Clash of the Elements

Distributed Systems – Project Proposal

Omar Lone, Valentin Scherer, Yan Wang, Jonas Purtschert, Miriam Tschanen
11-715-901, 12-932-380, 12-922-217, 11-949-278, 11-929-114
olone@student.ethz.ch, schererv@student.ethz.ch, yawang@student.ethz.ch, jonaspu@student.ethz.ch, tmiriam@student.ethz.ch

ABSTRACT

We present a marvelous cross-platform multiplayer game that allows up to 4 players to cooperate via local area network. The gameplay mechanics are based on the tower defense genre, where players build structures to defend a central location against hordes of enemies. We will use the Unity 5 engine, a free high-level game development framework, for our implementation.

1. INTRODUCTION

Tower defense games are very popular and there is a myriad of different variants available both for pc's and mobile platforms. They appeal to players because their mechanics are simple yet remain challenging because of the ever increasing difficulty level. Sadly, most adaptations are singleplayer games only. We think the genre could greatly benefit from a cooperation-based experience.

Our game features a top-down, 2D view and focuses on cooperation as it is a core gameplay characteristic. Each player chooses one of four elements at the start of the game. This choice influences what kind of towers they can build and what special abilities are at their disposal for the current match. Elements complement each other, thus forcing players to work together for maximum efficiency.

The player's objective is to defeat increasing numbers of enemies to keep them from reaching the player's base for as long as possible. The longer a group can hold out the more points they score.

The main technical challenge is the communication between the game clients. It is very important that the game's state is consistent across all devices at all times. To this end critical calculations are all executed on the master application and subsequently broadcast to the other clients.

2. SYSTEM OVERVIEW

2.1 Gameplay

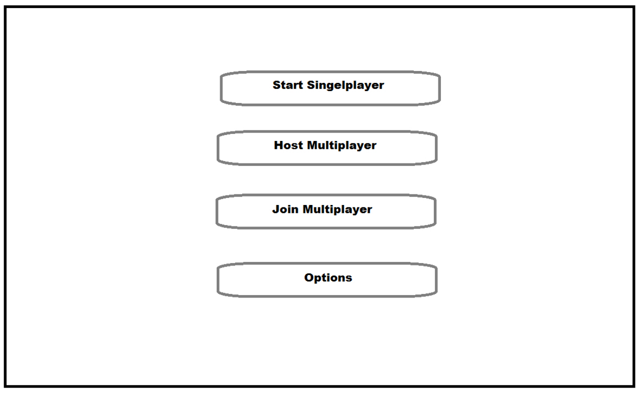

Figure 1: Main Menu

Figure 1 outlines the main menu. This is the application's starting screen which allows players to start a singleplayer game, open a multiplayer session or join another multiplayer game. Selecting "Host Multiplayer" will allow the user to select a name for the session and specify other options. Selecting "Join Multiplayer" opens the lobby screen (Figure 2) where the client lists available multiplayer sessions in the local network by name. Clicking on a name lets a player join that session. Once all players have joined, they can choose one of the four elements (Water, Earth, Air, Fire). Every element is only available once, making it impossible for two players to share the same powers. After the selection process has been completed, the host can start the game.

