CS 319 - Object-Oriented Software Engineering

Design Report

Hallway Rush

Group D

Caner Bozkurt

Doga Dikbayir

Ezgi Ebren

Mert Uygur

Table of Contents

1. Introduction

1.1. Purpose of the system

Hallway Rush's game system aims for player to have fun without worrying about complicated HUD[1] (heads-up display) that confuses player. The game's user interface is going to be implemented with simplistic manner that does not interrupt player with unnecessary buttons or screens. This allows players to simply focus the game and try to beat the high score.

1.2. Design Goals

Hallway rush is going to be implemented with latest Java Development Kit. Hallway Rush will take full advantage of Java; game will be platform independent, lightweight and robust.

Adaptability: One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.[2]

Efficiency: Gameplay of The Hallway Rush will be fluid and responsive since the main purpose of the video gaming is creating smooth images that entertain players. Thus, we are going to implement the game such that memory will be allocated efficiently to improve performance. Also Java is self-garbage collecting programming language, therefore memory leaks of the game will be minimal.

Reliability: The Game will be implemented using object-orientation principle, which includes encapsulation. Encapsulation will make the game

---

[1] Wikipedia contributors. "HUD (video gaming)." *Wikipedia, The Free Encyclopedia*.
[2] Wikipedia contributors. "Java (programming language)." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 10 Nov. 2014. Web. 13 Nov. 2014.

error free and robust since inner parts of the game that shouldn't be changed by user will be protected.

Extendibility: Since the game is implemented with object-oriented principle after the first version of the game implemented, game can be further improved by adding additional object classes.

1.2.1. Trade Offs

Functionality vs. Usability: Since Hallway Rush aims for an easy game-play, usability will have priority over functionality. This means, the game will only have fundamental functionalities such as quitting the game and keeping high score.

Efficiency vs. Memory: The game will use minimal memory by keeping the graphics or effects simple so that we can provide high-performance with smooth transitions.

2. Software Architecture

2.1. Subsystem Decomposition

The overall system will consist of subsystems in order to provide modularity and reduce complexity of the design. The three-tier architecture is adapted when decomposing the system into subsystems.[3]

In the User Interface level there is the HRInterface subsystem, which will contain GUI components like menus. The user will be able to interact with these components in order to communicate with the game. The user's inputs will be collected at this layer and will be sent to the next layer, Game Core & Logic. This layer contains controller subsystems, which take inputs collected by the HRInterface subsystem and process them in order to run the game properly. At this stage, the SettingsManagement subsystem plays a crucial role on handling game's settings. The user will interact with the settings menu in the HRInterface subsystem and the raw parameters will be passed to

[3] *Object-oriented Software Engineering: Using UML, Patterns and Java*. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2003. 291. Print.

the InputManagement subsystem. This component will process the raw input data and send it to the SettingsManagement subsystem. The subsystem will communicate with the GameManagement subsystem in order to set the games with the provided options. The inputs received during the game will again be collected by the HRInterface component and the raw input data will be processed by the InputManagement subsystem. Unlike the settings management process, this dynamic input data will be directly sent to the GameManagement subsystem in order to run the game.

Finally, to store high score data, the HighscoreManagement subsystem in the Game Core & Logic layer sends data to the HighscoreStorage subsystem located at the Storage layer.

