C++ project
Towerfall: Ascension
Documentation

Perttu Yli-Opas 590947
Joonatan Bergholm 507260
Essi Rantanen 589369
Jaakko Visti 530907

December 2017

Contents

1 Introduction

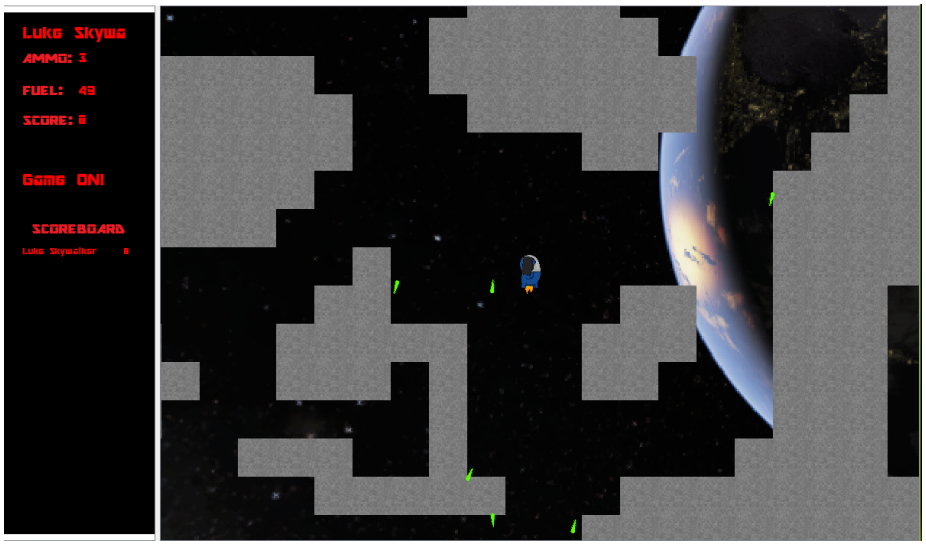

Figure 1: Main view of the game.

C++ towerfall ascension project -game is an online multiplayer 2D shooter platformer game (figure 1). The goal of the game is to hit as many opponents with green projectiles as possible while trying to avoid hitting them yourself. After 5 rounds of hectic combat winner will be the player with most kills.

2 Setting up the game

Before beginning the game players need to do some setting up. In this section we'll tell you what that is.

2.1 Server

One of the players needs to be the host. Host is responsible for setting up the server for other players to join in. After the host has executed server program a window should open (figure 2). In this window there are 2 buttons and 3 useful fields of information.

1. Ip address

Host's ip address needs to be shared with all players for them to join.

2. Connected players

Shows the names of all players that have already connected to server.

3. Start-button

Starts the game with current players. Once the game has been started no more players can join the game and button itself turns into pause/continue -button.

4. Regenerate map -button

If players are not happy with current map they are playing they can randomize a new one (this recommended to do before game starts or while it's paused).

5. Minimap

Shows silhouette of current map and position of all players (red dots).

