Cyber Sword Game

developed by Püspök-Kiss Balázs

Documentation

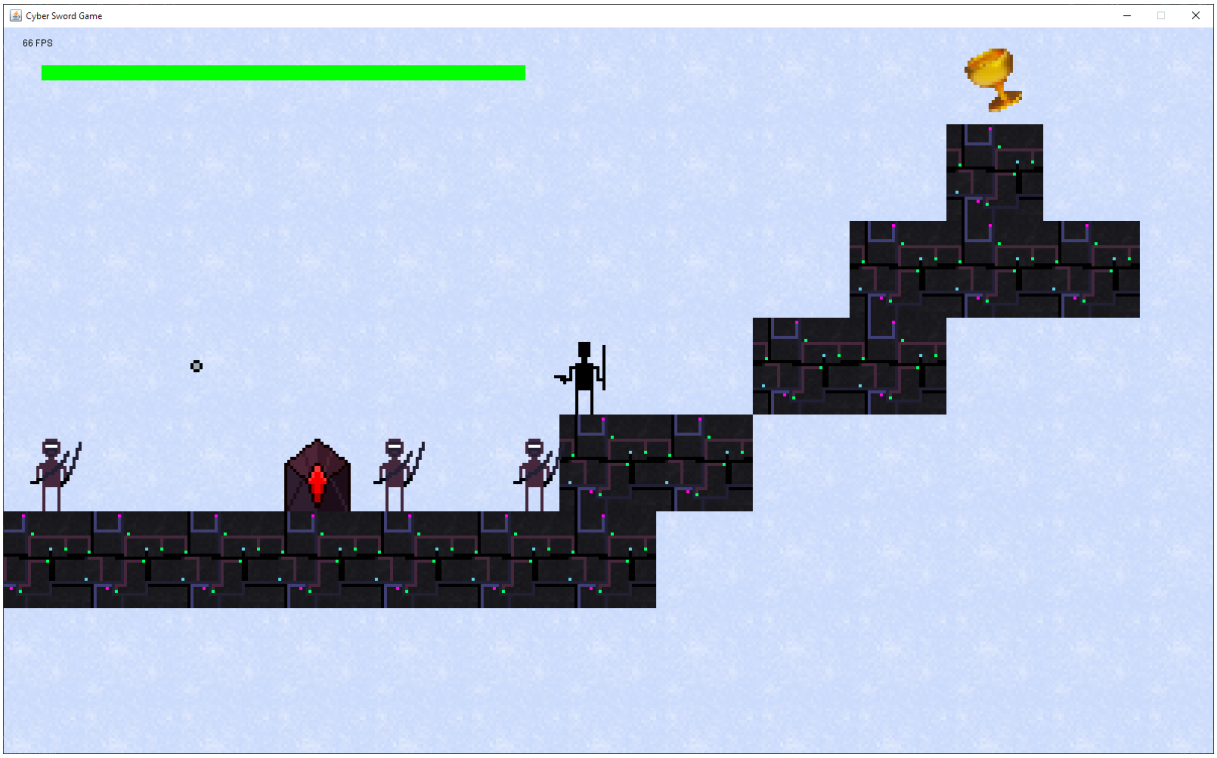
Contents

Task description ... 3
Editor description ... 4
Game description ... 6
Beginning of the game ... 6
Ending of the game ... 7
Losing ... 7
Winning ... 7
Use Cases ... 8
Game's use cases ... 8
Control Player ... 8
Move Player ... 8
Shoot Bullet ... 8
View Map ... 9
Control Enemies ... 9
Spawn Enemies ... 9
Move Enemies ... 9
Editor's use cases ... 9
Setting size of the game map ... 9
Placing actors onto Canvas ... 9
Saving ... 9
Loading ... 9
Class Diagrams ... 10
The Actors of the game ... 11
GUI Classes ... 12
Summary ... 13

Task description

The user controls a ninja warrior in a 2D platform game. The ninja have to get to a place (Chalice) inside the game's map, to win the game, but unfortunately, enemies stands in his way, with whom the player has to get through.
The ninja acquired a very handy pistol, which is able to shoot some bullets. These bullets can kill the enemies, helping the ninja clear the way to get to his destination.

