CENG 491

Computer Engineering

Design 1

Detailed Design

Report

Geeks In Action

M. Oğuz ŞEN	:	1395532
Talat ÖZER	:	1347806
Nur Muhammet ARINÇ	:	1448364
Cuma KILINÇ	:	1395193

Contents

Table of Figures

1 Introduction

1.1 Motivation

We are going to implement a 3D arcade football game for Linux for our senior project. Most of the current popular football games such as Pro Evolution Soccer series, Winning Eleven series, FIFA series and Championship Manager series are designed for Windows operating systems and playing these games on Linux is always a big problem. Thus, we wanted to make a football game for Linux. Main aim of our project is to develop well and entertaining a football game for Linux users and to satisfy the desire of football games running on Linux environment as much as we can. Also adding our game to Ubuntu repository is another big aim of our project, so that volunteer developers are going to be able to make further improvements on this game. The other problem of football games is that most of the games are too realistic. Making an enjoyable football game is another main aim of our project. In order to make an enjoyable football game, new game modes is going to be included in our game.

1.2 Project Description

Our senior project is a 3D game project, which have a genre of arcade football. Arcade behaviors consist of our main skeleton of the game. The game is going to be fully same with other football games. The main aim of our project is making a good alternative football game for Linux. So our game is going to both include classical football game modes and extra "for fun" modes such as "Japanese goal mode", "German goal mode" and "Street football". Each of these game modes will include their own rules. Number of players is going to be between 1 or 4 players. All players

can use only keyboard. The game can be played both single player and multiplayer via LAN. The game is going to be played in a 3D atmosphere. Atmosphere is going to be rendered as realistic as possible. Stadiums, outdoor game areas and 3D characters are going to form our game atmosphere. Artificial intelligence is also a part of our 3D football game. AI is not a main part of our 3D football game but AI is needed when only one player plays against computer and also when playing one or more players, AI is going to control footballers that human players do not control at that time. Initial implementation of AI is going to satisfy basic conditions, which we think that it is going to be enough for simulation of an enjoyable game. We are planning further development of AI as much as we could find appropriate time for it.

1.3 Purpose of Document

Purpose of this document is to give information about the detailed design steps of the game. These topics are introduced and explained in detail throughout the document.

1.3.1 Game Play

The game play is explained in detail at the "Game Descriptions and Mechanics" chapter of this document. We added some explanatory drawings in order to make everything clear.

1.3.2 Game User Interface

In order to build a powerful interface which is going to be understandable and functional, complex graphic components are used in modern games. Since it is not expected from team members to be experienced on tools such as Autodesk 3ds Max, Adobe Photoshop etc.; creating complex graphic components is not a primary objective of this project. In our project, we are going to design not so much complex but easy to understandable and functional graphical interfaces.

1.3.3 Game Concept

Our game is going to be both similar and different from classical football games. In other words, game is going to include both classical modes and different modes.

1.4 Design Constraints

1.4.1 Project Schedule

Gantt chart part explains the schedule of our project. According to this time chart, we have to design and implement the project in 8 weeks. We must scatter workload equivalently because we are supposed to make demonstrations; these are in fact small milestones for us.

1.4.2 Language Constraints

We will use an object oriented language to program the game. C++ will be our programming language. We chose it because team members` programming experience is mostly on C++.

1.4.3 Data Constraints

We need a huge amount of data because we have to store footballer names, footballer attributes, club names, 3D footballer characters, atmosphere components and saved games.

1.4.4 User Interface

User interface is going to be easy and simple in order to make it user friendly. It is going to be easy to start a game. In other words users do not need to enter a lot of sub menus to start a game.

1.5 Project Goals and Scope

Designing and implementing a 3D football game for Linux is the main goal of this project. During the development of project, we are going to follow the below methodology:

- ✓ analyzing the current football games
- ✓ analyzing requirements for game
- ✓ analyzing specifications for game
- ✓ design of a game according to the defined criteria
- ✓ implementation and testing of game
- ✓ documentation and technical support

In addition; since being fun and entertaining is a criteria for evaluation of a football game, we are going to develop our game considering this criteria as much as we can.

1.6 Team Organization

Since every member has nearly same experience about design patterns and concepts of this project, decisions about the project are going to be made by group consensus. This situation yields our team to have democratic decentralized structure. Communications within the group is going to be horizontal. Like many other software engineering projects, presence of a team leader is vital in game projects. We cannot evade this fact, so we chose a team leader for coordination and interactions within our team. While making decisions, our team leader will consult each team member in order not to go against our team structure.

Assignment of roles in a team to each team member is listed as below:

Team Leader and Initiator: M. Oğuz Şen

Initiator and Optimist: Cuma Kılınç

Devil's Advocate and Time Keeper: Talat Özer

Recorder and Gate Keeper: Nur Muhammet Arınç

1.7 Process Model

We are going to use Scrum design model in this project, which is commonly used with agile software development methodology. It involves iterative and incremental development and we are going to make use of it frequently while developing AI and graphics part of the project. Since it defines strict and irreversible steps, Waterfall model cannot be applied to our project.