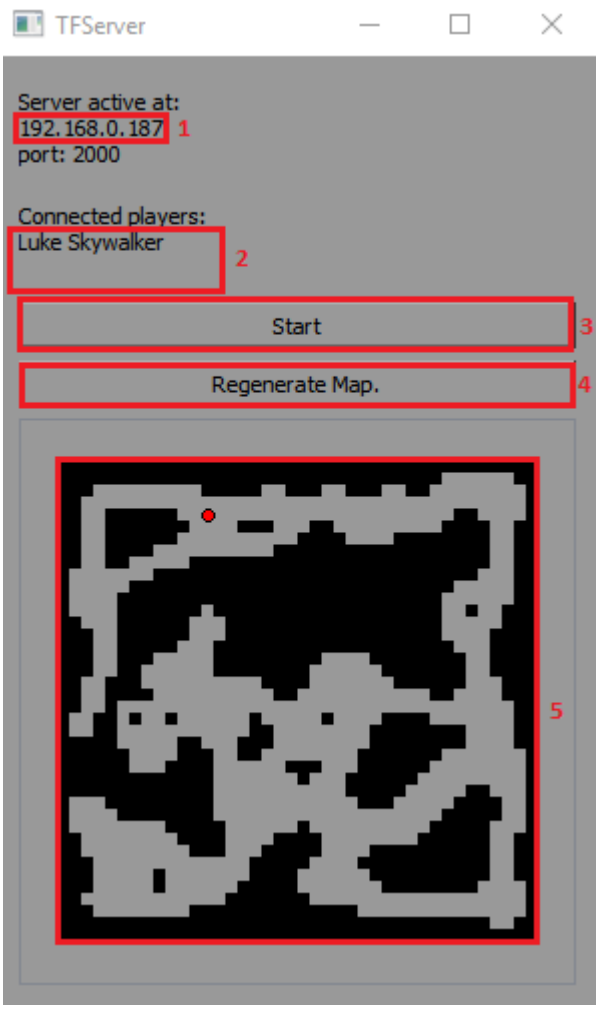

Figure 2: Server's setup window

2.2 Client

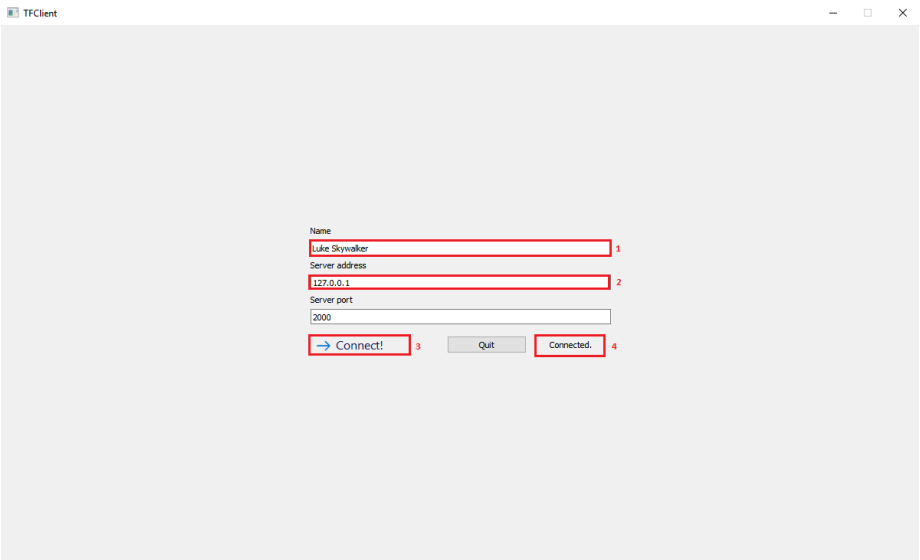

Figure 3: Client's setup window

Whilst only one player needs to execute the server program, all of them need to do it for client program. After executing a connection window should appear (figure 3.

1. Name

Write your name here before connecting(3).

2. Ip address

Write the ip address shared by the host here. Host themself can use the default address.

3. Connect-button

Once you haved entered your name and serever's address you are ready to join the server by clicking this button.

4. Connection status

Shows the current status of your connection. If this says connected you can just wait for the host to start the game.

3 Gameplay

In this section we'll tell you how you'll actually play the game once the set up is done.

3.1 Rules

There are 5 rounds in each match and the winner is the player who gets the most kills during these rounds. Before each round and after each pause there's a three second countdown for each player to get ready. Players can shoot green projectile's which will bounce few times off the wall's before stopping. They are dangerous and can cause a very brutal death to any player that touches them even after they have stopped. If player gets hit by a projectile they'll be dead for the rest of the round (figure 4). Player's can move, jump and use jetpack to avoid these projectiles. Ammo and jetpack's fuel are limited but they will regenerate over time. When there's only one player alive the round will end, player's are given new starting positions and a new round begins.

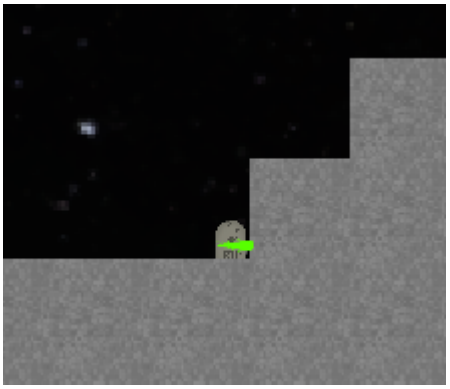

Figure 4: If player gets hit by a projectile they'll be dead for the rest of the round.

3.2 UI

As can be seen from the figure 1 there are to main parts of the UI. Part on the left (infobox) shows player information about the current game state including how much ammunition and fuel the player has, what's their score and the overall score situation in the game. Part on the right is the canvas which shows all the player interactions in the game.

