Specification and Design Document for the Risk

Author: Yichao Xu
Student ID: 201299092
Computer Department account: x7yx2
Date: 15 Nov. 2018

Specification Document

Yichao Xu 201299092

1. Project Description

"Risk" is a turn-based strategy board game for two to six players. Each player will have the fixed number of armies at the start of the game and will automatically receive additional army units in each turn [1]. Additionally, each player will be able to attack or ally with a territory controlled by opponent.

The project aim is to design, implement, test and evaluates a turn-based strategy game which is based on the rule of "Risk", with the both intelligent agent players and the human players. The game will be delivered on time and shall be reliable. This project is prepared for the final year project supervisor and the players.

The "Unity3D", the programming language "C++", "Selection and Mini-Max Tree Algorithm" will be used to solve the project aims. The "Unity3D Game Engine" and "C++" will be used to implement the basic game rule following then, the "Unity3D UI System" will be used to implement the two-dimension graphic user interface. Additionally, the "Selection Algorithm" and the "Mini-Max Tree Algorithm" will be used to implement different layer of AI player.

2. Statement of Deliverables

2.1 Anticipated Documentation

The anticipated document will be the "specification and design document", "interim report" and "user manual";

2.2 Anticipated Software **(Functional and Non-function Requirement)**

- There will be two to six computer or human players in each game and the game will be turn-based and turn rotates among players.
- There will be zero to six human players in each game and the human players will be able to decide the number of the computer players at the start of the game. Therefore, the game type will be the "Player versus Player", "Player versus Computer" and "Computer versus Computer".
- The human player will be able to select the type of computer player by methods of different Artificial Intelligence algorithms at the start of the game.
- The game map will be shown on the screen and it will depict a political map of the earth, divided into forty-two territories, which are grouped into six continents.
- The action of each player, such as dispatching armies, invading territories and attacking enemies, will be shown on game map.
- The position of the player in relation to other player, such as the number of the armies in each territory, the additional army units in each turn will be shown on the screen.
- Each player will have the fixed number of armies and the player will either have complete control of where their armies start by clicking the map or by an algorithm to randomly distribute on equal size territories at the start of the game [1].
- Each player will automatically receive additional army units based on the number of territories they control at the start of each turn.

- In each turn, a player will be able to attack a territory controlled by an opponent. The system will randomly generate a number based on the size of each player's army. The player with highest number wins and the losing player removes one army from the attacked territory.
- The player who loses all his territories will lose the game and the winner is the only one with territories left on the map. Additionally, the name of the winner will be shown on the screen.
- The user interface shall be user-friendly and shall make player easily comprehend game rule.
- The game map shall be well-design and will be shaping forward the different territories and the different continents. Additionally, it shall be visually well-designed too.
- The game-code shall be readable and self-documented. The programing language shall be Object-Oriented, and the software design pattern such as factory method, shall be used.

2.3 Anticipated Experimentation

The only anticipated experimentation should be the experiment in the evaluation phase. A series of "Computer versus Computer" game will be tested, and the results will be recorded following than, it will be used to evaluate the efficiency of the AI algorithm.

2.4 Methods for Evaluation

The game will have been evaluated by the programmer by means of identifying if the function and the non-function requirements are achieved or by comparing it with other online Risk game to estimate the game quality. The methodology in the software engineering, such as the "object point" and "function point" shall be similarly used to evaluate the game.

Furthermore, in order to analyze whether the game meets its aims, a group of volunteers will be used for game feedback. The project will strictly follow the requirements in the "Computer Science Student project 3rd party Evaluation Procedure" [2]

The game intelligent agent will be evaluated through analysis of the result of the "computer versus computer" game and also compared with the intelligent agent in other online Risk game. The "Evaluation Function", such as the heuristic evaluation function or the static evaluation function, shall be used to evaluate.

3. Conduct of the Project and Plan

3.1 Preparation

According to the introduction in the website of "Hasbro", Risk is a turn-based strategy board game which is designed in 1957 by Albert Lamorisse and the game emphasizes the diplomacy, conflict and conquest between players [1]. This project will transform the board game into a computer game.

The data, such as the game parameter from other online risk game and the player feedback, will be used in the project. However, the project will not require any "synthetic data", "real non-human data" or "real human data" and it will not require the human participants except for 3rd party evaluation of the project.

In order to implement the project, a user guide of the risk game, other online Risk game, the game engine and the game assets will be required. The user guide of Risk from "Hasbro", the publisher of the game, will be used in the project to elicit the function requirement. The "Warzone" and "Pogo Risk", online Risk game, will be used to compare and evaluate the game quality. The Unity3D personal, the free cross-platform game engine and IDEA, and the assets in the "Unity asset store" will be used in the project.

Additionally, the software engineering methodologies, such as “Object-Oriented Programming”, “Automatic Testing”, “Project Management” and “Software Design Principle”, will be used in the project.

3.2 Design stage

The design of the project will be object-oriented, and the design phase shall strictly follow the software design process, namely architecture design, abstract specification, interface design, components design, data structure design, algorithm design. In addition, it will also include evaluation design. The method, such as the UML diagram (including, but not limited to limit class diagram, interaction diagram, user cause diagram) and the Pseudo-code, will be used in this stage.

3.3 Implementation stage

In the implementation stage, the developing environment will be hardware requirement, graphics card with DX10 (shader model 4.0) capabilities and SSE2 instruction set support CPU, and the software requirement, Windows 7 SP1+, 8, 10, 64-bit versions only; macOS 10.11+ and the development environment of the Unity3D [3].

The project will use the object-oriented testing to test the program and it shall strictly follow a top-down integrate testing process, namely, operation test, class test, cluster test, system test; The test method, “White Box Test”, “Equivalence Partitioning” and “Program Flow Graph”, and the test tools, “Unity Test Tools” and “Apache Ant”, will be used in the test phase;

3.4 Risk Assessment

The estimation risk, in which time required to develop the game is underestimated, will be caused by the overestimation of the programming efficiency and the underestimation of the project size. It will lead to the project delay or the low-quality documents and program. In this project, the estimation methods such as the “function point”, “EOF” and the Gantt Chart will be used to avoid this kind of risk.

The requirement risk, in which the project requirements are capricious, will be caused by the misunderstanding of the project. It will lead to the worthless function or nullification of the whole project. In this project, the software engineering methods such as, the “viewpoint”, “Ethnography and prototyping” will be used.

The people risk, in which the developer is unavailable at the critical time, due to the medical or family difficulty. It will lead to the project delay. In this project, the tasks shall be finished before at least one week to avoid the risk.

In the project, the artificial intelligent algorithm and the programming language C# are the required new skills. Therefore, the AI algorithm used in the project, “Selection” and “Max-Min Tree”, will be obtained from the “Artificial Intelligence - Search & Logic”, an online course. The programming language C# and the usage of the Unity Engineer will be obtained from the online course, “C# Programming for Unity Game Development Specialization”;

4. Reference

[1] A. Lamorisse, Risk, 6th ed. MA 01915. Printed in U.S.A: Hasbro, 1993, pp. 2,11-13.