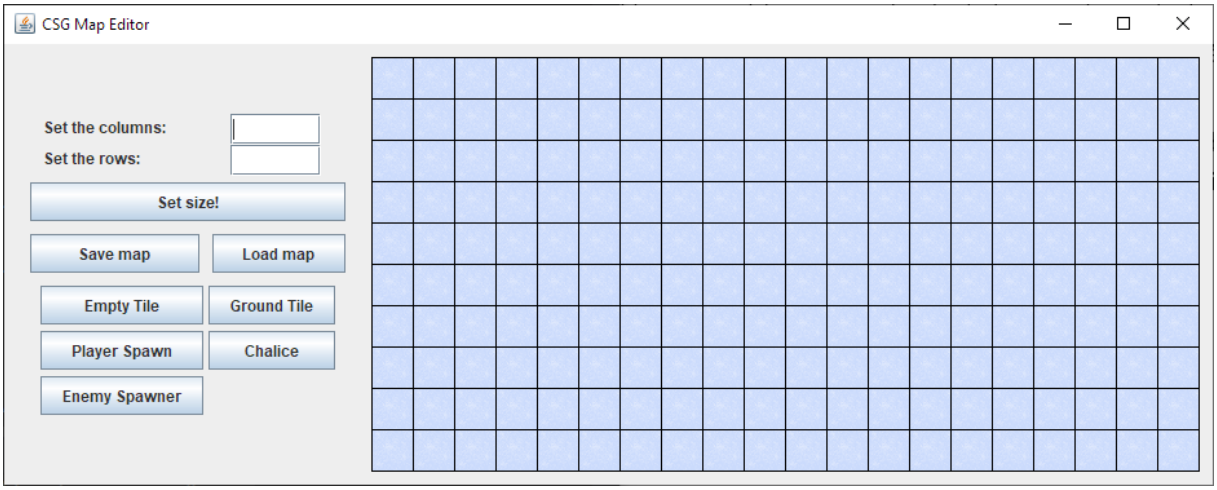
The enemies' goal is to kill the ninja, so they follow him everywhere. If the enemies kill the ninja, then it is game over for the player. The enemies can damage the ninja by punching him in close proximity.



1 – A screenshot of the game. On the north-east corner is the Chalice, the ninja has to get to.

Editor description

The user is able to make unique maps for the game, using the given map editor.

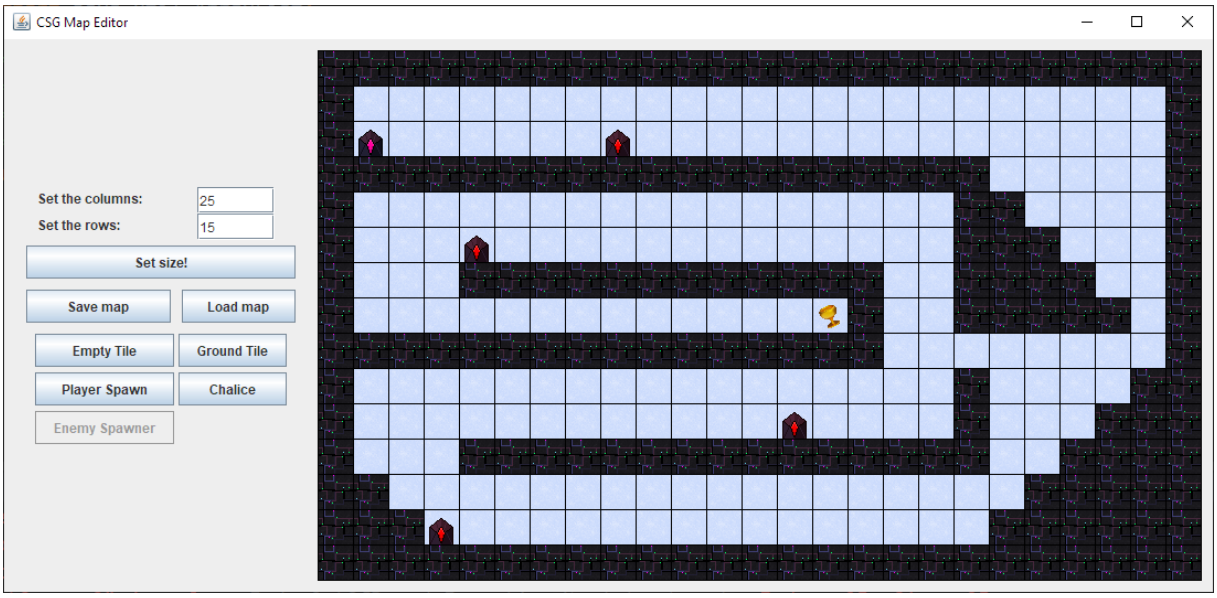


2 – Freshly opened map editor, filled with empty tiles.