3.3 Controls

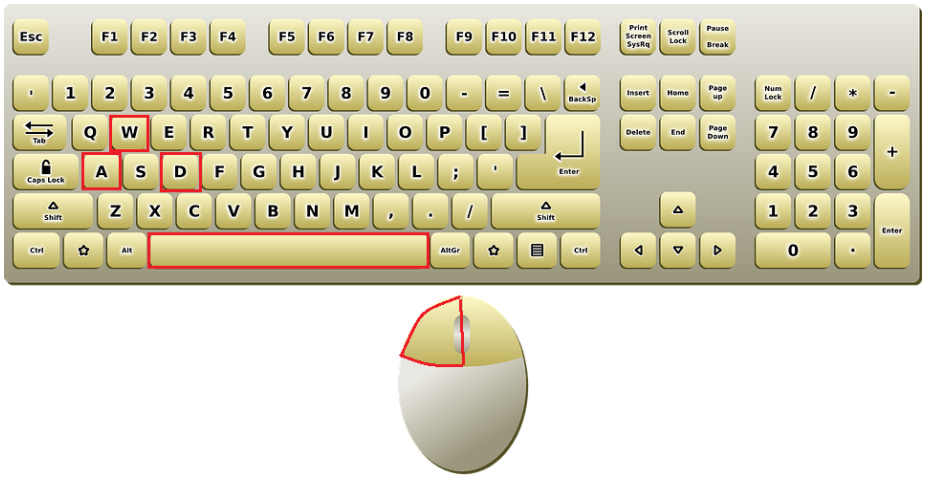

Figure 5: Controls presented on a keyboard

W	use jetpack
A	move left
D	move right
Spacebar	jump
Left mouse button	shoot

4 Structure

The program consists of a server and a client program. The connectivity between the programs is implemented using TCP sockets.

4.1 Server

The server handles all game mechanics. It receives players' keyboard and mouse actions and runs the game accordingly. It sends all object info to all clients. The architecture of the server is shown in figure 6.

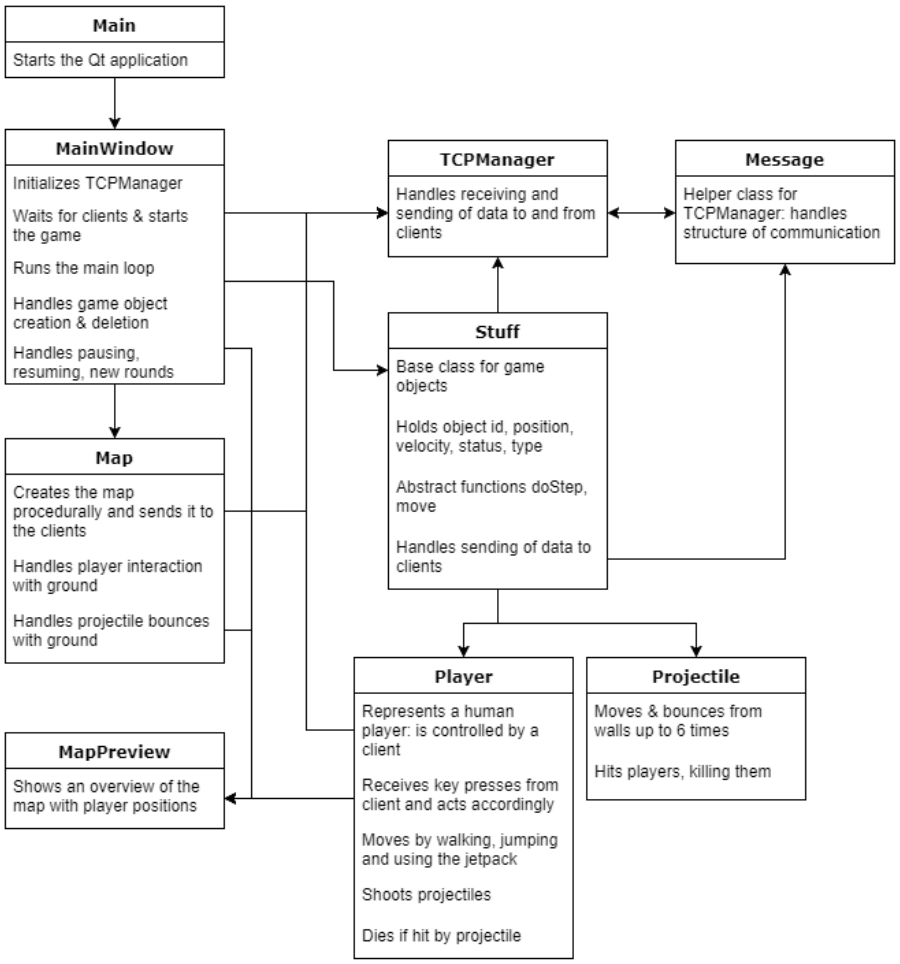

Figure 6: Server-side architecture

4.2 Client

The client is used for playing. All players play on different computers, being connected to server through the internet. It shows the game to the player, showing player's own character with other players' characters and other objects on screen with the game world as a background. It reads the keyboard and mouse and sends the player's actions to the server.

