***Magic Realm***

**Team 6**

**Couturier, Stefan**

**Lam, Ryan**

**Sayyeau, Christopher**

**COMP 3004**

**Date Due: April 8th, 2015**

**Table of Contents**

1. **Introduction**
   1. **Motivation**
   2. **Terminology**
2. **Game Rules**
3. **Requirements**
   1. **Functional Requirements**
   2. **Assumptions**
4. **Use Cases**
   1. **Use Case Diagram**
   2. **Use Cases**
   3. **Responsibilities**
   4. **Bound and Unbounded Use Case Maps**
5. **Design Decisions**
   1. **Decisions**
   2. **Structural Model**
   3. **Client/Server Functionality**
6. **Object Specifications**
7. **Interaction Diagrams**

Note: Some pictures and wording was taken from the example report for team 13-Antoine Bauza

**1 - Introduction**

Magic Realm was first released in 1979 by Avalon Hill. It was designed as a fantasy adventure board game by Richard Hamblen. It is a very complex role-playing war game that can be played from 1-16 players over the course of several hours. The rules were rereleased in a second edition in 1986 and a third edition was released by fans after the company went out of business in 1998.

* 1. **- Motivation**

This report will briefly cover our interpretation and implementation of this board game. It will outline the functional and non-functional requirements of the software, as well as its use cases and the assumptions made throughout its construction.

The goal of the project was to turn Magic Realm into a networked computer game. The official 3rd edition rules were followed as closely as possible to create the software.

* 1. **- Terminology**

A number of key terms are used throughout this report, which will be listed alphabetically in this section. Definitions are included, however if the term is covered in the Game Rules it will be covered in its own section (2).

|  |  |
| --- | --- |
| Term | Definition |
| Chit  Example 2 | This is a token with symbols on it representing some tangible object in the game universe.  Example 2 |

1. **- Game Rules**

The official third edition game rules for Magic Realm.

We will copy in all pages that we reference later

1. **– Requirements**

This Section contains what features must be implemented. Each requirement has its own identifier and a short description. The source of traceability may either be from the official game rules, assumptions, other requirements, or team decisions.

* 1. **– Functional Requirements**

Functional requirements define what behavior and functionality the software must have. They have been categorized below, based on different game states or areas of functionality. All of the functional requirements specified in the CorrectionGrid file provided are represented below.

|  |  |  |
| --- | --- | --- |
| **ID** | **Functional Requirements** | **Traceability** |
| FR-1 | Display correct titles of the board. | A-1 |
| FR-2 | Board is scrollable for each client. | A-1 |
| FR-3 | Networking capabilities. Capable of having multiple players. The actions of one player are represented in each window of his opponents. | A-3 |
| FR-4 | Initial character selection offers 2 or more distinct characters. |  |
| FR-5 | Cheat Mode allows user to assign 4 types of Map Chits( V, W, M, C) to specific tiles. |  |
| FR-6 | Can give the 5 Map Chits to Lost Castle. |  |
| FR-7 | Can give the 5 Map Chits to Lost City. |  |
| FR-8 | Can manually place Lost Castle. |  |
| FR-9 | Can manually place Lost City. |  |
| FR-10 | Sound and Warning chits are correctly displayed to clients. |  |
| FR-11 | Support choosing starting location for characters. | Location of game rule or assumption,section 2 |
| FR-12 | Support of moving within the tile. | GR- |
| FR-13 | Support of moving across tiles. |  |
| FR-14 | Support Captain’s extra phase when he is in a dwelling. |  |
| FR-15 | Support Amazon’s extra Move Phase. |  |
| FR-16 | Support the choice of dice roll results in cheat mode. | A-4 |
| FR-17 | Support hiding successfully (1-5). |  |
| FR-18 | Support hiding unsuccessfully (6). |  |
| FR-19 | Selection of fight and move counters for combat. Then resolve combat. |  |
| FR-20 | Can choose shield direction. |  |
| FR-21 | Support Multiple Rounds of combat between 2 characters. Resulting in a death or no death. |  |
| FR-22 | Support a 2 effort limit per round of combat. | A-5 |
| FR-23 | Support random order of player turns. |  |
| FR-24 | Support Dwarf only having 2 phases a day. |  |
| FR-25 | Support Black Knight |  |
| FR-26 | Support the recording of players phases and then implementing them after all players have finished recording. |  |
| FR-27 | Support the viewing of objects in clearing. | A-2 |
| FR-28 | Support the alerting of weapons. |  |
| FR-29 | Support using different stats of an alerted weapon. |  |
| FR-30 | Support combat with un-alerted weapons. |  |
| FR-31 | Support fatiguing counters. |  |
| FR-32 | Support wounding counters. |  |
| FR-33 | Support resting fatigued counters. |  |
| FR-34 | Support resting wounded counters. |  |
| FR-35 | Support locating of treasure. |  |
| FR-36 | Support looting of treasure. |  |
| FR-37 | Support Great treasures. |  |
| FR-38 | Support specification of treasure’s notoriety and fame. |  |
| FR-39 | Player’s current gold, fame, notoriety updated regularly. |  |
| FR-40 | Support cheat mode for searching tables. |  |
| FR-41 | Support locating of secret paths and passages. |  |
| FR-42 | Support of Cloak of Mist. |  |
| FR-43 | Support of Magic Spectacles. |  |
| FR-44 | Support Cave phase restriction. |  |
| FR-45 | Support of random dice rolls in normal mode. |  |
| FR-46 | Support monsters prowling to other clearings. |  |
| FR-47 | Support missile attacks. |  |
| FR-48 | Support viewing of clearing values. | A-1 |
| FR-49 | Support monsters. |  |
| FR-50 | Support character death. Creating pile of his belongings. |  |
| FR-51 | Support Looting of his belongings |  |
| FR-52 | Support victory point selection. |  |
| FR-53 | Support victory point calculation. |  |

