Swording around

Project title	**Swording Around**
Author(s)	**Hulpe Miruna, Stoica Tudor**
Group	**30423**

1. Task description

Our game is a 2D multiplayer game made in java and java swing. The purpose of the game is to attack and eliminate other players. The characters are robots, and their weapon is a sword. They can be moved using the arrows. The app consists of 4 frames. The first 2 are a sign in and a sign up menu. The third one is a menu where you can select the type of game mode you want to play (being just a beta version, for now you can play only one game mode). The last frame is the game.

In order for the application to work, the Server app needs to be runned first, as it commands the database and sends and receives data across all players.

Whenever a user starts the SwordinAround app, a mini server client opens; this server client is intending to send the user's credentials across the server, waiting for validation.

In this way, if the player presses the login button, a DatagramPacket will be sent to the server. The data will be extracted, the intention of the packet will be calculated and if the credentials fit the AppData object (this object memorizes all players and their credentials), a DatagramPacket, along with a message which enables the sign in will be sent back to the user. In the case of sign up, no verification will be done -> the data will be introduced in the database right away. The database stands on the server side and an AppData object with all players credentials will be created, by the time the server opens.

Since we want to further develop this project, the Server contains several lobbies. Whenever a player presses the START GAME button, a DatagramPacket with the information REQUEST_TO_JOIN_GAME (due to time constraints, I implemented just the 1 v 1 game type, more on that later) will be sent to the server. The server will analyze the packet and look for available lobbies. If no lobby is found, the server will create a new lobby. The player will be then sent a SUCCESSFUL CONNECTION packet which will enable it to start the Game class.

Due to performance issues, every game instance will be processed by each player's computer, thus not over requesting server's capacity.

The server acts just like a mailbox. Whenever a player moves or swings its sword, it will send this information to the server. Whenever an in game information is sent from a player to the server, the server will send that information to all the other players. We are doing this, because I do not want the game to be processed on a single cpu, instead, we want every game to be processed on every user's pc. Server's only job is to receive and further deliver datagram packets. Of course, I intend to assign in game tasks for the server to, for instance, to analyse whether the game must finish or not, but we will leave this feature for the future.

When a game starts, the lobby becomes the server for the players. The lobby will be responsible for the mailbox task. Listening to DatagramPackets is done on separate threads. We are using the UDP protocol, since it works the best for video games.

The game seems to become laggy when the second player joins the lobby; this is the reason why we designed the game as a 1 v 1 duel: more players seem to render even a intel core I5 gen 9 to 100% and a nvidia geforce gtx 1650 to 30%. Maybe the java swing method for painting graphics is not suited for video games. Intriguingly, the second player in the lobby seems to have an effect on the input control -> the robot is no longer 100% responsive to inputs.

Due to simplicity terms, the game will stop when a player gets killed.We have used a Timer which repaints all the graphics, once at 10ms. Also, the players move() and send packets to the lobby (server) every 10ms.

Regarding the task management, Tudor worked on the connection (server) part, and Miruna on the database, UI and character movement. In the end, Tudor connected all the parts of the project, in order to work and fit together.

2.1. Features:

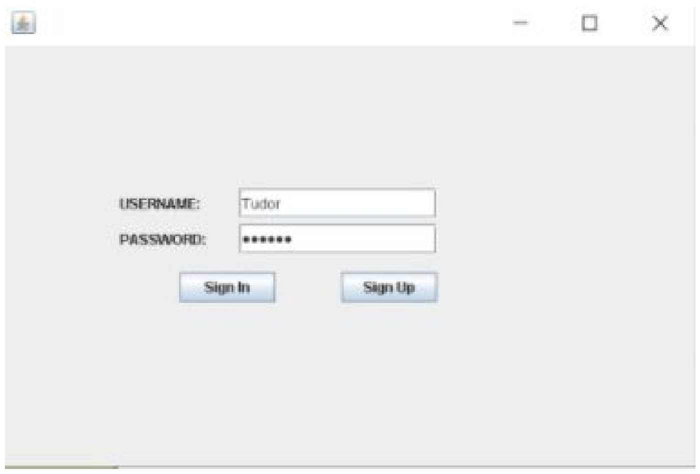
- Create an internet connection between players: create a lobby with panding players
- Make the database.
- Make the connection to the database.
- Making the character moving on the map.
- Making the character dash.
- Making the character swing it’s sword.
- Making the character shoot.
- Perform interaction between players. (Damage on hit)
- Create skins for characters. (For now we only have 2 colors - blue and black)
- Create map skins.(For now we only have one)