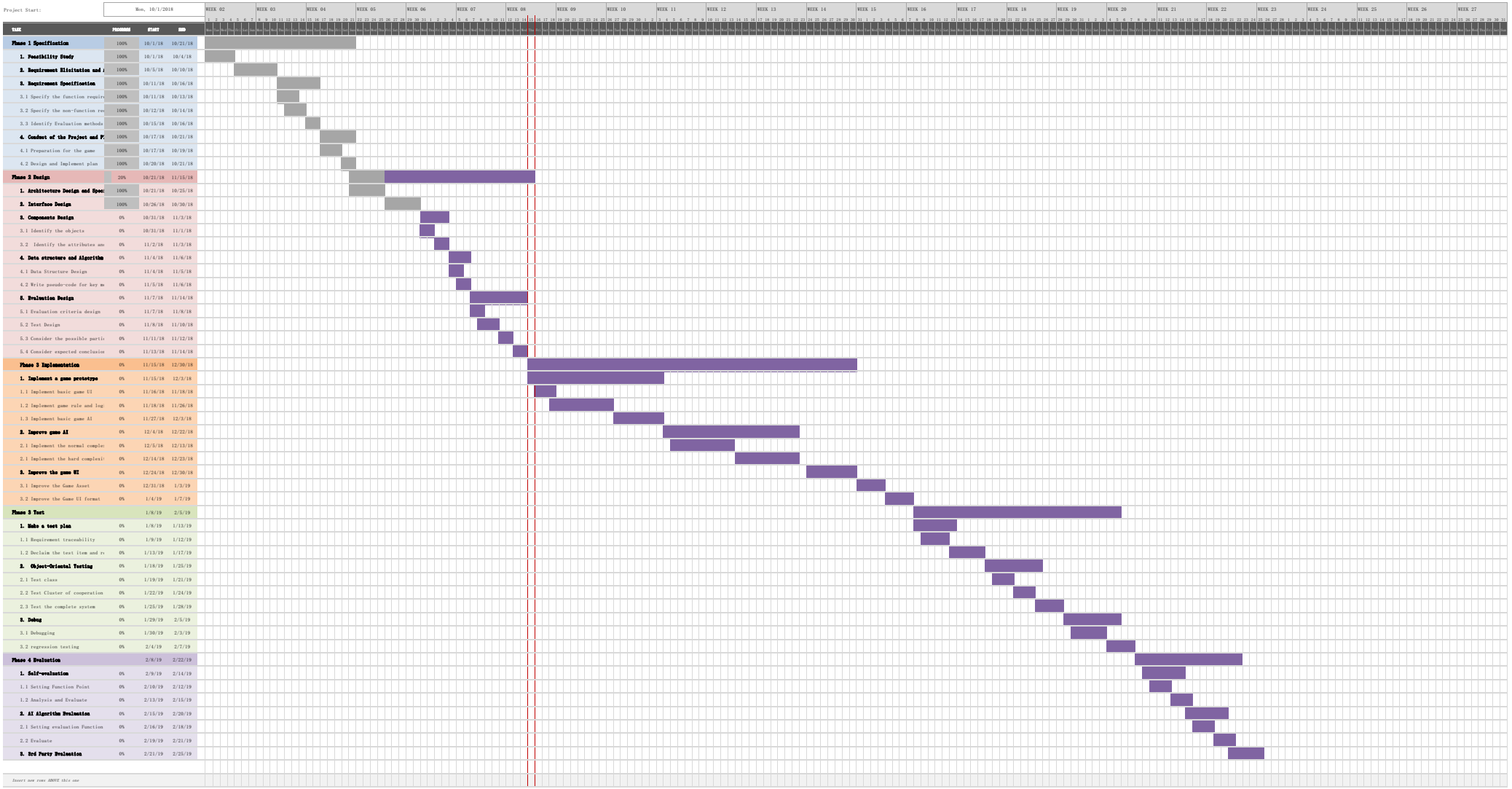
[2] Computer Science Student project 3rd party Evaluation Procedure, 1st ed. Liverpool: University of Liverpool, pp. 2-3.

[3] "Unity - System Requirements", Unity, 2018. [Online]. Available: https://unity3d.com/unity/system-requirements. [Accessed: 02- Nov- 2018].

RISK GAME

University of Liverpool
Yichao Xu

Project Start: Mon, 10/1/2018



TASK	PROGRESS	START	END
Phase 1 Specification	100%	10/1/18	10/21/18
1. Feasibility Study	100%	10/1/18	10/4/18
2. Requirment Elicitation and A	100%	10/5/18	10/10/18
3. Requirment Specification	100%	10/11/18	10/16/18
3.1 Specify the function requir	100%	10/11/18	10/13/18
3.2 Specify the non-function re	100%	10/12/18	10/14/18
3.3 Identify Evaluation methods	100%	10/15/18	10/16/18
4. Conduct of the Project and P	100%	10/17/18	10/21/18
4.1 Preparation for the game	100%	10/17/18	10/19/18
4.2 Design and Implement plan	100%	10/20/18	10/21/18
Phase 2 Design	20%	10/21/18	11/15/18
1. Architecture Design and Spec	100%	10/21/18	10/25/18
2. Interface Design	100%	10/26/18	10/30/18
3. Components Design	0%	10/31/18	11/3/18
3.1 Identify the objects	0%	10/31/18	11/1/18
3.2 Identify the attributes an	0%	11/2/18	11/3/18
4. Data structure and Algorithm	0%	11/4/18	11/6/18
4.1 Data Structure Design	0%	11/4/18	11/5/18
4.2 Write pseudo-code for key m	0%	11/5/18	11/6/18
5. Evaluation Design	0%	11/7/18	11/14/18
5.1 Evaluation criteria design	0%	11/7/18	11/8/18
5.2 Test Design	0%	11/8/18	11/10/18
5.3 Consider the possible partic	0%	11/11/18	11/12/18
5.4 Consider expected conclusion	0%	11/13/18	11/14/18
Phase 3 Implementation	0%	11/15/18	12/30/18
1. Implement a game prototype	0%	11/15/18	12/3/18
1.1 Implement basic game UI	0%	11/16/18	11/18/18
1.2 Implement game rule and log	0%	11/18/18	11/26/18
1.3 Implement basic game AI	0%	11/27/18	12/3/18
2. Improve game AI	0%	12/4/18	12/22/18
2.1 Implement the normal comple	0%	12/5/18	12/13/18
2.1 Implement the hard complexi	0%	12/14/18	12/23/18
3. Improve the game UI	0%	12/24/18	12/30/18
3.1 Improve the Game Asset	0%	12/31/18	1/3/19
3.2 Improve the Game UI format	0%	1/4/19	1/7/19
Phase 3 Test		1/8/19	2/5/19
1. Make a test plan	0%	1/8/19	1/13/19
1.1 Requirement traceability	0%	1/9/19	1/12/19
1.2 Declaim the test item and r	0%	1/13/19	1/17/19
2. Object-Oriental Testing	0%	1/18/19	1/25/19
2.1 Test class	0%	1/19/19	1/21/19
2.2 Test Cluster of cooperation	0%	1/22/19	1/24/19
2.3 Test the complete system	0%	1/25/19	1/28/19
3. Debug	0%	1/29/19	2/5/19
3.1 Debugging	0%	1/30/19	2/3/19
3.2 regression testing	0%	2/4/19	2/7/19
Phase 4 Evaluation		2/8/19	2/22/19
1. Self-evaluation	0%	2/9/19	2/14/19
1.1 Setting Function Point	0%	2/10/19	2/12/19
1.2 Analysis and Evaluate	0%	2/13/19	2/15/19
2. AI Algorithm Evaluation	0%	2/15/19	2/20/19
2.1 Setting evaluation Function	0%	2/16/19	2/18/19
2.2 Evaluate	0%	2/19/19	2/21/19
3. 3rd Party Evaluation	0%	2/21/19	2/25/19