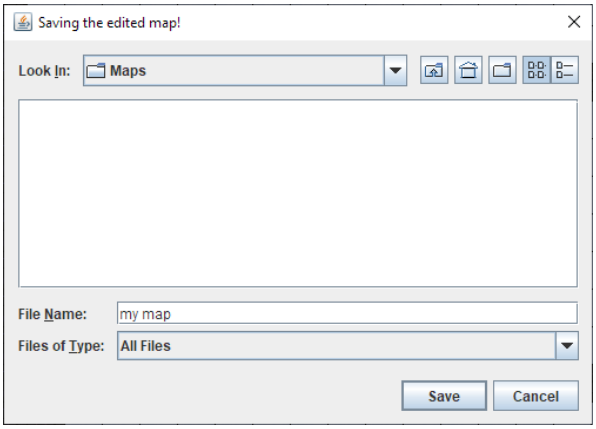
The user can paint tiles onto the game map in the editor by pressing and dragging the mouse across the game map's canvas.
Although some elements of the editor's elements behaves differently.
For example, here only can be (and must be) one Player Spawn on the map, but enemy spawners' and Chalices' quantity is endless.

The user is also able to save the map using the "Save map" button, where the user can choose a location where to save the map. The maps are saved using the Serializable interface. The user is also able to load in maps, so they can refine the experience even more.

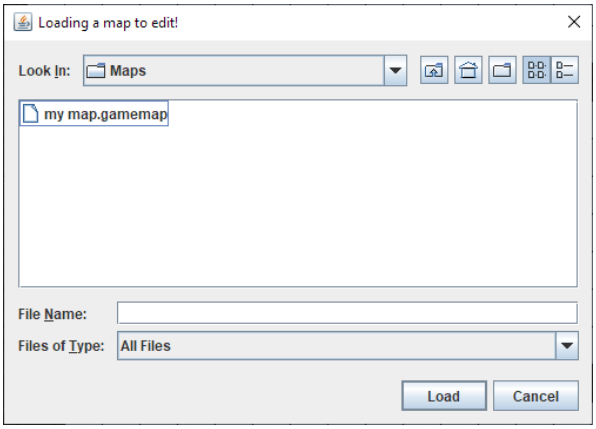
Also, the user can set a custom size to the map using the "Set size!" button, don't forget to set a proper value into the columns and rows text field! The editor will notify you if you set those values too high or too low.



3 - A well edited map, with custom size and all elements inside.

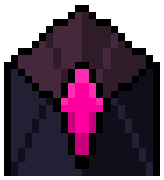


4 - Saving the game map.



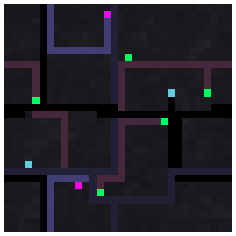
5 - Loading in a game map.

6 - Enemy spawner.



7 - Player spawn.

8 - Empty tile.