2.2. Future features:

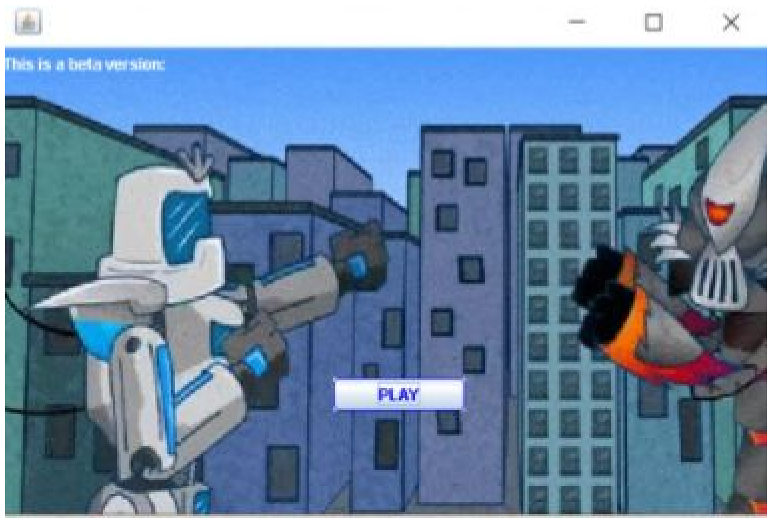
- If a lot of time has passed and 10 real players didn't ghater in the lobby, summ bots.
- Implement AI to bots.
- Create map skins.
- Animation skins.
- Implement game sounds.
- Option menu with control change.
- Log in portal.
- Statistics(total kills, average kills)

3. Screenshots from the app

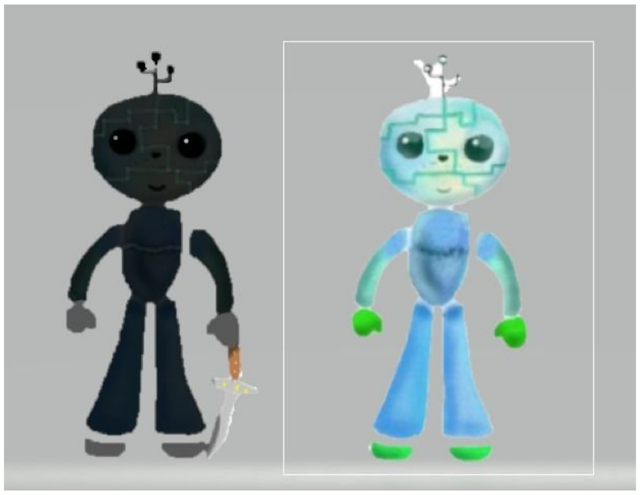
- Sign in menu



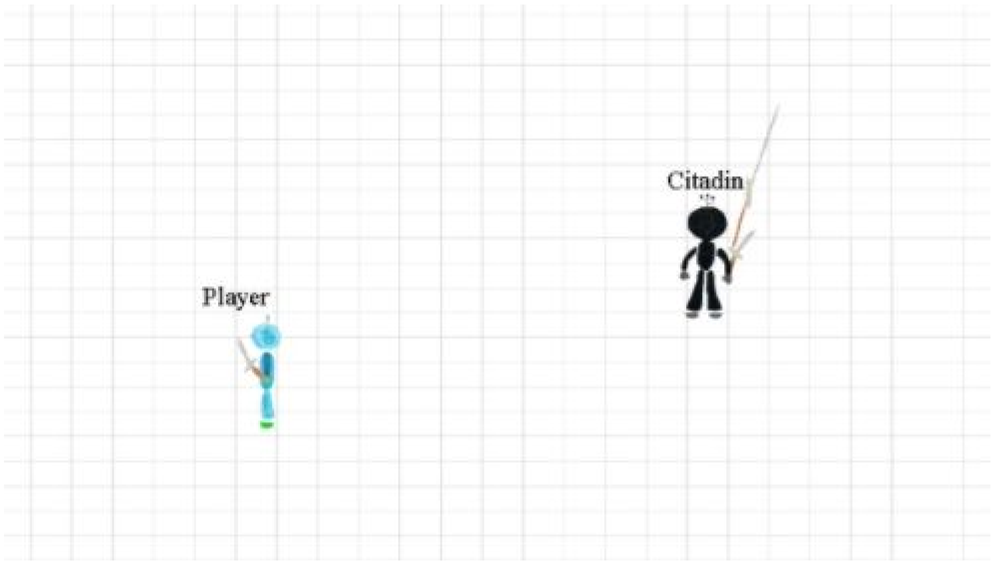
- The main manu:



- Players:



- Game mode:


Citadin
Player