Mario Ramos
Alex Paredes

Memòria Projecte CanColapi

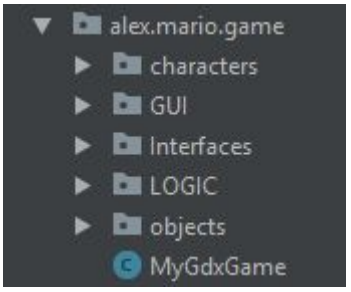
Es tracta d'un projecte que té com a objectiu aprendre a crear i gestionar objectes amb una estructura de dades orientades a POO. Se'ns ha demanat fer un joc amb els components de Java Swing. Nosaltres hem volgut anar més enllà i afegir-li una interfície gràfica més còmode per l'usuari, fent-ne ús d'imatges, mapes i sons amb LibGDX.

La lògica del joc està programada amb el mapa, és a dir, el motor de joc que hem creat és autosuficient per a adaptar-se a qualsevol mapa creat. És per això que el desenvolupament ha sigut més extens del que ens esperàvem.Podem parlar de que hem creat un motor de joc, el qual és autònom i qualsevol modificació/ampliació de mapa.

El motor parteix de la base que ens ofereix LibGDX, amb mètodes com **public void create()** o **public void render()**. La llibreria ens ha servit per a dibuixar en pantalla (figures, fonts i sprites) i per a l'estructura principal del joc. Tota l'estructura i lògica del joc ha sigut dissenyada i desenvolupada per nosaltres.

Esquema general ordre dels fitxers

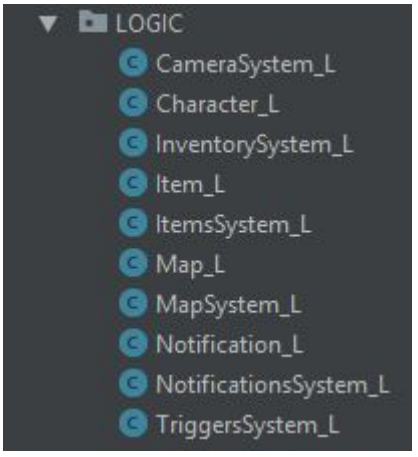
Per tal de separar la part lògica de la part visual hem dividit tots les clases en dos parts, sent la part lògica la base per a la part gràfica. Malgrat que la nostre idea inicial era fer una separació total (és a dir, que la part lògica funcionés 100% sense la part gràfica), no ha sigut possible del tot i per tant la separació és parcial.



MyGdxGame

És la classe principal del joc, la qual s'encarrega d'instanciar els corresponents sistemes i de controlar-ho tot. Hereda d'**ApplicationAdapter**, classe que pertany al sistema de LibGDX. Aquí podem trobar els controls del jugador, WASD per moure's, E per agafar items, SPACEBAR per utilitzar l'item seleccionat, I per obrir l'inventari, Q per seleccionar el següent item etc…

Carpeta LOGIC



Systems (xSystem_L)

Conté totes les classes BASE que gestionen la lògica dels sistemes i classes creades. Totes les classes implementen l'interfície iSystem_L que té el mètode **void update()**. Aquest mètode s'encarrega d'actualitzar l'objecte.

Character (Character_L)

Classe base que representa un personatge. La classe implementa l'interficie iCharacter_L que conté els mètodes bàsics per als presonatges.

Carpeta GUI

Systems (xSystem)

Conté les parts visuals dels sistemes. També inclou certes funcions que fan ús de parts visuals d'altres sistemes. Totes els sitemes que es troben a la carpeta GUI implementen l'interficie iSystem que té el mètode **void draw()**.

Character (Character)

Classe base que representa un personatge. La classe implementa l'interficie iCharacter_L que conté els mètodes bàsics per als personatges.

SoundSystem i TextureSystem

Aquests sistemes han sigut especialment creats per gestionar les fonts externes com els sons o les textures. Aquesta gestió la fa per a optimitzar els recursos de la memòria i facilitar la crida de resources.
Ambdós sistemes funcionen de la mateixa manera, però ambdós carreguen els resources i ho guarden a un HashMap del tipus <String, File>(TexturesSystem) i <String, Music>(SoundsSystem) de manera estàtica.

