**War of Robotcraft**

**Design Document**

Team: A3

Team Members:

|  |  |  |
| --- | --- | --- |
| **Name** | **NSID** | **Student ID** |
| Fu, Chen | chf354 | 11183491 |
| He, Jiahuan | jih889 | 11183346 |
| Wang, Shisong | shw940 | 11157916 |
| Xie, Ruida | rux793 | 11194258 |
| Yang, Chen | chy202 | 11183550 |

Date: Oct 21, 2016

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Number** | **Change Request Number** | **Description of Changes** | **Approved Date** |
| 0.1 | 0 | First Draft | 2016-10-21 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Document History Log:

# Introduction

# Architecture

# Detailed Design

## 3.1 Packages and Classes

### 3.1.1 Model Package

**3.1.1.1 Coordinate Class**

This class is used to create a coordinate for an absolute position in the map. Since this game map is hexagonal, each coordinate consists of three dimensions, which are X, Y and Z. Every object in the map such as the hexagon inside the map and the robot on the map has an absolute coordinate.

**Fields:**

X: int

Y: int

Z: int

**moveXPositive(): void**

**Summary:** this method is to update the coordinate when move forward x-positive direction by 1 step.

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:** x is increased by 1; y is decreased by 1; z is the same.

**moveXNegative(): void**

**Summary:** this method is to update the coordinate when move forward x-negative direction by 1 step.

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:** x is decreased by 1; y is increased by 1; z is the same.

**moveYPositive(): void**

**Summary:** this method is to update the coordinate when move forward y-positive direction by 1 step.

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:** x is the same; y is increased by 1; z is decreased by 1.

**moveYNegative(): void**

**Summary:** this method is to update the coordinate when move forward y-negative direction by 1 step.

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:** x is the same; y is decreased by 1; z is increased by 1.

**moveZPositive(): void**

**Summary:** this method is to update the coordinate when move forward z-positive direction by 1 step.

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:** x is decreased by 1; y is the same; z is increased by 1.

**moveZNegative(): void**

**Summary:** this method is to update the coordinate when move forward z-negative direction by 1 step.

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:** x is increased by 1; y is the same; z is decreased by 1.

**3.1.1.2 Game Class**

This class is used to create a new game including a list of players, the number of all players, and the current player who takes control of a play. Whenever a player executes an action that is moving a robot or shooting at a specific direction, the game will be updated the status. Then controller will send these updates to corresponding view objects, and the view object displays them.

**Fields:**

playerList: LinkedList<Pair<int, Player>>

playerNum: int

currentPlayer: Player

**setPlayerPosition(): void**

**Summary:** this method is to set all player’s position to corresponding side of the game board.

**Precondition:** the game is created.

**Input:** nothing

**Return:** void

**Postcondition:**

Set players to position Red and Green if the number of players is 2;

Set players to position Red, Yellow and Blue if the number of players is 3;

Set players to all positions if the number of players is 6.

**goNextPlayer(): void**

**Summary:** this method is to let the next player take control of the game

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:** the current player is set to the next player

**runPlay(): void**

**Summary:** this method is to run a play

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:** a new play starts

**3.1.1.3 Map Class**

This class is used to create a map model of the game. The map model is consisted of a list of coordinates that determine an abstract position in the map. Any robot shows up on the map has a abstract coordinate. When a robot gets the permission to move or shoot, the relative position for the robot is present on the game board, and the relative position is determined by the abstract coordinate.

**Fields:**

coordinateList: LinedList<Coord>

mapSize: int

**updateMist (): void**

**Summary:** this method is to update the mist on the map according to the robots’ positions.

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:** the mist on the map is updated.

**3.1.1.4 Player Class**

The player class stores the data of player. The player’s name, score, the robots the player has, as well as the death status are stored as attribute in this class. Moreover, the player class is also responsible for determining the status of the player and switch turns between players and give feedback to the gameController class. The gameController class gets messages from view classes then manipulate player data. When the game is on progress, the player class is to figure out which robot should move in this play, and move to next play.