We are going to perform an object oriented approach in the project development progress. We are going to focus on the modularity to provide efficient development in future. Since this project is going to be open source and our game is going to be uploaded to Ubuntu repository, we are going to create components and packages in a way to provide Linux developers easy to understand and develop our project.

1.8 Tools

Certain needs will arise throughout the implementation of our game by means of different aspects. After some research and analysis, we decided on following issues.

- Our project is expected to run on Linux operating system. The reason is that Linux is open source and team members are supporters of open source and free software.
- We are going to develop our game on C++ .
- In terms of graphics, we are going to use C++ for implementation and Irrlicht as graphics library and game engine.
- In terms of graphics, OpenGL are going to be used for graphics rendering. The reason is that all of team members are experienced about C++ programming

language and OpenGL.

- We are going to use FMOD as a sound library. Since FMOD is a very powerful library and it is free to use it in non commercial projects; we chose to use it in our project.
- For creating and editing images for texture, we are going to use GIMP, since it works also with Linux environment.

2 Constraints of Development Process

2.1 Constraints Related to Members of the Project Team

Our group, Geeks in Action consists of 4 senior computer engineering students of Middle East Technical University's Computer Engineering Department. The project is being manufactured for senior project course of the mentioned department. Here are some important issues about project:

- The process started on October 2009 and is going to end in June 2010. Currently, we have 6 months left from 9 months of development.
- Senior project course schedule bounds development team with deadlines of reports, phases and etc.
- Other important courses and academic works of team members critically limit their effective project development time.

2.2 Constraints Related to Implementation

We don't have any restrictions about networking, graphics and artificial intelligence implementation.

2.3 Constraints Related to Licensing and Environment

Since our project is intended to be an open source project and expected to run on Linux operating systems, we are limited in using tools that have licenses compatible with GPL. We are going to license our game with GPL, so our game should be designed in a way that is not going to violate this licensing. Also tools and

software development kits that are developed for Windows environment is not going to be used, since we have no intention of running this game under Windows operating systems.

3 Game Description and Mechanics

3.1 Game modes:

In this part, game modes are explained. The properties of game modes are listed below.

3.1.1 Traditional Mode:

Traditional mode is going to be played by normal football rules. The number of players is 22; each team has 11 players and 7 substitute players. Each team is going to have minimum 3 and maximum 7 substitutions. There is going to be two halves; each half length is going to be equal. Match length is going to be between 5 and 30 minutes. There is going to be a referee. There is going to be rules like penalty, throw in, corner, goal kick. Match length is going to be adjusted by users. Match length options are 5, 10, 15, 20 and 30 minutes. 45 and 90 minutes are not considered because in that case, the match is going to be long and boring for players; so these options are not preferred by users. There is going to be a referee and players can choose accuracy of referee decisions. Match properties such as offside rule, injury probability of a player etc. are going to be adjusted by players. Rules like penalty, throw in, goal kick, corner kick etc. are also present in this mode.

Figure 1: A screenshot from FIFA 2010 game

Figure 2: A screenshot from PES 2010 game

3.1.2 Japanese goal mode

The first "for fun" mode is Japanese goal mode. Japanese mode has different rules from traditional mode. In this mode, there is not going to be a goal keeper for a team. Size of a goal is going to be smaller than traditional mode. Each team is going to have 2 or 3 players. There is not going to be a referee. If a team performed 3 corner kicks, there is going to be a penalty shoot out. There is not going to be throw in and goal kick. The match is going to be finished when a team scores a defined amount of goals.

This mode is actually based on the skills of the players. More talented players can have superior advantage against their opponents since AI for goal keepers is not defined in this mode, scoring for a team is going to be dependent to the skills of footballers only. And also, reduced goal size is going to force the player to perform shooting in a more accurate way.

Figure 3: An actual performance of a Japanese goal match

:

3.1.3 Street football mode

Another "for fun" mode is the street football mode. This game mode is going to include from 2 vs. 2 to 4 vs. 4 playing. There is going to be only one goal and one goalkeeper. There is not going to be a match time. There is going to be a goal limit to win the match, for example; the player who reaches 3 goals first, is going to be winner of the match. The game is not going to be played in a stadium. There is going to be different place choices such as streets and suburb areas. We are planning to use different footballer models for this mode. (i.e. small children)

Figure 4: Screenshots from Urban Freestyle Soccer game

Figure 5: A photograph of children playing street footbal

Figure 6 : Another photograph of children playing street football

3.1.4 German goal mode

Another "for fun" mode is German goal mode. The first "for fun" mode is "German goal" mode. There is going to be a single goal and its goal keeper. This goal keeper is not going to belong to any teams on the field. As a different property; for each technique used in scoring a goal, is going to be rewarded with distinct extra

score values. (For example bicycle kick have 3 score points, volley have 2 score points.) Like street football mode, the game is not going to be played in a stadium. There is going to be different place choices such as streets and suburb areas.