Item

La classe **Item** és una extensió d' **Item_L**, que conté tota la lògica de l'objecte, com per exemple els setters/getters, mètodes per quan es troba al terra:**touch()**, **untouch()**, **useGround()**; quan es troba a l'inventari del personatge: **use()**.
Aquesta classe actualment no implementa cap interfície, però seria una molt bona millora fer-ne una interfície per tal de afegir rigor a les subclasses i a l'utilització d'objectes.
Tots els objectes per defecte es poden agafar, utilitzar i llençar.
La classe **Item** és la base dels objectes que apareixen al joc, per tant és una classe abstracta i és la base de les següents classes:

Bone

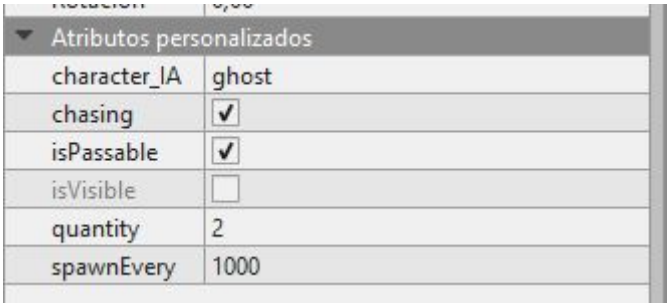
Representa l'os que distreu al Dog. No incorpora cap funcionalitat extra.

Bullet

Representa les bales que llança la ouija. El seu mètode update ha sigut patit un **Override** per a poder incorporar la lògica d'una bala: moure's en la direcció disparada i atacar si impacta amb algún personatge, si impacta amb una paret s'autodestrueix.

CharacterSpawner

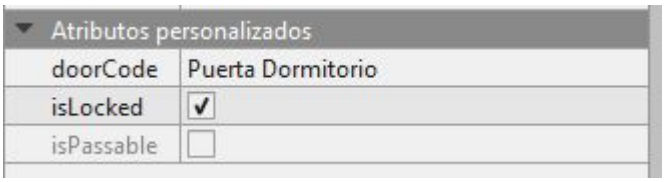
És un objecte utilitat per a poder "afegir" personatges mitjançant Tiled. Aquest objecte instanciará un objecte amb la classe configurada. Aquest és un exemple de CharacterSpawner:



La propietat character_IA conté una string que está vinculada amb un personatge_ia del joc. Podem declarar altres característiques, com si és atravessable, o si està perseguint al jugador.

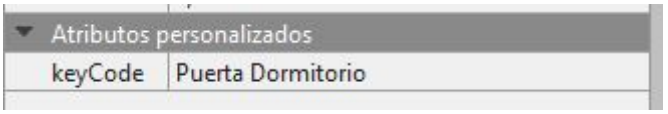
Door

La porta representa una barrera per al jugador, aquesta té un codi que representa la clau per a obrir-la. Si l'usuari fa click a la E o selecciona una clau i fa click al SPACEBAR la porta comprovará que els codis de la clau i la porta coincideixen, si coincideixen la porta s'obrirà: es cambiarà la textura i l'Item serà atravessable. Un cop la porta està desbloquejada es pot tornar a tancar sense haver d'utilitzar la clau un altre cop.



Key

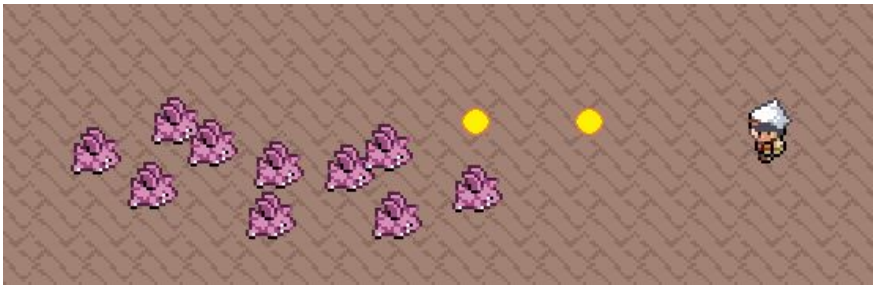
La clau és essencial per a obrir les portes bloquejades. La clau inclou un codi que servirà per a obrir la porta adient.



