

**Cs 319**

**Object-Oriented Software**

**Engineering Project**

**Design Report (First Draft)**

Section 2 / Group 2K

Ege Darçın

Alptuğ Albayrak

Özgün Özkan

Furkan Erdem

Instructor: Bora Güngören

**TABLE OF CONTENTS**

1. **Introduction**
   1. **Purpose of the System**
   2. **Design Goals**
      1. Well-Defined Interface
      2. Ease of Use And Learning
      3. Performance
   3. **Maintenance**
      1. Portability
      2. Reliability
      3. Modifiability
      4. Understandability
      5. Extendibility
   4. **Trade-Offs**

* Functionality vs. Usability
* Flexibility vs. Efficiency
* Performance vs. Portability

1. **Software Architecture**
   1. **Overview**
   2. **Architecture Style**
      1. Architecture Type
   3. **Subsystem Decomposition**
   4. **Hardware / Software Management**
   5. **Persistent Data Management**
   6. **Boundary Conditions**

* Initialization
* Termination
* Failure

1. **Subsystem Services**
   1. **Player**
   2. **Input Manager**
   3. **Interaction / Physics**
   4. **User Interface**
   5. **Game Manager**
   6. **Menu Data**
   7. **Ball**
   8. **Background**

**1) Introduction**

**1.1Purpose of the System**

Head Soccer is a 2-D arcade game. By designing heads of character’s much bigger compared to their size, providing power ups and optimizing character’s controls, main purpose is to provide users with a game in which they can have an entertaining experience. In addition, in order to provide user with more variety, thanks to random mode, different ball types, different conditions and goal sizes can be set each round. Head soccer aims to offer the players an entertaining game with variety and good design.

**1.2 Design Goals**

**User Criteria:**

**1.2.1) Well-Defined Interface**

While designing the user interface, our main purpose is to make the game as desirable and enjoyable as we can. User interface plays a key role in creating a positive first impression on players as much as it directly influences how desirable the game is. Game will be consisted of two backgrounds which are normal football field and a green pitch with tribunes. In addition, players are going to be provided with a clear description of the location and the intended purpose of the buttons which aims to provide users with conveniences while playing the game.

**1.2.2) Ease of Use and Learning**

Easiness of usage and learning the game rules in an easier way, provides player with getting use to the game in a short time which directly effects player’s entertainment while playing the game. In order to accomplish that, help section will be provided to the players which clearly enlightens players related with the purpose and the gameplay of the game.

**1.2.3) Performance**

Since the performance of the game, directly effects the quality, there are some certain features that our program should support such as fast loading. By coding the program wisely and creating a program with a high run time performance, we will make the game smooth which will also increase the entertainment of the game.

**1.3) Maintenance**

**1.3.1) Portability**

In a software design, portability is an essential issue which helps the program to be reached by a large number of users. In order to accomplish that, our program is going to be implemented on Java which is supported by lots of OS, devices and versions. As a result, our program will be able to play in most of the system which are suitable for running JVM.

**1.3.2) Reliability**

Considering the various inputs that can come from the users, our main purpose is to make sure that our program will not be crushed. After determining the possible error cases, errors are going to be solved by using exceptions and giving notifications to the user in the case of occurrence of an error.

**1.3.3) Modifiability**

During the implementation stage it is inevitable to make some changes and modify the Project. Our existing functionalities will be open to some modifications. By minimizing the coupling, our aim is to make sure that a small change in the code does not affect the entire program in a considerable way.

**1.3.4) Understandability**

Since we have to consider the ones who uses and analyze the Project, we need to design our Project in such a way that it is understandable for anyone. By trying to express every detail in a clear and understandable way in our Project reports, we aim to accomplish this purpose. When it comes to implementation stage, we are aiming to provide some comments to express some details of our Project which will increase the understandability.

**1.3.5) Extendibility**

While designing a software, it is always significant to design a system which is suitable to extensions such as new functionalities (i.e. new Modes, new power-ups etc.) Providing a system which is suitable with extensions, our main purpose is to provide users with an entertaining adventure.

**1.4) Trade-Offs**

* **Functionality vs. Usability**

Headball is easy and simple game in which players can control the footballers with arrow keys, left, up and right, or w, a and d. This game appeals players from different range of ages. Game doesn’t so much functionalities and complexities in terms of playing; on the other hand, it appeals players with its simplicity. It can be easily understandable, playable. We have sacrificed more complex functionality; however, we get more funny game which doesn’t require lots of mental and physical efforts.

* **Flexibility vs. Efficiency**

We used open architecture (transparent layering) in which each layer can call operations from any layer. It provides run-time efficiency, while closed (opaque) architecture allows program to be more flexible and maintainable. We have sacrificed these features by choosing open architecture.