Figure 7: A photograph of children playing German goal

3.1.5 Indoor Mode (Saloon Football)

Our final "for fun" mode is indoor mode. As expected, the game area is an astro pitch and rules are going to be same as an astro pitch football game. The number of players is limited as 6 for each team. Matches are going to be played on 2 halves; each half length is going to be equal. There is not going to be substitution for all teams, so that injury possibility for players is going to be turned off automatically in this mode. There is not going to be a referee, so there are no rules such as throw in, goal kick and corner kick for this mode. There are different indoor stadium choices. We are planning to use different indoor stadium models. So that the users are going to be able to choose the saloon that they want to play the match.

Figure 8: A photograph of people performing saloon football

3.1.6 League Mode

In the league mode, the users are going to play matches according to the schedule of the preferred league. League mode is going to cover only traditional mode, other game modes are going to be played as single matches. In the league mode; different league choices are going to be involved, such as Turkish Super League, Italian Serie A Calcio, Spanish La Liga, German Bundesliga and French Ligue One. Players and teams from these leagues are going to be arranged according to 2009 – 2010 football season. (A future development of this project might include updates on this database regarding changes in real life.)

Each league choice is going to have a specific league cup and each league is going to have certain number of matches in one season period. There is going to be a ranking table which is going to show each team's current position in the league and other information related with that team. (such as matches won, matches lost etc.) This table can be viewed by player after a match. The player can control only one

team in the league. Remaining matches (the matches that are not played by the user) are going to be scored in a manner such that stronger teams are going to have higher probability values in the creation of random score values for matches. The team controlled by the player is going to receive points with respect to the score of a match. The team is going to gain 3 points after winning a match and 1 point after a match resulted in draw. The team is not going to gain any points after losing a match.

Some properties of the footballers are going to be changed in a season. For example, a footballer's stamina is going to decrease, after playing in more than five successive matches. Also a footballer's booking status is going to be recorded. For example, if a footballer is punished with a red card in a match, he is going to miss next 2 matches. A footballer is going to be miss the next match, if he is punished a yellow card in both previous and current matches. If a footballer is punished with a yellow card twice in the same match, he is punished with a red card and he is going to miss the next match.

3.1.7 Training Mode

In additional to all these modes, a training mode is also going to be included in the game. Main aim of this mode is to provide users a platform where they can improve their playing skills. There is going to be a training menu for the selection of a specific training mode. In this menu, following training modes are going to exist.

Shooting: Users can learn tricks of shooting by performing shooting.

Penalty Shootout: Users can improve themselves by performing a lot of penalty shootout training

Corner kick: Users can learn how to make corner kicks into a scoring opportunity.

Free kick: If a player is good free kick taker, most of the free kicks can be resulted in a score. By that reason; a user can become a fatal free kick taker with proper free kick training.

Different training places can be chosen by users, such as stadiums, streets and

indoor places.

Figure 9: A screenshot of a training menu from Pro Evolution Soccer game

Figure 10: A screenshot of training mode game play screen from Pro Evolution Soccer game

3.2 Edit Mode

In this mode, users can create their own players and use these created players in the game. Also users can change the properties of current players. When a user creates a character; physical attributes of players can be chosen by user. Player skills such as dribbling, attack and defense can also be determined by user.

Figure 11: Screenshot from edit mode of Pro Evolution Soccer game

3.2.1 Footballer Properties

Every football player has unique attributes such as;

- ✓ Name
- ✓ Age
- ✓ Condition
- ✓ Shooting
- ✓ Current team
- ✓ Height
- ✓ Defense
- ✓ Speed
- ✓ Weight
- ✓ Position
- ✓ Price
- ✓ Agility

✓ Attack	✓ Short Passing	✓ Injury
✓ Reputation	✓ Long Passing	✓ Constitution

The values of these properties are between 0 and 20. (except current team, injury, age and height) According to these properties, each footballer is going to have a power between 0 and 20. Properties such as injury and constitution are temporary variables that are used in actual game play; so that the user is not going to be able to change these attributes.

Figure 12: A screenshot of footballer properties screen from Pro Evolution Soccer game

3.2.2 Team Properties

Each team has unique properties such as;

- Offense
- Defense
- Speed

- Strength
- Overall point

Properties are between 0 and 20. Overall properties are going to be decided according to average of other properties.

3.3 Weather Condition

The game is going to have three main weather conditions: fine, rainy and snowy. Each weather condition is going to have different effects on outdoor football environment. Each weather condition is going to have different graphic properties.

