1. Introduction

1.1 Purpose of the system

The “HeadBall” is a soccer game in which the player tries to score in the restricted time or score count against another player. The aim of the game is to successfully have score advantage against competitor. There will be in game power-ups to enhance the gameplay. Obstacles will prevent the ordinary ball rotation to create spontaneity in the game. The pitch will be small to enhance player and game interactions. In this game, user can select their soccer player which are visualized in funny appearances (big heads). The “HeadBall” has a potential to attract attention of soccer fans and players who likes competitive games. Our aim is to make a sport game to give users enjoyable time by competitive matches. Players will control characters by the dedicated keyboard buttons, players have the ability to jump, strike the ball with leg and go left or right directions.

1.2 Design goal

The main goal for a computer game is to entertain the player. To achieve this task, it is needed to focus on little details which are not directly noticeable at a first glance. This section details the design goals of the system such as end user criteria, maintenance criteria, performance criteria that come with our chosen way of implementation

Maintenance criteria

Extensibility:In the lifecycle of a software program, it is important to maintain the ability to add or remove features. As object oriented software engineering principles imply, the first goal in development is to create a highly maintainable software.

Portability: In today’s software and hardware world, everything is changing rapidly. However, it has a solution, cross-platform application/game development which runs platform independent. Java is a platform independent programming language such that no matter what operating system or processor architecture is system used. “Write once, run everywhere”

Modifiability: Since our system is going to object oriented base and it has several subsystems with the minimum level of coupling and high level of cohesion, it should be easy for us to modify the system. The reason we want to change the system is that as every game HeadBall is also required to change some of its features and user interface to keep people's attention on the game. So our system has to have high modifiability goal.

Performance criteria

Response Time: Even though we do not have any heavy data base system, still we need to have decent amount of response time to play the game. As you know, games require a lot of reaction time so that is why when the button pressed system should recognize it immediately. Also it is same as the object interaction with each other. When the player hit the ball, ball should react to touch and change its direction immediately.

Throughput: Since the system can be played as multiplayer and single player, system should be able to accompolish several takes in a short period of time. For instance, when the both players press the button which is responsible for the jumping, both players should jump at the same time.

Smooth Graphics: Main character, power-ups, goalkeepers and the whole level environment is going to be visualised in a retrospective way. Therefore, at this point, we will not experience a noticeable loss of speed in order to process the graphics and move with the game engine.

End user criteria

Utility: In today's World people want to socialize while they are spending time in front of computer while they have leisure time since they need to socialize. In today's World people like to play multiplayer games since use their leisure time as socializing tool. Therefore, in today's World e-sports and other online competitive tournaments are becoming more important. Our application gives user a change user to play competitive matches with his/her friend. Therefore, it is an opportunity for users to socialize.

Usability: In our application we use very basic controllers like other games are used. Moreover in our user interface when user enters the application they will find what they want to do in the game very easily since user can understand how to move one screen to other very easily by reading labels that are on the buttons written by very basic and understandable English.

Ease of learning: In our game we used "wasd" and arrows as initial controls like most of the other games. Therefore, user will high probably imagine how he/she can control the game even if he/she cannot imagine they can see the controls by reading instructions in the game. Moreover, instruction includes a lot of information about the game. By reading the instruction user will have enough knowledge about game. Instruction includes a lot of information that explains the purposes and process of tournaments, PvP games and single player game. If user will see that he/she should make some practice to expertise in this game he/she can practice in training mode.