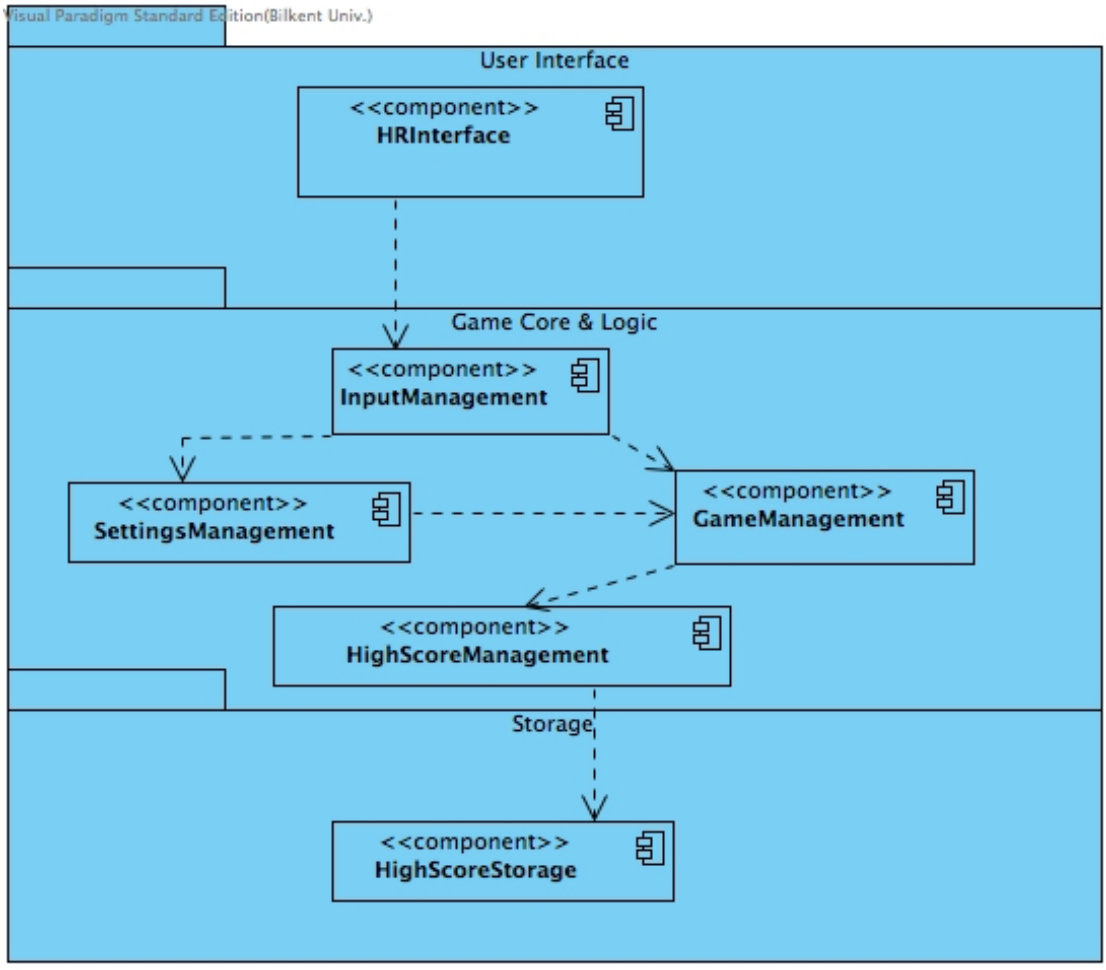

Figure 1- Subsystem Decomposition shown using UML component diagram (layers are shown as UML packages)

2.2. Hardware/Software Mapping

Hallway Rush is going to be implemented in Java programming language. The game needs keyboard in order to take inputs from player. Since we will implement the game in Java and system requirements are going to be minimal, a computer with software installed such as an operating system and Java Virtual Machine that runs java associated programs. Also, we will benefit from java's platform independency; therefore game will run on most operating systems.

As the system decomposition shows there is Game Core & Logic layer that includes InputManagement subsystem. InputManagement serves as a bridge between hardware and software.

2.3. Data Management

Our game will only store high-score data in the computer's hard disk. The high score data will have a format in order to allow the HighscoreManagement subsystem parse it and use it properly in the system. The formatted high score data will be stored in a text file and the system will read from it and write to it when it is necessary.

2.4. Accessibility and Security

Hallway Rush is designed to be a single player game. Therefore there are not any network security protocols to be handled by the system. The game does also not have a user database so there are no authentication credentials to be secured by the system. The only concern at this stage may be accessibility, which could corrupt the system's data and cause problems in the runtime. So the accessibility to the core system files should be limited. However this limit should not prevent the user from experimenting on the game to a certain level.

2.5. Boundary Conditions

2.5.1. Initialization

Hallway Rush does not have an executable file so it does not require installation. It is executed as a simple .jar file.

2.5.2. Termination

Termination of Hallway Rush occurs in these cases:

- If the player decides to quit before, during or after the game; they can terminate by clicking the “x” button on the upper right corner of the game panel, or by choosing “quit game” option from the main menu.
- If a failure occurs during the game which requires termination and fixes.

2.5.3. Failure

If any of the files used in the game are corrupted at any point before the initialization or during the execution of the game, the game will be terminated with an error message and corrupted data will be fixed. However, minor errors in performance will not result in termination. Instead, the user will be informed about the nature of the problem and advised to re-initialize.

3. Subsystem Services

In this section further information about every subsystem introduced before will be given at the system design level. The details about the classes used by the subsystems will be introduced and explained in the object design phase.

HRInterface: This subsystem will display the user interface to the player and collect the raw input data from the user in order to send it to the Game Logic layer. The subsystem will consist of GUI components. Below the services provided by the subsystem are listed with short descriptions:

- Play Game: A new game is initialized with the settings handled by the SettingsManagement subsystem.
- Change Settings: The settings menu is displayed in order to let the player communicate with the SettingsManagement subsystem.
- Display High Scores: The high scores page is displayed
- Display Credits: Credits page is displayed

InputManagement: This subsystem will process the raw input data collected by the HRInterface and pass it to the other control subsystems. The InputManager class will serve this purpose using event listeners.

SettingsManagement: This subsystem will set the system's settings variables to parameters passed by the user through HRInterface and InputManagement subsystems.

The services this subsystem provides, map the options in the settings menu:

- Change character's gender: The subsystem will set the image file's link associated with the character's gender according to the input passed by the player
- Turn music on/off: The subsystem will set the associated boolean variable according to player's choice
- Change ambient: The subsystem will set the background image file's link according to player's input
- Change game mod: The subsystem will set the game mod variable to standard/god/hardcore according to user's choice

GameManagement: This subsystem contains the most important services in the system which determine the gameplay. This component is in concurrent communication with the InputManagement subsystem to process player's input through the GameEngine:

- Game Engine: This service is the core of the program. It performs all the logical operations in the game with the parameters passed by other subsystems and it runs the game loop.
- Game Panel: This service visualizes the changes made by the game engine on game objects and the game map.
- Game Elements: This service will declare the game components and interact with the game engine

HighScoreManagement: This subsystem is basicallay responsible from determining which score is a high score and which high score will be recorded with a gold medal.

- High Score Manager: This service will take information from the game engine such as the score and if the boss is beaten. It will also collect

player's nickname. Finally it will format and send the formatted data to the storage layer.

HighscoreStorage: This subsystem is located at the storage layer of the system architecture. It is responsible from storing formatted high score data in a text file.

- High Score File: This service will be designed as a class which contains information to initiate the high score file and parse data from it in order to display it on the game screen.

The subsystem services are introduced and explained at this system design phase. Further information about the actual Java classes, which are going to be used in the services, will be introduced and explained at object design phase later.