Figure 13: A screenshot showing snow covered pitch

Figure 14: A screenshot showing fine weather condition

3.4 Environment Properties

The game is going to contain certain game areas that are suitable as a football field. In traditional mode; game areas are going to be stadiums. For German goal mode and street football modes; game areas are going to be different outdoor places such as streets and suburb areas. For the indoor mode and Japanese goal modes, matches are going to be played on an astro pitch. There are going be various stadium options (such as Nou Camp, Anfield Road, Inonu stadiums) for traditional mode.

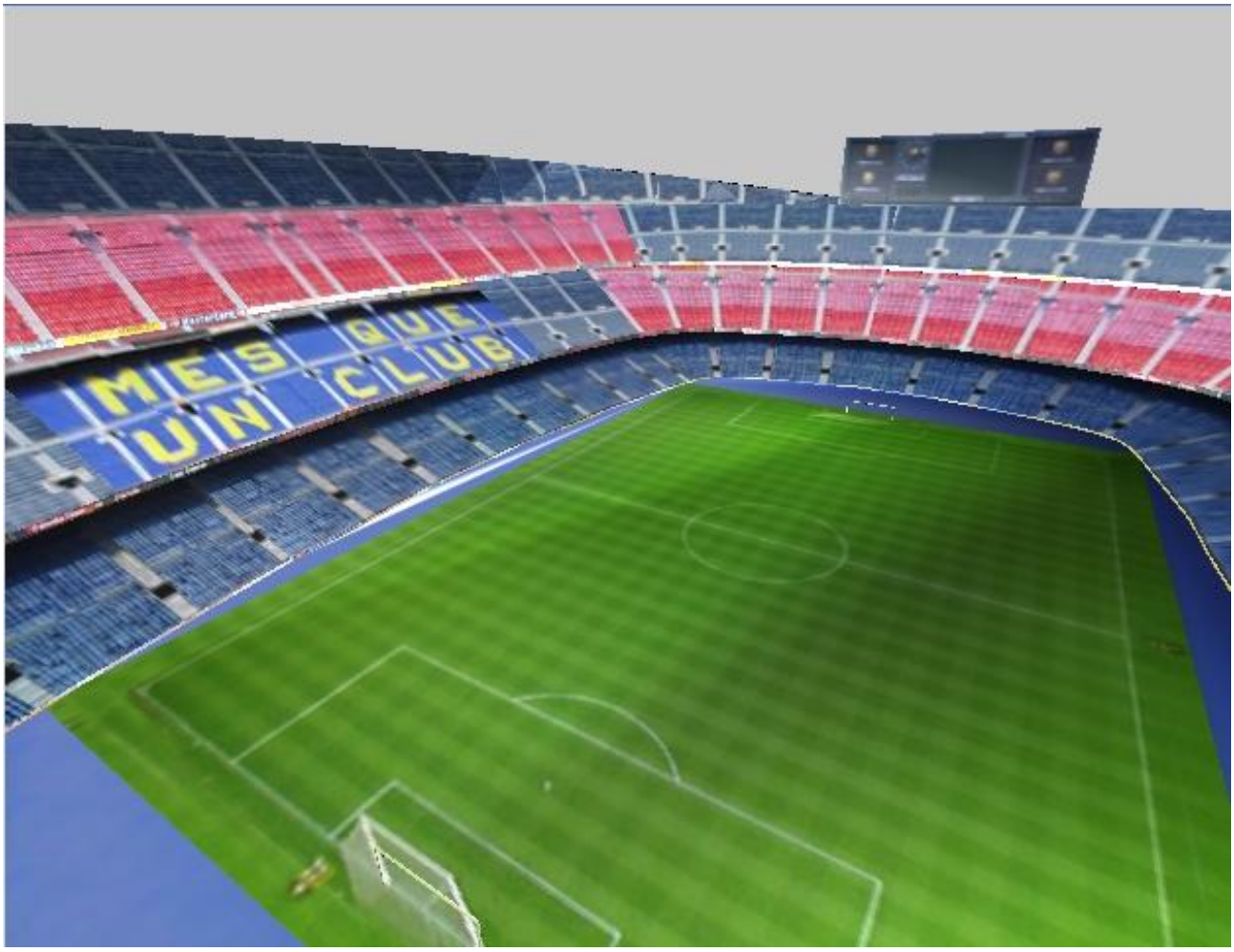


Figure 15: A Screenshot of Nou Camp Stadium from Our Game

3.5 Sound Properties

Sound properties listed below are going to be in the game

- Goal celebrations
- Tackle sound
- Whistle sound
- Ball hit sound
- Audience sound

4 Project Requirements

4.1 Functional Requirements

Functional requirements of a software project are the functions required to be implemented and working in the release version of the software.

