Distributed game - user documentation

Tadej Vozlič
UP FAMNIT, RIN
Koper, Slovenia

Tilen Jesenko
UP FAMNIT, RIN
Koper, Slovenia

Domen Vake
UP FAMNIT, RIN
Koper, Slovenia

1 INTRODUCTION

The document includes the documentation for the project, developed for the RIN course "Project seminar". The project that was implemented is a distributed version of the Atomic Bomberman game, that was introduced to the market in 1997. The main difference in the implementation concepts of the original game and the game presented in this document, is the client-server architecture.

The project does not require a central server to function. The game is completely distributed and every player presents a node in a distributed network. Every node is both a server and the client. Players that play the game are all equal and communicate through a distributed message passing algorithm to exchange information.

The game includes a move verify agent that checks for validity of every move of every player to see weather any of the players are cheating.

2 PREREQUISITES

The game is containerised within a docker environment. Docker allows us to distribute the game to any machine and guarantees a equal working environment for all clients. To install docker run the following command:

```
sudo apt install docker-ce docker-ce-cli
```

The installation command is usable in all Linux environments. If you are doing the installation on windows, consult the documentation found on https://docs.docker.com/.

The second prerequisite is installation of git. Similar to the docker installation you may use the *apt* package to install git, using the command below:

```
sudo apt install git-all
```

Once the installation of *git* has completed you may download the porject from the GitLab repository.

The repository download is available via the command:

```
git clone https://gitlab.com/t33ll3n/p2p-
    bomberman-clone.git
```

3 RUNNING

To run a client, first navigate into the folder structure of the cloned project:

```
cd p2p-bomberman-clone
```

Once within the folder, use the script, that comes with the project to run the client. But first before running the script, you have to give the script the required permissions to run on your sistem, by using the following command:

```
chmod +x runDocker.sh
```

Once the *runDocker.sh* script you may run it in order to run the client.

```
./runDocker.sh
```

You may run any number of clients (up to 1024) on your system at the same time, as long as your system has the capacity to run them. Each node requires a minimum of 20MB of ram and the consumption may increase on usage. In the case of a memory leak the maximum RAM usage of the client is 4GB.

4 USAGE

The once the game is run, we need to connect to a game. There are two ways to get into a game.

- Join an existing game
- Host a room

Joining an existing game is effectively the same as hosting a game, except the host is someone else. The game itself works and is played in the exact same manner since it's a distributed system game and all the players have the same tasks.

Upon opening a client, we are automatically connected to a lobby server. The lobby server is the only centralized part of the project, but is completely optional and is not necessary to play the game. It is only there to make the connecting to a game easier.

To connect to a game, a client has to know the IP and the port of a player, that is currently in some room. We can connect to that player and initiate a handshake protocol and we will be connected to the game.

If we don't know any clients that are currently playing the game, we can use the lobby server. If we host a game we can post the data of the game (our IP, PORT and name of the room) to the lobby server. The lobby server stores the data and will provide it to any client who asks for it. Its purpose is to distribute the data of running games to any client that is looking for a game.

We can fetch all the data from the lobby server, by writing the following command into the terminal of the running client:

```
/lobby fetchRooms
```

The lobby server will respond with a list of the running servers. That is the connection data of the hosts of rooms. Being a host is nothing more, than a public entry point into the game.

Once we have the data of all the rooms we can enter a room with the command:

```
/connect <room name>
```

This will only work if we have acquired the hosted rooms data from the lobby server. If we did not do that, we have to enter the room pragmatically, by using the IP and PORT directly.

From that point on, we can play the game as we want.

The game gives you the freedom to control one so-called Bomberman. Bomberman is a character that presents the player in the game. Your task in the game is to blow up other player, i.e. other bombermen. The environment where you are doing that is an rectangle arena. The playing board is bound from all four sides. You may move freely in the arena using the controls we will explain later. Also a player has an option of dropping a bomb, that will in a few seconds explode in a cross shape. The goal is to hit other players with the explosion reach, but not yourself.

For controlling your bomberman you may use the following controls:

- **W**: the key will move you upwards while you hold it, and stop moving when the key is released.
- **S**: the key will move you downwards while you hold it, and stop moving when the key is released.
- **A**: the key will move you to the left while you hold it, and stop moving when the key is released.
- **D**: the key will move you to the right while you hold it, and stop moving when the key is released.
- **F**: when the key is pressed, the bomberman will release a bomb on the location it is currently standing.

This will give you all the control you need to play the game.

```
\disconnect
```

When you are done playing the game, you may exit the game room, by either terminating the process of writing the above command into the client terminal.