Not Alone

The story of the game

The game begins in a fantastic intergalactic multiverse that emerged as a result of the god Chaos's decision to manifest his love into a spiritual-material form for his seven important deities. It was intended to represent an indestructible force in which there were seven realms: the realm of shadows (dedicated to the god Hades), the realm of fairies (the god Erebus), the realm of goblins and giants (the god Deimos), the realm of light (the god Aether), the realm of fire (the god Hephaestus), the realm of waters (the god Nereus), and the realm of air (the god Aeolus). All seven gods had full rights over the gifted world, but they were not allowed to interfere with the others. However, they had the ability to know every particle and characteristic of each realm. Each realm is connected to the others through a portal, which, in the realm of fairies, was hidden under a veil of protective magic by the seven oldest and most powerful fairies. This was done to prevent passage through the portal from both sides, thus ensuring the safety of their people.

Every fantastic being that passes through a portal is compelled to traverse the other realms in order to return to their place of origin, knowing that they will be in enemy territory. The most powerful beings are the shadows and the fairies from the seventh generation, the heirs to the throne. The intrigue of the story begins when the queen of the fairies' only granddaughter and successor to the throne, named after the mighty goddess Hecate, disappears while playing in the Forbidden Chamber. In a moment of weakness of the chamber's magic, Hecate is able to see and hear the malevolent call of the mystical portal. Enveloped and under its powerful spell, the fairy steps through it and is teleported to the next world.

Upon arriving in the realm of shadows, she realizes that she loses her ability to fly, a portion of her magical powers, and that she is in great danger, with no apparent path back home. Frightened, she embarks on a long journey, attempting to find ways to return home, aided and guided by the god Erebus. After traversing the arduous path, being hunted by enemies and forced to fight them, she finally reaches the portal that leads between worlds. Passing through it, she arrives in another realm where her journey continues, eventually reaching a new portal that will lead to another world or back home, depending on what the fairy will do in that particular world. In the end, she reaches her homeland and fully recovers her powers, which have become stronger, earning her the prestigious role of the sole protector of the portal and queen of her people.

Game presentation

The game is a single-player type, and the player must go through an unknown number of stages (corresponding to the story) where they are required to defeat enemies using the character's attacking abilities (with the help of the staff received from the god Erebus) and overcome the obstacles in their path in order to reach a portal that represents the passage to the next level of the story.

Game rules

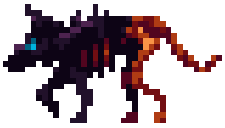
The game involves navigating through obstacles and battling antagonists that come in the character's path. The player has the power to kill enemies using the magical ability of the protagonist and can only be killed when their life reaches 0 in the health bar displayed above the character.

The player's health level and the number of attacks will increase if the player collects potions placed on the map. If the total possible value calculated does not exceed the predetermined maximum value (100), then the health will increase, and the number of attacks will increase regardless. The player must defeat all antagonistic characters encountered in their journey to pass through the portal to the next level.

Winning and completing the game occurs when the player passes through the portal of the last map and returns to the realm of fairies. Transitioning from one level to another will save the current map or game state in the database, allowing the game to be loaded for continuation. The game will resume from the last save point reached if the player falls into the abyss.

The player will have a set number of attacks (initially 4), which can be depleted but will replenish every 5 seconds from the moment the power to attack was last charged (if I attacked twice in a row, one attack will replenish after 5 seconds, and the next one after 10 seconds), and can only attack if they are not moving and haven't already attacked.

Game characters:

Name	Description	Image
Hecate	She the main character, the granddaughter of the fairy queen and the seventh heir to the throne, making her the most powerful living being. She has the ability to defend herself by using the rod of God Erebus.	
Goblin	Attacks the player by attacking it with a sword.	
Ghoul	Attacks the player by burning.	
HellDog	Attacks the player by biting and burning.	

Game board

The game items are:

- passive items:
- the ground: the surface where the player and its enemies move;
- pillars: passing by jumping over;
- trees, pillars, statues, stones, doors: map components the player will not interact with.
- active items:
- portal: represents the door between the realms. It determines the next level to be loaded or the end of the game;
- potion: item that will increase the life of the player.

Image 1. Map 1

Image 2. Map 2

Image 3. Map 3

The items and characters are placed so that the difficulty increases from level to level and from the beginning of the map to its end.

Game controls

The player can move as follow:

- *Left Arrow / A key*: walk to the left;
- *Right Arrow / D* key: walk to the right;
- *Up Arrow / W* key: jump over obstacles and enemies.

The player can attack using the **C** key**.**

Commands for testing:

- ***F9*** key: remove all enemies for generating the portal.

The interaction of objects will be determined by the collision they have.

- the interaction between the earth and player creates the platform visual effect, where the player cannot pass through the map objects.
- the interaction between characters will decrease the life and decide what happens to the character after.
- the interaction between the player and the potions will increase the player lifespan.

The enemy action is based on the distance to the player:

- follow the player and increase the speed;
- attack the player;
- return to the established path to walk.

The life box of characters modifies depending on the hit they get or the potion benefits (for the player).

Game sprites

Image 4. Tilesets for the first map.

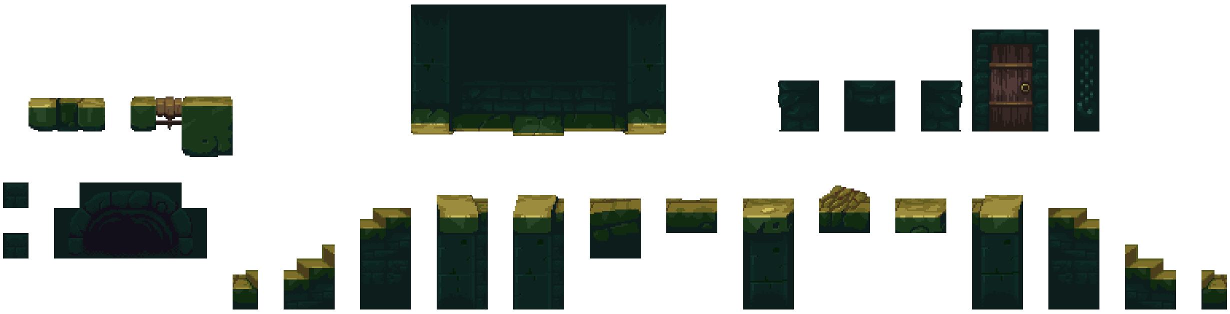


Image 5. Tilesets for the second map.

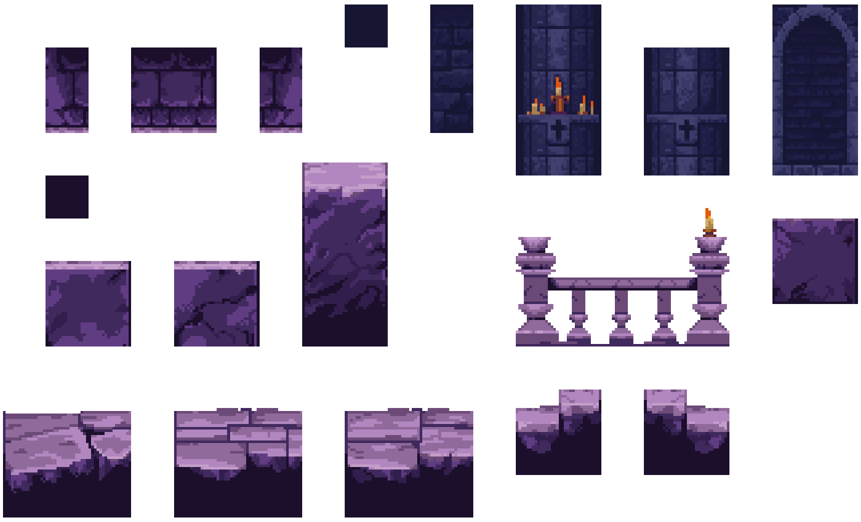


Image 6. Tilesets for the third map.