- Selecting game modes.
- Viewing credits.
- Joining and creating match modes.
- Changing settings.
- User training.
- Multiplayer games on local area network

4.2 Operational and Structural Requirements

These are basic requirements of the game. These objects affect the reality of the game, so they give user intentness to play more. We examined these requirements here: graphic engine, sound, AI, physics and networking.

4.2.1 Graphic Engine

Since our project is a 3D football game, graphics is the main requirement for our project. In a football game; players, ball, stadium and other visual aspects require high quality 3D graphics. Our goal in graphics aspect of the project is to give players a realistic game playability. To maintain this goal, a graphics engine should be used. Here are some graphic engines we considered for our project:

4.2.1.1 Irrlicht Engine

The Irrlicht Engine is a cross-platform high performance real-time 3D engine written in C++. (http://irrlicht.sourceforge.net) It features a powerful high level API for creating complete 3D and 2D applications such as games or scientific visualizations. It comes with an excellent documentation and integrates all state of the art features for visual representation such as dynamic shadows, particle systems, character animation, indoor and outdoor technology and collision detection. All this is

accessible through a well designed C++ interface, which is extremely easy to use. It has also high performance real time 3D rendering using Direct3D and OpenGL and extensible material library with vertex, pixel, and geometry shader support platform. Lastly it runs on Windows, Linux, MacOS X, Solaris and other platforms independently.

4.2.1.2 Torque 3D

The Torque Game Engine (TGE) is a fully featured AAA game engine with award winning multiplayer network code, seamless indoor/outdoor rendering engines, state of the art skeletal animation, drag and drop GUI creation, a built in world editor, and a C-like scripting language. (http://www.torquepowered.com) Unlike most commercial game engines, as part of the low cost license, you receive all C++ source code to the engine, so you can make any additions you need for your game. The game features a terrain engine which automatically creates level of details of the ground, so that it renders the fewest polygons necessary at any given time. The terrain is automatically lit and textures applied to the terrain can be blended together seamlessly. The model supports loading of 3D models in the .DTS file format and the .DIF file format.

4.2.1.3 OGRE

OGRE (Object-Oriented Graphics Rendering Engine) is a scene-oriented, cross-platform, flexible 3D engine for hardware-accelerated 3D graphics software. It is licensed under LGPL. (http://www.ogre3d.org). It only provides graphic capabilities but it doesn't impose restrictions on other aspects of software so any library can easily be used with it. It also has a large and active community and extensive documentation which complements its design driven approach. The engine can take advantage of latest hardware for its advanced features. Simple API eases overall integration.

4.2.1.4 Id Tech 3

id Tech 3 is a game engine developed by id Software for Quake III Arena and has been used in many games under the "Quake III Arena engine" and "Quake III: Team Arena engine" branding. (http://www.idsoftware.com/business/idtech3/) During its time, it competed with the Unreal engine; both engines were widely licensed. Id Tech 3 loads 3DModels in the MD3 format. The format uses vertex movements (sometimes called pre-vertex animation) as opposed to skeletal animation in order to store animation. The animation features in the MD3 format are superior to those in id Tech 2's MD2 format because an animator is able to have a variable number of key frames per second instead of MD2's standard 10 key frames per second. This allows for more complex animations that are less "shaky" than the models found in Quake II. Another important feature about the MD3 format is that models are broken up into three different parts which are anchored to each other. Typically, this is used to separate the head, torso and legs so that each part can animate independently for the sake of animation blending (i.e. a running animation on the legs, and shooting animation on the torso). Each part of the model has its own set of textures.

We are going to use Irrlicht engine because of licensing issues. Moreover it is a flexible 3D engine written in C++, designed to make it easier and more intuitive for developers to produce applications utilizing hardware-accelerated 3D graphics. Our game is going to be open source, so that volunteer developers will be able to develop this game further for Linux community.

4.2.3 Sounds

Our game is going to have different sounds during game and waiting for the next session. The sounds are going to change between the steps. Audience, referee, announcer and environment sounds are going to be used during games. Sound options are going to be controlled by user via settings. We considered two sound libraries for

our project, which are OpenAL and FMOD.

4.2.3.1 The Open Audio Library, OpenAL

OpenAL is an environmental 3D audio library that supports just about every major platform. (http://connect.creativelabs.com/openal/default.aspx) It aims to provide an open replacement for proprietary (and generally incompatible) 3D audio systems such as EAX and A3D. OpenAL can add realism to a game by simulating attenuation (degradation of sound over distance), the Doppler Effect (change in frequency as a result of motion), and material densities. OpenAL has been used in several Linux game ports, including Heavy Gear II and Sid Meier's Alpha Centauri.