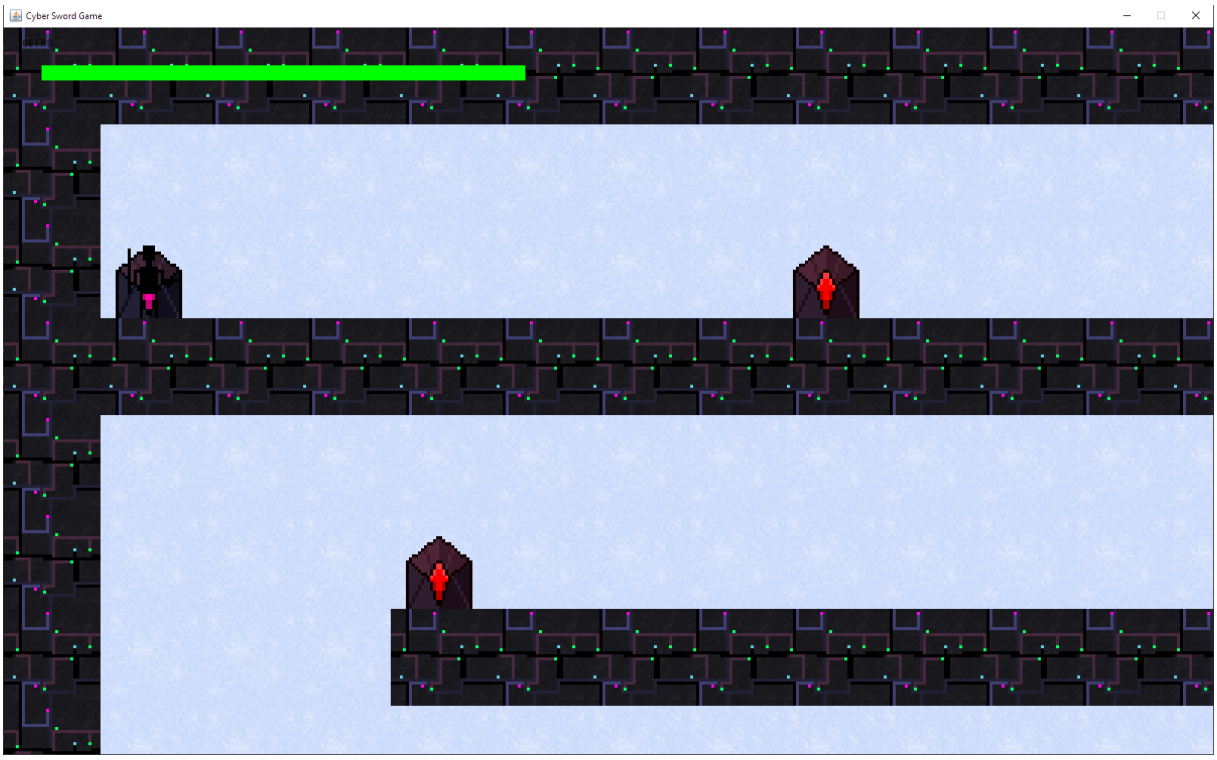
9 - Ground tile.

10 - Chalice.

Game description

Beginning of the game

If the user wants to play a game, first, the user has to choose a game map to load in. Game maps can be made in the editor. They have to contain 1 player spawner to spawn the player.



11 - The player spawns at the player spawn. Notice, that the camera clips onto the sides of the game map.

The player spawns at the player spawn at the beginning of the game. The user can control the player. He can jump with ‘Space’ and go side by side with the ‘A’ and ‘D’ keys.

If the player is close to the enemy spawners, then they activate and spawn enemies periodically. These enemies then follow the player.

The player’s health bar is at the north-west corner of the canvas, pictured by the green bar. It indicates if the player’s health is low or full, in picture 11, it is full.

Entities which are the player and the enemies, can collide with the ground, so they can stand on the ground and move. Entities can also jump, and while not grounded, they fall down by gravity.

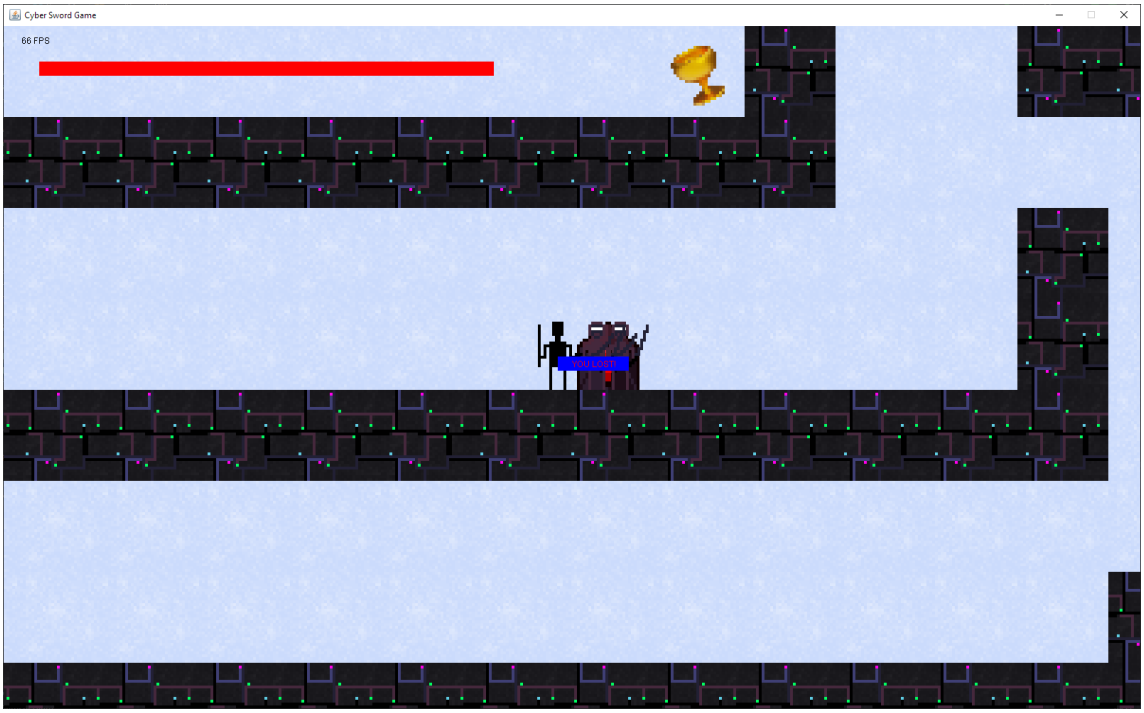
The player can shoot a bullet, which is destroyed upon impact. If the bullet hits a ground tile, then it is just destroyed and gone from the scene. But if it hits an enemy, then it deals a damage to the enemy. If the enemy suffers critical damage, then it dies.