* 1. **– Assumptions**

While building the software many assumptions were made about the rules so that the rules could be captured the software requirements. Many of the requirements above will trace to these items. The following table will identify them along with their justification.

|  |  |  |
| --- | --- | --- |
| **ID** | **Assumption** | **Justification** |
| A-1 | Each player has a labelled scrollable map. | Players will want to be able to remember quickly where everything is located. |
| A-2 | Players want to view the contents of a clearing. | Players will want to know what treasure, monster or other character is in the clearing so they can make informed choices. |
| A-3 | Game must have minimum of 1 player and a maximum of 6. All should be able of viewing the others actions. | Game becomes longer the more players are added. |
| A-4 | Player wants to be able to choose Dice result. | By fixing the result, it guarantees access to all portions of the game. |
| A-5 | Put a 2\* limit on combat rounds. | This keeps the game moving quickly. Also ensures new players don’t accidentally kill their characters in round 1. |
| A-6 | ADD MORE AS NEEDED |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. **– Use Cases**

A use case covers a scenario. It details the path of events needed for this scenario.

* 1. **– Use Case Diagram**

The following diagram corresponds to the use cases in 4.2. Actors are depicted as persons. Ellipses represent use cases, dashed arrows with an <<include>> stereotype represent an “includes” relationship, and regular lines represent association.

****

* 1. **– Use Cases**

Each use case is detailed in its own table row, which describes its sequence of events.

