# Description: https://lh5.googleusercontent.com/ykiT9AqltWwZhr4yy0I8EMArov3nDMgonw73zz3xpW55Pjnk8xUHN8UQ9cSBvnTYiNAypNe7dAXlTfHdscP_Uzi7ewyNNO03_D5Zd9cClbcOOZwZ3EI

CS 319 - Object-Oriented Software Engineering

System Design Report

Hurdle Race

Group Members

Güneş Batmaz

Rabia Ecem Afacan

Selim Mıdıkoğlu

Haluk İncidelen

Table of Contents

[1. Introduction 3](#_Toc446870361)

[**Maintenance Criteria:** 4](#_Toc446870362)

[**Performance Criteria:** 4](#_Toc446870363)

[**Trade Offs:** 4](#_Toc446870364)

[1.3 Definitions, acronyms, and abbreviations 5](#_Toc446870365)

[**Abbreviations:** 5](#_Toc446870366)

[1.4. References 5](#_Toc446870367)

[1.5. Overview 5](#_Toc446870368)

[2. Software Architecture 5](#_Toc446870369)

[2.1. Overview 5](#_Toc446870370)

[2.2. Subsystem Decomposition 6](#_Toc446870371)

[**2.3.1 Layers** 8](#_Toc446870372)

[**2.3.2 Model View Controller** 8](#_Toc446870373)

[2.3. Hardware / Software Mapping 8](#_Toc446870374)

[2.4. Persistent Data Management 8](#_Toc446870375)

[2.5. Access Control and Security 9](#_Toc446870376)

[2.6. Boundary Conditions 9](#_Toc446870377)

# Introduction

#### 1.1 Purpose of the system

Hurdle Run is 2- D game which includes multiple player game mode as well as single player mode in advance. Two players can compete at the same time. Significant properties of the system are to provide high quality graphical game play process, to create challenge between two players by the futures of Hurdle Run, to create hand-eye coordination according to time and the game play scenes and the most important one is to create a enjoyable game for the players. Additionally the system is created to be easy, both players or just single player can manage to direct the game with couple of buttons and with an efficient interface which we provide.

#### 1.2 Design Goals

The main goal of design process comes from the quality, property and the main features that we want from system to have them. So In analysis process we declared our non-functional requirements. We will use these requirements to implement more usable and efficient designed system. The main goals of design given below.

##### End User Criteria:

###### The Efficiency of The Managing:

The main aim of the design is to create an interactive system with pure interface with easy usage. The player can manage system with couple of clicks which provides easy basic system usage. These clicking actions include keyboard inputs and Mouse inputs which provides action on the game. Mouse will navigate the menu and the keyboard will manage operations, controls and actions.

The Ease Of the Learning:

The properties and features of the game are so easy to understand. But since the use has no clue about them we will provide instructive and informative document which will inform the user how to play the game , what to do in the game or what are the basics of the game.

### **Maintenance Criteria:**

Extendibility:

In the lifetime of the Project it needs to be provided that, the system should be able to get new features, external properties. That is why we create extendible system which can be added new features in future steps like (new hurdles, new player modes).

Portability:

To make our system platform independent we use java for implementing our system. In further step if our game reaches more and more people, this means our game needs to work in different platforms. JVM[1] solves this. So for the sake of portability we create our game with Java.

Modifiability:

For possible feature change, the software system can be modified.

### **Performance Criteria:**

##### Response Time:

To not interrupt user’s joy and fluent game process we need to response every action immediately on the game stage. This is a must for every single game system. This provides smooth gameplay with no interruptions and immediate responses.

### **Trade Offs:**

##### Easy and Efficient Usage Vs. Functionality:

Since we do not create any complex game system like users should control system with bunch of different control platforms., we are aimed to create really easy system which user can manage, direct and play easily and efficiently. So we did not add complex components which may cause more functionality less efficient and easy usage or learning. So in this manner our system does not provide too much functionality.

##### Performance Vs. Memory:

The main component we want to have in our system is performance , since we are working with graphical displays, actions, effects and transitions, the system we want should work smoothly that is why we did not put too much into memory. We only want to keep data of scores, wins and player records. These do not affect performance too much. So we have given more importance to performance more than memory.

## 1.3 Definitions, acronyms, and abbreviations

### Abbreviations**:**

MVC: [2] Model View Controller

JDK: [1] Java Development Kit

JVM: [1] Java Virtual Machine

## 1.4. References

[1] <http://en.wikipedia.org/wiki/Java_(programming_language)>

[2] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2010, ISBN-10: 0136066836.

## 1.5. Overview

In this section, we represented purpose of the system, which is basically entertaining the player as much as possible, to achieve this purpose we defined our design goals in this part. Our design goals are determined according to provide the portability, ease of use, ease of learning, high performance, high maintainability. In this respect of course we made some trade-offs to realize our goals. We sacrificed from functionality to make our system simpler and understandable, also we sacrificed from memory to gain performance on playing smooth animations and effects.