The structure of the client is shown in figure 7.

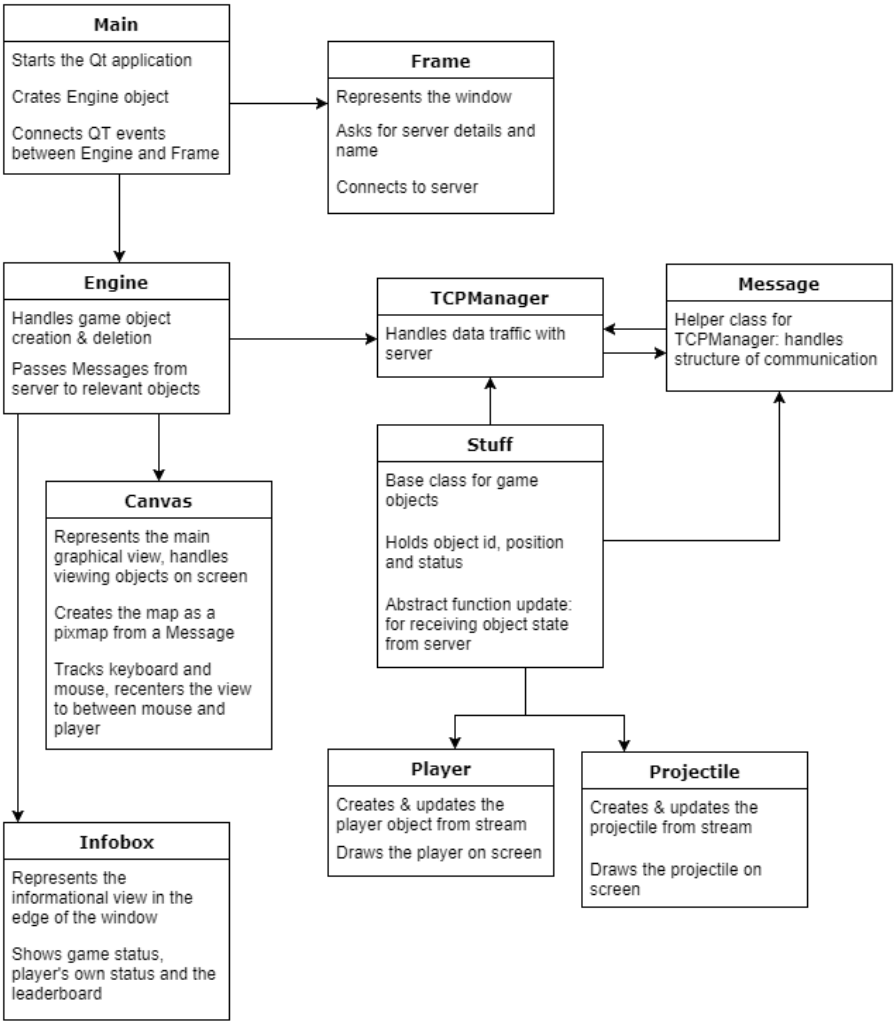

Figure 7: Client-side architecture

5 Compiling the program

```
# Building

Building our software is a simple process.

1. Goto either TFClient or TFServer folder.
2. Run `qmake`.
3. Run `make`.

Now you have finished building either client or
server depending on which folder you chose. To build
the other one, repeat the process, but choose the other folder.
```

6 Work log

```
###### Week 1
Basic structure for server and client.

###### Week 2
Player class.

###### Week 3
Constructing the map.

###### Week 4
Moving.

###### Week 5
Projectile class.

###### Week 6
UI and gameplay improvements.

## Workload
| Jaakko Visti   | Joonatan Bergolm   | Perttu Yli-opas   | Essi Rantanen   |
|:--------------:|:------------------:|:-----------------:|:---------------:|
| 50h            | 50h                | 45h               | 40              |
```

Figure 8: Work log

Figure 8 shows how the work was divided between each week. Everything started from creating the basic structures and after that the basic classes for the game was made and gradually different classes were created to form a complete game.

Perttu and Joonatan were responsible of the basics of the program, whereas Jaakko focused on player and projectile class and Essi on creating the map and

graphics. The workload for those is also represented in figure 8. However, the work hours don't directly correspond the amount of hours that was spent to write the actual code. Due to the fact that this group consisted of people with different coding backgrounds and for some Qt was already familiar whereas for some it wasn't. That is why for example a great deal of Essi's workload consists on searching information and studying to do graphics and using Qt which doesn't necessarily show in the amount of code made for the game.

7 Testing

Testing of different classes was done by running the program and using qDebug().