Image 7. Player sprite sheet.

Image 8. Goblin sprite sheet.\

Image 9. Potion images.

Image 10. Ghoul sprite sheet.

Image 11. HellDog sprite sheet.

Levels description

- Level 1

  Represents the first map and it is the shadows realm. The player has to jump over the abysses, piles and to fight against the Ghoul and HellDog enemies.
- Level 2

  Represents the second map of the game and the world of goblins, according to the narrative. As in the first map, the player will have to perform the same tasks, only he will fight against goblins.
- Level 3

  Represents the third world, the fire world. The player will have to defeat the fire category enemies.

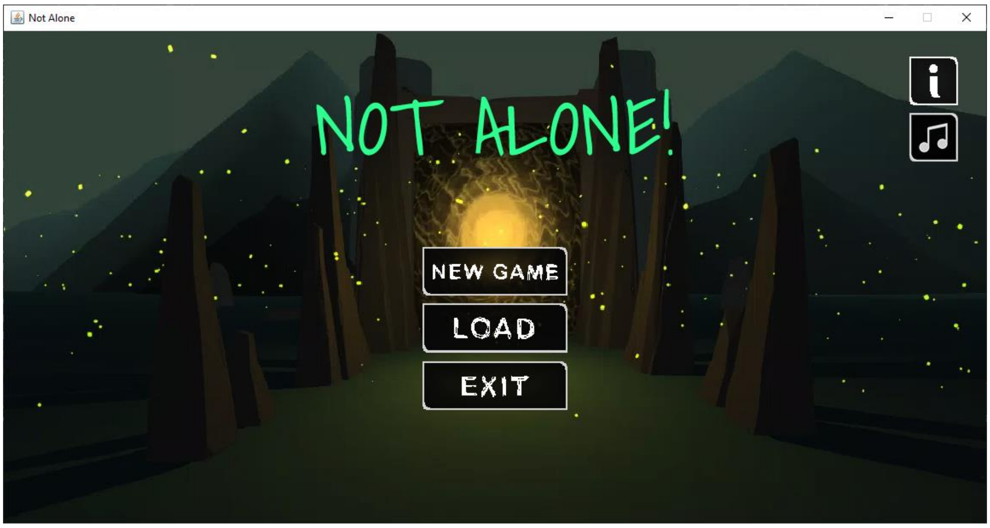
Menu description

The interaction with the menu window is done with the mouse.

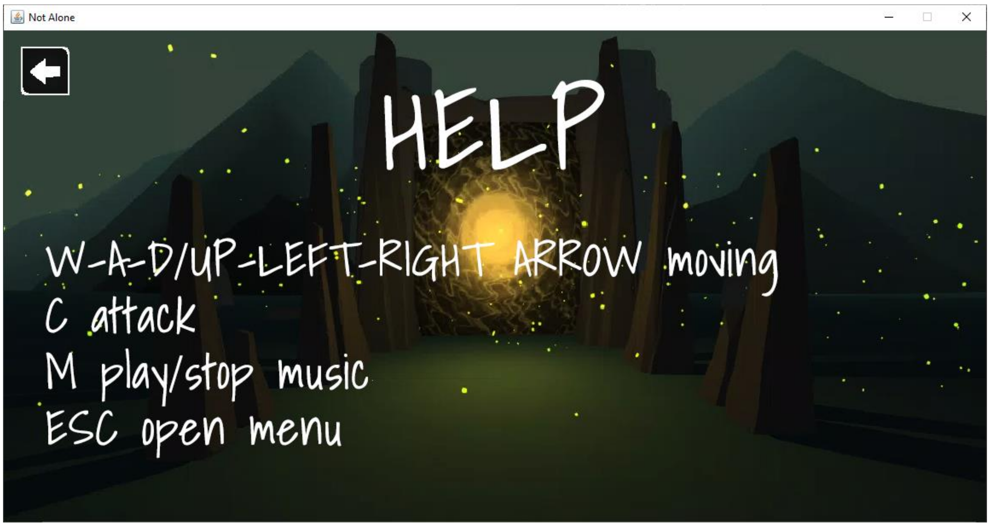
The menu has four buttons:

- NEW GAME: starts the game, resets the current data, and loads the first map in the game window;
- LOAD: starts the game from last session (if the game has been won, this won't be allowed);
- HELP: shows the game help interface;
- MUSIC: starts/stops the background music;
- EXIT: closes the game after saving the current data (the current level)

Game screenshots:



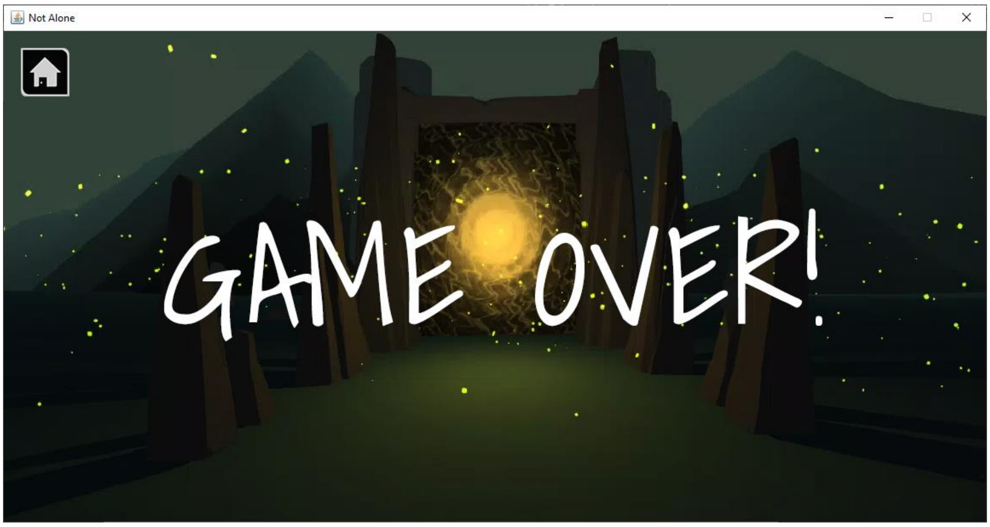
Imagine 1 - Menu state



HELP

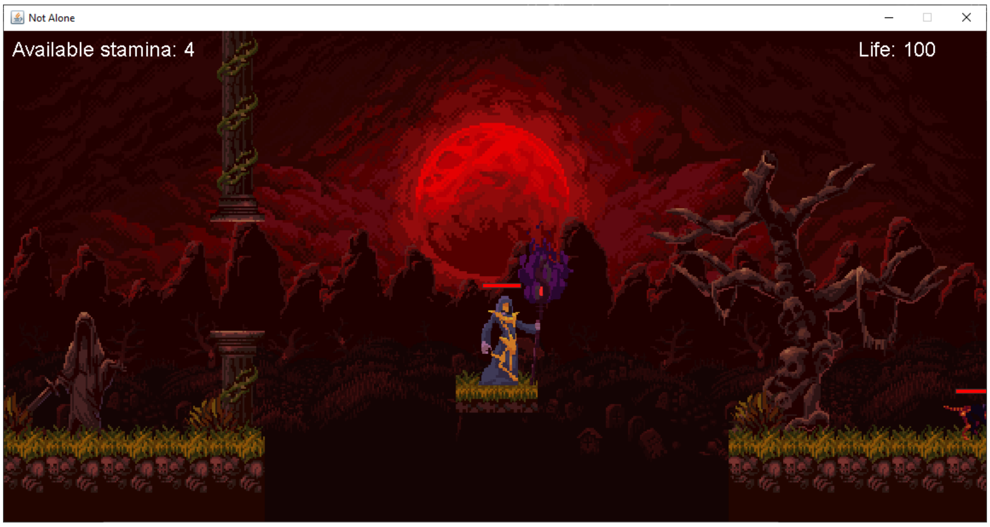
W-A-D/UP-LEFT-RIGHT ARROW moving
C attack
M play/stop music
ESC open menu