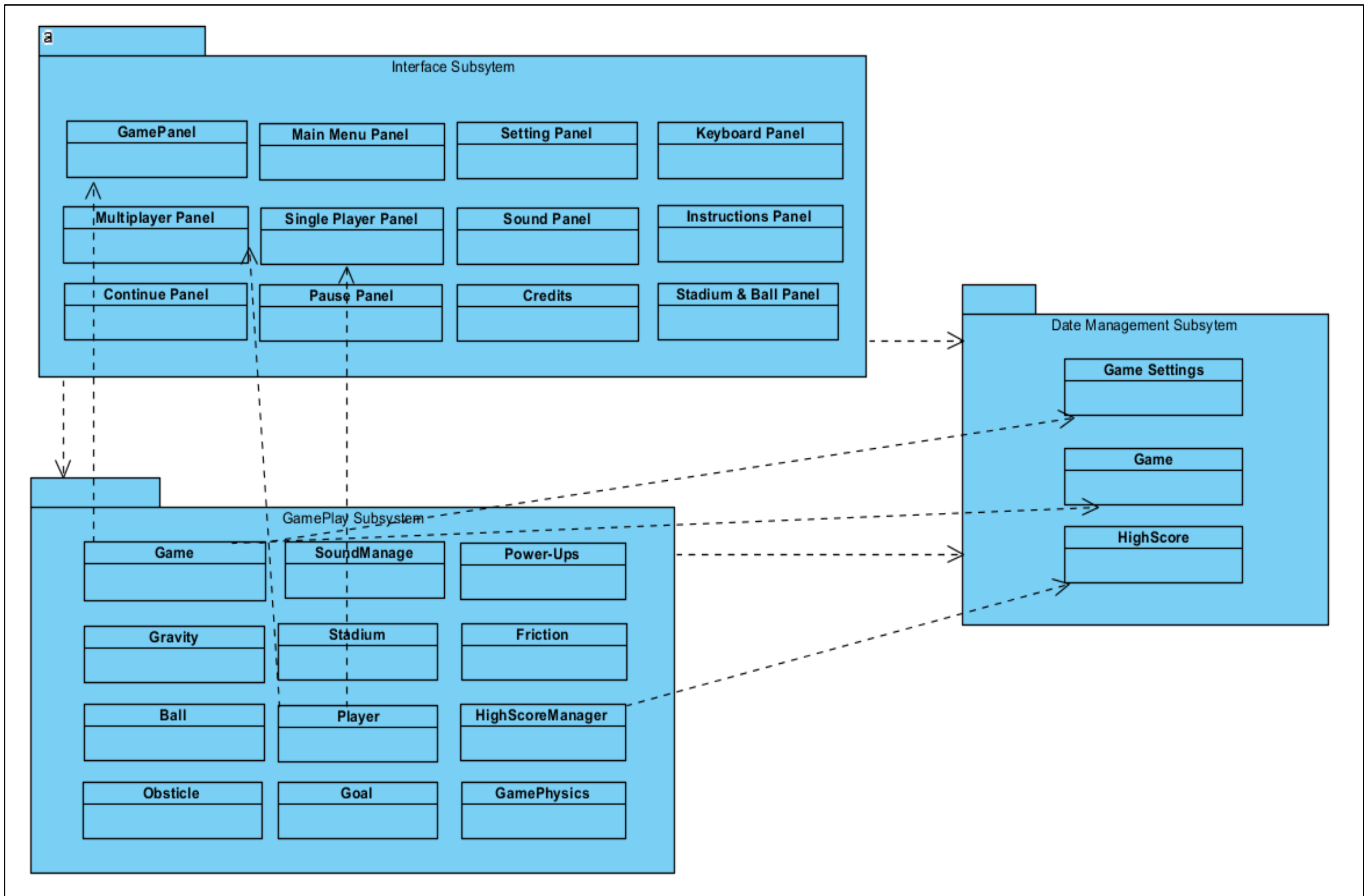
2.1

The software is composed of three subsystems. These are User Interface Subsystem, Data Management Subsystem and Gameplay Subsystem. The criteria between these subsystems are maximum high coherence and low coupling. User Interface Subsystem controls the user interaction with system by consisting of menu bar and information screen so it provides navigational functionalities to the user.

User Interface Subsystem takes user input and transmits to the Gameplay Subsystem and shows the output of Gameplay Subsystem to the user.

Data Management Subsystem handles storing and accessing game settings, game mode and high score so the interface provided by Data Management Subsystem handles storing settings and retrieves game mode and data of high score which are basic text files.

