COSC2440 Software Architecture: Design and Implementations

Assignment

Multiplayer Turn-based Battle Arena Game

Design Proposal

**Semester:** 2013A

**Group Member:** Huy, Vuong Do Thanh (3342135)

Nghia, Nguyen Quoc Trong (3343711)

1. **Idea:**This is a game called Infinity Defense, which is similar to MOBA games (DOTA 2, LOL, HON) but it is a turn based game. It has some DOTA 2’s basic concepts which are 2 teams, creeps, fountain and towers. The game has two sides which are The Light and The Dark. Each team will try to destroy the enemy’s fountain. Furthermore, there will be creeps and defensive towers fighting along allies. When a hero kills an enemy hero, the game will drop supportive item(s) on the ground so that victorious hero can pick up and use as an advantage. After one fountain is destroyed, the game will end. The system will record each player score and make a record to player’s database.
2. **Logic/Rule:**The player can select each of his heroes (in a limited time) and choose an action (use an ability, pick up an item, use an item, etc.) to perform, and then select “End Turn”. The game engine will process the actions.
3. **Features:**
   1. *Maps:*

* The game will have one default map which is similar to DOTA 2 map.
* Map will be implemented using 2-d array of characters and tiled layer.
* The map can be customized.
* Map customization is done by editing the array.
* Import feature is implements to import a text file of correct format into the array to form the map.
* There are rules on editing/making new map: total obstacles on the map cover maximum of 45% of the total map area.
* Spring framework and LibGDX is used intensively to help code the sprite and tiled layer.
  1. *Heroes:*
* The game will have 10 basic heroes.
* Each hero will have 3 different abilities; one of them is the default attack with hero’s weapon.
* Each ability has cool down time of 3-5 seconds: the more power the ability is, the longer cool down time it has.
* Hero can move freely on moveable area on the map.  
  Hero can pick up item thrown out right after an attack.
* Heroes will be implemented using Sprite with transparent .gif or .png images.
* Spring framework and LibGDX is used intensively to help code the heroes and its relevant methods.
  1. *Instant Messaging (IM):*
* Players can chat with each other. (Connection via the server, not P2P)
* Group chat among nicks is available. Message broadcasting is available.
* Chat module on server will be multi-threaded. Each session is a thread.
* Login is required to use IM. Login is ID-Password combination.
* ID is sent in plain-text. Password will be sent in encrypted form. That encrypted form will then be compared on the server to do the login process.  
  1. *Client - Server*:
* Socket will be used to implement the client-server architecture: compared to RMI, socket is more generic and can be later ported into other programming languages. It also allows us to fully customize the protocol instructing the communication between client and server.
* Socket connection between server and client will be on any available IP at port 7777. Ports under 1024 are reserved, so we choose alternative port 7777.
* Data transfer will happen in plain-text format. Those data include, but not limit to:
  + chat (credential information, messages, online/offline status, etc.)
  + game (login credential information, game commands to attack heroes or collect items, game instructions to initialize map).
* Server will handle client connections in multiple threads. Each client connecting to server and requesting game is a thread.  
  1. *Login:*
* Before starting the game client, user must login to the server using ID and correspondent password.
* This will be implemented using HyperSQL and JDBC on the server module.
* Password saved in database’s table is encrypted using BCrypt package in Java.
* Login table should be included following information: id, passwd, fullname, availability.
  1. *Pick Hero:*
* At the start of the game, each player chooses his heroes.
* Decision should be made within the time limited: 30 seconds. If no choice is made, then the computer will choose randomly among available heroes.  
  1. *Music & Sound Effect:*
* During the time when the game is running, there will be background music playing.
* Each time user press a button, or a key, a beep sound will be triggered.
* Each time an action is performed by the hero, a sound is triggered.
  1. *Animation for attacks:*
* When an attack is performed, an animation is display from the source hero to the target.
* Animation is done using Sprite with the support of Spring framework and LibGDX.
  1. *Multiple players:*
* Multiple players can play the game at the same time.
* They can play individually or in group.
* Whichever form must follow the turn-based mechanism.
  + In group mode: one group can have 2-5 players. 2 groups maximum, corresponding to 2 sides of the game.
  + In single mode: maximum 2 players corresponding to 2 sides of the game.
  1. *“Fog of war” concept:*
* The game follows the “Fog of war” concept.
* The concept is that an enemy hero can hide behind the tree without being seen or cannot be seen if that hero is too far from hero sight.  
  1. *Action points:*
* User can use mouse or keyboard to give command for a hero to perform.

1. **Library and reuse Code/Materials:**

Spring Framework.

LibGDX.

BCrypt.

HyperSQL.

1. **Mockup UI:**
2. **Game’s Communication Diagram:**

|  |  |
| --- | --- |
| **CLIENT** | **SERVER** |
|  | Server starts |
| Client runs the program |  |
| Client sends login request with his/her ID and password | Server receives id and password, checking whether the ID and password are matched. |
| Client joins the lobby with other players | If matched, redirecting client to lobby |
| Client hits the “Send Message” button |  |
| System sends the request to append text to lobby | Checking request |
| Clients can read their text and other players’ text | If accepted, appending player’s name and text on the lobby screen. |
| Client hits the “Create game” button |  |
| System sends request to create a game with a default map or custom map information | Checking request |
|  | If yes, create map with provided information. |
| Clients are able to choose a hero | Redirect player to the choosing hero section |
| Client hits “Play hero” button |  |
| System sends client id and his/her selected hero | Server checks whether if the hero is available or not. |
| Repaint the selected heroes with gray color to notice other players choose available heroes. | If that hero is available, lock that hero so that other players cannot choose that hero |
| Done choosing, System send request to start game | Checking request. |
| Game starts | If yes, start the game and start to count the clock and record player score. |
| If fountain is destroyed, sending request to end game | Checking request |
| Appending recorded information | If yes, send recorded information |
| Client clicks “Close button” |  |
| System sends closing game information | Checking request |
| Remove closed game and display opened game on the screen | If yes, closing the game. Delete finished game |