Imagine 2 - Help state

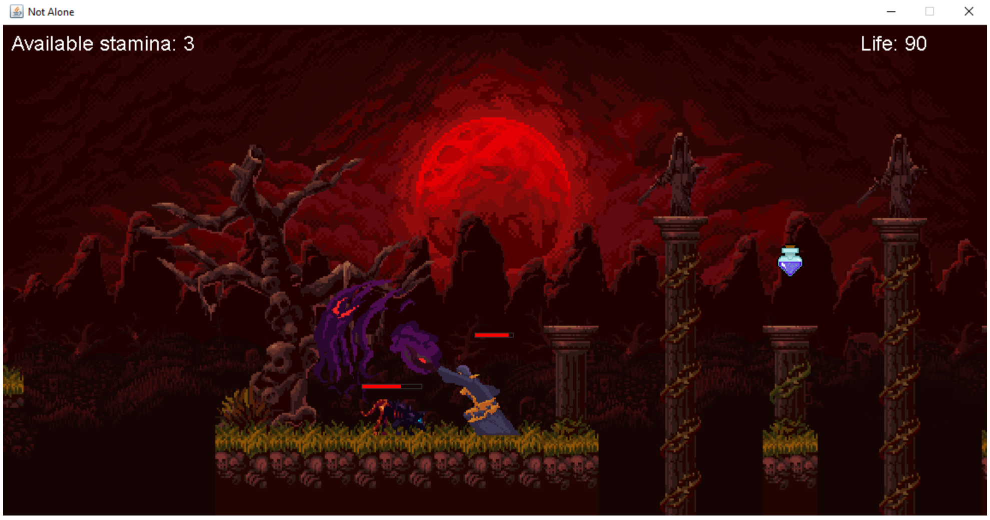


GAME OVER!