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Description** |
| UC-1 | Player Plays a Game | A player starts and plays a game of Magic Realm.  **Actors:** Player, System  **Triggering Event:** A new game of Magic Realm is started.  **Pre-Conditions(s):** The player is not in the game.  **Main Sequence:**  1. Player arranges a game with 1-6 players (execute UC-3 if hosting or UC-4 if player is a client).  2. Until everyone has played 28 Game Days  2.1. System displays the current game state.  2.2. A day will pass (Execute UC-2).  3. Calculate victory points based on players choices when creating the game.  4. Display the total score.  5. Player then leaves the game.  **Post-Condition(s):** Game has ended. Player is not in a game.  **Resulting Event:** The Player leaves the game.  **Alternative Scenario:**  Player leaves game Early – can happen at any time by closing application or quitting. Player is not replaced by another and is removed from game.  **Traceability:**  FR-53 |
| UC-2 | Game Day | During a game of Magic Realm all players progress through the day concurrently.  **Actors:** Players, System.  **Triggering Event:** The system enters a new day round.  **Pre-Condition(s):** There exists a player. The game has not ended yet.  **Main Sequence:**   1. Player records what he intends to do on his turn. 2. Monsters roam a little in their tiles. 3. Players are chosen to do their recorded turns in a random order. 4. After all players have done their turn. If there are possible player vs player combats execute UC-5. 5. If possible the player may trade. 6. All weapons become un-alerted.   **Post-Condition(s):** It is no longer the current day.  **Resulting Event:** The game state has been advanced.  **Traceability:**  FR-12 to FR-18, FR-23, FR-24,  FR-26 to FR-29, FR-31 to FR-46, FR-48 to FR-51 |
| UC-3 | Player Creates a Game | A player sets up a game of Magic Realm.  **Actors:** Player, System.  **Triggering Event:** The player chooses to host a new game of Magic Realm.  **Pre-Condition(s):** The player is not in a game.  **Main Sequence:**   1. The host player starts the server, specifies the number of players and then waits for them to connect to his game. 2. He will then specify whether the game is in cheat mode or not. 3. If in cheat mode he will then place all map chits. 4. All players will then choose their character, their starting location and their victory points. 5. Once all players have chosen the game will commence.   **Post-Condition(s):** The player is in a session.  **Resulting Event:** The players begin a session.  **Traceability:**  FR-1 to FR-11,  FR-25, FR-37, FR-38, FR-42, FR-43, FR-52 |
| UC-4 | Player Joins a Game | A player joins a game of Magic Realm as a client.  **Actor:** Player, System  **Triggering Event:** The player chooses to join a game of Magic Realm as a client.  **Pre-Condition(s):** The player is not in a game.  **Main Sequence:**   1. The player finds and specifies a game to join that is available. 2. Will then do steps 3-5 of UC-3.   **Post-Condition(s):** The player is in a session.  **Resulting Event:** The players begin the session.  **Traceability:**  FR-1 to FR-11,  FR-25, FR-37, FR-38, FR-42, FR-43, FR-52 |
| UC-5 | Player vs Player Combat | A player fights against another player.  **Actors:** Player, Player, System  **Triggering Event:** Two players end the day in the same clearing.  **Pre-Condition(s):** Both players are alive and in the same clearing. Neither player has already fought today.  **Main Sequence:**   1. Both players choose how they will attack their opponent, their shield defense and their defensive maneuver. 2. Player with the best attack attribute will attack first.    1. Determine if attack hit.    2. Determine attack damage.    3. Resolve damage. 3. If second player is still alive he will attack. Repeat step 2 using second player. 4. If both players are still alive execute UC-5.   **Alternative Scenario:**  Both players choose to do nothing- combat ends.  **Post-Condition(s):** One player is dead or both players are alive.  **Resulting Event:** Combat is finished, the day may resume.  **Traceability:**  FR-19 to FR-23, FR-26,  FR-28 to FR-34, FR-47, FR-50 |

* 1. **– Responsibilities**

These are created from the use cases above; they will be used in the use case maps in section 4.4. The reference for the respective use case is on the right.