* **Performance vs. Portability**

We have chosen Java for implementing our program. Java is portable language which can work on any computer with Java Virtual Machine. Java language is also known by our entire group members; writing codes wrote in different programming language and merging them would be so hard and time consuming for us, so we chose Java to implement our program. However, Java brings also some disadvantages. For instance, JAVA does not have a good running time performance compared to other languages. Game will also have 2d graphics which allows game to be run by any computer with minimal requirements of hardware.

**2) Software Architecture**

**2.1) Overview**

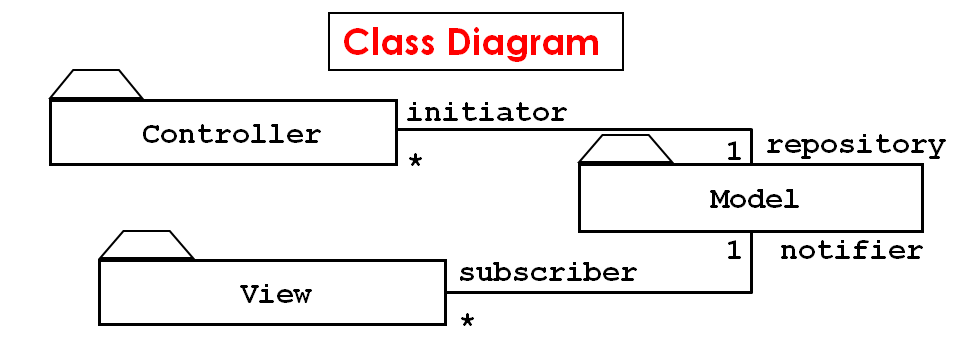
This section will be mainly on structure of the system. In the first part of the report, we have identified design goals. Now, we are modeling system design as a set of subsystems. We have decomposed the overall system into manageable parts by using the principles of cohesion and coherence. This part and third part will include identification of subsystems, services, and their association to each other. Objects and classes in the previous analysis report are ‘seeds’ for our subsystems. And uses cases in previous report allow us to define services in this part.

**2.2) Architecture Style**

We have chosen the MVC (Model-View-Controller) architectural style. In this style, we classified subsystem into three different types.

Model subsystem is responsible for application domain knowledge. This subsystem expresses the system’s behavior in terms of the application domain. View subsystem handle display related issues. Controller subsystem interacts with user, gets input and does the computations needed to be done behind and modifies both Model and View subsystems.

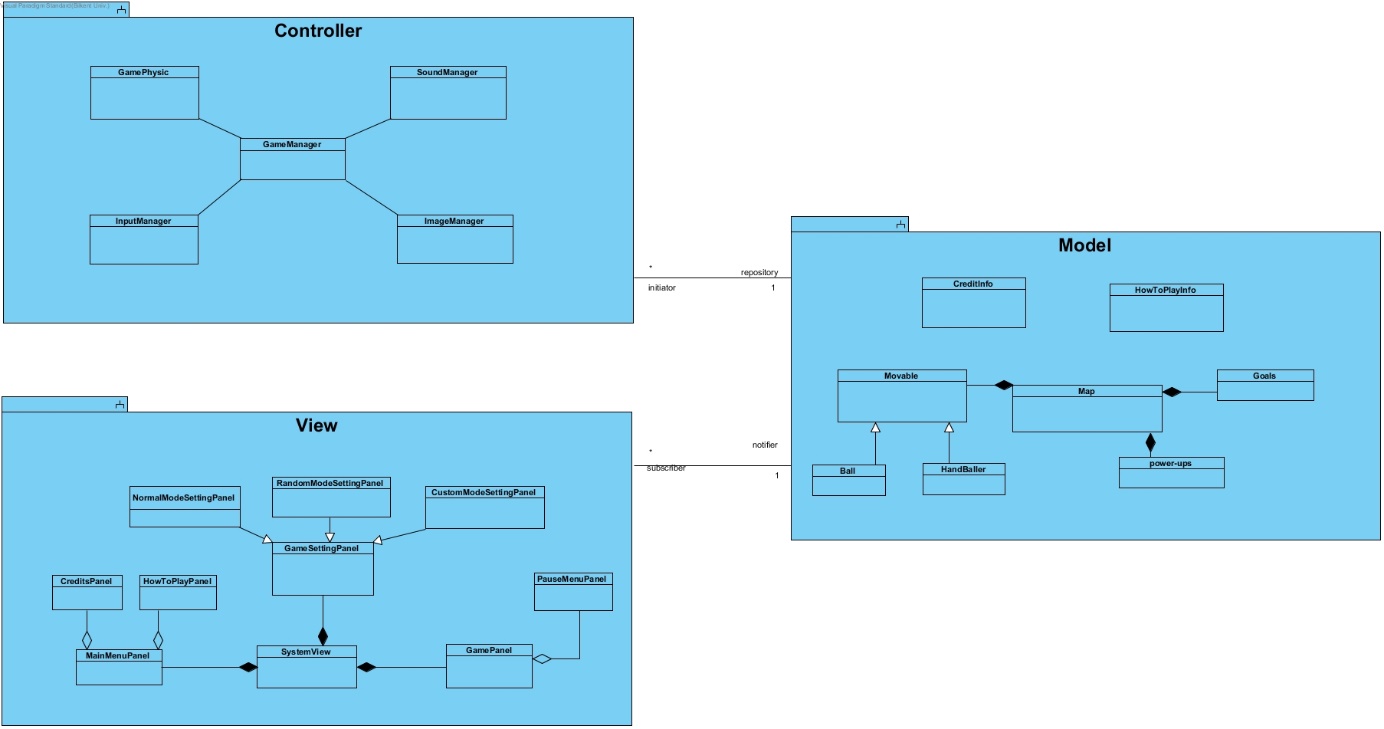
This style provides high coherence by allowing classes to perform similar tasks and to be related with each other, and provides low coupling by decreasing dependency between subsystems and providing that they have less information about each other. This style brings also some tradeoffs which were mentioned in the previous part.