4.2.3.2 FMOD

The FMOD sound system is a revolutionary audio engine for game developers, multimedia developers, sound designers, musicians and audio engineers. (http://www.fmod.org) The development of FMOD is based on the years of experience of Firelight Technologies' previous product FMOD 3. FMOD is intended to push the creative boundaries of audio implementation for games and the like, whilst using minimal resources and being fully scalable. FMOD gets the mix right the first time - by mixing directly on the target hardware. It tweaks the mix at run-time by connecting designer to FMOD application over a network.

We are going to use FMOD as our sound library. FMOD provides us nearly all of our needs. The most important feature of FMOD sound library is that using minimal resources and being scalable. We can play some game music and environment sounds at the same time.

4.2.4 Networking

For our game, users from different computers can play same game via local area network and internet. We chose to use RakNet after considering these networking APIs.

4.2.4.1 UDP (User Datagram Protocol)

UDP (User Datagram Protocol) is a simple OSI transport layer protocol for client/server network applications based on Internet Protocol (IP). (http://en.wikipedia.org/wiki/User_Datagram_Protocol) UDP is the main alternative to TCP and one of the oldest network protocols in existence. UDP network traffic is organized in the form of datagrams. A datagram comprises one message unit. The first eight (8) bytes of a datagram contain header information and the remaining bytes contain message data. The UDP datagram *size* is a count of the total number of bytes contained in header and data sections. As the header length is a fixed size, this field effectively tracks the length of the variable-sized data portion (sometimes called payload).

4.2.4.2 RakNEt

RakNet is a networking API that is a wrapper for reliable UDP and higher level functionality on Windows, Linux, and UNIX. (http://www.jenkinssoftware.com) It allows any application to communicate with other applications on the same computer, over a LAN, or over the Internet. Although it could be used for any networked application, it was developed specifically for rapid development of online games and the addition of multiplayer to single player games. It is also free for non commercial products.

4.2.4.3 Zoidcom

The Zoidcom network library is a high-level, UDP based networking library providing features for automatic replication of game objects and synchronization of their states over a network connection in a highly bandwidth efficient manner. (http://www.zoidcom.com) This is achieved by multiplexing and demultiplexing object information from and into bit streams, which make it easily possible to avoid sending redundant data. Booleans only take one single bit, integers and floats are stripped down to as many bits as needed.

4.2.5 Physics

The function of physics engine is computing movement of the ball according to user input and physics laws. When a collision happens between ball and players or ball between poles, engine defines what is going on next. In our project there is going to be "street football" mode, and in this mode there are some calculations needed according to the ball's movement. We chose ODE as a physics engine.

4.2.5.1 ODE (Open Dynamics Engine)

ODE is an open source, high performance library for simulating rigid body dynamics. (http://www.ode.org) It is fully featured, stable, mature and platform independent with an easy to use C/C++ API. It has advanced joint types and integrated collision detection with friction. ODE is useful for simulating vehicles, objects in virtual reality environments and virtual creatures. It is currently used in many computer games, 3D authoring tools and simulation tools.

4.2.6 Artificial Intelligence

While playing against computer or in multiplayer mode; artificial intelligence is required for non-controlled players. Also the goalkeeper needs AI for ease of playability. The tactics, passing, pressing or shooting of opponent team requires AI when someone plays against computer.

4.3 Non-functional Requirements

Non-functional requirements support functional requirements about performance requirements, security, quality standards or design constraints.

4.3.1 Usability and Playability

Since games are for fun, it must be easy to learn and use so as not to be boring for the user. So we thought that, menus must be designed as understandable as possible. Also we planned to follow the traditional approaches in many designing menus, because players will compare our game to other football games that he/she has played earlier. So in design, we are inspired by some other games on market. We

are going to add new sections for the game, like “Japanese goal”, “German goal”.

Playability is also essential point. User must not get bored when playing a football game. The game will give users a good feeling while playing it. Also, the game will not be too hard or too easy.

4.3.2 Reliability and Security

Debugging of game is going to be done very carefully. In order not to cause any problem to users, the product will be checked in many aspects. Since our game is also a multiplayer game, security is important for us. Used protocols will be integrated to system very carefully. During and after plays there will not be any open backdoor left on computers of players.

4.3.3 Portability

This project is not going to be a platform independent game. There is not much entertaining football games on Linux operating systems, so our aim to make a funny and entertaining soccer game for Linux.

4.4 Software Requirements

A distribution of Linux operating system and related libraries are going to be required since the game is expected to run on Linux environment.

4.5 Hardware Requirements

The minimum hardware requirements for our game are listed as:

- ✓ Computers with Intel Pentium Core 2 Duo or AMD Athlon 64 X2 dual core CPU.
- ✓ A graphics card with Nvidia GeForce 6 series or ATI Radeon HD series GPU.
- ✓ Onboard sound card.
- ✓ 1 GB of RAM.
- ✓ 500 MB of available disk space.
- ✓ An onboard 10/100 Mbit/s ethernet card for network connection

5 Functional Modeling

5.1 Level 0 of Data Flow Diagram

Level 0 of data flow diagram shows the interactions between the user and game system. User initiates control inputs and game system responds with visual and audio outputs.