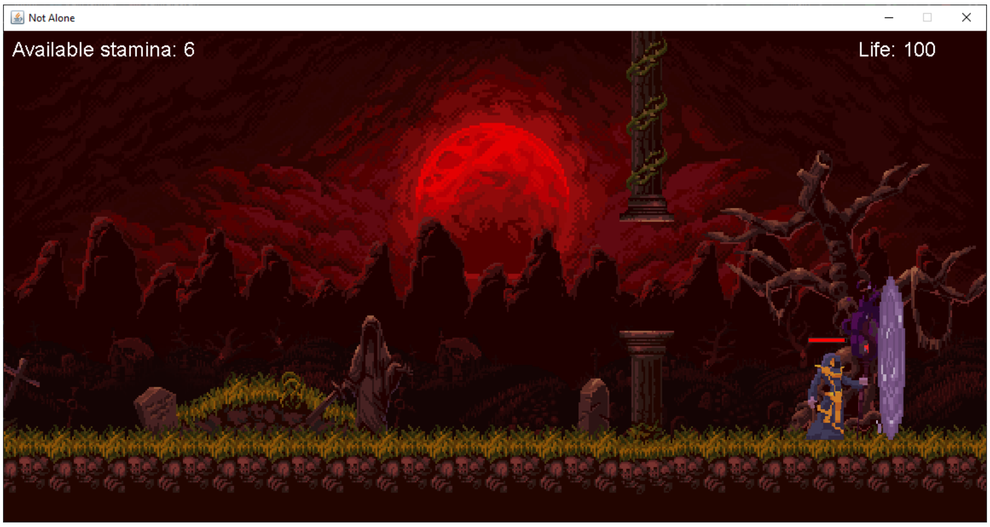
Imagine 3 - Game Over state



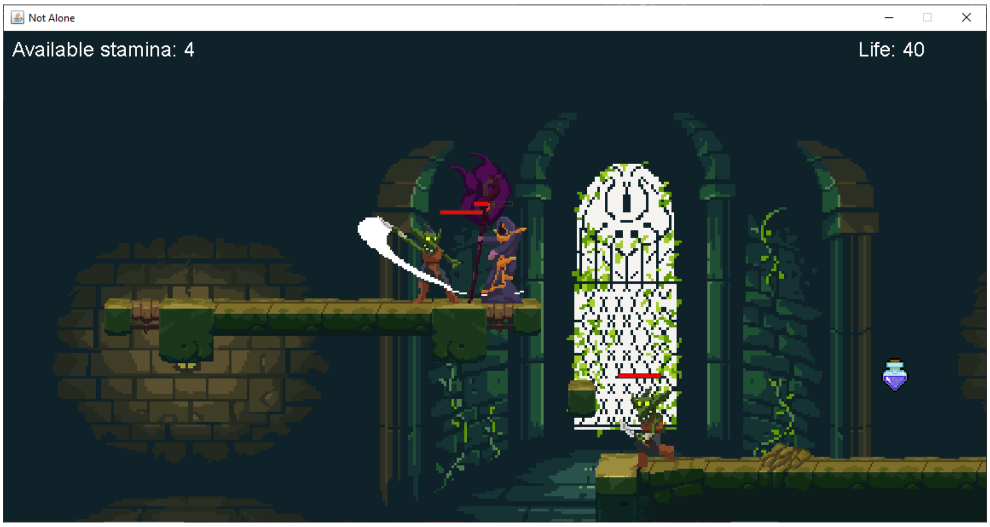
Imagine 4 - First map



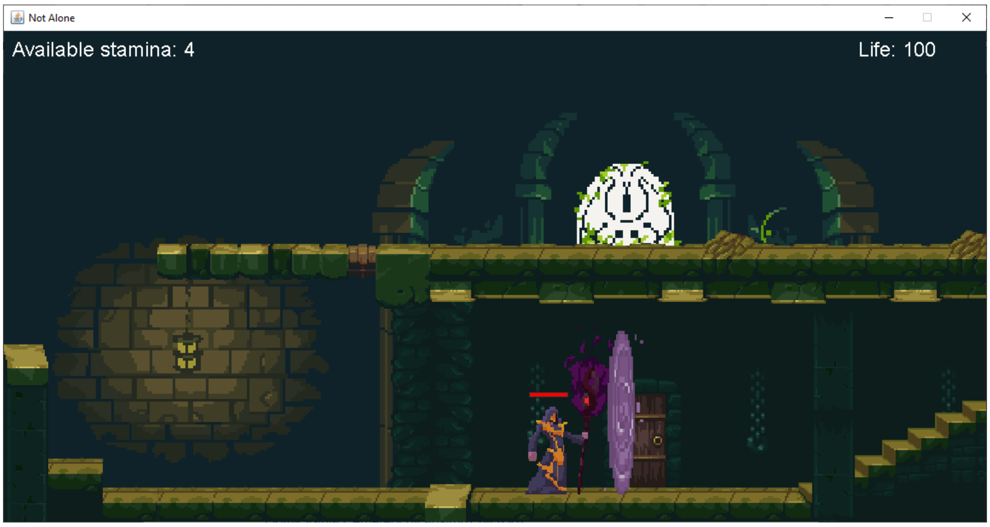
Imagine 5 - Player attacks an HellDog



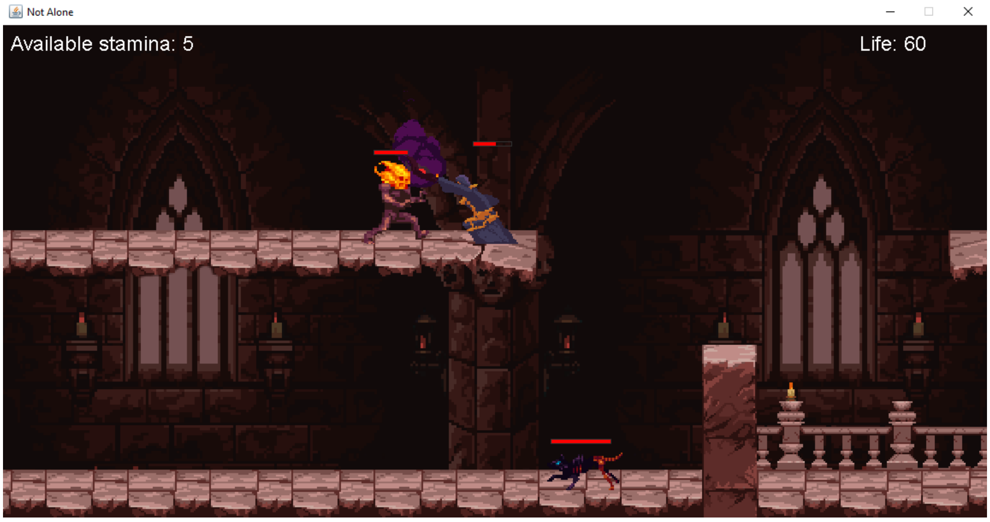
Imagine 6 - The player reaches the portal of the first map



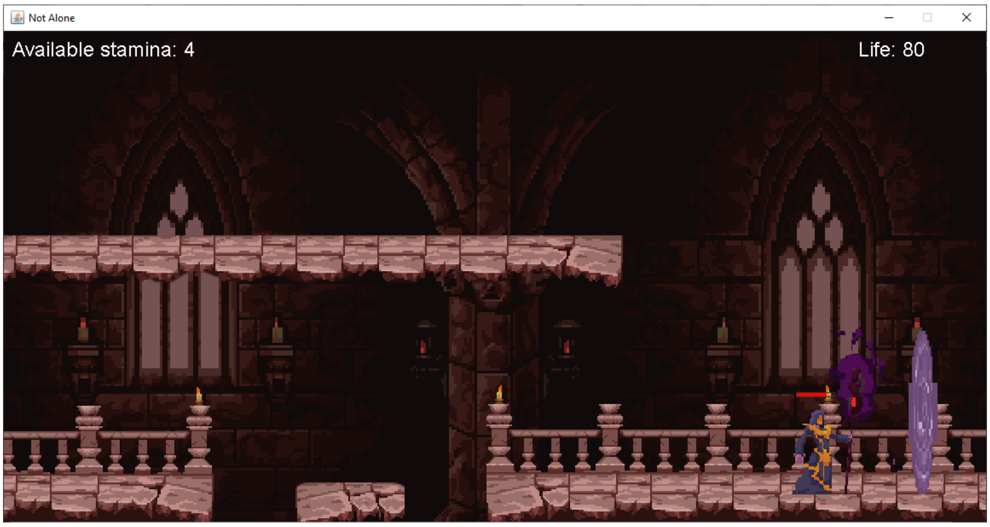
Imagine 7 - The player is attacked by a Goblin



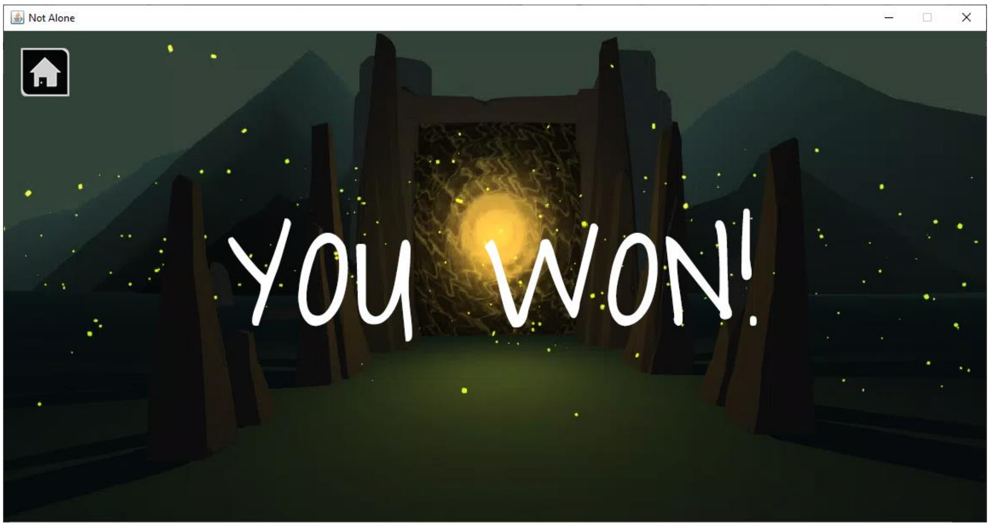
Imagine 8 - The player reaches the portal of the second map



Imagine 9 - The player attacks a Ghoul



Imagine 10 - The player reaches the portal of the third map



Imagine 11 - Won state

Data model:

The file formats used in the application are: .java, .db, .png, .wav, .txt, .jar.

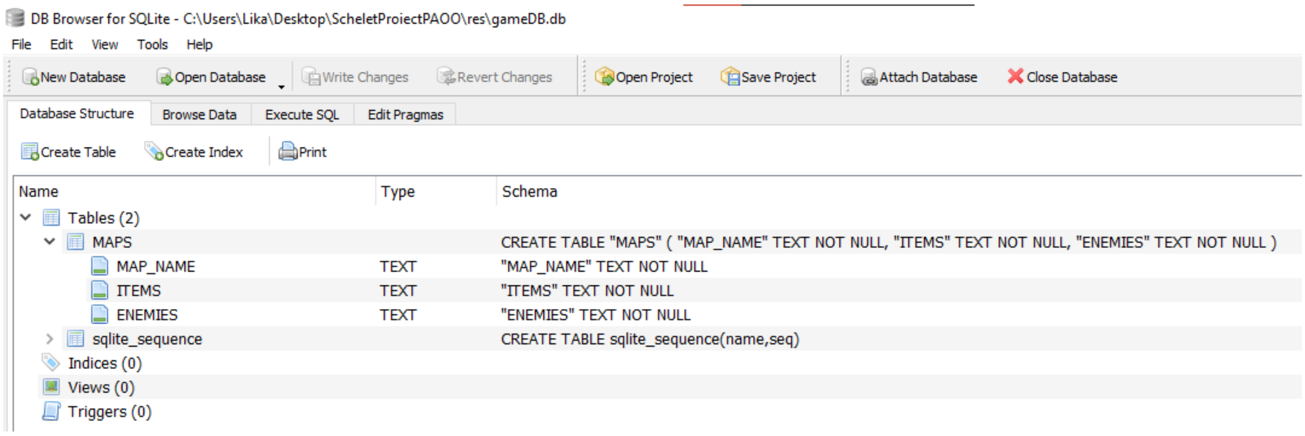
Global data structures:

- the image vectors in the Assets class;
- **ArrayList<Block> gameMap** - the blocks of the map.