Insert new rows ABOVE this one

Design Document

Yichao Xu 201299092

Index for the Design Document

1. Summary of Proposal

1.1 Brief Statement

As described in the specification document, “Risk” is a turn-based strategy board game for two to six players. Each player will have the fixed number of armies at the start of the game and will automatically receive additional army units in each turn [1]. Additionally, each player will be able to attack or ally with a territory controlled by opponent.

The project aim is to design, implement, test and evaluates a turn-based strategy game which is based on the rule of “Risk”, with the both intelligent agent players and the human players. This document of the project is prepared for the final year project supervisor and the players.

Additionally, the artificial intelligence algorithm is changed from the selection algorithm to a advanced searching algorithm

1.2 Summary of The Research and Analysis

The document designs the game program following the software engineering design process. At first, it describes the use cases of the game by a use case diagram with a dictionary to describe each use case.

Then, the document uses the context model to describes the architecture of all subsystems. The system will use the Model-View-Controller architecture pattern and the parallel controlling model.

In the abstract specification, the document discusses the details of these subsystem and the design patterns, such as the singleton, delegate and notification, will used in the system.

In the interface design, the document specifies a user interface of the game. the Unity asset “Earth-3D” will be used to generate the game map. In addition, the document provides the image of the game map which is automatically generated by the asset.

In the component design, the document uses the class diagram to describe the classes consisting of the system. Additionally, the document focuses on the design of the system model and system controller.

In the data structure design, the document describes the HashMap, LinkedList and an extension for the General Tree. And in the algorithm design, the document implements the evaluation algorithm and alpha-beta pruning algorithm.

In the test design, the document writes the possible test cases and in the evaluation design, it describes the details of the three evaluation methods in the specification document.

2. Project Design

This project will firstly describe the requirements in the project by the use case diagram and then it will follow the software design process, namely, architecture design, abstract specification, interface design, components design and algorithm design.

2.1 Use case diagram

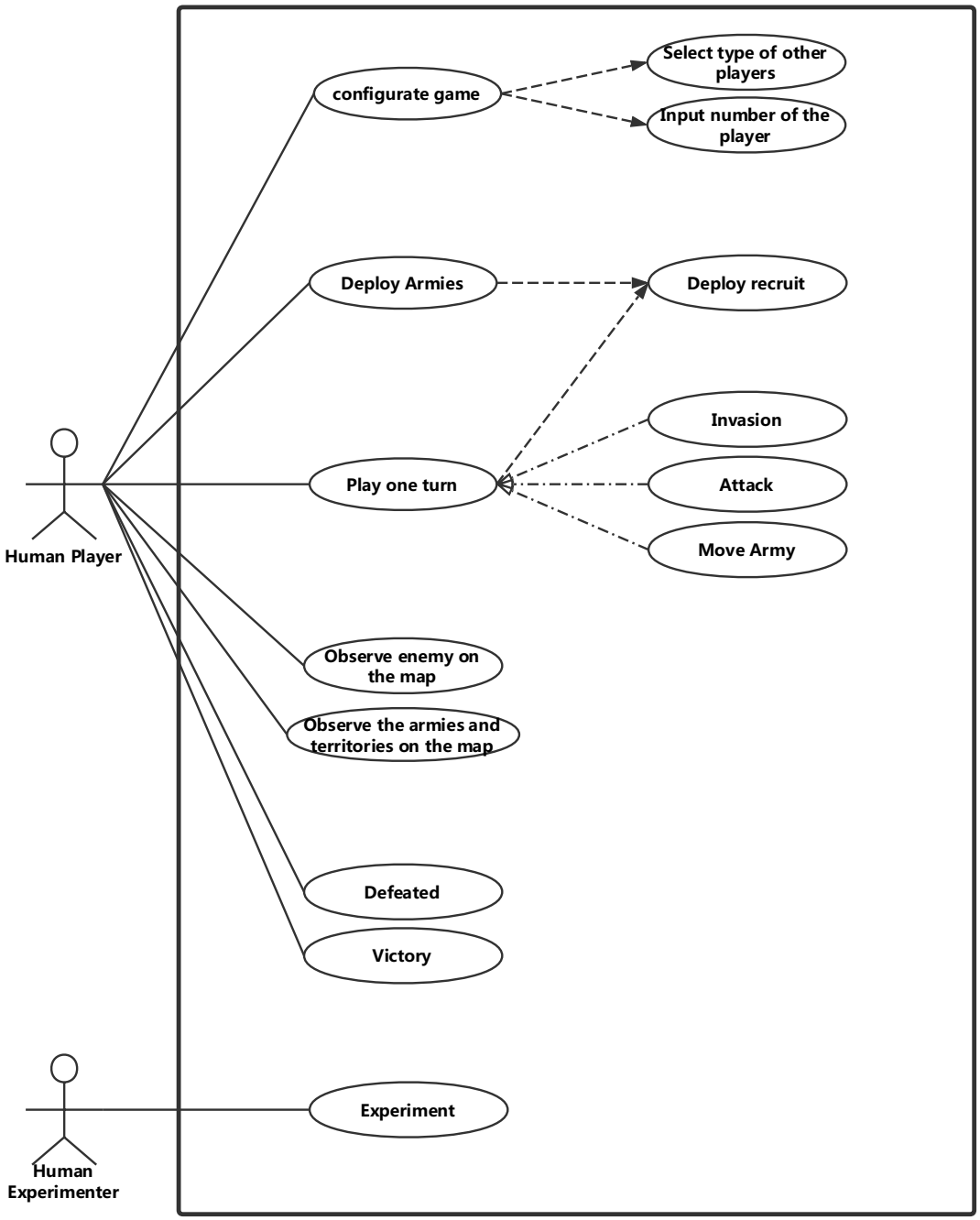


Figure 1 Use case Diagram

ID	UC1
Name	Configure game
Description	The user configures the game before starting a new game.
Pre-conditions	There is not any information about the number and the type of player.
Event flow	The game system shows the user a guide of configuration. include UC2() include UC3() Generate the game based on the data from UC1 and UC2
Post-condition	The game has been generated according to the collected data.
Includes	UC2, UC3
Extensions	None
Triggers	A game user click button “Start Game”