|  |  |  |
| --- | --- | --- |
| **ID** | **Responsibility** | **Use Case(s)** |
| RESP-1 | The system displays the current game state. | UC-1 |
| RESP-2 | The system calculates victory points for the players. | UC-1 |
| RESP-3 | The system checks for any human players that have exited. | UC-1 |
| RESP-4 | The system removes any exiting human players. | UC-1 |
| RESP-5 | The system determines whether 28 game days have been played. | UC-1 |
| RESP-6 | The system notifies the player that the game has terminated unexpectedly. | UC-1 |
| RESP-7 | The system displays the scoreboard to each player. | UC-1 |
| RESP-8 | The player exits the game. | UC-1 |
| RESP-9 | The player records his actions for this day. | UC-2 |
| RESP-10 | The system moves the prowling monsters around their tiles. | UC-2 |
| RESP-11 | The system randomly selects a player to do their turn. | UC-2 |
| RESP-12 | The system executes the player’s turn. | UC-2 |
| RESP-13 | The system checks if there is possible combat then calls UC-5. | UC-2 |
| RESP-14 | The system checks if there is possible trading. | UC-2 |
| RESP-15 | The system un-alerts all weapons. | UC-2 |
| RESP-16 | The system displays the partially updated game state. | UC-2  UC-5 |
| RESP-17 | The system determines whether every player has completed their turn during this day. | UC-2  UC-5 |
| RESP-18 | The system advances the game state. | UC-2  UC-5 |
| RESP-19 | The player will choose where treasures, sounds, ghosts, dwellings, lost castle and the lost city are placed if this is cheat mode. | UC-3  UC-4 |
| RESP-20 | The player will choose their character, his starting location and the victory points needed. | UC-3  UC-4 |
| RESP-21 | The host player waits for a guest player to join. | UC-3 |
| RESP-22 | The host checks that there are enough players. | UC-3 |
| RESP-23 | The host begins the session. | UC-3 |
| RESP-24 | The host will choose if the game is in cheat mode or not. | UC-3 |
| RESP-25 | The system verifies whether or not the game to be joined is accessible. | UC-4 |
| RESP-26 | The system connects the player to the game as a client. | UC-4 |
| RESP-27 | The guest player waits for the host player to begin the session. | UC-4 |
| RESP-28 | The system initializes the session on behalf of the host player. | UC-4 |
| RESP-29 | The players selects attack direction and strength, shield direction, evasion direction and strength. | UC-5 |
| RESP-30 | The system verifies if the attack hits the other player. | UC-5 |
| RESP-31 | The system verifies if any damage is done to the other player. | UC-5 |
| RESP-32 | The system applies damage to the other player. | UC-5 |
| RESP-33 | The system will verify that the player is still considered alive. | UC-5 |
| RESP-34 | The players decide to leave the combat. | UC-5 |
| RESP-35 | The player is removed from play. | UC-5 |
|  |  |  |

**4.4 – Bound and Unbounded Use Case Maps**

The bound maps correspond to the unbounded ones directly below them; while the unbounded maps correspond to section 4.2. The Triggering and resulting events are labeled in the tables following them.

SEE Example

1. **– Design Decisions**

This section will cover the design decisions that were taken with respect to classes and objects chosen for the system. The UML diagram will be in section 5.2.

* 1. **– Decisions**

|  |  |  |
| --- | --- | --- |
| **ID** | **Design Decision** | **Traceability** |
| DD-1 | **Strategy Pattern**  The Strategy Pattern is extremely useful for implementing the game chits of all kinds and their Behaviours. We wanted to be able to reuse generic classes, rather than implement the various classes separately. Using the Strategy pattern, we could do this and keep the code very clean and simple, organizing each individual classes actions into its own inherited class.  This was also used for MapTile types, charater profile types and the different kinds of items a person could carry. | Group Decision |
| DD-2 | **Singleton**  ?Not sure if we have any of these? | Group Decision |
| DD-3 | **Relevant statistics for each player**  The game records all the resources, victory points, and other statistics with tallies of how much the player has accumulated for each. This decision reduces interface clutter, and lets the player or system instantly know how many of a type of resource they have instead of trying to count all the symbols they have. | Group Decision |
| DD-4 | **Testing of behaviors and strategies**  We were able to test how our functions were behaving and what strategies they were using. To do this, we added instances of them into a running game and looked at what they were outputting. As they were active debugging information was printed through their consoles, aiding us greatly in fixing their implementations. | Group Decision |
|  | Strategy Pattern, Mediator Pattern, Observer Pattern,Adapter pattern, template method pattern, singleton pattern, state pattern, composite pattern, decorator pattern, singleton pattern, etc |  |
|  | models we used  and why we used them |  |

* 1. **– Structural Model**

UML diagram of project, already created, just needs unscrambling and posting here, if the server side and client side classes were clearly evident we can erase 5.3 below

* 1. **- Client/Server Functionality**

This section will detail which functionality resides on the server side of the code and what code resides on the client side.

|  |  |
| --- | --- |
| **Client** | **Server** |
| On the client side their exists these classes, this section wouldn’t be needed if the UML gets cleaned up properly. |  |

1. **– Object Specifications**

SEE TABLES, IN EXAMPLE

1. **– Interaction Diagrams**

Below are the UML Unteraction Diagrams. Each corresponds to the previous bound use case maps in section 4.4.

SEE TABLES, IN EXAMPLE