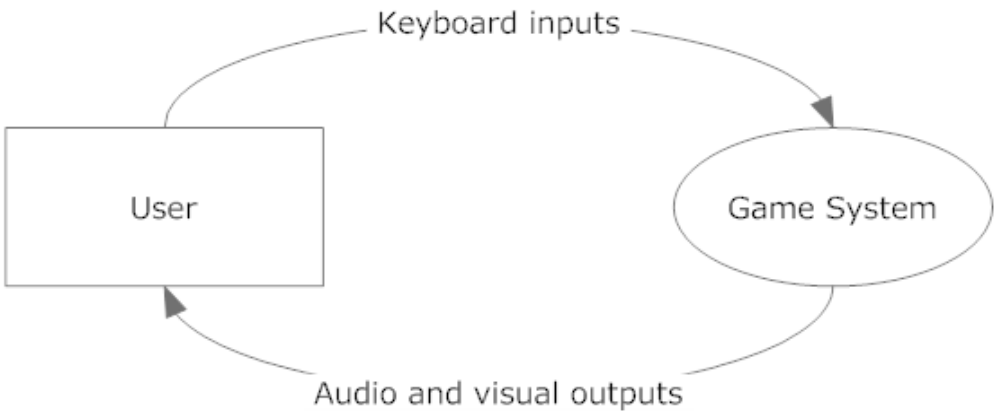


Figure 16: level 0 DFD

5.2 Level 1 of Data Flow Diagram

Level 1 of data flow diagram shows the details of the game system, describing the relations and interactions between its main components such as graphics engine, sound engine, physics engine etc.

