TheGame: Real-time network multiplayer game

Shitole Onkar Popat [2022MCS2060]
Divyanka Chaudhari [2019CS50429]
Rajan Singh [2022MCS2066]

Aim

To create a real-time network multiplayer game.

Goals

- Create a library that contains the code to be shared commonly by the client and server.
- Create clients and servers which can be played on different machines.
- Players must interact with each other in some form.
- The game must be able to resolve conflicts arising due to asynchrony.
- Allow rollback for high latency connections.
- Client-side option to simulate high latency connection.
  (delaying the sending of each message to the server and delaying acting on each message received from the server.)

Game Overview

- Limited Grid Based game.
- Two players positioned at opposite ends of the grid have to bring back a crystal from their opponent's end to their end.
- Game has 3 phases.
    - Character select/ placing
    - Weapon selection/placing
    - Cell Reveal
    - Run phase (Character can use special ability here)
    - Victory / Next / Death
- Each player has a set of
    - Weapons: which they can place over the grid to slow down/destroy/harm the enemy. Without any special power, weapons are in general invisible to the opponent.
    - Characters: with varying characteristics like lives/strength/magical abilities which they can use to navigate through the grid and back

(note that the player is different than a character, each player can have multiple characters).

Detailed Game Plan

Point System

- A player wins if they bring back the crystal first or if their opponent loses all characters before them.

Grid

- The size of the grid varies by level.
- The grid could have pre-placed blocking/any other characteristic objects like trees (symmetrical to the center so no player is at a disadvantage).
- One cell is 60 x 60.
- Games runs at 60 FPS

Weapons

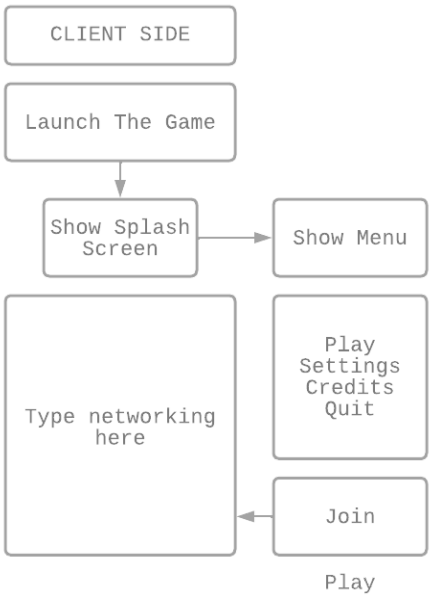
- Each player gets a set of weapons at each level which can be placed on their half of the grid which will be invisible to the opponent.
- Weapons:
    - [1 x 1] Wall: Blocks the character.
    - [1 x 3] Slippery floor: Slips the character 3 grid boxes forward.
    - [1 x 1] Mine: take one heart/life point of character.
    - [1 x n] Fireball Machine: Shoots fireballs in a column.
    - [1 x 1] Bomb : detonate after fixed time
    - [1 x 1] FeezeBlast: Freeze opponent for certain time

Characters

- Each player gets a set of characters at each level. They can use the characters in any order they want. When the character steps on any invisible weapon, it becomes visible.
- Each character can have the following characteristics:
    - Number of heart/lives
    - Speed
- Each character can take damage by weapons equal to the number of hearts after which it dies.
- Current Characters:
    - Spy:
        - Can reveal 1 extra square nearby.
    - Magician:
        - Has one extra heart/life.
    - BomberMan:
        - Detonates bomb on will.
    - Cloner:
        - Can create and place a fake clone.
    - Runner/Flash:
        - Fast but always moves 2 squares at a time.(2x speed)
    - Knight/Horse:
        - Runs like chess knight, can jump through obstacles.