ID	UC2
Name	Input number of the player
Description	The user input the number of the game player.
Pre-conditions	The system doesn’t know the number of players.
Event flow	At first, the game system requires the number of players from the user. After the user types a number, the system will check the validation. if it is valid, the system will store the number of the player. else the system will notify the user, the number is invalid.
Post-condition	The number of the player has been stored in the system.
Includes	None
Extensions	None
Triggers	A game user click button “Start Game”

ID	UC3
Name	Select type of other players
Description	The user selects the type of another player in the game.
Pre-conditions	The system doesn’t store any information about other players The system has stored the number of players.
Event flow	for all other players, the user selects the player type from (human, simpleAI, normalAI, HardAI). and then, it will be recorded in the system
Post-condition	The system has recorded the type of all players.
Includes	None
Extensions	None
Triggers	UC2 has been finished.

ID	UC4
Name	Deploy Armies
Description	The human player deploys its armies at the start of the game.
Pre-conditions	A player doesn’t deploy the armies and the game is still in the configuration phase.
Event flow	for each recruit in the configuration phase include UC5()
Post-condition	The player doesn’t have undeployed armies.
Includes	UC4
Extensions	None
Triggers	The configuration turn of the player starts.

ID	UC5
Name	Deploy recruit
Description	A human player deploys its recruit on the map.
Pre-conditions	The player has an undeployed recruit and in the recruiting phase.
Event flow	select a province on the map if it has already been occupied by the player, the number of the armies on the province plus one. if it has already been occupied by another player, abort the deployment. if it is unoccupied, allocate the province to the player set the number of armies on the province to be one.
Post-condition	The size of the player's armies plus one.
Includes	None
Extensions	None
Triggers	The player clicks the allocate button.

ID	UC6
Name	Invasion
Description	A human player's armies invade the other player's territory.
Pre-conditions	One of the player's armies is on the boundary and in the attack phase.
Event flow	The user selects an army on the boundary. After that, move the army into an enemy's province which is ungarrisoned Then, the player occupies the province.
Post-condition	The number of the player's provinces plus one
Includes	None
Extensions	None
Triggers	The player drags an army on the belligerent's ungarrisoned province

ID	UC7
Name	Attack
Description	A human player's one of the armies attacks the enemy and in the attack phase.
Pre-conditions	One of the player's armies is nearby the enemy.
Event flow	The user selects an army on the boundary. After that, move the army into an enemy's province which is garrisoned Then, the system calculates the result of the battle The armies of the defeated party will be annihilated The province will be occupied by the winner.
Post-condition	One of the belligerents' armies on the province is annihilated
Includes	None
Extensions	None
Triggers	The player drags an army on the belligerent's garrisoned province

ID	UC8
Name	Move Army
Description	A human player moves the army between own provinces.
Pre-conditions	The number of the army of the player is not zero and the player is in the attack phase
Event flow	The user selects an army. After that, move the army into an own province Then, the number of the garrison plus one.
Post-condition	The allocation of the player's armies is changed
Includes	None
Extensions	None
Triggers	The player drags an army on the belligerent's garrisoned province

ID	UC9
Name	Play one turn
Description	A human player plays one turn of the game.
Pre-conditions	The player doesn't be defeated.

Event flow	The game turn starts.
	the player enters the Deploy Phase,
	for each recruit do
	include to UC5();
	for each army in the map do
	if the user decides to invade
	include to UC6();
	if the user decides to attack
	include to UC7();
	if the user decides to move
	include to UC8();
	end the game turn

Post-condition	The player ends the turn.
Includes	None
Extensions	UC5, UC6, UC7, UC8
Triggers	The player turn starts.

ID	UC10
Name	Observe enemy on the map
Description	A human player observes the location and the number of armies or provinces of other players.
Pre-conditions	The player currently doesn't be defeated.
Event flow	Observe provinces and armies on the map.
Post-condition	The player has observed the enemy.
Includes	None
Extensions	None
Triggers	The player is going to observe the enemy.

ID	UC11
Name	Observe own armies and provinces
Description	A human player observes the location and the number of own armies or provinces.
Pre-conditions	The player currently doesn't be defeated.
Event flow	Observe provinces and armies on the map.
Post-condition	The player has observed own armies and province.
Includes	None
Extensions	None
Triggers	The player is going to observe own armies and provinces.

ID	UC11
Name	Victory
Description	A human player wins the game.
Pre-conditions	The player is the last player of the game.
Event flow	The game system find that player is the last player of the game. After that, the system shows an information to notify the player.
Post-condition	The game is over, or the player is removed from the game.
Includes	None
Extensions	None
Triggers	UC2 require remotely authenticated service

ID	UC12
Name	Defeated
Description	A human player is defeated.
Pre-conditions	The number of the player's armies and provinces is zero.
Event flow	The player's last province has been occupied by an enemy. The system checks the number of human players. if the number is zero, Game over and conclude the result of the game; else game continues, but the player will be removed from the game;
Post-condition	The game is over, or the player is removed from the game.
Includes	None
Extensions	None
Triggers	UC2 require remotely authenticated service

ID	UC13
Name	Experiment
Description	A human experimenter to exam the AI system.
Pre-conditions	The game system is running.
Event flow	The experimenter clicks the "Simulating" button on the main menu. The experimenter decides types of two computer players. The system automatically generates and logs the result of the game.
Post-condition	The game is over, and the game log has been produced.
Includes	None
Extensions	None
Triggers	The experimenter clicks the "Simulating" button on the main menu.

2.2 Architecture Design

The design pattern, Model-View-Controller, will be used in the project as the architecture design pattern. The subsystem of the game will be the "User Interface System", "Game Control System", "Game Model", "Artificial Intelligence System" and "Notification Centre". Additionally, the controlling model should be the "Parallel Model".