This chapter kind of describes the use cases of the game elements as well.

Ending of the game

Losing

If the player dies from the enemies, then the user lost the game. The health bar is full red, indicating the player has no health points left.



12 - The player died... Notice how the camera follows the player on the middle.

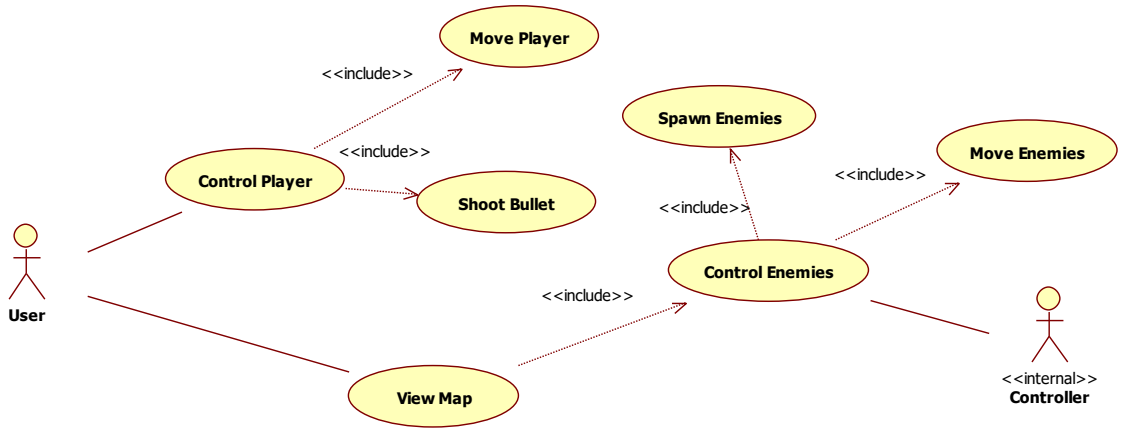
Winning

(I discovered a little flaw of the map, so using the editor tool, I fixed it. The flaw was that the player couldn't reach the chalice because the jump height is around 2 blocks and not 5.)

Now, the player reached its destination and the user won.

Use Cases

Game's use cases



Control Player

A player can suffer from a hit, meaning its health points go down by a damage amount. Enemies can damage the player by getting close to him. Although the player has an invincible window, while he can escape from taking another amount of the damage from an enemy.

If the player suffers critical damage, he dies and the game ends.

If the player reaches a Chalice alive (collides with it), the game is won by the user and the game ends.

Move Player

The use is able to move the player with his keyboard. Pressing 'A' moves the player left, pressing 'D' moves the player right. If a ground tile wall stands in the way, the player collides with it and the wall doesn't let it move forward.

The player is able to jump by pressing 'Space'. If the player jumped, he loses its grounded state and cannot jump unless it is grounded again. The player gains vertical velocity (in the Y axis) and the gravity starts affecting him. That means, that the player loses its initial velocity and starts falling down. If landed on a ground tile, then the player collides, and the player gets the grounded state.
If the player moves off an edge, the empty block, in which the player is fully in removes its grounded state if the block under it is empty.

Shoot Bullet

The user is able to shoot bullets from the player by pressing the left mouse button of the mouse. The direction of the bullet depends on the cursor position relative to the player position. If the cursor is on the left side of the player, then the bullet will go left from the player for example.