Linked data structures*:*

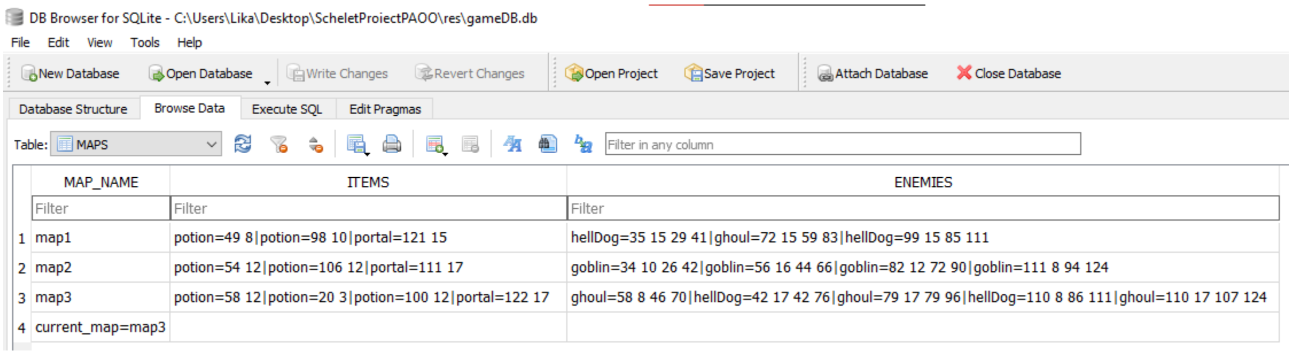
- **CopyOnWriteArrayList<ItemBlock> items** - the list of items to be checked for collision;
- **CopyOnWriteArrayList<Enemy> enemies** - the list of enemies to be checked for collision.

Database:



Name	Type	Schema
Tables (2)		
MAPS		CREATE TABLE "MAPS" ("MAP_NAME" TEXT NOT NULL, "ITEMS" TEXT NOT NULL, "ENEMIES" TEXT NOT NULL)
MAP_NAME	TEXT	"MAP_NAME" TEXT NOT NULL
ITEMS	TEXT	"ITEMS" TEXT NOT NULL
ENEMIES	TEXT	"ENEMIES" TEXT NOT NULL
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (0)		
Views (0)		
Triggers (0)		

Image 12. The structure of the MAPS table.



	MAP_NAME	ITEMS	ENEMIES
	Filter	Filter	Filter
1	map1	potion=49 8|potion=98 10|portal=121 15	hellDog=35 15 29 41|ghoul=72 15 59 83|hellDog=99 15 85 111
2	map2	potion=54 12|potion=106 12|portal=111 17	goblin=34 10 26 42|goblin=56 16 44 66|goblin=82 12 72 90|goblin=111 8 94 124
3	map3	potion=58 12|potion=20 3|potion=100 12|portal=122 17	ghoul=58 8 46 70|hellDog=42 17 42 76|ghoul=79 17 79 96|hellDog=110 8 86 111|ghoul=110 17 107 124
4	current_map=map3		

Image 13. Data in the table.

The database consists of a table with the following columns:

- MAP_NAME - contains the name of the map and has records in the following format:
  - map% - which represents the field after which a map will be loaded;
  - current_map=map% - which represents the field in which the game state will be updated.
- ITEMS - contains the elements of the map, and the records are in the format:
  - element_name=index_X index_Y
  - each element is separated by the "|" character.
- ENEMIES - contains information about the enemies of the map, and the data has the structure:
  - enemy_name=index_X index_Y left_limit right_limit
  - each element is separated by the "|" character.

Code documentation:

The documentation is attached in HTML format, generated using the Doxygen program.

Used design patterns:

Singleton

A class that implements the Singleton pattern has a private field of the class type and provides a public method to obtain its unique instance.

The classes that implement this pattern are: PlayState, MenuState, WonState, GameOverState, HelpState, Player, Game, MouseHandler, Assets, Collision, DBHandler, Camera.

State

The abstract class that represents the interface is GameState and it contains the main methods draw and update.

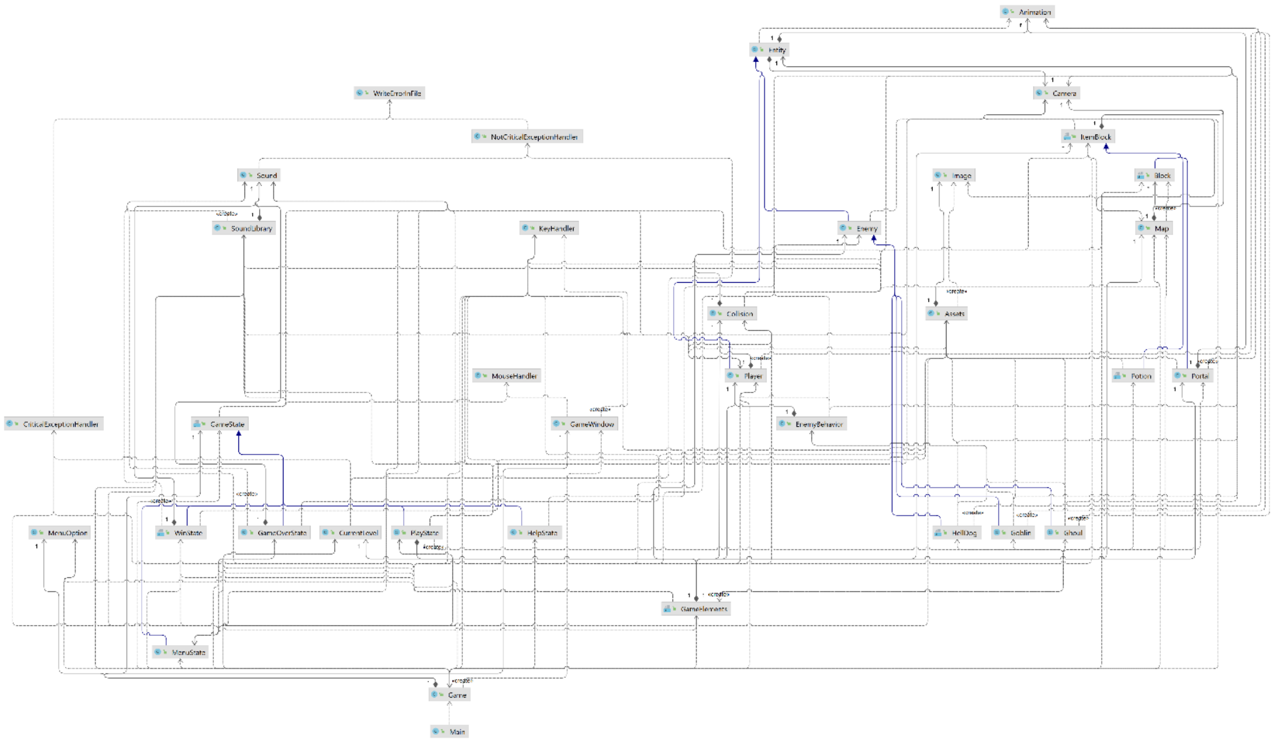
The classes that represent the concrete states are: PlayState, MenuState, WonState, GameOverState, HelpState.