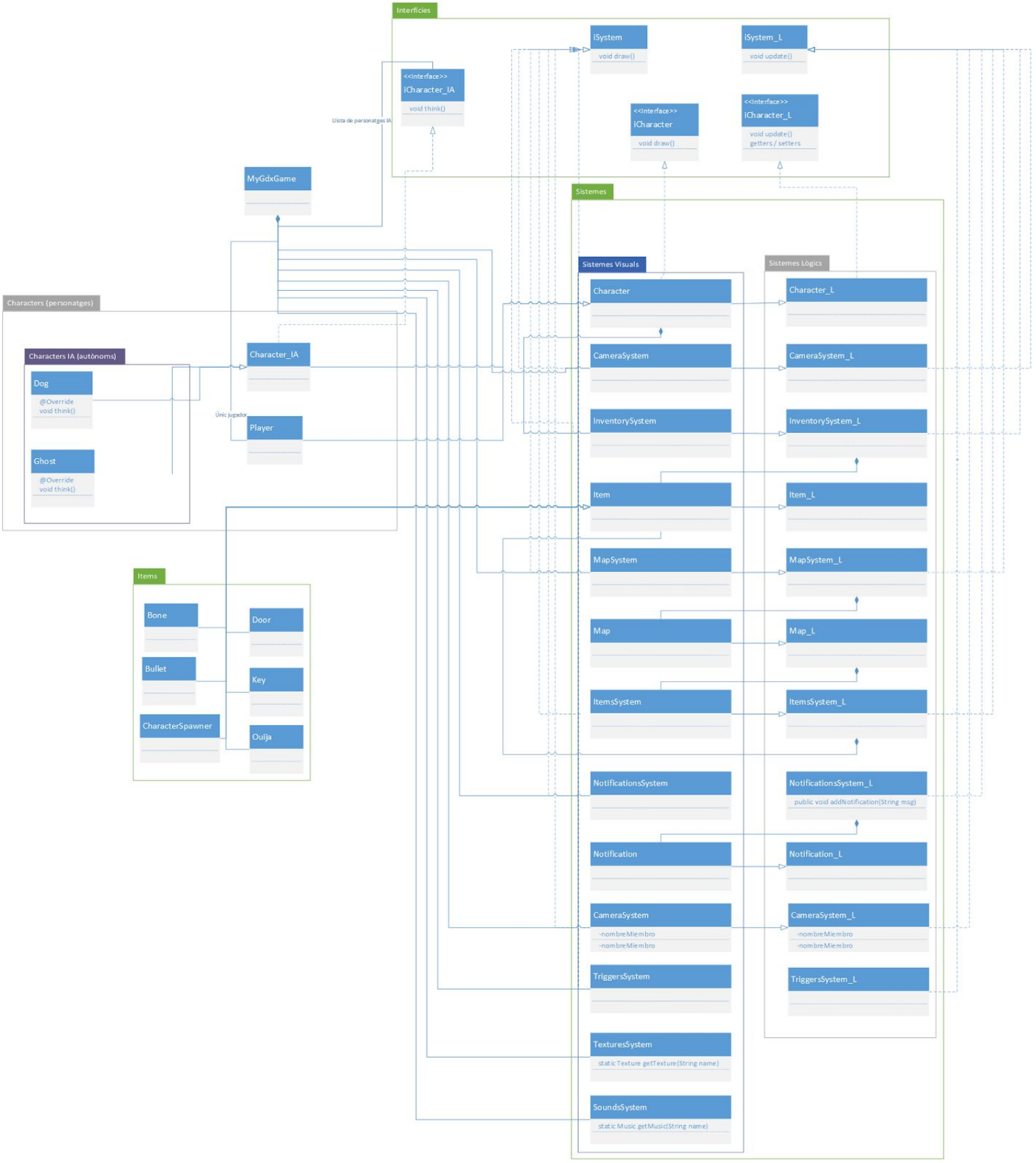
A més, si la clau està a l'inventari del jugador, aquest pot veure el seu codi:

Ouija

És l'arma que permet derrotar als **Ghost**. Té la textura d'una **Ouija** i al prèmer el botó SPACEBAR disparará una **Bullet** a la direcció on miri el jugador.