Gameplay Subsystem presents the most essential functionalities because it manages all game actions. So Gameplay Subsystem provides initialisation of stages, creation and destruction of monsters, creatures, food and weapon in a harmonious way with user input so rules of two different stages. Gameplay Subsystem interacts with User Interface Subsystem for getting user input and interacts with Data Management Subsystem for getting game mode and settings.



2.2 Hardware software mapping

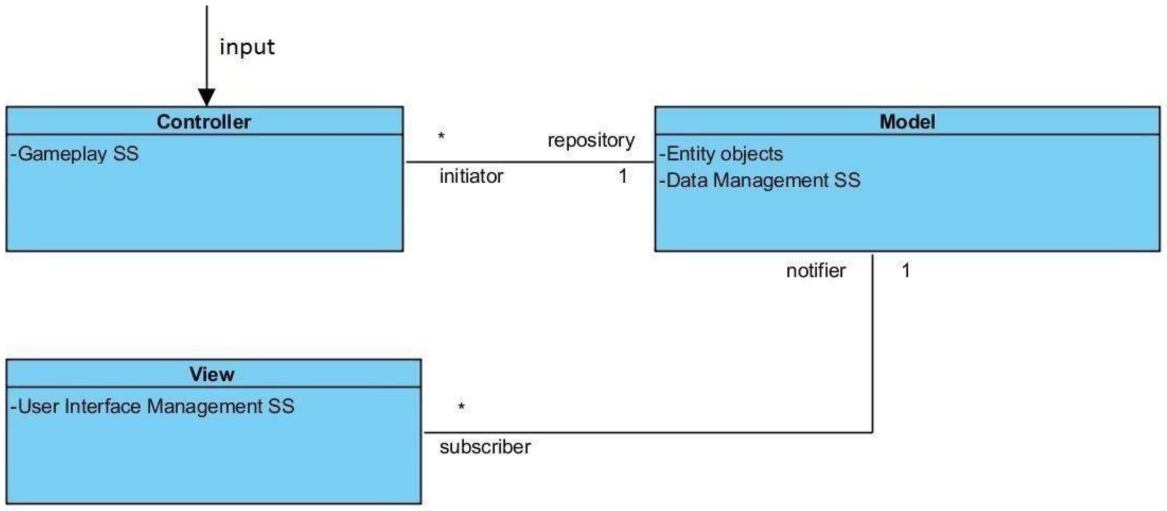
Our game is developing by using Java language as development language. Therefore, in order to run this game user should have java run time environment in his/her computer. In order to control the game user what user will need are only keyboard and mouse.

In order to run the game successfully almost every computers that are produced in last decade can be used since required systems are very low