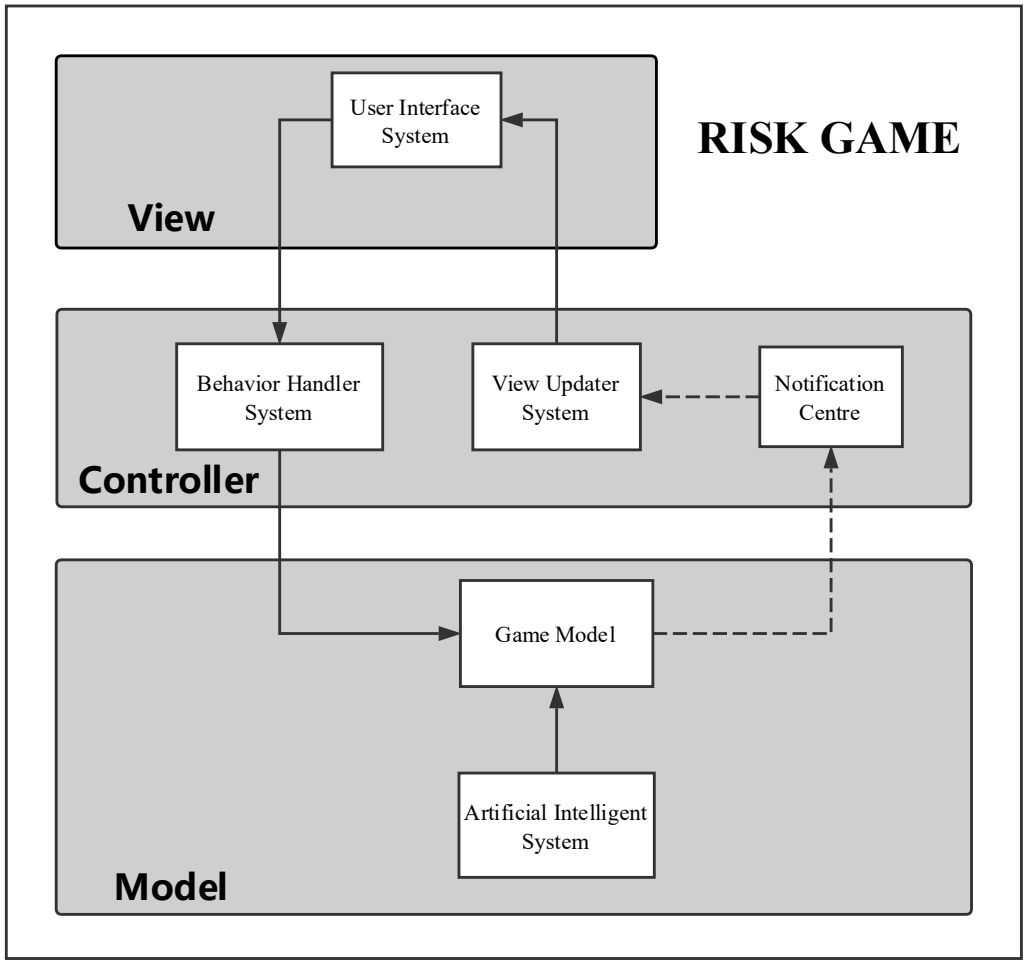


Figure 2 Context Model of Risk

2.3 Abstract Specification

The "User Interface System" is the game view. The subsystem contains all user interface components such as "button", "game map" and "army icon". When the user interacts with (clicks or drags) these components, the subsystem will send a message to the "Behavior Handler System".

The "Behavior Handler System" is a part of the game controller. They should consist of a set of the Unity scripts which bind with game components in the user interface system. When the subsystem receives a message from the user interface system, it will send a message to the "Risk Game Model".

The "Game Model" is the game model for handling the player's action (from the human or computer player). It can receive messages from either the behavior handler system or the artificial intelligence system. After that, the subsystem will update the game model and send

the changes to the “Notification Centre”. The “Artificial Intelligence System” is similarly a part of the game model. It will generate a series of actions by a minimax algorithm and deliver these actions to the game model.

The “Notification System” is a part of the controller. It will receive data from the game model and then, generate a series of notifications. After that, the subsystem will broadcast these notifications to the “View Updater System”

The “View Updater System” is a part of the game controller and it similarly consists of a set of the Unity scripts. The system will periodically check the broadcasted notification to update components in the game view.

2.4 Interface Design

In this part, the document will describe the design of the game user interface. The user interface of the game will consist of two parts namely, the game map and the game functional components.

The game map will show the player the provinces, continents, armies and enemies. In this project the “Earth-3D Asset” from the “Unity Assets Store” will be used. The asset will automatically generate a world map which is divided into series provinces as the figure3.

Figure 3 Game Map [2]

The game functional component is the component in the game which provides different functions to the player. As shown on the figure4, the game functional components of the game are three buttons and a text field on the top of the screen.

Figure 4 Functional Component

The text field shows the current turn of the game and the three buttons are the tools for making the game UI more user-friendly. The “NEXT” button will end the current player’s game turn and move to the next human player’s turn. The “ATTACK” button will highlight the boundary of the player and the “RECRUITS” button will highlight the player’s provinces.

Figure 5 Activate the button "RECRUITS"