If a bullet reaches a ground wall or enemy, then it does something to it and gets destroyed. If it collides with an enemy, then the bullet deals damage to it then gets destroyed. If the bullet hit a wall, then it just simply gets destroyed.

View Map

The user is able to view the map and what the map contains. The player character, enemies, player spawn, enemy spawners, tiles and the Chalice all gets drawn on the display in their positions.

Control Enemies

An internal controller takes care of controlling the enemies. The enemies set the player as their target, and they all start to follow him. An enemy can be damaged and if it suffers critical damage, then it dies and disappears from the map.

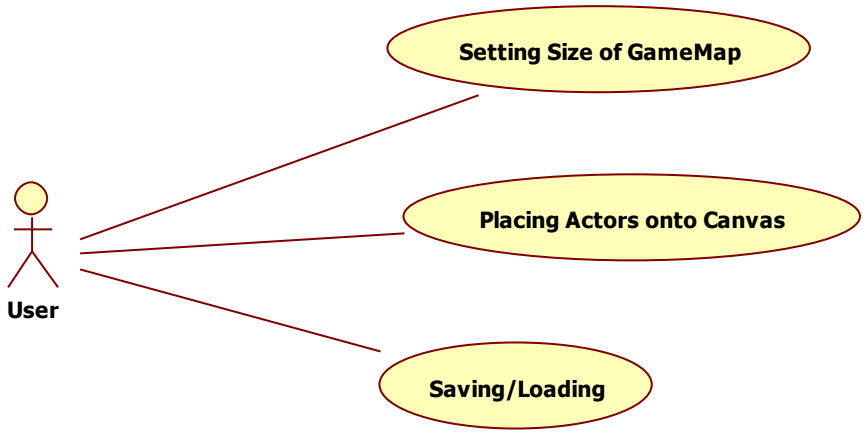
Spawn Enemies

The internal controller (in this case an EnemySpawner) decides if it should spawn an enemy or not, every tick (in game update). If yes, it created an Enemy at the spawner's position.

Move Enemies

The enemies can move left and right and have the same collision methods as the player has (meaning they can fall and have a grounded state, handled by tiles, also can collide with walls in the same manner the player does).

Editor's use cases



Setting size of the game map

The user is able to set the size of a game map with restrictions. The number of rows and columns have to be between some values. If the player makes a mistake, then the program will notify him in a dialog message.

Placing actors onto Canvas

The user is able to use his brush to "draw" actors onto the game map. He can draw multiple actors onto a map with the appropriate brush.