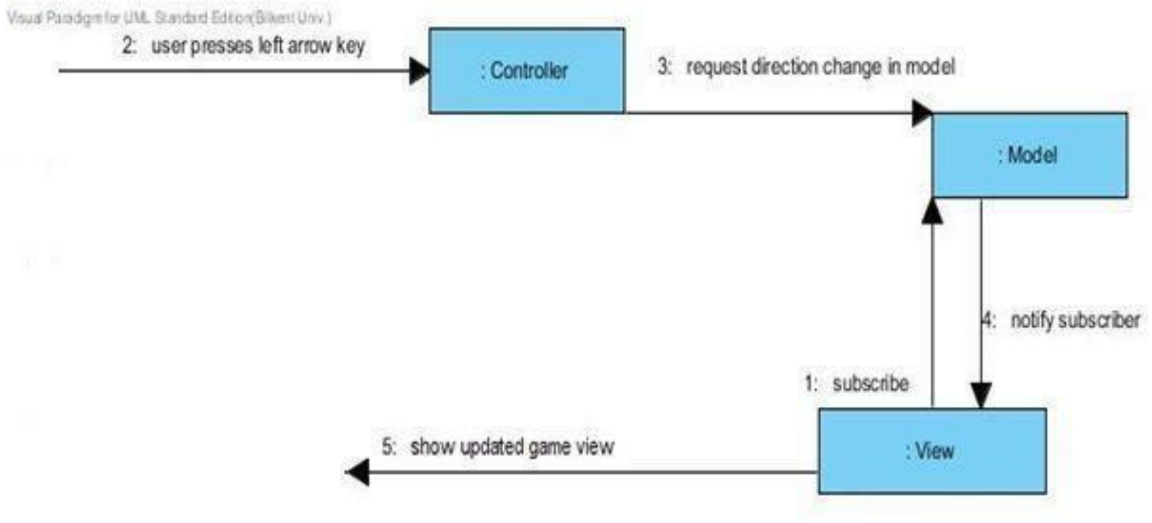
2.3

We used 2 architectural design patterns in our design, which are Façade design pattern and MVC (Model-View-Controller) design pattern. We need to divide our project into subsystems to reduce the complexity of the system. We do this by using MVC pattern. Each part of the project is belongs to controller, model or view part.

We used Façade design pattern for making software libraries easier to use and readable also to reduce dependencies of the system. It provides an interface to a long code sample like a library or a subsystem. This action is done by creating an intermediate class which takes the requests and handles them by using the collection of classes that needs to be simplified. While simplifying the classes and subsystems, it also provides reusability, flexibility, extensibility and easy maintenance. We used Façade design pattern in out interface and gameplay subsystem

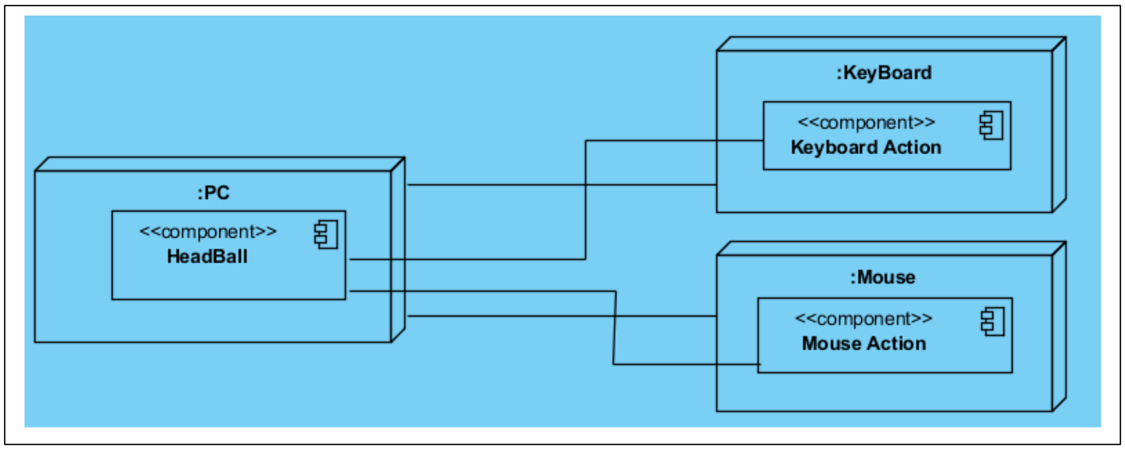


Below diagram depicts the sequence of events in MVC. It models how model, view and controller response to a user input during HeadBall.


Visual Paradigm for UML Standard Edition(Bilkent Univ.)
2: user presses left arrow key
: Controller
3: request direction change in model
: Model
4: notify subscriber
1: subscribe
5: show updated game view
: View

2.4 Persistent Data Management

Since our application does not use any online database system, we use users' hard disk to store the data. Most of the data that should be stored are images and sounds and it stored inside the project folder. Moreover, scores of the games is stored inside a txt file inside the project folder.



2.5 Boundary Conditions

If any of a required data in the game corrupted the game will crush and it will give some error. While playing a game dependent on the game mode that is chosen's limits exceeded the game will be ended. For instance if user plays a goal limited game the game will be over if goal limit will be reached.

Façade pattern:

Façade pattern is very important to build an appropriate communication among the classes. Thanks to this system complexity of the project get simpler and

2.6 Global software Control