2.5 Components Design

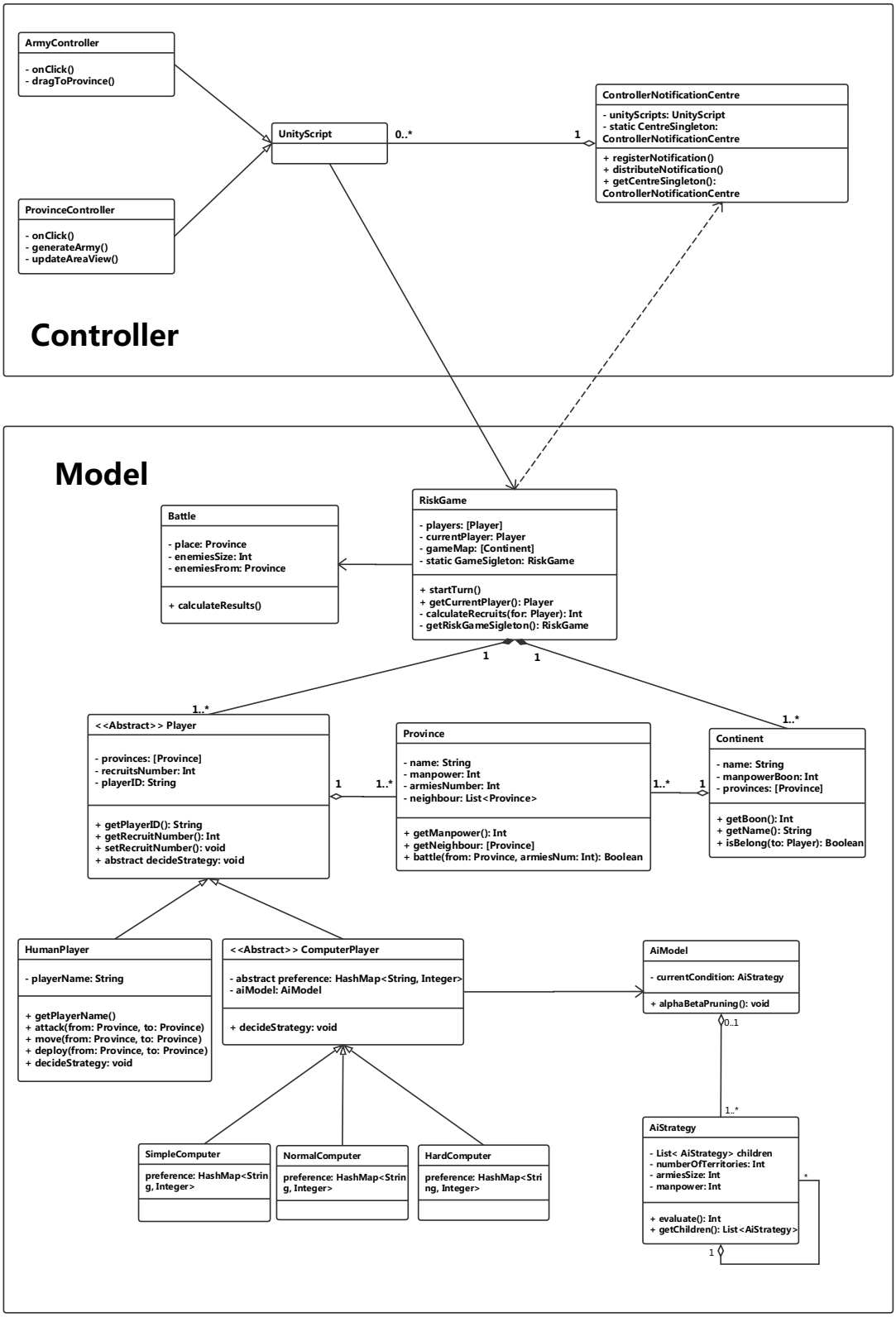


Figure 6 The class diagram

2.5.1 Controller

In the system, the controller consists of the Unity-script and notification centre's instances.

The basic Unity-script class provides two different functions: firstly, it will provide a group of interfaces for system components, such as the button, to handle the user behaviours; secondly, these Unity-scripts implements the delegate pattern, the Unity engine will periodically (per frame) invoke the method of the Unity-script. Therefore, in the project, both of the behaviour handler system and view updater system consist of the Unity-script's instances. They will invoke the game model's methods after receiving a message from the user interface system and update the game view after receiving a notification from the notification system. Additionally, both of the "Army Controller" and "Province Controller" inherit the Unity-script class. Two classes will handle the behaviour of the player for the armies and provinces on the game view respectively.

"Notification Centre" implements the observation and singleton pattern and it is the implementation of the notification system. It will receive data from the game model system and broadcast a notification to these Unity-scripts (All instances of Unity-script shall register itself into the system).

2.5.2 Model

The system model includes the instances of "Risk Game", "Battle", "HumanPlayer" and "ComputerPlayer", "Province" and "Continent".

The "Risk Game" class will be the interface of the game model. Its public methods will be invoked by the Unity scripts and it will automatically send the result to the notification centre. It contains all continents on the game map and all players of the game.

The battle class represent a battle of the game. It stores the location and size of the battle and provides an interface to calculate the result.

The "Player" class represents a game player. It is an abstract class which describe all possible behaviours of a player namely, "move armies", "deploy recruits" and "attack enemy". Additionally, the class stores some information of a player such as the player's id and the number of the recruits.

The "Continent" class represents a specified area on the map. The class includes the name of the continent, the manpower boon when a player has occupied all of these provinces and the pointers of all provinces in the continent.

The "Province" class represents a minimum geographic unit of the game. The class includes a series pointer for the nearby provinces, the number of manpower, the province will produce in each turn and number of the armies deployed on the province. In addition, it will provide a method to calculate the result of a battle.

The "HumanPlayer" and the "ComputerPlayer" extend the "Player" class respectively. The "ComputerPlayer" class is an abstract class which describe how the computer player makes a strategy and constraints that all subclasses should implement their preference function. It contains an instance of the "AiModel" which will implement an algorithm for the game AI. The "SimpleComputer", "NormalComputer" and "HardComputer" class implement the class.

The "AiModel" is the core of the AI system. The class implement the alpha-beta pruning algorithm. It contains an instance of the "AiStrategy" which implement the general tree node and the evaluation algorithm.

2.6 Data structure Design

HashMap, LinkedList and GeneralTree will be used to implement the game

2.6.1 HashMap and List

The HashMap will be used in the "NotificationCentre" to register the controller and similarly in the "ComputerPlayer" to store a preference function.

The List will be used in the class "RiskGame", "Continent" and "Player". In the "RiskGame", two different Lists will be used to respectively store the continents and players of the game. And in both of the "Continent" and "Player" class, a List will be used to store instances of "Province" class.

2.6.1 General Tree Node

The instances of the class consist of the tree structure. The tree nodes can have an arbitrary number of children and it provides a method to access all children of the node.

```
class GeneralTreeNode {
    private List< GeneralTreeNode> children;
}
```

Code 01: The design of the abstract class General Tree.

In the project, the class "Strategy" will extend the GeneralTreeNode. It contains a series of pointers for all possible strategies and provides a method to evaluate the result of these strategies. It also contains a set of attributes which will be used for evaluation.

```
class Strategy {
    private List< Strategy> children;
    private HashMap<String, Integer> attributes;
    public int possibleNextStrategies(){}
    public int evaluate(Strategy strategy){}

}
```

Code 02: The design of the Strategy class.

2.7 Algorithm Design

The project will use a kind of adversarial search algorithm to implement the intelligence agent. The system algorithm design focuses on two different parts namely, the alpha-beta pruning algorithm design and the evaluation algorithm design.

2.7.1 Evaluation Algorithm

The evaluation algorithm will value the player's strategy in a game turn. the project will use the following expression to generate the evaluation algorithm.

$$Eval = w_1 \times F_1 + w_2 \times F_2 + \cdots + w_n \times F_n$$

Function01: the expression for the evaluation algorithm [3]

The expression contains two kinds of different variables: the variable F_n represents a function to transfer a parameter in the game into a value and the variable w_n represents the weight of

the parameter. Therefore, the evaluation function will describe the value to a series of parameters and their weights.

As shown in Table01, the evaluation method will contain three different parameters

Parameter	Function	Preference (weight)
Territory	Return number of the territory	2
Army	Return size of the army	3
Manpower	Return value of the manpower	2.5

Table 1 The function and weight for the risk game parameters in an evaluation function

Therefore, the design for the evaluation function will be:

```
public function eval(List<Province> territory, Int sizeOfArmy, Int manpower){
    return preferences.get("Territory") * territory.size()
    + preferences.get("Army") * sizeOfArmy
    + preferences.get("Manpower") * manpower;
}
```

Code 03: The design of evaluation algorithm

2.7.2 Alpha-beta Pruning Algorithm

The alpha-beta pruning algorithm will be used in the project to produce the strategy based on the results of the evaluation algorithm.

