DESIGN SPECIFICATION
for
PokéSnowdown

Version 1.0

Produced by Matthew Zang, Nicholas Cage, Chiraq Obama, 4Evar Young

Contents

Update Log

This log will hold all the updates/git pushes to our game, showing our current progress/what needs to be completed, etc...

1 System Description

1.1 Core Vision

Our plan for A6 is to create a Pokémon Showdown spinoff using `OCaml` using key concepts presented in class (concurrency, mutability, functional programming) as well as integrating functions of `OCaml` not presented in class, including producing a suitable GUI and server to aid in development of the game. To set us apart from Pokémon Showdown, we are incorporating a single-player story mode in the form of a "Tournament mode", as well as allowing for full 1-Player, 2-Player, and No-Player functionality (by developing an intelligent bot that can fully play the game).

1.2 Key Features

We hope to have the following implemented in our completed project:

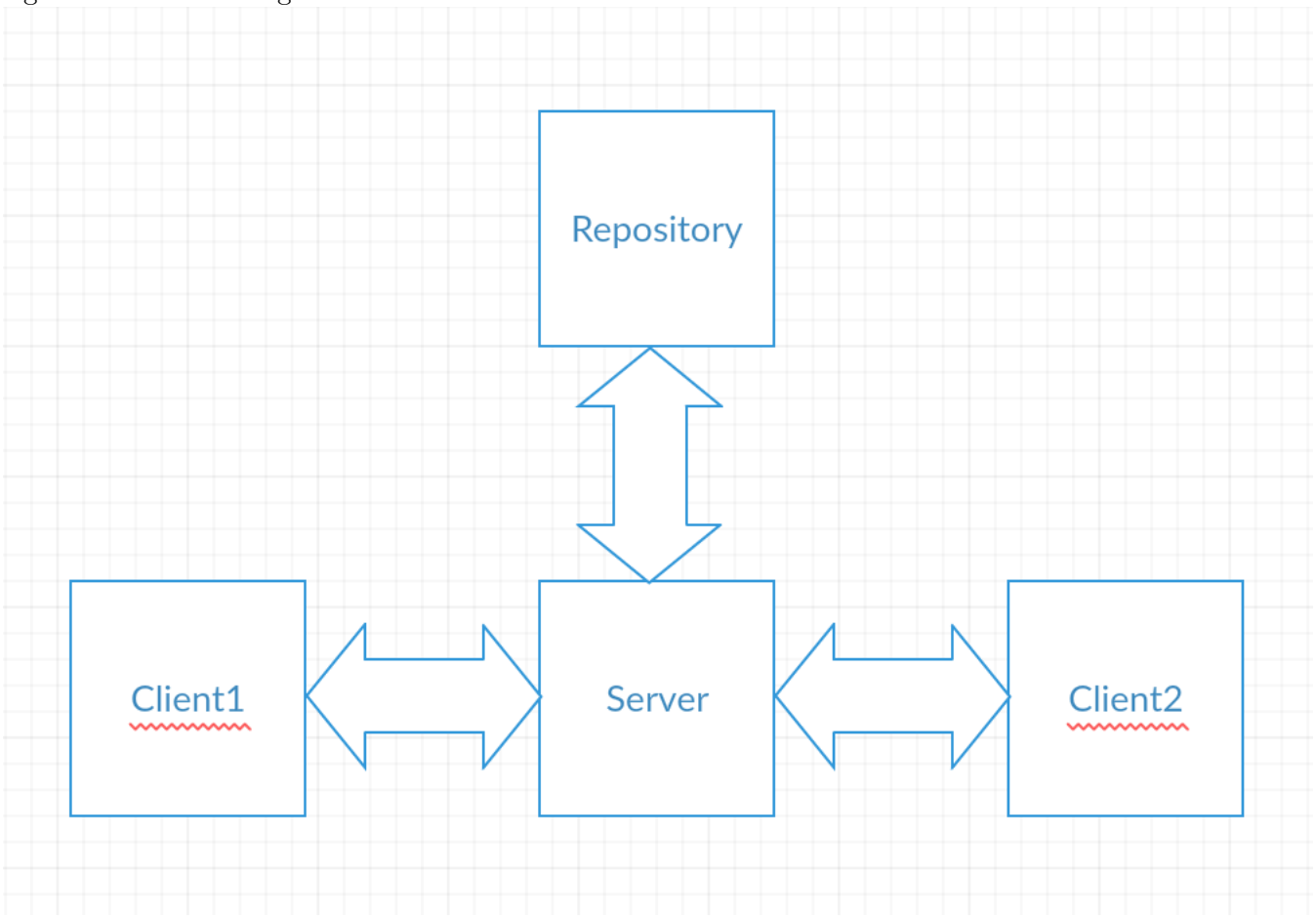
1. A fully functional GUI that allows the players to see the current state of the game. The GUI will most likely show the Pokémon but the attack animations might be too complex to complete.
2. All the current Pokémon will be included in the game, as well as items that are commonly used in competitive battling will also be included.
3. Multiple game options. We are hoping to have the following modes.
    - For 1-Player, we hope to incorporate three different modes.
        a) **Random Battles**: This is the same game mode as commonly found on Pokémon Showdown where teams are randomized. However, the player will be playing against the AI.
        b) **Tournament mode**: This is the single player story mode that we plan on incorporating.
        c) **Preset Battles**: The player gets to create his own Pokémon team and play against AI that use pre-made Smogon team compositions.
    - For 2-Player, we hope to have a single game mode.
        a) **Friend Battles**: Players have the options of choosing a Random team or a Preset team to head into battle with their friend.
    - Lastly, for No-Player, we hope to have a single game mode.
        a) **SkyNet Battles**: Player has option of choosing the Pokémon team for two bots, and watch them as they face off.
4. We wish to provide a Pokémon team editor that allows a player to choose Pokémon and movesets. This team is saved and can be loaded up for future use. These teams will be used in the game modes described above.
5. We wish to integrate mustic into our game based upon the game mode as well as the current battle status.
6. We want to include an interactive story mode/dialogue that will describe the underlying storyline in our Tournament mode.
7. We will have to include concurrency for two player battling, meaning that there will have to be a server to respond to player requests as well as the GUI.

1.3 Narrative Description

This game will be an upgraded version of Pokémon Showdown, suitable and to the scale for a small console video game. It will have the main aspects of Pokémon Showdown including random battles between two players, as well as premade team battles between two players and a fully functional Pokémon team editor. However, it will also include single player capabilities, such as random battling or premade battling against an AI. The AI must be able to battle effectively and use advanced Pokémon techniques in the battle. In addition to this, there will be a Tournament mode which is essentially a story mode. The player has to play against several bots before proceeding to the final boss. This means that bots should have different levels of intelligence in order for this game mode to be challenging.

2 Architecture

We are aiming to produce a client-server architecture. The game server will keep track of the current game state, make sure all the rules of Pokémon are followed, update the GUI, etc... while the client (the players) send messages to the server to retrieve information as well as request to perform certain game actions. There will be at most three clients (the GUI that displays the current state of the game, and the other two clients are the players). When the game is launched, it waits until it establishes a connection with the GUI through a different port allowing multiple people to watch instances of their own GUI. Then it spawns a thread while waiting for either two players (either real players or AI to connect). It initializes the players and then starts the game. The C&C diagram is shown below.



3 System Design

3.1 Important Modules

1. Game.ml

4 Module Design

5 Data

6 External Dependencies

7 Testing Plan