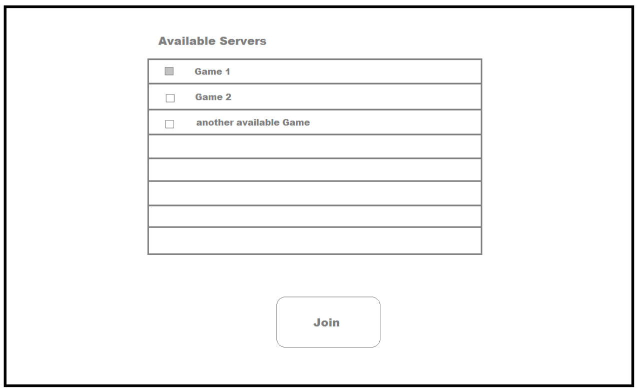

Figure 2: Local Multiplayer Lobby

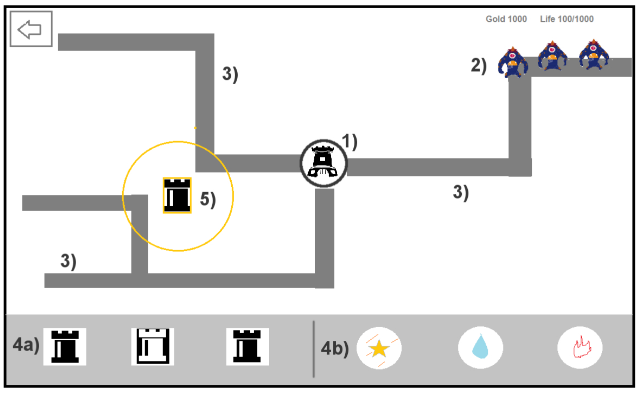

Figure 3: Gameplay Mockup

Figure 3 shows a mockup of the gameplay screen. The central structure (1) is the player's base. It must be defended against waves of attacking monsters (2). Any monster that reaches the base alive decreases the collective player live counter (as indicated in the top right corner) by 1. The game ends when the counter reaches 0. The monsters' movement is restricted to monster lanes (3) with the players' base serving as the ultimate destination. Players can spend resources (viewable in the top right corner) to build different kinds of towers (4a) or cast spells (4b). Towers are permanent structures that can attack monsters or otherwise impede their progress. Spells are instant, one-time effects that can be cast on the battlefield. Resources are earned by killing monsters. Each player has his own pool of resources but all incoming resources are split equaly across the players and a player may choose to share some of his resources with the others. Towers can be built in certain areas on the battlefield (arranged in a regular grid, excluding the monster lanes). (5) shows one such tower currently selected by the player. The yellow circle indicates the tower's firing range. A tower can only attack monsters within its range. A tooltip (not shown in Figure 3) provides additional information about the structure, such as firing rate and damage per hit. The element chosen at the start of the game determines which types of towers and spells are available to the respective player.

2.2 Network Architecture

The main challenge of this project is implementing the distributed system parts. The device on which the multiplayer session is hosted acts as the server, while all other participants run client applications. Information must be shared in such a way that the state of the game's important events remain the same on all devices. Player actions (building towers, casting spells etc.), monster health, tower firing events and loss of player lives classify as that sort of events. The exact position of monsters on the screen and particle effects for tower projectiles are not critical and can therefore be calculated locally.

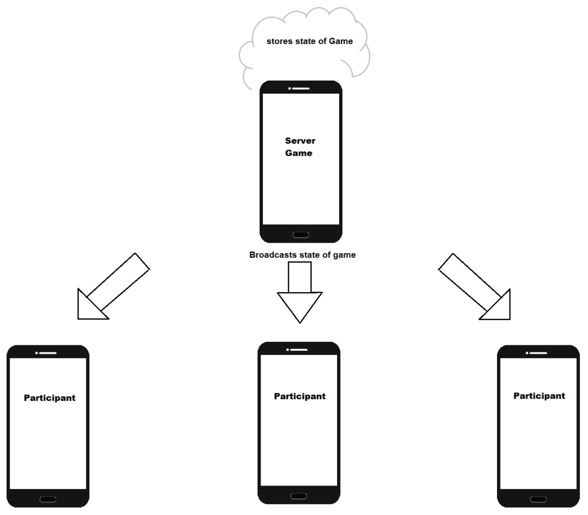

Figure 4: Broadcasting of game state from server to clients

All critical calculations are done by the server application. The game state is then broadcast to all other participants at regular intervals, using the UDP protocol for efficiency. Packet loss is not an issue because each packet contains the complete game state, meaning a client can update to the latest state even when messages are missed. The state of such an unlucky participant's session will be out of synchronization with the server until a new up-to-date packet arrives. Since the expected RTT of small packages (<20 Kb) is very short compared to human reaction speed, the resulting lag should not be noticeable to the user. Identifying an unstable or lost connection will further ensure that any network problems are handled and displayed to the user. Figure 4 illustrates a broadcast from the server to 3 client applications.

Due to the fact that there is only a single broadcast transmitter in the system, correct message ordering can be achieved with simple versioning. Clients will reject packages with lower version numbers than the latest one received.

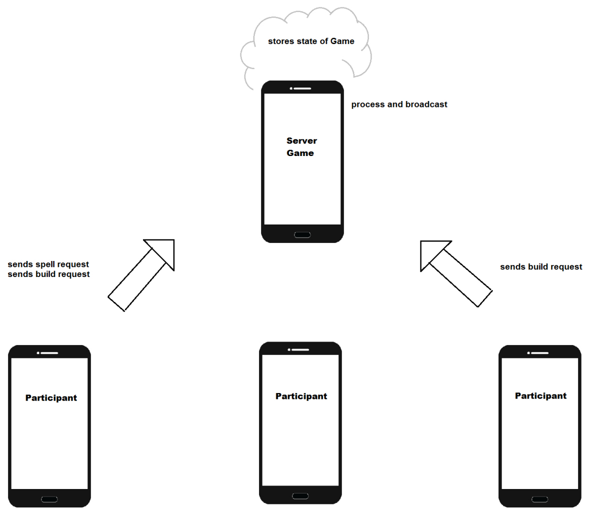

Figure 5: Clients submit requests to server

Player actions are submitted as requests to the server. The server evaluates a request, performs the necessary calculations and broadcasts the result on the next update. Request ordering is not critical, since it is relatively rare that conflicting actions occur (e.g. players trying to build a tower in the same location) at the exact same time. Requests are implemented using the TCP protocol, since packet loss would cause the game to ignore player input. The slightly higher response time is acceptable in this case. Figure 5 shows clients submitting requests to the server.