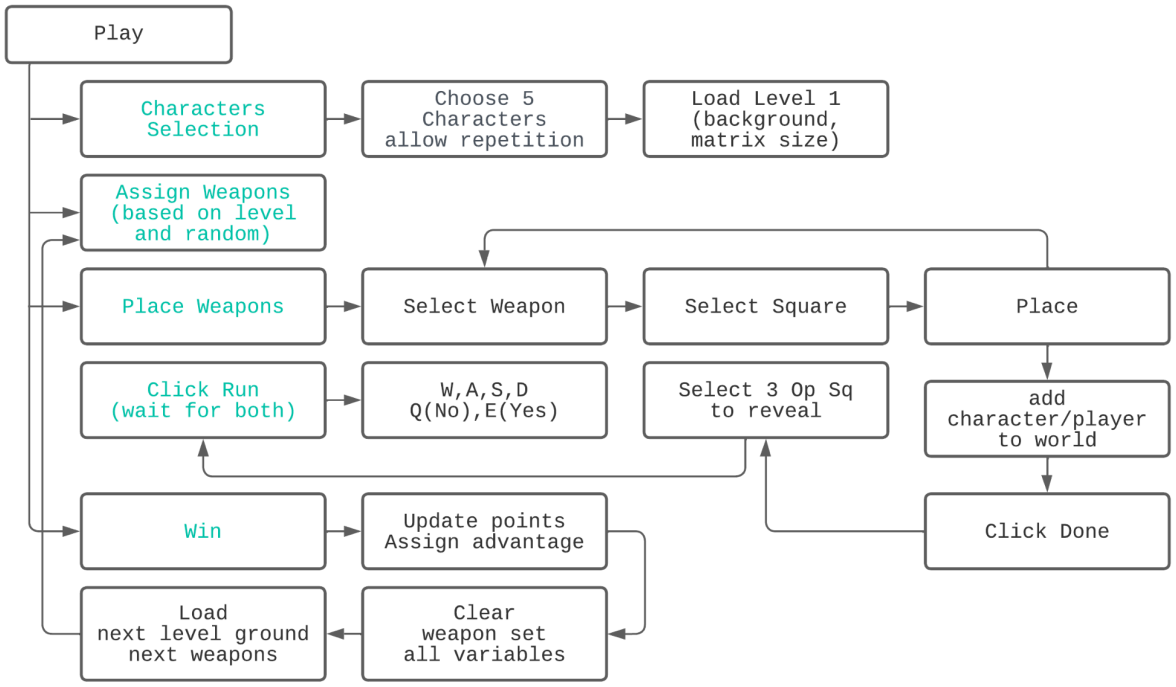
Game Flow

Status Flow



1. Menu: The menu screen is showing.
    a. Play (in gray so not accessible yet)
    b. Join (click to join to server and then the Play button will become active).
    c. Settings (Take game to settings screen and then back button to menu)
    d. Quit (Properly close all connections and close the game)
2. Join: Just enter IP and press enter/play.

Playing Flow



- On play (assume everything is initialized for level 1).
- Launch character select. Finish Character select.
- Loading:
    - Load level background
    - Load level matrix, inventory, put everything on screen.
- Weapon Place Phase:
    - On click check cell , then change state to something selected,
    - if character selected then launch character place.
    - if weapon selected, on cell select gray out weapon in inventory and add to cell.
    - Change the matrix accordingly. On run send matrix to server and wait for back input
- Run Phase:
    - [W A S D] to move.
    - [E Q] to interact yes/no.
    - Pick up crystal [E]
    - Run back to goal
    - Place the crystal [E].
    - Use ability [Q].

Additional Requirements Satisfied

1. Used Distributed Version Control: https://github.com/Silvmor/TheGame
2. Created automated documentation using Sphinx
3. Packaged into a redistributable: https://test.pypi.org/project/TheGame/
4. Implemented Rollback feature

Software Architecture

The application is based on two software architecture models.

A. Component-Based

The different system processes are placed into separate components so that all of the data and functions inside each component are semantically related. The calling component does not need to know about the inner workings of the component (implementation) in order to make use of it. The animations, characters, client, server, and game world are all separate components. This architecture is chosen to keep visually different sections logically separate too. To reduce inter dependability in order to reduce error-prone situations.

Benefits: Components are modular and cohesive. Increased code reusability. Better Information hiding.

B. Event-driven

The system changes its state on mouse and button events, and all modules and components are managed by these events. The game has a total of 6 states driven by events - Object placing, Sending Map over to Server, Receiving map from the server, Ready to play, Play/Run Phase, and Pause State. This architecture is chosen to keep all event handling and changes localized in a single place to ease navigation among themselves and build all-inclusive logic.

Benefits: Close coordination between vital components. Localized error handling.