The Code04 is a pseudo code for the alpha-beta algorithm. In the pseudo code, the parameter `strategy` is a node of a general tree which stores all possible strategies of the player. The parameter `depth` will identify the depth of the DFS. Two parameters `α` and `β` are used to cut off the subtree and they store the maximum and minimum value of the current strategy. The Boolean value, `isMyTurn`, is used to identify the player and it will change the behaviour of the algorithm.

```
int alphaBeta(Strategy strategy, int depth, int α, int β, boolean isMyTurn){
    if depth = 0 or the gameover, then
        return eval (node)
    if isMyTurn, then
        value := −∞
        for each strategy in strategy. possibleNextStrategies do
            valueOfNextStrategy := alphabeta(child, depth − 1, α, β, false)
            value := Math.max(value, valueOfStrategy)
            α := Math.max(α, value)
            if α ≥ β then break (cut-off)
        return value
    else it is enemy's turn
        value := +∞
        for each strategy in strategy. possibleNextStrategies do
            valueOfNextStrategy := alphabeta(child, depth − 1, α, β, false)
            value := Math.min(value, valueOfStrategy)
            β := Math.min(β, value)
            if α ≥ β then break (cut-off)
        return value
}
```

Code 04: The design of alpha-beta algorithm [4]

2.8 Test Design

As described in the specification document, the project will use the object-oriented testing to test the program and will strictly follow a top-down integrate testing process, namely, operation test, class test, cluster test, system test.

2.8.1 System Test

At first, the black-box test will be used to test the whole system. The test case in the phase will rely on the use cases as shown in the following tables.

Title	Configure game
Description	A user configure the number and type of the game player
Match use case	UC1, UC2, UC3

Test Steps	The game system requires the number of players from the user.
	After the user types a number, the system will check the validation.
	if it is valid,
	    the system will store the number of the player.
	else
	    the system will notify the user, the number is invalid.
	and then for all other players,
	    the user selects the player type.

Expected Result	Game start with the correct number and type of players.

Title	Play One Turn
Description	A human player plays one turn of the game.
Match use case	UC5, UC6, UC7, UC8, UC9
Test Steps	The game turn starts with 3 recruits. the player enters the Deploy Phase, for each recruit do test the recruit behaviours test the attack behaviours test the move behaviours test the invasion behaviours end the game turn
Expected Result	The player ends the turn and both of the locations of the player and results of the battle are correct.

Title	Victory
Description	A human player wins the game.
Match use case	UC11
Test Steps	The human player eliminates all enemies and occupy all provinces
Expected Result	A message about victory is shown on the screen.

Title	Defeated
Description	A human player loses the game
Match use case	UC12
Test Steps	The human player loses all provinces
Expected Result	A message about defeated is shown on the screen.

2.8.2 Cluster Test

The project will test the interaction between subsystems in this phase. The system will test the message communication between the “Behavior Handle System” (BHS), the “Game Model” (GM), the “Artificial Intelligence System” (AIS) and the “Notification Centre” (NC), namely, the communication between BHS with GM, AIS with GM and GM with NC.

Title	Communication between BHS (or AIS) with GM
Description	The message sends from the BHS (or AIS) to GM
Test Steps	The test program invokes the methods in the game model from BHS or (or AIS). Collects the results of the invoked methods.
Expected Result	The GM generate the correct response.

Title	Communication between GM with NC
Description	The data send from the GM to NC
Test Steps	`The test program requires the GM to send the results of modelling to the NC`
Expected Result	The NC receive the correct data.

2.8.3 Class Test and Operation Test

In the class and operation test, all classes and the public fields of these classes will be tested. In addition, the test methods such as "Equivalence Partitioning" and "Program Flow Graph" and the test tools such as "Unity Test Tools" and "Apache Ant", will be used in the phase.

2.9 Evaluation Design

As described in the specification document, the project will use three different methods to evaluate the game, namely, self-evaluation, 3rd party Evaluation and the Evaluation Function.

2.8.1 self-evaluation

In the project, the programmer will be required to evaluate the game quality by "function point". In the function point evaluation, the game will be divided into a series of functions in different aspects and each aspect will be allocated a weight. The programmer needs to count the number of the finished functions in each aspect to calculate the final value of the game. The table02 describes the different aspects in function point and their weight.

Function	Simple		Normal		Hard		Total
	Point	Num	Point	Num	Point	Num	
Game logic	2		3		5		
User interface	1		2		3		
Game Controller	2		3		4		
Artificial Intelligence	3		5		8		

Table 02: The table for the function points

The game logic, user interface, game controller and artificial system respectively represents the function in the game model, game controller and the artificial intelligence system. The simple, normal and hard represent the complexity of these function.

Additionally, the programmer will use the functional and non-functional requirements as the criteria to compare the risk game online such as PoloRisk and WarGameRisk.

Function	Marks (0 ~ 5)		
	The project	WarRisk	PoloRisk
Is the game online available?			
Is there a visual user interface?			
Can the player attack an enemy?			
Can the player move the army?			
Can the player deploy recruit?			
Does the game provide multiple game model?			
Does the game provide different kinds of computer player?			

Table 03: The table for the criteria

2.8.2 3rd party Evaluation

In this part, the project will design a survey to collect feedback from the player. In addition, the project will strictly follow the requirements in the "Computer Science Student project 3rd party Evaluation Procedure" [5]

Questionnaire	Marks (-5~5)
How likely is it that you would play this game on your computer or smartphone?	
Is the game user-friendly?	
Is the UI well-designed?	

Table 04: The table for the Questionnaire

2.8.3 Evaluation Function

The game intelligent agent will be evaluated through an analysis of the result of the "computer versus computer" game and also compared with the intelligent agent in other online Risk game. The "Evaluation Function", such as the heuristic evaluation function or the static evaluation function, shall be used to evaluate.

Parameter	Function	weight
isWin	if win return 1 else return -1	10
GameOverTurnNumber	Return 10 - the number	1

Table 04: The table for the Parameter of the evaluation

Therefore, the design for the evaluation function will be:

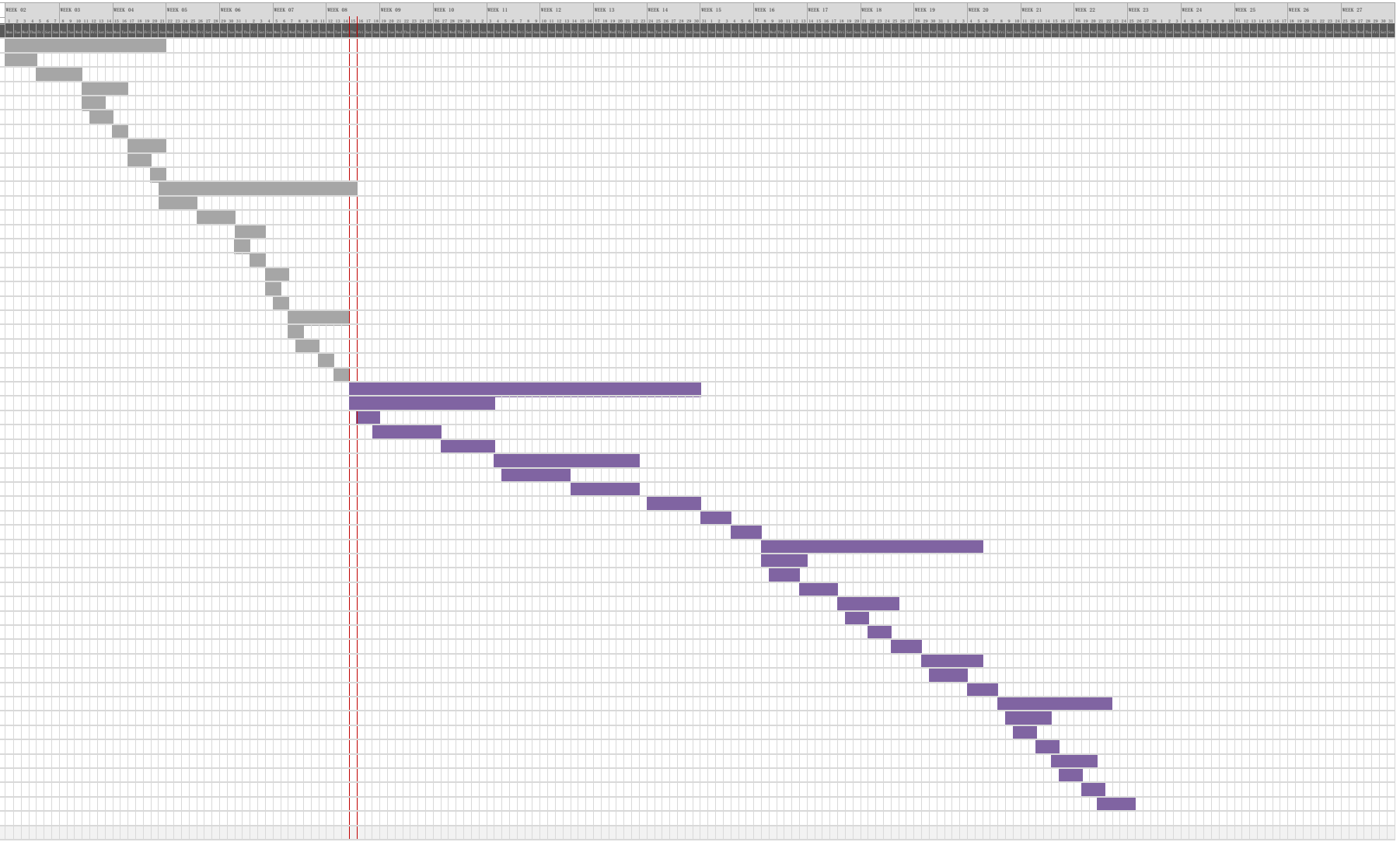
```
public function eval(Boolean isWin, Int numOfTurns){
    int isWinValue = ;
    return weight.get("Turn")* (10-numOfTurns)
         + weight.get("isWin")* ((isWin)? 1:-1);
}
```

Code 04: The design of the evaluation function.

RISK GAME

University of Liverpool
Yichao Xu

Project Start: Mon, 10/1/2018



TASK	PROGRESS	START	END
Phase 1 Specification	100%	10/1/18	10/21/18
1. Possibility Study	100%	10/1/18	10/4/18
2. Requirment Elicitation and A	100%	10/5/18	10/10/18
3. Requirment Specification	100%	10/11/18	10/16/18
3.1 Specify the function requir	100%	10/11/18	10/13/18
3.2 Specify the non-function re	100%	10/12/18	10/14/18
3.3 Identify Evaluation methods	100%	10/15/18	10/16/18
4. Conduct of the Project and P	100%	10/17/18	10/21/18
4.1 Preparation for the game	100%	10/17/18	10/19/18
4.2 Design and Implement plan	100%	10/20/18	10/21/18
Phase 2 Design	100%	10/21/18	11/15/18
1. Architecture Design and Spec	100%	10/21/18	10/25/18
2. Interface Design	100%	10/26/18	10/30/18
3. Components Design	100%	10/31/18	11/3/18
3.1 Identify the objects	100%	10/31/18	11/1/18
3.2 Identify the attributes and	100%	11/2/18	11/3/18
4. Data structure and Algorithm	100%	11/4/18	11/6/18
4.1 Data Structure Design	100%	11/4/18	11/5/18
4.2 Write pseudo-code for key m	100%	11/5/18	11/6/18
5. Evaluation Design	100%	11/7/18	11/14/18
5.1 Evaluation criteria design	100%	11/7/18	11/8/18
5.2 Test Design	100%	11/8/18	11/10/18
5.3 Consider the possible partic	100%	11/11/18	11/12/18
5.4 Consider expected conclusion	100%	11/13/18	11/14/18
Phase 3 Implementation	0%	11/15/18	12/30/18
1. Implement a game prototype	0%	11/15/18	12/3/18
1.1 Implement basic game UI	0%	11/16/18	11/18/18
1.2 Implement game rule and log	0%	11/18/18	11/26/18
1.3 Implement basic game AI	0%	11/27/18	12/3/18
2. Improve game AI	0%	12/4/18	12/22/18
2.1 Implement the normal comple	0%	12/5/18	12/13/18
2.1 Implement the hard complexi	0%	12/14/18	12/23/18
3. Improve the game UI	0%	12/24/18	12/30/18
3.1 Improve the Game Asset	0%	12/31/18	1/3/19
3.2 Improve the Game UI format	0%	1/4/19	1/7/19
Phase 3 Test		1/8/19	2/5/19
1. Make a test plan	0%	1/8/19	1/13/19
1.1 Requirement traceability	0%	1/9/19	1/12/19
1.2 Declaim the test item and r	0%	1/13/19	1/17/19
2. Object-Oriental Testing	0%	1/18/19	1/25/19
2.1 Test class	0%	1/19/19	1/21/19
2.2 Test Cluster of cooperation	0%	1/22/19	1/24/19
2.3 Test the complete system	0%	1/25/19	1/28/19
3. Debug	0%	1/29/19	2/5/19
3.1 Debugging	0%	1/30/19	2/3/19
3.2 regression testing	0%	2/4/19	2/7/19
Phase 4 Evaluation		2/8/19	2/22/19
1. Self-evaluation	0%	2/9/19	2/14/19
1.1 Setting Function Point	0%	2/10/19	2/12/19
1.2 Analysis and Evaluate	0%	2/13/19	2/15/19
2. AI Algorithm Evaluation	0%	2/15/19	2/20/19
2.1 Setting evaluation Function	0%	2/16/19	2/18/19
2.2 Evaluate	0%	2/19/19	2/21/19
3. 3rd Party Evaluation	0%	2/21/19	2/25/19

Insert new rows ABOVE this one

4. Reference

[1] A. Lamorisse, Risk, 6th ed. MA 01915. Printed in U.S.A: Hasbro, 1993, pp. 2,11-13.

[2] "Asset Store", *Assetstore.unity3d.com*, 2018. [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/23399. [Accessed: 15- Nov- 2018].

[3] K. Lee and S. Mahajan, "A pattern classification approach to evaluation function learning", 2018.

[4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, N.J.: Prentice Hall/Pearson Education, 2003.

[5] Computer Science Student project 3rd party Evaluation Procedure, 1st ed. Liverpool: University of Liverpool, pp. 2-3.