

CS 319-Object-Oriented Software Engineering

Analysis Report

Royal Road

Group 17

Kadir Ata Dönmez

Veysel Alperen Ceylan

Yakup Kılıç

Table of Contents

1 Introduction**1**

2 Overview2

3 Gameplay Elements3

4 Functional Requirements4

5 Non-functional Requirements5

6 Pseudo Requirements6

7 System Models**1**

7.1 Use Case Model2

**1 Introduction**

Royal Road is purely Java based adventure game and the basic fundementals of the game like movement, level exploration and action will be inspired from Hotline Miami, such as UI design, playability and art style.

**Hotline Miami link:**

http://store.steampowered.com/agecheck/app/219150/?l=turkish

Primary aim of this game will be heavily on Player-to-Object and Object-to-Object interactions like killing creatures by shooting bullet at them, passing corridors and collecting items and such.

**2 Overview**

The player controls a single chracter to explore rooms and corridors that are filled with creatures and dangerous traps. The player will be able to move freely all directions except corridor lines and map borders. Shooting capibality is at the direction that only looking way. There will be many items that will upgrade and enhance player’s abilities. These abilities may be temporary or permanent. Some creatures/enemies would be coded behavior, some of them would have basic AI. There would be two type of objects that are independent and dependent.

**Gameplay Elements**

**• Player**

The player controlled character is a King in Royal Road in the game. Character has health and can have 2 different weapons also can pick items from chests.

**• Enemies**

Enemies will have two different types of behavior set to them: melee and ranged. Melee attacks when it close to player and ranged shots from range. They also can patrolling in certain areas or coming closer when they see the player.

1. Homeless: The weakest enemy. They are melee class enemy. They attack with punch and they take down player’s one health. They are patrolling in certain areas.

2. Ninja: Slightly stronger counterpart of Homeless, they can attack from range. They throw iron shurikens and they take down player’s one health. They can patrolling in certain areas or start shooting when they see the player.

3. Waylayer: This guy hits harder and he is harder to take down then the Ninja, but they attack slower than ninja. They attack with sword that take down player’s two health, so they are melee class enemy. They are coming closer when they see the player.

4. Pirate: The hardest enemy in game. They use guns which take down player’s two health. They can patrolling in certain areas or start shoting when they see the player.

**• Objects (Independent)**

Independent objects execute their behavior with/without player’s intervention.

1. Spike Traps: Melee traps on the floor.

2. Fire Traps: Fire version of Spike Traps. When Player get on it, his health decrease one.

**• Objects (Dependent)**

Dependent objects, contrary to independent objects, will not execute their predetermined behavior unless player gives any kind of input to them. However, they are the only way to change the existing behavior of the independent objects as well.

1. Locked Door: If the player has a key, it will open, giving access to new room.

2. Chest: Player can get health, key or weapon from chests.

3. Door: Unlike Locked Doors, these can be opened by triggering mechanisms associated with this door to grant access to other parts of the current room.

**• Terrain**

Terrain types are usually for determining the graphics that a tile will use, but some terrain might have special perks attached to them, the main example spike traps and fire traps, which is still a walkable floor, but it will damage the players when walked on.

1. Floor: Standart terrain type with nothing special about it.

2. Wall: Standart impassable terrain.

**• Others**

Other objects which do not belong to any specific category will belong here, in this case:

1. Gold: Golds will spawn depends from room to room at predetermined spot. If player picks up the gold, player’s points increase.

2. Key: Player need key to open locked door.

**4 Functional Requirements**

**1. Play Game**

The main purpose of the game is to reach the final room where you will encounter the final level, you will win the game. The players will need to find keys to gain access to new rooms by completing certain objectives depending on the room type (combat/puzzle/both). Combat includes wiping out all enemies in the room which will in turn spawn the key at the end of the room.

**2. Change Options**

The user can alter one option:

-Screen size

The system would provide two different size that can be changeable by user.

**3. Display Credits**

Display of credits for any external sources we might use and list of our names will be displayed on the screen.

**4. Display Instructions**

The game have some complexities about controling. The user might desire to access list of controls before starting the game. The menu would list them out on screen.

**5. Open Bestiary**

There would be different types of creatures and also their behavior would be completely unknown for the initial time players. Hence, the Bestiary would involve info about all enemies the player would encounter in the game.

**6. Pause Game**

System also allows to give a short break to player whenever is wanted.

**5 Non-functional Requirements**

**• Game Performance**

Royal Road would require to hold low latency from each keys so as to make the playability as smooth as possible. Object rendering would also be a concern for the performance since game will offer higher number of objects on the screen simultaneously.

**• Graphics Performance**

As explained above, rendering them as efficiently as possible is a notable.

**• User-friendly Interface**

The user should know what is going all times, that means the UI ought not be included with useless info and make it easy for user to figure out what is going on at their screen while they are entertaining the game.

**• Reusability**

The game has different maps and enemies so that users can create their own special replayability.

This kind of modular system can allow the game to be modified and extended in the future.

**6 Pseudo Requirements**

1. The code would be written in Java only

2. Desktop only

3. Level editing should be easy to learn.

**7 System Models**

**7.1 Use Case Model**

**Change settings**

**Primary Actor:** Player

**Interests:** The Player wants to change screen size.

**Pre-condition:** Player has to be in Main Menu

**Post-condition:** If player has changed any setting, new setting will be applied during the menu screen.

**Entry Condition:** Player clicks on "Change Settings" in the main menu.

**Exit Condition:** Player clicks on ““Return to Main Menu” in the settings screen.

**Open Enemies**

**Primary Actor:** Player

**Interests:**

Player wants to get info about creature/enemy types at the main menu

System displays Bestiary panel on the screen

**Pre-condition:** Player has to be in Main Menu

**Post-condition: -**

**Entry Condition:** Player clicks on “Open Enemies” in the main menu

**Exit Condition:** Player clicks on “Return to the Main Menu” in the enemies screen

**Display Credits**

**Primary Actor:** Player

**Interests:** Player desires to reach info about developers of the game and external sources used in game development.

**Pre-condition:** Player has to be in Main Menu

**Post-condition: -**

**Entry Condition:** Player clicks on “Display Credits” in the main menu.

**Exit Condition:** Player clicks on “Return to Main Menu” in the credits screen.

**Pause Game**

**Primary Actor:** Player

**Interests:** The Player wants to pause the game

**Pre-condition:** The player is in the game

**Post-condition:** -

**Entry Condition:** The player presses the corresponding key that opens the “Pause Menu”

**Exit Condition:** The player presses “Resume Game”

**Instructions**

**Primary Actor:** Player

**Interests:** Players want to see different keybindings and general information on how to play the game

-System displays all of these on the screen

**Pre-condition:** Player has to be in Main Menu

**Post-condition: -**

**Entry Condition:** Player clicks on “Instructions” in the main menu.

**Exit Condition:** Player clicks on “Return to Main Menu” in the instruction screen.

**Play Game**

**Primary Actor:** Player

**Interests:** The Player aims to reach last level and pass the last level.

**Pre-condition:** The system would use default settings if not the player has changed the settings beforehand.

Player has to be in Main Menu

**Post-condition**: -

**Entry Condition:** Player clicks on “Start Game” in the main menu.

**Exit Condition:** Player clicks on “Quit Game”.

-Player reaches to final exit

-Player loses all life

The use case diagram can be the best summarized by the fallowing figure below. (figure-1)