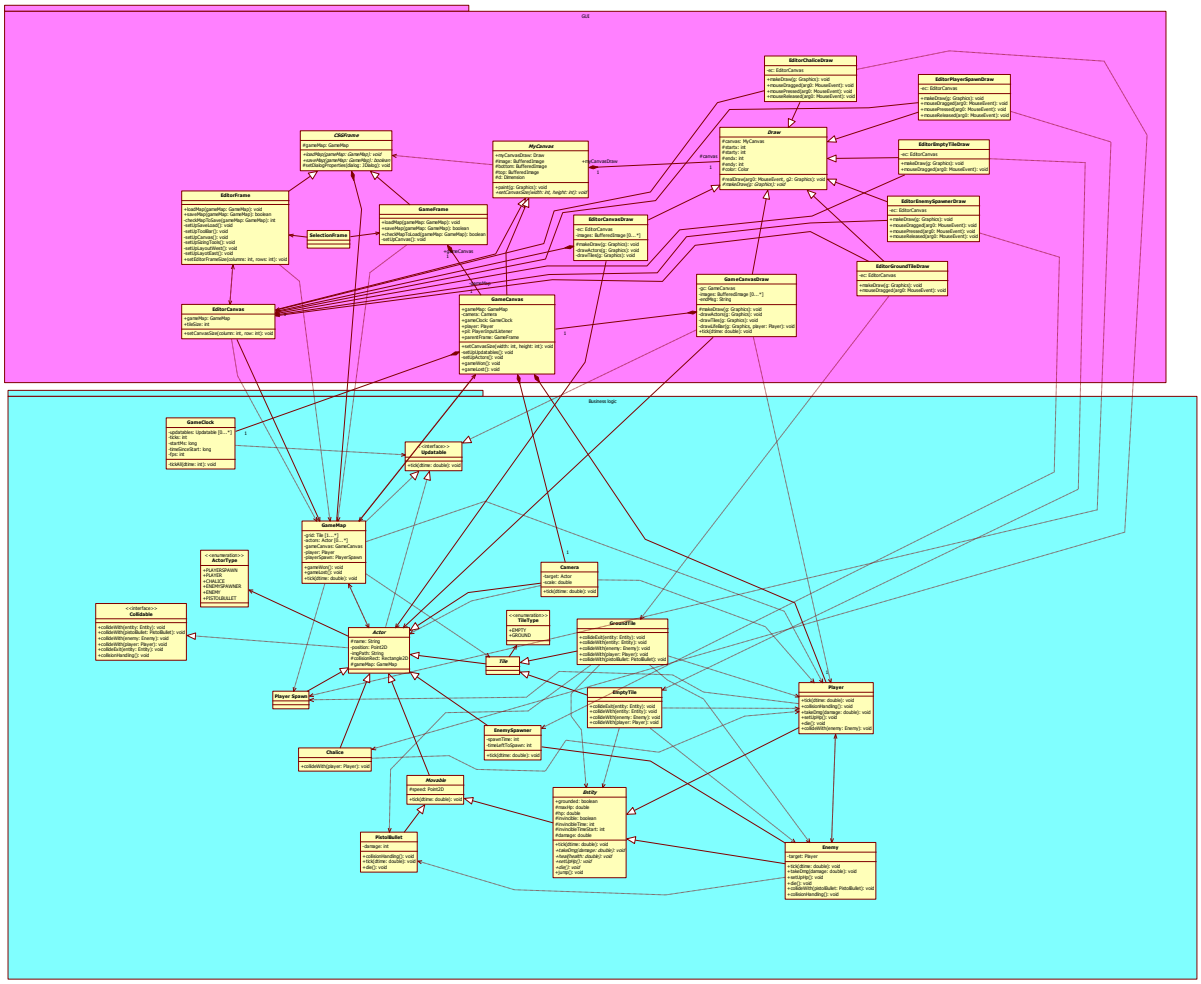
Saving

The user is able to save his progress, after he finished drawing the game map with the actors.

Loading

The user is able to load in a previously saved map into the editor. The loadable maps are ending with ".gamemap".