3. REQUIREMENTS

This game will be developed using the Unity3d Engine. Unity offers a useful graphics and physics library for game development and also a networking system (UN) which will be used to implement the multiplayer cooperative gaming feature. Alternatively, the Photon Unity Networking (PUN) is taken into consideration, since it offers some advantages and provides optimized socket communication. The main target platform will be Android, but due to easy cross-platform development provided by Unity other platforms could potentially be supported as well. In order to play the game an Android phone is required, Tablets and other devices will not be supported directly.

4. WORK PACKAGES

Note that the worker entries can and will probably change. These are just guidelines to who will do what in the process of developing this game.

- **WP1/Unity tutorial**: Becoming acquainted with Unity.
  Workers: All
- **WP2/Map creation**: Create a map and display it.
  Workers: Miriam, Valentin
- **WP3/Moving figures**: Get figures to follow a certain path on the map.
  Workers: Miriam, Valentin
- **WP4/Towers and Monsters**: Implement the classes of towers and monsters.
  Workers: Miriam, Valentin
- **WP5/Interacting**: Player being able to interact with the map and build towers.
  Workers: Miriam, Valentin
- **WP6/Shooting Towers**: Towers being able to choose a monster and shoot at it.
  Workers: Miriam, Valentin
- **WP7/Spells**: Allow the player to cast a spell and choose an area to release it.
  Workers: Miriam, Valentin
- **WP8/Interface**: Creating splash screen, main menu, sub menu.
  Workers: Miriam, Valentin
- **WP9/Core features of the game**: Allow the players to spend gold on various towers and select an element to enhance them. Scale difficulty and rewards with the number of waves defeated.
  Workers: Miriam, Valentin
- **WP10/Connecting and joining**: Having the option to create a game or join one.
  Workers: Miriam, Valentin
- **WP11**: Establishing a connection with main player.
  Workers: Yan, Jonas, Omar
- **WP12/TCP UDP Comparison**: Compare the advantages and disadvantages of UDP and TCP and then decide which one to use. Alternatively using both, UDP and TCP, is an option.
  Workers: Yan, Jonas, Omar
- **WP13/Content of packages**: Decide and implement what information network packages that are sent to the server and user should contain.
  Workers: Yan, Jonas, Omar
- **WP14/Timing of packages**: Decide and implement how many client-updates are sent to the server per second. Decide in what order, if there is an order, the clients should be served.
  Workers: Yan, Jonas, Omar
- **WP15**: Main player starting the game and others see the events.
  Workers: Yan, Jonas, Omar
- **WP16/Main player broadcasting**: Implement the broadcasting of the running game to the clients.
  Workers: Yan, Jonas, Omar
- **WP17/Client interaction**: Implement and test the interaction of clients with the server.
  Workers: All
- **WP18/Error handling**: Implement error handling for common errors that can occur while playing the game.
  Workers: Yan, Jonas, Omar
- **WP19/Additional features**: Add any additional features that enhance the gameplay.
  Workers: All
- **WP20/Game art**: Include stuff and more.
  Workers: All
- **WP21/Play the final game**:
  Workers: TAs.

5. MILESTONES

- **17. November 2015** Start work.
- **24. November 2015** Basic interaction with game.
- **01. December 2015** Communication working.
- **08. December 2015** Game finished.
- **16. December 2015** Testing Complete.
- **17. December 2015** Send presentation slides.
- **18. December 2015** Send code.

6. REFERENCES

[1] Giantbomb - Tower Defense. http://www.giantbomb.com/tower-defense/3015-413/.

[2] Unity - Asset Store. https://www.assetstore.unity3d.com/en/.

[3] Wikipedia - Tower Defense. https://en.wikipedia.org/wiki/Tower_defense.

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Embedded and Ubiquitous Computing. https://books.google.ch/books?id=gOgFCAAAQBAJ&pg=PA1135&lpg=PA1135&dq=udp+Rtt+paper+lan&source=bl&ots=j-dYdyqNFe&sig=txdX6qyfFkdgfAsKxI_vToRB1q8\&hl=en\&sa=X\&ved=0CD4Q6AEwBWoVChMI1fOKh8mLyQIVhBcsChOmTACr\#v=onepage\&q\&f=false.