The class that holds the main state is Game. The state changes in the following classes:

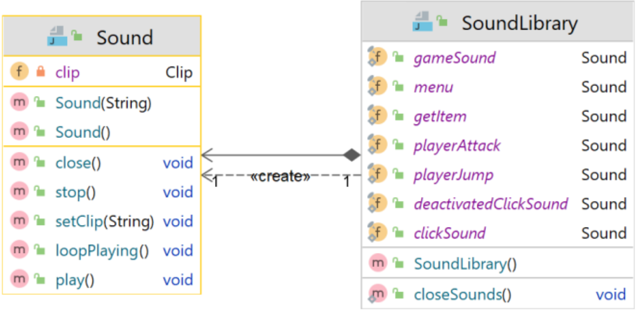
- PlayState - when the ESC button is pressed;
- MenuState - when a button is clicked;
- WonState, GameOverState, HelpState - when ESC is pressed or a button is clicked within the window.

UML Diagrams

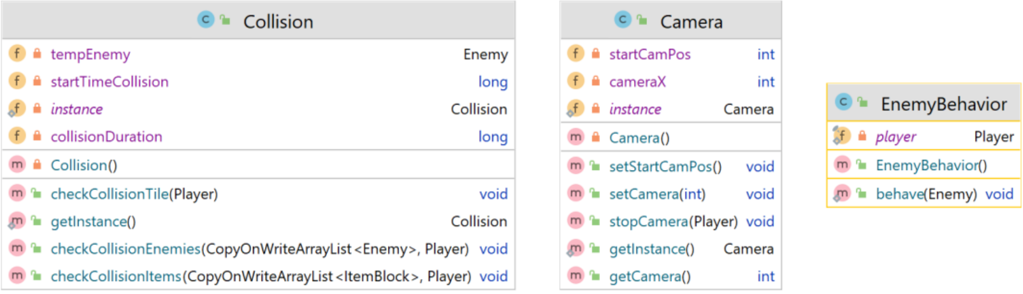
All classes UML diagram:



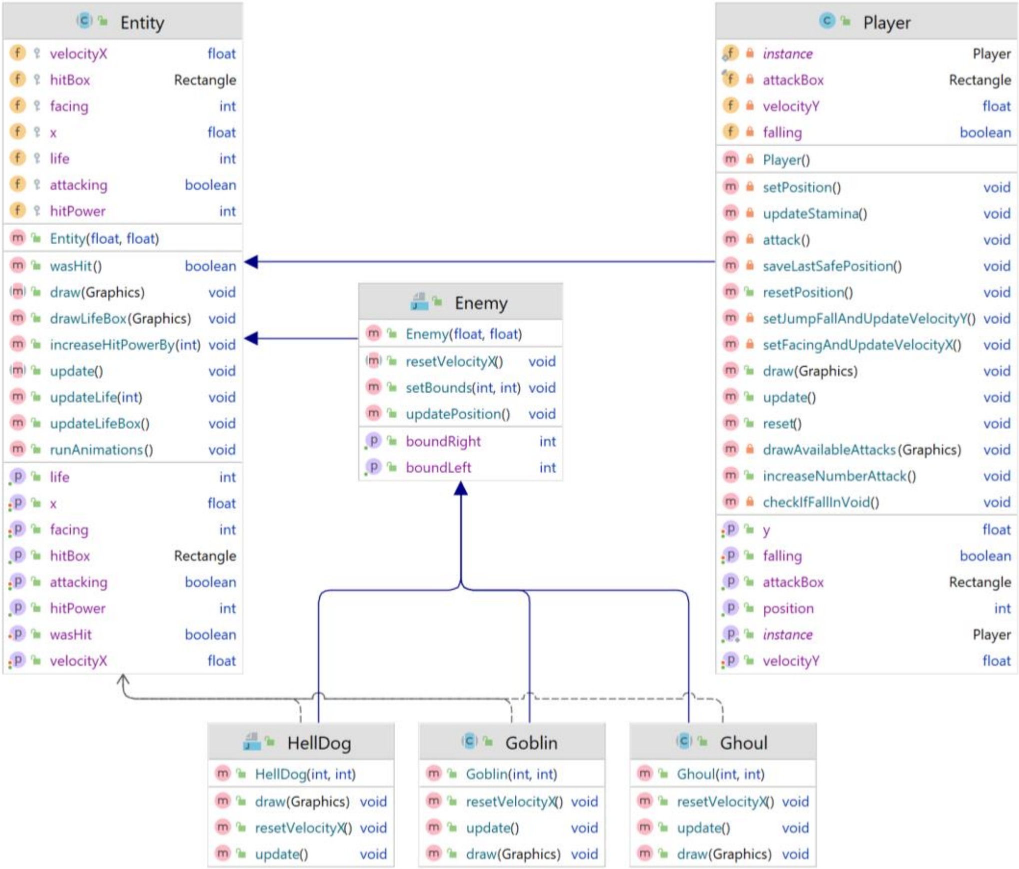
UML Diagram for the Audio Package



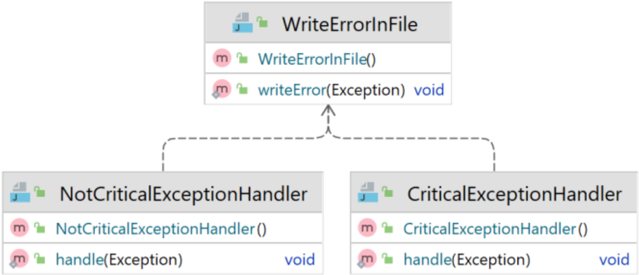
UML Diagram for the Control Package



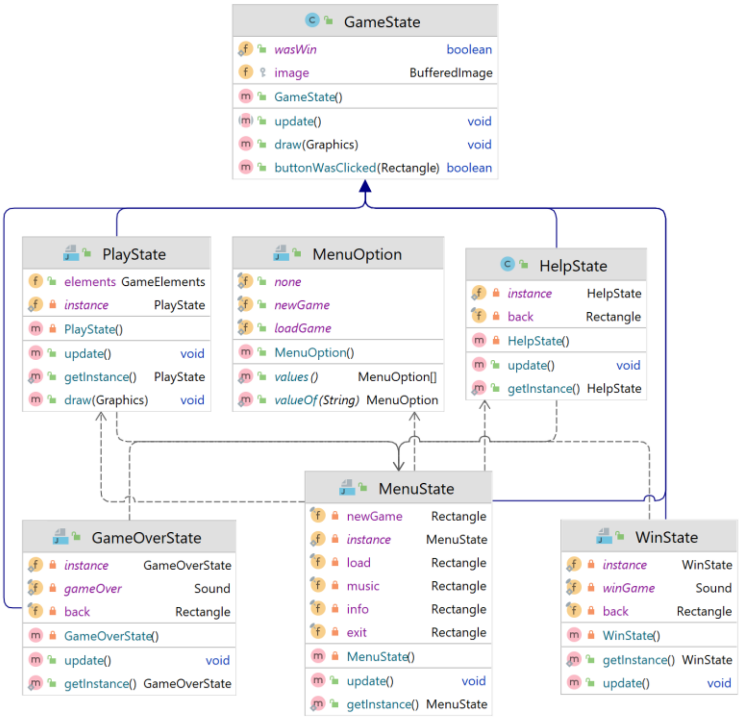
UML Diagram for the Entity Package



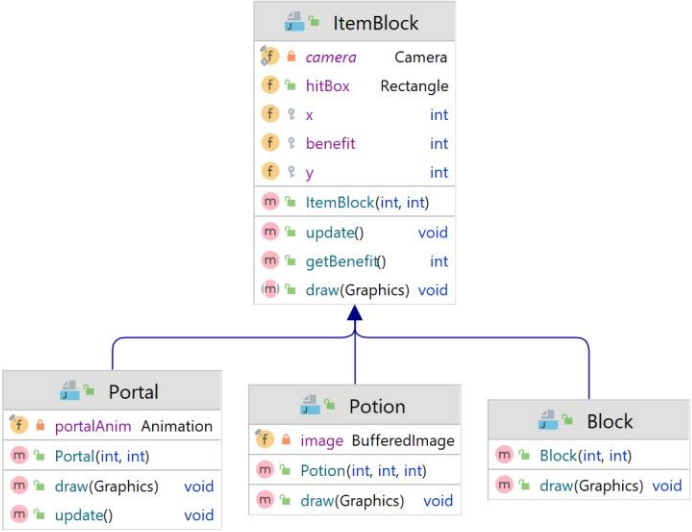
UML Diagram for the Exceptions Package


WriteErrorInFile
WriteErrorInFile()
writeError(Exception) void
NotCriticalExceptionHandler
NotCriticalExceptionHandler()
handle(Exception) void
CriticalExceptionHandler
CriticalExceptionHandler()
handle(Exception) void

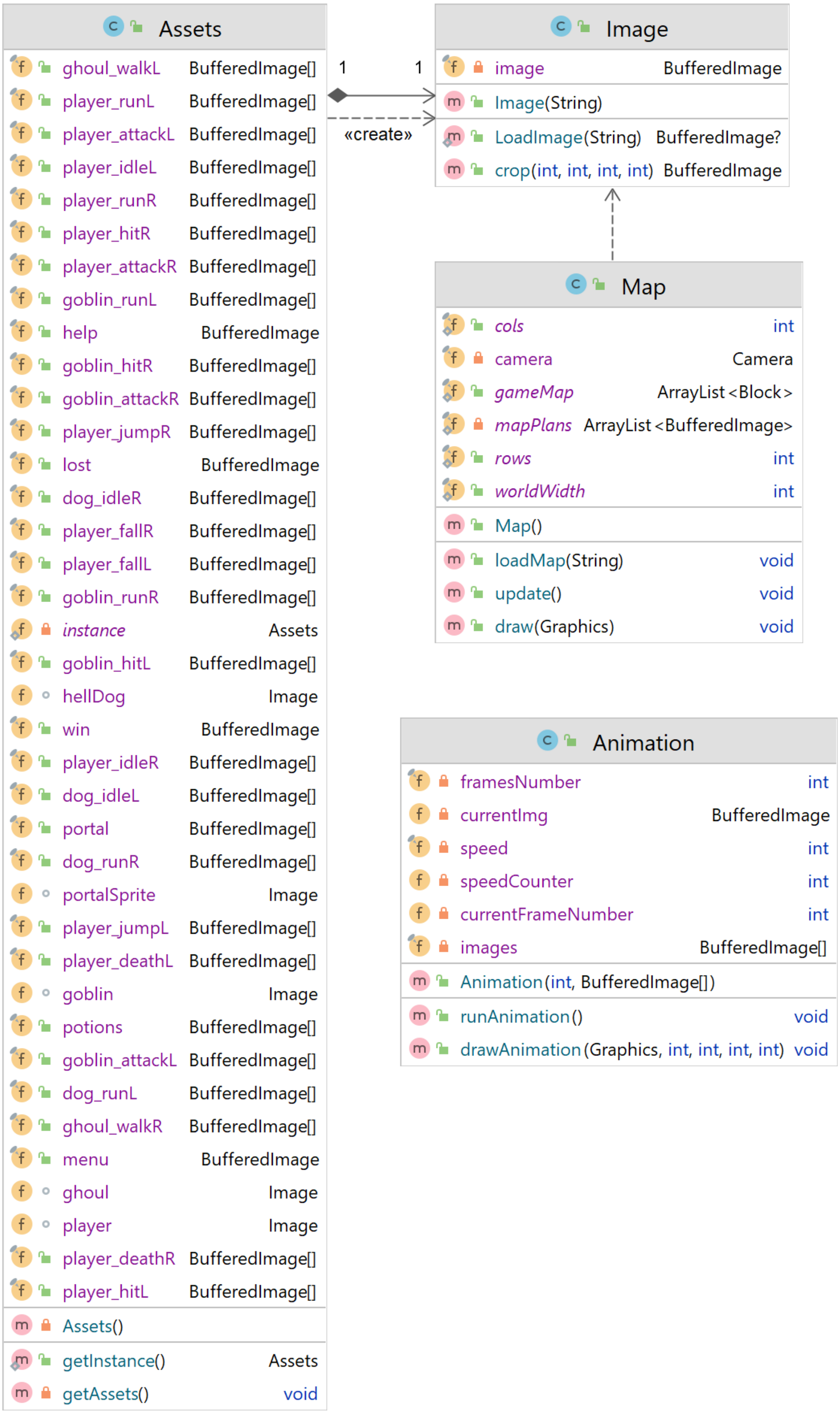
UML Diagram for the GameState Package



UML Diagram for the Items Package



UML Diagram for the Graphics Package



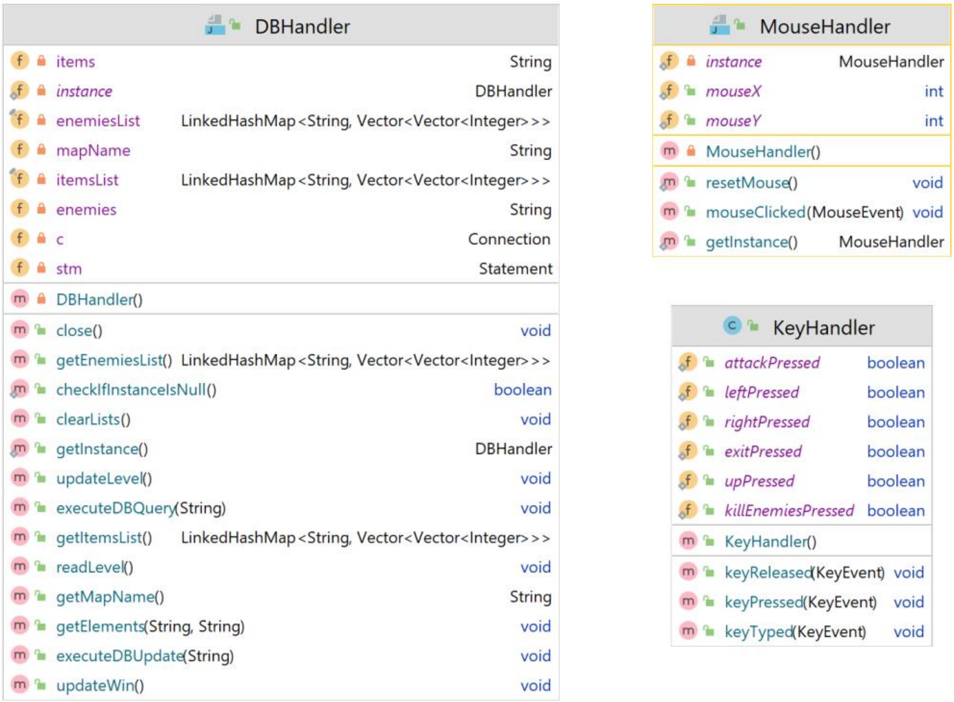
Assets	
ghoul_walkL	BufferedImage[]
player_runL	BufferedImage[]
player_attackL	BufferedImage[]
player_idleL	BufferedImage[]
player_runR	BufferedImage[]
player_hitR	BufferedImage[]
player_attackR	BufferedImage[]
goblin_runL	BufferedImage[]
help	BufferedImage
goblin_hitR	BufferedImage[]
goblin_attackR	BufferedImage[]
player_jumpR	BufferedImage[]
lost	BufferedImage
dog_idleR	BufferedImage[]
player_fallR	BufferedImage[]
player_fallL	BufferedImage[]
goblin_runR	BufferedImage[]
instance	Assets
goblin_hitL	BufferedImage[]
hellDog	Image
win	BufferedImage
player_idleR	BufferedImage[]
dog_idleL	BufferedImage[]
portal	BufferedImage[]
dog_runR	BufferedImage[]
portalSprite	Image
player_jumpL	BufferedImage[]
player_deathL	BufferedImage[]
goblin	Image
potions	BufferedImage[]
goblin_attackL	BufferedImage[]
dog_runL	BufferedImage[]
ghoul_walkR	BufferedImage[]
menu	BufferedImage
ghoul	Image
player	Image
player_deathR	BufferedImage[]
player_hitL	BufferedImage[]
Assets()	
getInstance()	Assets
getAssets()	void