**2.2.1) Architecture Type**

We have divided our subsystems horizontally into several independent subsystems. They have mutual knowledge about each other. While some of them have compile-time dependency, some of them have also run-time dependency. We also use open architecture in which each layer can call operations from any layer.

**2.3) Subsystem Decomposition**

****

**2.4) Hardware / Software Management**

Our game will be coded in JAVA mainly because everyone in the group has an experience with JAVA and furthermore JAVA has some very beneficial libraries to use. Images and sound effects will be stored in .png and .wav format.

On the hardware part, our game will need a keyboard and since our game’s system requirements will be very low any PC with JAVA environment can run the game without any problems.

**2.5) Persistent Data Management**

The data for the game will be stored in the user’s hard drive and the game will only store some basic stats like highest number of goals in a match, best victory etc. because of this no database will be used.

**2.6) Boundary Conditions**

**Initialization**

Our game will not need an installation to work. The game will work as a .jar executable file. When this executable clicked the game will be opened in a new window.

**Termination**

There are multiple ways to terminate our program. The most basic way is to simply click the “Exit” button in the main and pause menu. In addition to this you can close the program by clicking “X” at the top of the window or ending the task with task manager.

**Failure**

There are a few cases where there can be a failure, each of them will have an error message specific for that error. These failures consist of: missing program files, absence of JDK on the computer.

**3) Subsystem Services**

**3.1) Player**

The player component is in the model and view of our MVC architecture. This component includes HeadDesign, Jersey, Coordinates and Status classes. Link between Interaction/Physics component and Player component provide the data about location that a player has. Player component takes the data that about the coordinations it and transfer the view controller about the location. Moreover, Player class also provides data about which animation is required for an interaction.

**3.2) Input Manager**

Input Manager component handles the inputs of the users and it is in the controller of our MVC architecture. Basically, it collects the input data of the users from keyboard and manipulates our model, which controls the movements of their player.

**3.3) Interaction / Physics**

Interactions component is in the model architectural design. It includes Collision, Power-up, Goals and Sound classes, and also implements Physics class. The data about interactions that player has during the game play is in the interaction component. It interacts with game manager and player component. The interaction/physics component provides the data about coordination and physics which is needed for both view and control architectures. In game, Powerups and physics are controlled and transferred to UI. Moreover, when required, sound class is used for notifying the users, like a shoot, goal or menu navigation sounds to add a new sense to the game.

**3.4) User Interface**

The user interface component is in the view of our MVC architecture. This component takes data from all other components on condition that direct and indirect. Player, Ball, Game manager, input manager and menu data components have the direct interaction with user interface component. The input manager provides hardware inputs, such as movements, powerup usages and menu option button usage to the user interface component. All movements of the players, balls etc. are displayed here, with control of the interaction/physics.

**3.5) Game Manager**

The game manager is in the model of MVC architecture. It handles all the game play features. The game manager interacts with interaction/physics and user interface components. It takes the game data from interaction component and sends the data of the game to user interface component.

**3.6) Menu Data**

Menu Data will be in the model of MVC architecture. This system consists of Help, Settings, Credits, Modes and Pause Menu and classes. Menu Data component will allow players to choose what type of game options they want to pick, namely characters, goal sizes, ball settings, randomizing options etc. Also, reflecting changes of settings and transition between screens, Menu Data provides information for different menu screens (Pause menu, main menu, etc.) to user interface component.

**3.7) Ball**

Ball will be in the Model and view of MVC architecture. This system consists of Shape and Properties classes. The ball will be main focus on the gameplay and also be in the design. The location and physics of the ball will be calculated via interaction/physics and its data will be sent to UI. Different types of balls will be used and their properties will be checked on here. Moreover, the goal check will be used from here, since the coordinates of the ball defines the goal.

**3.8) Background**

Background will be in View of MVC architecture. This system consists of animation, Scoreboard and goals classes. Since the shapes of goals and the medium change during each round in the random mode, the change and properties will be controlled by here. Also, scores will be displayed in the background and this makes it interact with the interaction and UI classes.