COMP 504 Final Project Milestone 1 Documentation

Yue Jiang (yj25) and Yuchang Shen (ys54)

Game Related Thoughts

Prose description of how your game will work, including typical game play with winning and losing.

Game Design Idea

Our idea of the game design comes from a real-world game called the Scavenger Hunt. In the real world, the scavenger hunt is played by several groups with usually 1 to 3 people per group. One popular variation of the game is that, each group will be given some clues at the beginning of the game, and the clues will lead the group to the first checkpoint of the game. When the group reach the first checkpoint, another piece of clue will be available there and will lead the group to the the second checkpoint. And same for the rest of the checkpoints. The clues given at the checkpoints are usually not straightforward information that directly indicate the location of the next checkpoint. Instead, it is usually a quiz or some description about the location of the next checkpoint. The players need to decode the information and get to the next location.

Game Description

Since we are playing the scavenger hunt in the digital world, things will be changed a little bit. Since the game will be implemented based on the GIS package which provide an interactive view with the geographic information, our digital scavenger hunt will be played as the following:

> Each player will be given a location information (geographic coordinate for example) indicating where the next checkpoint is when they click the button at the current checkpoint and correctly answer the little quiz question that pops up after you click the button. The player or the team that first hit the final checkpoint wins the game, and other teams will be considered as losing the game.

Team Play

Our game design allows players to collaborate in the scavenger hunt. Up to **three players** allowed in each team, and up to **five teams** allowed in the game. Any member of the team can answer the quiz question and as long as one of the team members gives the correct answer, the checkpoint is checked and the information for the next location will be revealed.

API Implementations

A prose description of how your design will implement the required common API as well as any features specific to your implementation.

Same Top Level Interface for the Game Player and the Game Server

In the peer-to-server structure of the game design, the “peer”, which is the player in the game, and the “server”, which is the game server dealing with game related instruction in the game, are doing the same thing in an abstract level: they are all sending and receiving messages and they are all executing commands according to the messages. In our API, the *IUser* interface is designed to be implemented as the functioning “player” and “server”. However, the player and the server are playing different roles in the game application since the server knows everything about the game but the player knows nothing detail about the game. This is the reason why the implementations of these two roles in the game should have the ability to distinguish themselves from each other when receiving messages and executing commands.

Different Receivers for Different Roles

We are using a special receiver to connect the *IUser* that serve as the game server and the game room, and we call this special receiver the *host receiver*. The commands held by this host receiver is different from other receivers. These commands are pre-installed in the *IReceiver* object held by the server *IUser* following the instructions from the game. By doing so the server *IUser* will have all the information about the game. Since the receivers are different, the server *IUser* will respond differently from the player *IUsers*. This is how we distinguish the server and the player under the same top lever interface and let them do their own jobs.

Team Management

Team management are achieved by message passing between the player and the server.

Join A Team

We are allowing the *IUser* interface to receive messages in our design. The join a team process is achieved by a player directly connecting to the game holder. The game holder needs to start the server function (i.e. initiate the host receiver and teams etc.) before any player connecting to the server. Once a player connect to the server, the server will provide a team list to the player’s GUI and the player can choose which team to join.

To invite a player to a team, the server need to get the invited player’s stub, send a message to that player via the receive method and invoke the select team process for the player.

When a new player joins a team, all the other team members will be notified and get a copy of the receiver corresponds to the new player.

Leave A Team

The leave a team process will be the same as the leave a chat room process in HW08.

Change Team

The player send a requesting message to the server, the server will remove the player’s receiver from the current team, add a copy of the player’s receiver to the new team, and at the same time notifies all the players in both teams.

Inter-Team Message Passing

Since the server is the only entity in the system that has the knowledge of all teams, this process will be achieved by the server routing messages from team to team.

Intra-Team Message Passing

The intra-team message passing can be achieved the same way as we did in the chat room in HW08.

Game Installation

Game installation can be done via sending a game factory from the server to the players.

UML Diagram