Nicolas C. Broeking & Josh Rahm
CSCI 4448
March 17, 2014

Risk: Project Part 2

I.) Project Summery

This is a PC/ Mac game that allows users to play risk against each other. We will allow users to log into a server and connect them to others trying to play. The app will connect computers and allow them play a game of risk. A stretch goal is to allow the user to play against a machine and to make it 3d.

II.) Project Requirements

1.) Must be supported on at least Mac and Linux.
2.) Must allow for up to 2 consecutive players at a time. (More players is a stretch goal)
3.) Must allow users to start a game.
4.) Must allow users to join a game.
5.) Must allow users to interact with game state.
6.) Must allow users to place troops.
7.) Must allow users to attack countries.
8.) Must allow users to end attack phase.
9.) Must allow users to move troops.
10.) Must allow users to end move phase.
11.) Must have the system manage the game state.
12.) Must have the system organize turns.

III.) Users and Tasks

Our system will have two kinds of users. All players will log in, as a user but a user that creates a game will have more privileges than a user that is joining the game. Each user will take turns going through the phases of risk. The system is in charge of organizing the game and managing the game state.

Common Uses
1. Plays a game.
2. Host disconnects.
3. Player disconnects.
4. Initialization phase
5. Reinforcement phase.
6. Attack Phase.
7. Move phase.
8.

Problems:
1. If the connection is dropped or one player loses a connection.
2. If a host player gets disconnected.

3. If the game state that the server has is different than what the graphical state displays.
4. A user may try and do an illegal move.
5. A user may cheat by creating his own client
6. User could not have the correct libraries.
7. Improper hardware.

The official Use Case Documents are in the Use Case Folder.

IV. Activity Diagram

The activity Diagram is included in the Diagrams Folder.

V. Architecture Diagram

The architecture Diagram is included in the Diagrams Folder.

We will have two kinds of systems, Server systems and client systems. The servers and clients will communicate by sending XML through sockets to each other. The client side will use SDL and openGL to display the graphics and run the event loop.

VI. Data Storage

We do not need to save data in any way but we do need to send information between the server and the client. We are going to do this using XML.

VII. UI Mockups

There are two kinds of views for risk. You can either play risk using a gui that opens up to the game with a hud located at the bottom or you can play using the command line.

The UI Mockup is in the UI Mockups Folder.

VIII. UI Interactions

The user interacts with the system in one way. They must click on a region to attack. When clicked the event loop sends a signal to the mouse handler object. This is shown in the User Interaction Sequence Diagram found in the Diagrams folder. Then the system must figure out what the user is trying to do and update the game state. This is shown in the update server sequence diagram. Then the server must validate the move and then update the game state. This is shown in the server validation sequence diagram.

VIX. Class Diagram

The class diagram is in the Class Diagrams Folder.