Image	
image	BufferedImage
Image(String)	
LoadImage(String)	BufferedImage?
crop(int, int, int, int)	BufferedImage

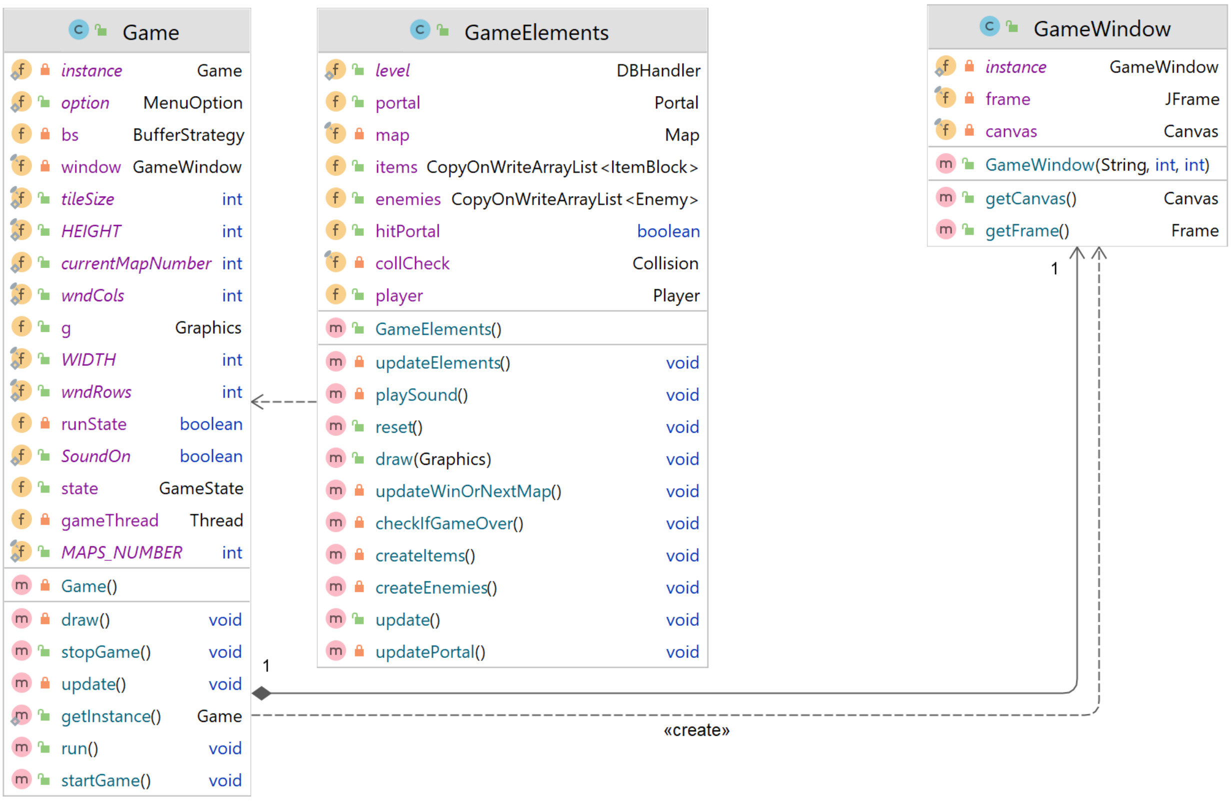
Map	
cols	int
camera	Camera
gameMap	ArrayList<Block>
mapPlans	ArrayList<BufferedImage>
rows	int
worldWidth	int
Map()	
loadMap(String)	void
update()	void
draw(Graphics)	void

Animation	
framesNumber	int
currentImg	BufferedImage
speed	int
speedCounter	int
currentFrameNumber	int
images	BufferedImage[]
Animation(int, BufferedImage[])	
runAnimation()	void
drawAnimation(Graphics, int, int, int, int)	void

UML Diagram for the Input Package


DBHandler
items String
instance DBHandler
enemiesList LinkedHashMap<String, Vector<Vector<Integer>>>
mapName String
itemsList LinkedHashMap<String, Vector<Vector<Integer>>>
enemies String
c Connection
stm Statement
DBHandler()
close() void
getEnemiesList() LinkedHashMap<String, Vector<Vector<Integer>>>
checkIfInstanceIsNull() boolean
clearLists() void
getInstance() DBHandler
updateLevel() void
executeDBQuery(String) void
getItemsList() LinkedHashMap<String, Vector<Vector<Integer>>>
readLevel() void
getMapName() String
getElements(String, String) void
executeDBUpdate(String) void
updateWin() void
MouseHandler
instance MouseHandler
mouseX int
mouseY int
MouseHandler()
resetMouse() void
mouseClicked(MouseEvent) void
getInstance() MouseHandler
KeyHandler
attackPressed boolean
leftPressed boolean
rightPressed boolean
exitPressed boolean
upPressed boolean
killEnemiesPressed boolean
KeyHandler()
keyReleased(KeyEvent) void
keyPressed(KeyEvent) void
keyTyped(KeyEvent) void

UML Diagram for the MainGame Package



Resources:

- State design pattern: https://refactoring.guru/design-patterns/state
- Singleton design pattern: https://refactoring.guru/design-patterns/singleton
- Images resources:
  - GothicVania Church Pack by ansimuz (itch.io)
  - GothicVania Cemetery by ansimuz (itch.io)
  - Metroidvania Demo [Godot] by PedroVMVictor (itch.io)
  - EVil Wizard 2 by LuizMelo (itch.io)
  - https://lornent.itch.io/heart-shaped-potion-bottle
  - https://elthen.itch.io/2d-pixel-art-portal-sprites
  - https://nectanebo.itch.io/menu-buttons
- Sounds resources:
  - https://ellr.itch.io/universal-ui-soundpack
  - https://mixkit.co/free-sound-effects/coin/ - Fairy arcade sparkle
  - https://mixkit.co/free-sound-effects/game-over/ - Fairytale game over
  - https://mixkit.co/free-sound-effects/fairy/ - Ethereal fairy win sound
  - https://psionicgames.itch.io/10-scary-horrortunes-pack-1
- Algorithms:
  - https://youtu.be/Icd2gAHDSfY - for detect collision with the map elements
- Other programs used for this project:
  - Photoshop – for creating and editing the map images