Class Diagrams

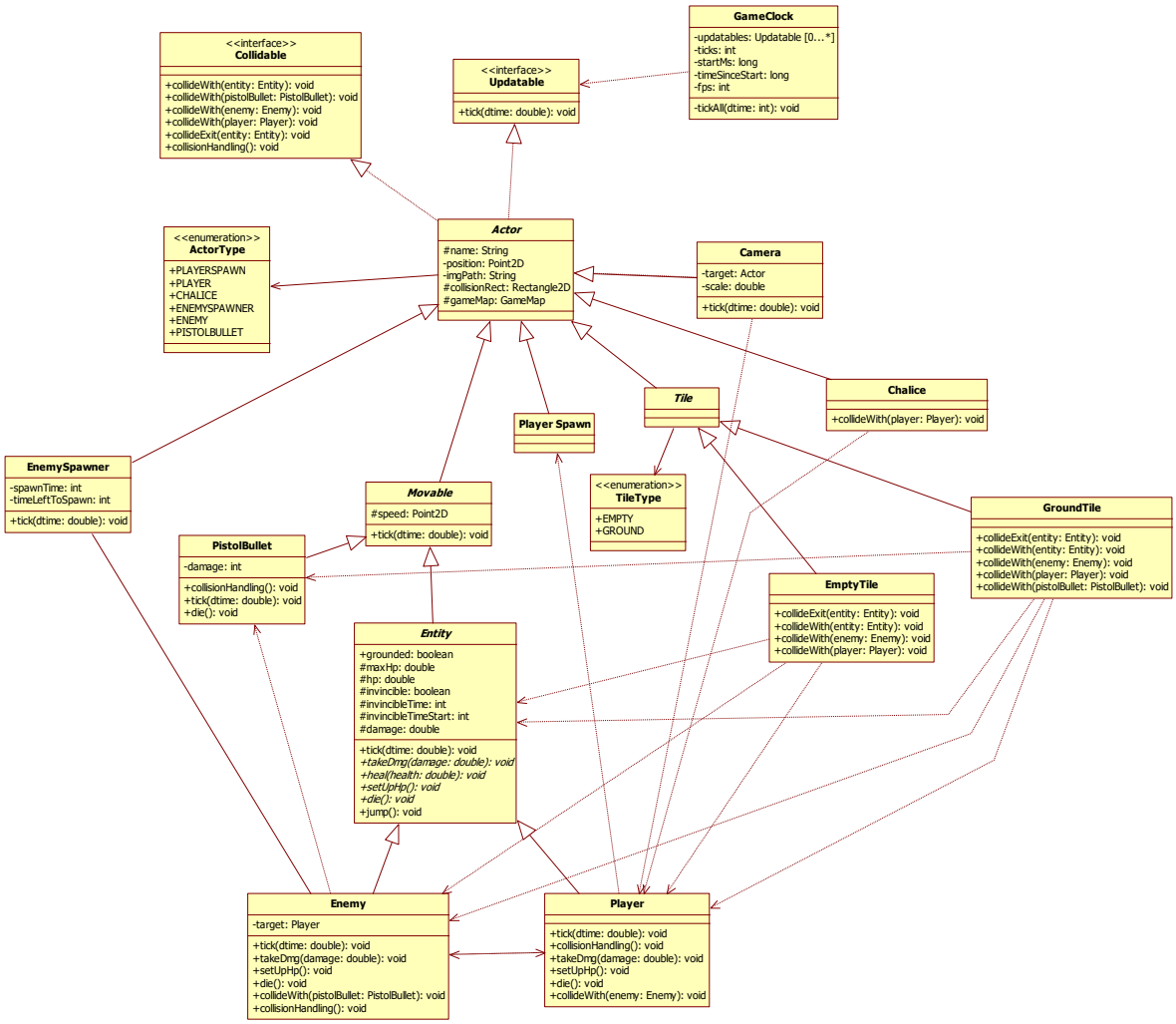


13 - the whole project's class diagram made in WhiteStarUML. A more refined version is available in the project files.

Whole Project.png

The Actors of the game

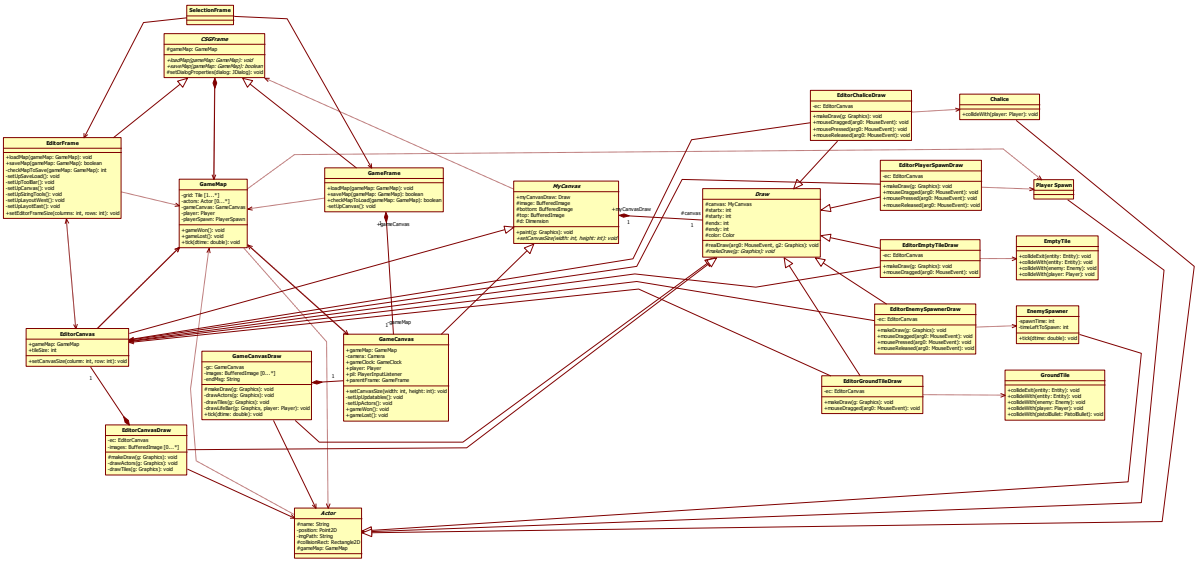
The actors are key elements to the game, as they are the thing which interact with each other. They almost deserve a full package.



14 - Game Actors.png

GUI Classes

These classes draw the elements onto the screen and from these classes forms the Graphical User Interface.



15 - GUI Classes.png

Summary

More description of the classes and structures are available, I JavaDoc-d the whole source code, and generated the JavaDoc documentation.

All of the pictures in this documentation and more is available in the document resources folder.

(And for last, I know, Cyber Sword Game didn't implement swords. ☹.)