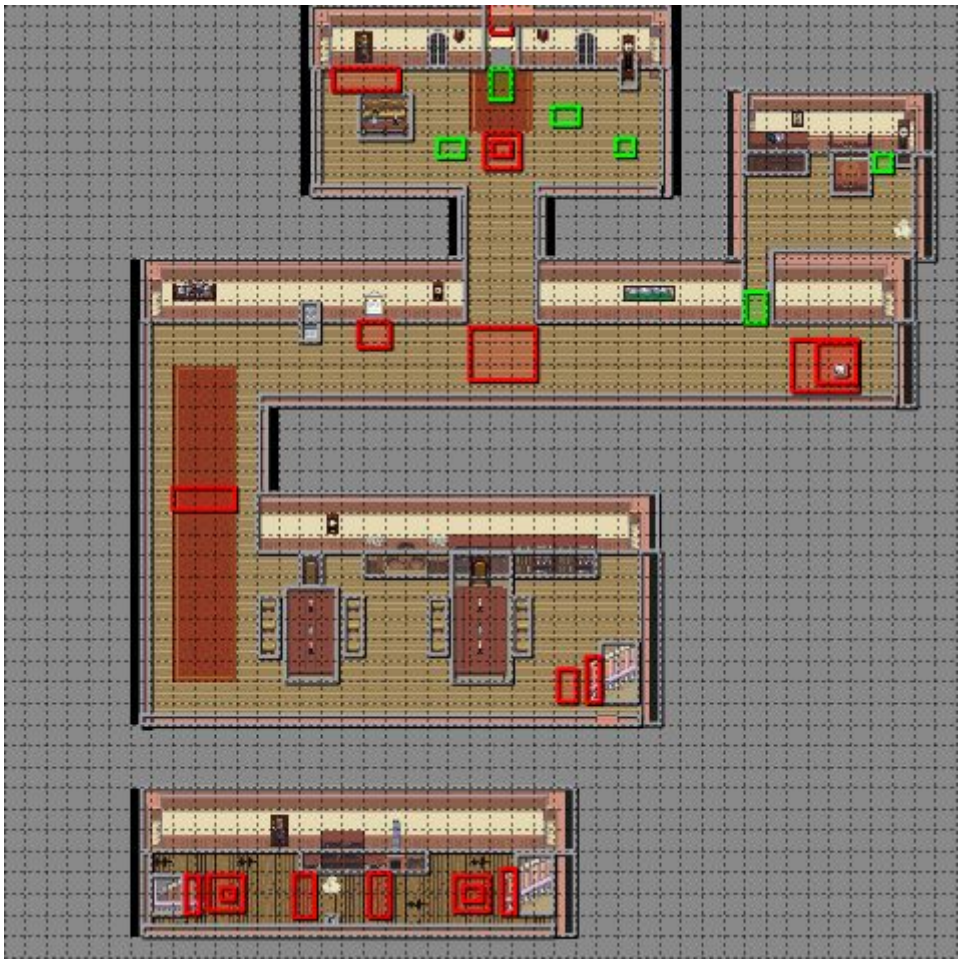


UML


Interficies
iSystem
void draw()
iSystem_L
void update()
<<Interface>>
iCharacter_IA
void think()
Llista de personatges IA
<<Interface>>
iCharacter
void draw()
<<Interface>>
iCharacter_L
void update()
getters / setters
MyGdxGame
Sistemes
Sistemes Visuals
Sistemes Lògics
Characters (personatges)
Characters IA (autònoms)
Dog
@Override
void think()
Ghost
@Override
void think()
Character_IA
Únic jugador
Player
Items
Bone
Bullet
CharacterSpawner
Door
Key
Oulja
Character
CameraSystem
InventorySystem
Item
MapSystem
Map
ItemsSystem
NotificationsSystem
Notification
CameraSystem
-nombreMiembro
-nombreMiembro
TriggersSystem
TexturesSystem
static Texture getTexture(String name)
SoundsSystem
static Music getMusic(String name)
Character_L
CameraSystem_L
InventorySystem_L
Item_L
MapSystem_L
Map_L
ItemsSystem_L
NotificationsSystem_L
public void addNotification(String msg)
Notification_L
CameraSystem_L
-nombreMiembro
-nombreMiembro
TriggersSystem_L

Creació dels mapes