**Fields:**

name : string

score : int

scoutRobot : Robot

sniperRobot : Robot

tankRobot : Robot

**Methods:**

**isDead() :** bool

**Summary:** this method is to return if the robot is dead.

**Precondition:** the game is not ended.

**Input:** None

**Return:** bool

**Postcondition:** The value is returned.

**getCurrentRobot() : Robot**

**Summary:** this method is to return the robot on current play.

**Precondition:** the game is not ended, the robot and the player are not dead.

**Input:** None

**Return:** robot object

**Postcondition:** The current robot is returned.

**goNextRobot() : void**

**Summary:** this method is to go to the next robot. (Run into next play)

**Precondition:** the game is not ended, the robot and the player are not dead.

**Input:** None

**Return:** None

**Postcondition:** The game runs into next play(It’s next robot’s turn to play the game).

HumanPlayer extends Player

* + move() : void
  + turn() : void
  + shoot() : 3DCoordinate

#### 3.1.1.5 AIPlayer extends Player

#### 3.1.1.6 Robot

* + turn(int direction) : void
  + move() : void
  + shoot(int distance) : 3DCoordinate
  + damaged(int attackPoint) : void
  + isDestroyed() : bool

### 3.1.2 View package

#### 3.1.2.1 GameStartView Class

The game start view welcome players and give the choice that start new game, manage the robots and exit the game. This class is used to create the game start frame including game name label, start game button, garage button, and exit button. When players click or press these button, the controller will respond these events, and send the result to set game mode view or garage view.

**gameStartView():void**

**Summary:** this method is to the game start frame, with game name, a button to start game, a button to go to garage and a button to quit the game.

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:**

The game start view constructed.

#### 3.1.2.2 SetGameModeView Class

The set game mode view displays the various mode, give player choices to set game mode. This class is used to create the view frame including a list of radio button of game mode, a list of radio button of player’s number, a list of combo box of player type, and a start button. After a player chooses the start game button, the controller will create the set game mode view. And any choice in this view will send to the controller.

**setGameModeView(): void**

**Summary:** this method is to construct the set game mode frame, a list with three radio button to set game mode, a list of three radio button to choose player’s number, a list of combo box to set player type, and a button to start game.

**Precondition:** nothing

**Input:** nothing

**Return:** void

**Postcondition:**

The set game mode view constructed.

#### 3.1.2.3 GameBoardView Class

#### 3.1.2.4 PlayerStatusView Class

#### 3.1.2.5 GarageView Class

### 3.1.3 Controller

#### 3.1.3.1 WarOfRobotcraft

**Fields:**

game : Game

**ActionPerformed(ActionEvent e) : void**

**Summary:** handle all the button clicked event

**Precondition:** button is clicked in views

**Input:** e is an ActionEvent from clicked button

**Return:** none

**Postcondition:** a new SetGameModeView instance created if startButton clicked; a new GarageView instance created if garageButton clicked; game quit if exitButton clicked; a new GameBoardView instance created if start game button clicked; switch to next play if endPlayButton clicked; the specific player marked died if surrenderButton clicked; go back to GameStartView if homeButton clicked; a new robot registered if registerButton clicked; the specific robot revised if reviseButton clicked; the specific robot retired if retireButton clicked

**keyTyped(KeyEvent e) : void**

**Summary:** handle all the key typed event

**Precondition:** key is pressed in views

**Input:** e is a KeyEvent

**Return:** none

**Postcondition:** enter turning mode if pressed “T”; enter moving mode if pressed “M”; enter shooting mode if pressed “S”;the playing robot’s direction rotate n (n is the number of pressed number key) unit if pressed number key and in turning mode; the playing robot shoots n (n is the number of pressed number key) unit of range if pressed number key and in shooting mode

**startTimer() : void**

**Summary:** start a new timer to be displayed in GameBoardView

**Precondition:** none

**Input:** none

**Return:** none

**Postcondition:** a new timer is started and is showed in GameBoardView

### 3.1.4 WarOfRobotcraft

### Entry

## 3.2 UML

# Different