# 2. Software Architecture

## 2.1. Overview

In the software architecture section, we separate our software system to subsystems. The purpose of this is making the relationships between subsystems more clear and increasing cohesion of subsystem components. Therefore our main goal is apply Model View Controller(MVC) architectural style to our system by decomposing.

## 2.2. Subsystem Decomposition

The purpose of this part is dividing system into relatively independent parts. By doing this organization of the system become more clear and understandable. The effects in division is protection the main features of system which are performance, modifiability and extendibility. In order to meet non-functional requirements of our system dividing into relatively independent parts is crucial.

User Interface, Game Management and Game Entities are names of our main focused subsystems which shown in figure-1. These subsystems work on different cases, however they are connected to the each other in the possible changes for future. The details of this design is shown in figure-2 where connection between these subsystems can be seen. For example changes in User Interface will affect the Game Manager because it has a role for control of interface.

The main goal of this section is make the our software more understandable and have a good design before we start implementation of our software. Also dealing with possible changes in our software without errors is important for future.

2.2. Architectural Styles

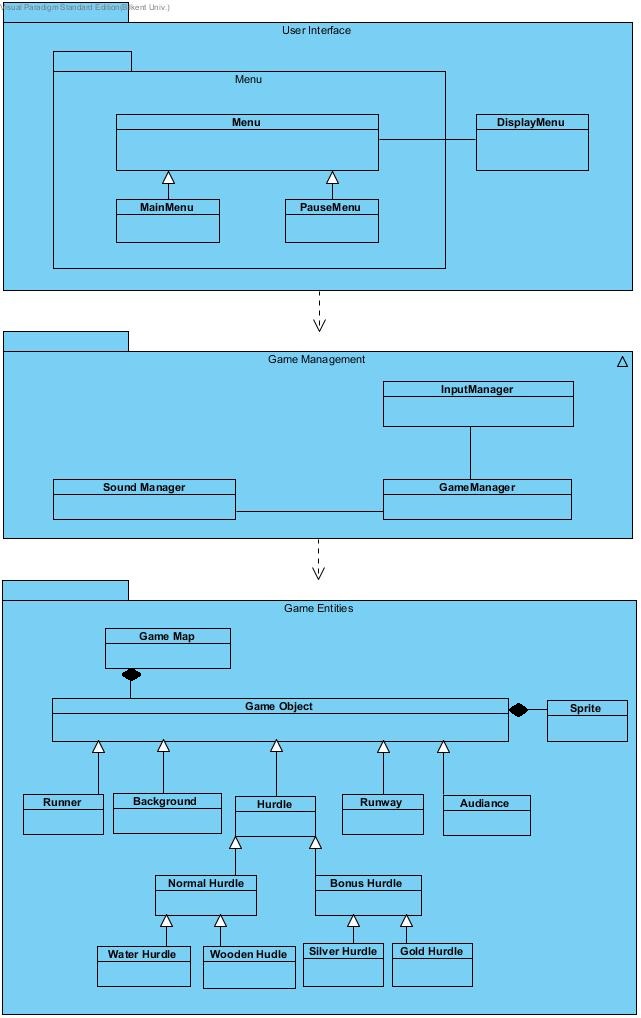


Figure 1

### **2.3.1 Layers**

In system decomposition we divide our system into three layers which are User Interface, Game Management and Game Entities. The hierarchical relationship among these layers is as follows: our top layer is User Interface which is not using any other one, however it is interactive with user action. The layer after comes User Interface is Game Management which has control of our main game logic. The last layer is Game Entities which has our necessary objects. We use closed architectural style in our decomposition in which a layer can only access to layer below. (See figure-3)

### **2.3.2 Model View Controller**

Main purpose of this architectural system is dividing system to three subsystems which are main, view and controller. In control part we use Game Management class which has main objects of our system. In view part we use User Interface layer which has interaction between user. The advantage of using this architecture is achieving changes on the interface does not affect the model of system.

## 2.3. Hardware / Software Mapping

Hurdle Race will be implemented in Java programming language (latest: Java 8). Our game needs a basic keyboard (for control of player by space and up keys) and mouse for user inputs as hardware configurations. System requirements for our project is a computer with installed software like operating system and a java compiler (Eclipse, NetBeans etc.) to write and run java files (.java). Our software is not require the Internet connection.

## 2.4. Persistent Data Management

In Hurdle Race we will store the data in client hard disk drive, we will not use any database, because we use data which is accessed in real time. Moreover, when the system requires; the necessary files will be loaded on the system. Data we use in our system are: background images, objects’ images, buttons for choices and display screen images.

## 2.5. Access Control and Security

In Hurdle Race there is not any personal account and profiles, therefore we do not have any personal security problem. Also Hurdle Race does not require any kind of network connection, only requirement to play Hurdle Race is initialization of game.

## 2.6. Boundary Conditions

##### Initialization

Hurdle Race does not require an install, since game will come with an executable .jar file.

##### Termination

Hurdle Race can be terminated by clicking “Quit Game” button in main menu. If a user wants to quit while playing, system has a pause button by which player can return to main menu and perform quit.

##### Error

If there is a problem caused by missing data such as sound or images the game will not start.