Per la creació dels mapes hem utilitzat Tiled que ens permet crear mapes de diferents mides, amb diferents capes de til·les i objectes. Tot el mapa l'exportem en un arxiu .tmx i l'importem a Java mitjançant una llibreria inclosa en libGDX.



Mapa “Planta1”

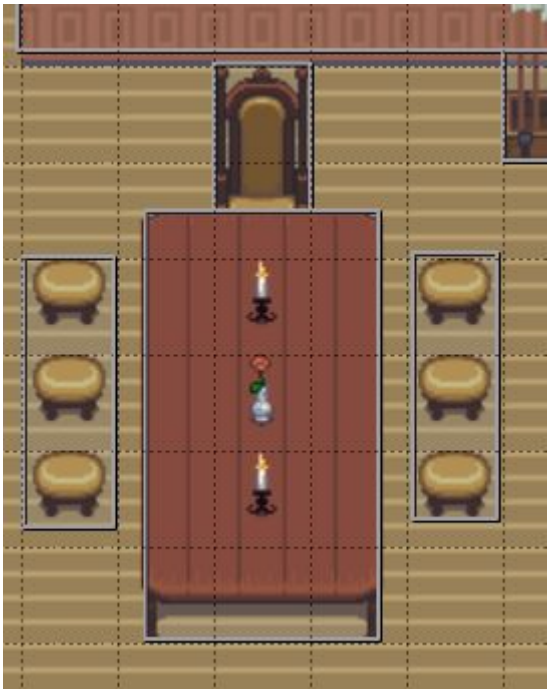
Diferents objectes del mapa

Com es pot veure a la imatge superior, el mapa disposa de vàries capes de til·les i d'objectes. A continuació explicarem per sobre les funcionalitats de cada capa d'objectes.

Colisions

És la capa d'objectes que s'encarrega d'indicar que les zones marcades tenen col·lisions amb el jugador i les bales. Si la següent posició del jugador s'interposa amb qualsevol objecte col·lisió, no es permet el moviment.

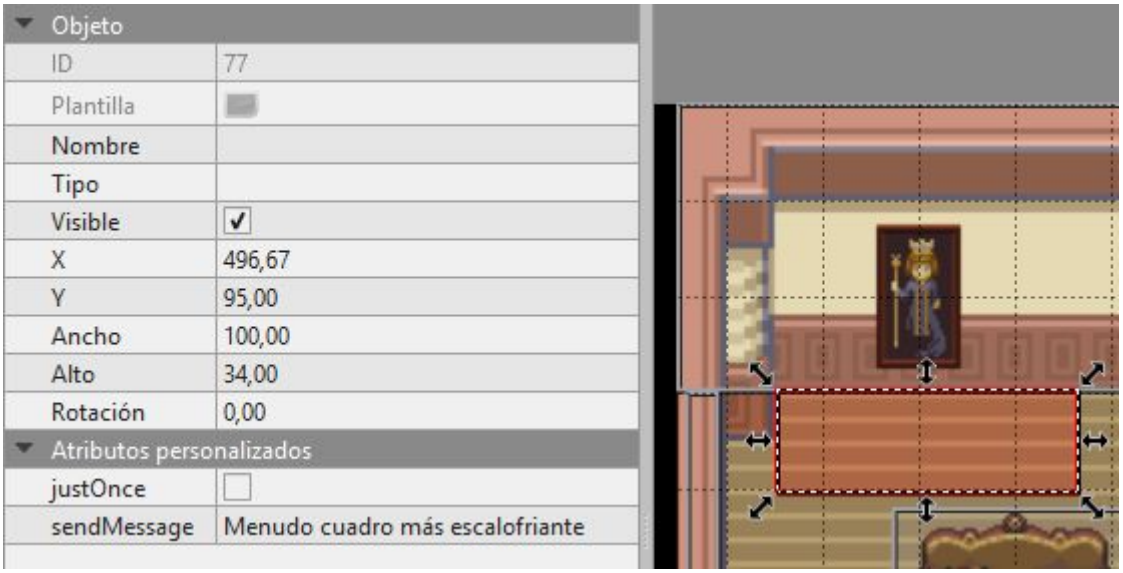
Hem optat per fer les col·lisions així perquè fer col·lisions per til·les amb tots els que disposem, seria una feinada. També comentar que així és molt més optim, ja que es redueixen considerablement el nombre d'àrees col·lisionables.

