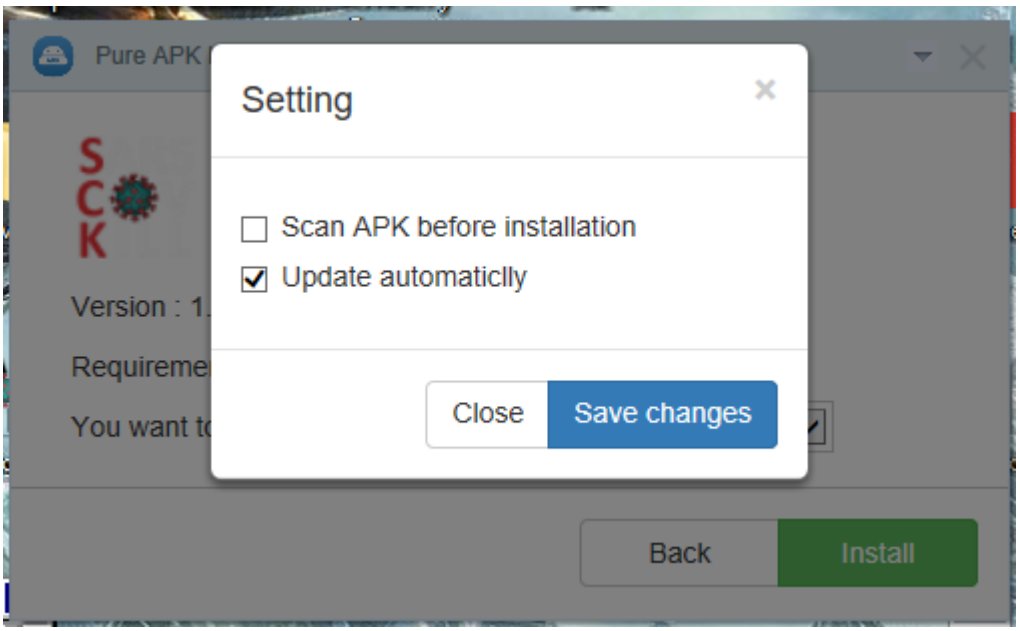
En el caso de no tenerla instalada o bien no querer instalarla. He generado un ejecutable (.exe) que contiene la máquina virtual incorporada.
Por motivos de tamaño (ocupa mas de 400MB) no enviaré el ejecutable en sí, pero si tendréis un enlace de drive para poder descargarlo.

## 5.2 Despliegue de la aplicación para Android (Windows)

Para el despliegue de la aplicación móvil hay varias alternativas, se puede hacer sobre una máquina virtual android o bien sobre el propio dispositivo móvil (conectándolo mediante usb), para la instalación el el dispositivo móvil se puede hacer uso de varios programas entre los más famosos se encuentra Android Studio.
En el caso de no tenerlo instalado en el sistema recomiendo instalar PureAPKInstall que puede descargarse desde el siguiente link (No obstante lo incluyo con la APK). Este programa es ligero (8MB) y eficiente para instalar APKs en el dispositivo desde un sistema operativo Windows.

### 5.2.1 Configurar nuestro dispositivo para instalar la APK

1. Por motivos de firma, habrá que activar el el dispositivo android la opción de: **Permitir instalar aplicaciones distintas de market** (o algo similar ya que depende del dispositivo) Suele estar en la ruta:
   Ajustes > seguridad > orígenes desconocidos (Hay que actívala).
   Android te dará un aviso con los riesgos que asumes, tendrás que aceptarlos.
2. **Activar depuración usb**
   Entra en Ajustes > Sistema > Acerca del teléfono > Número de compilación
   Pulsar el número de construcción no menos de siete veces.
   Hasta que veas una advertencia de que el modo de desarrollador ya está activado.
   De nuevo ir a "Sistema", encontraremos las "Opciones de desarrollador".
   Entrar y activar las **"Opciones del desarrollador"** y **“Depuración USB”**
3. Instalar Pure_APK_Install_setup.exe en nuestro equipo con Windows y una vez finalizado conectar el dispositivo Android con depuración USB.
4. Ejecutar Pure Apk Install y abrir las opciones, desmarcar la opción “scan APK before installation”, guardar los cambios.




5. Buscar nuestra apk e instalar en el dispositivo Android (nos dará la opción de instalar en la memoria interna o bien el la externa de nuestro dispositivo).

# 6 ERRORES CONOCIDOS

Para la versión Desktop de SarsCovKill no se han encontrado errores en la ejecución de la aplicación.

Para la versión de Android, la apk se instala perfectamente.
Al ejecutarla y después de pasar del menú hacia la pantalla de juego cargan correctamente todos los componentes, pero hay algún tipo de bug que afecta a la ejecución de algún componente de la pantalla de de juego, provocando el cierre de la aplicación a los pocos segundos de entrar a jugar.

Estoy trabajando para solucionarlo, pero aun no he encontrado la incompatibilidad, intentaré antes de la exposición de proyecto tener solucionado el bug.

# 7 PROPUESTAS DE MEJORAS O TRABAJOS FUTUROS

- Entre las propuestas de mejoras o trabajos futuros lo primero que tengo en la mira es solucionar el bug en ejecución que se produce en Android.

- Otras mejoras que quiero implementar es añadir nuevas mecánicas de juego, nuevos adversarios, nanobots, etc.

# 8 BIBLIOGRAFÍA

Entre los principales recursos usados en la creación de SarsCovKill destaco los siguientes que enumero a continuación.

Extending the Simple Game - libGDX

Pantalla (API libgdx)

Actor (libgdx API)

Motor libgdx para Android y Java

ScalingViewport (libgdx API)

LibGdx en español: Acciones

apuntes:libgdx [Programación Multimedia y Dispositivos Móviles]

Stage (libgdx API)

Sondeo del módulo de entrada LibGDX - programador clic

timedelta - valor delta en el método de renderización libgdx

Programación de Juegos para Android - Eventos de entrada InputAdapter

Touchpad.TouchpadStyle (API libgdx)

Asset Manager - LibGDX Tutorial - Game Development

Rectball – makigas

java - ¿Qué son los vectores en programación (LibGDX)? - Desbordamiento de pila

Home · libgdx/libgdx Wiki

Framewok libGDX para programación de videojuegos – Academia Android

apuntes:libgdx [Programación Multimedia y Dispositivos Móviles]

# 9 ANEXOS

Se adjunta con el pdf una imagen “UML Clases.png” que contiene el diagrama del proyecto.