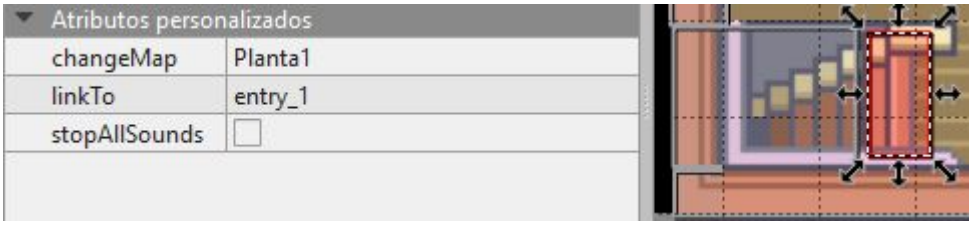


Triggers

Quan el jugador trepitja un objecte trigger, la classe TriggerSystem s'encarrega de tractar aquest trigger analitzant els atributs personalitzats que afegim a mitjançant Tiled. Disposem dels següents atributs:

- **JustOnce [bool]**: El trigger només saltarà una vegada o no.
- **SendMessage[String]**: S'envia una notificació a l'usuari amb el text posat al mapa.
- **ChangeMap[String]**: Canvia el mapa amb el text posat al mapa.
- **LinkTo[String]**: Teleporta al jugador a la posició d'un trigger "isLinkedTo" amb el mateix valor. S'usa pels teleports en el mateix mapa.
- **isLinkedTo[String]**: Valor per parejar amb LinkTo. Si es combina amb ChangeMap, serveix per establir una posició d'inici.
- **PlaySound[String]**: Reprodueix un àudio
- **isLooped[bool]**: (Sempre amb PlaySound) Indica si l'àudio es repetirà.
- **StopAllSounds[bool]**: Para tots els àudios que s'estiguin reproduint.





Path

Per fer el moviment dels fantasmes hem optat per ficar lis en camins pre dissenyats per poder adaptar totalment les rutes i la jugabilitat. El fantasma és un objecte autònom que s'instància amb direcció nul·la. Per adaptar les direccions, hem optat per fer una capa d'objectes "Path".

Aquests triggers són activats només pels fantasmes i agafen la direcció posada en les propietats d'aquest trigger.

Gameplay sencer del joc

https://youtu.be/dxLMn9W7JRc

Crèdit d’assets

Les fulles d'sprites han estat descarregades d'Opengameart.com, excepte les de Pokémon que han sigut descarregades d'unes fonts una mica menys "open source".

TILES
https://opengameart.org/content/lpc-dungeon-elements
https://opengameart.org/content/sewer-tileset
https://opengameart.org/content/dungeon-tileset
https://schwarzenacht.deviantart.com/gallery/61654676/RM-XP

MUSICA
https://opengameart.org/content/4-atmospheric-ghostly-loops
https://opengameart.org/content/church-bell

Tutorial ràpid d'importació del projecte a IntelliJ (o Android Studio)

https://github.com/libgdx/libgdx/wiki/Gradle-and-Intellij-IDEA

PD: Joc exportable a altres motors com Unity