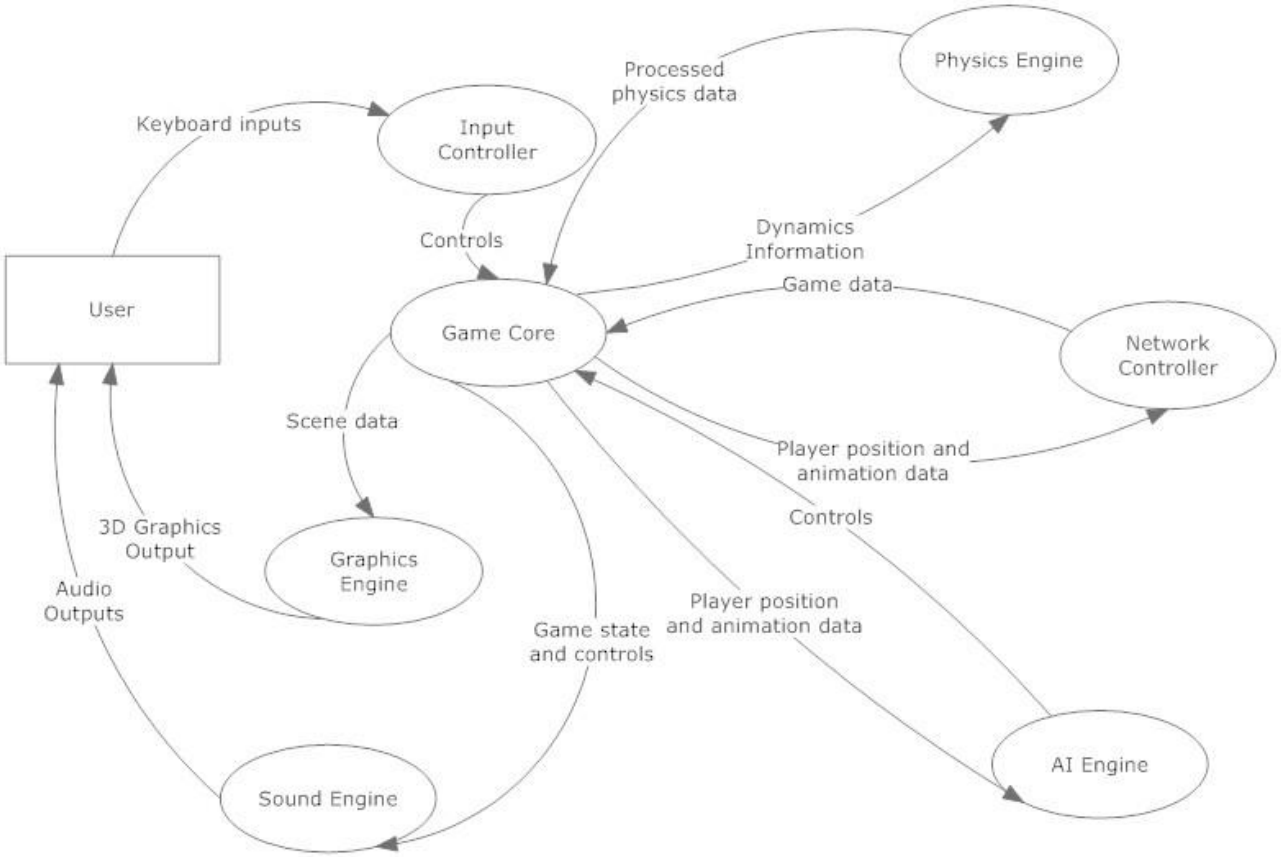


Figure 17: Level 1 DFD

5.3 Level 2 of Data Flow Diagram

Level 2 of data flow diagram shows the details of the game core, describing its components. Also interactions of these components among each other and with other main components of the game system such as graphics engine, sound engine, network engine etc. are also illustrated in this diagram.

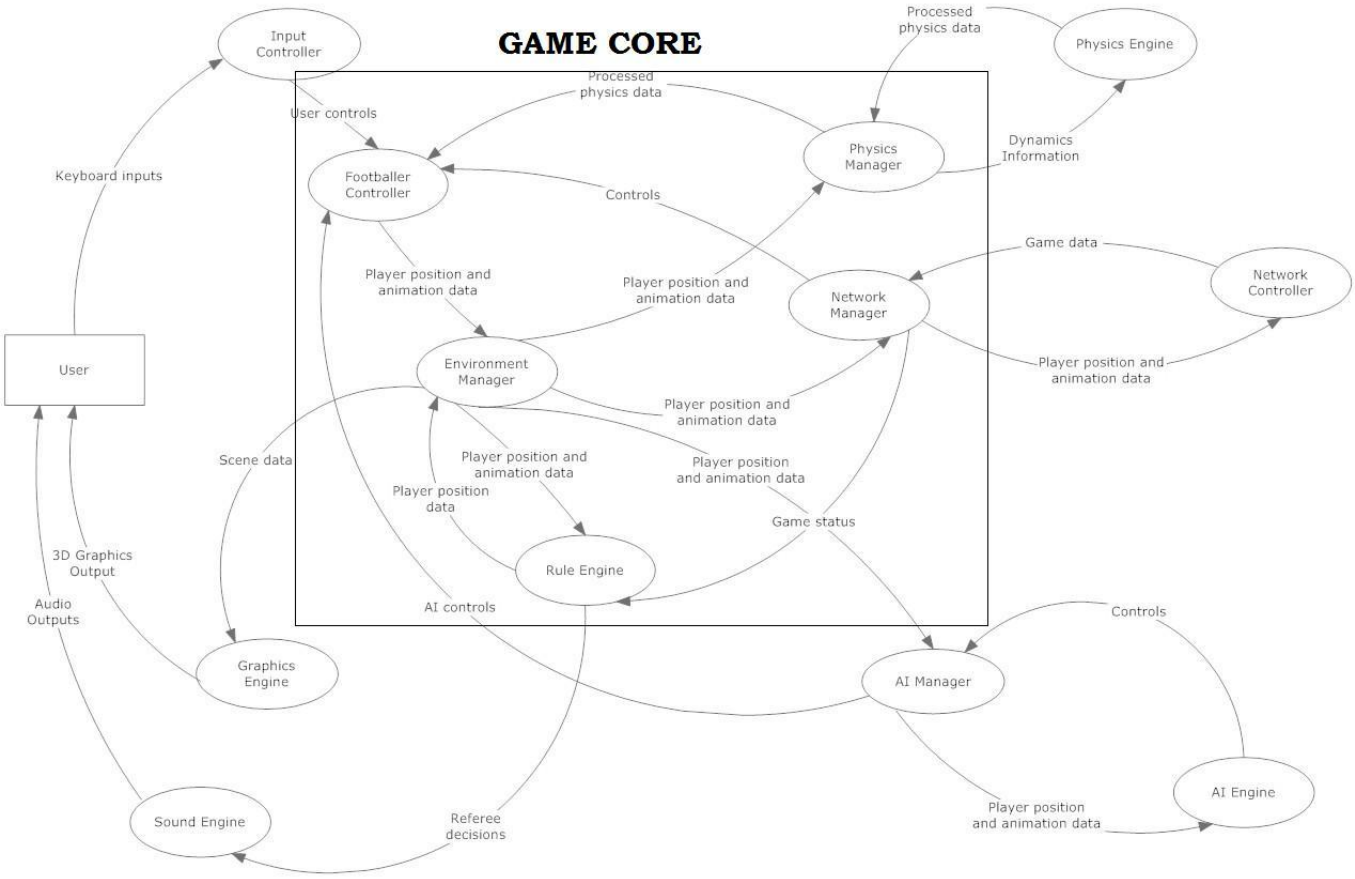


Figure 18: Level 2 DFD

5.4 Entity Relationship Diagram

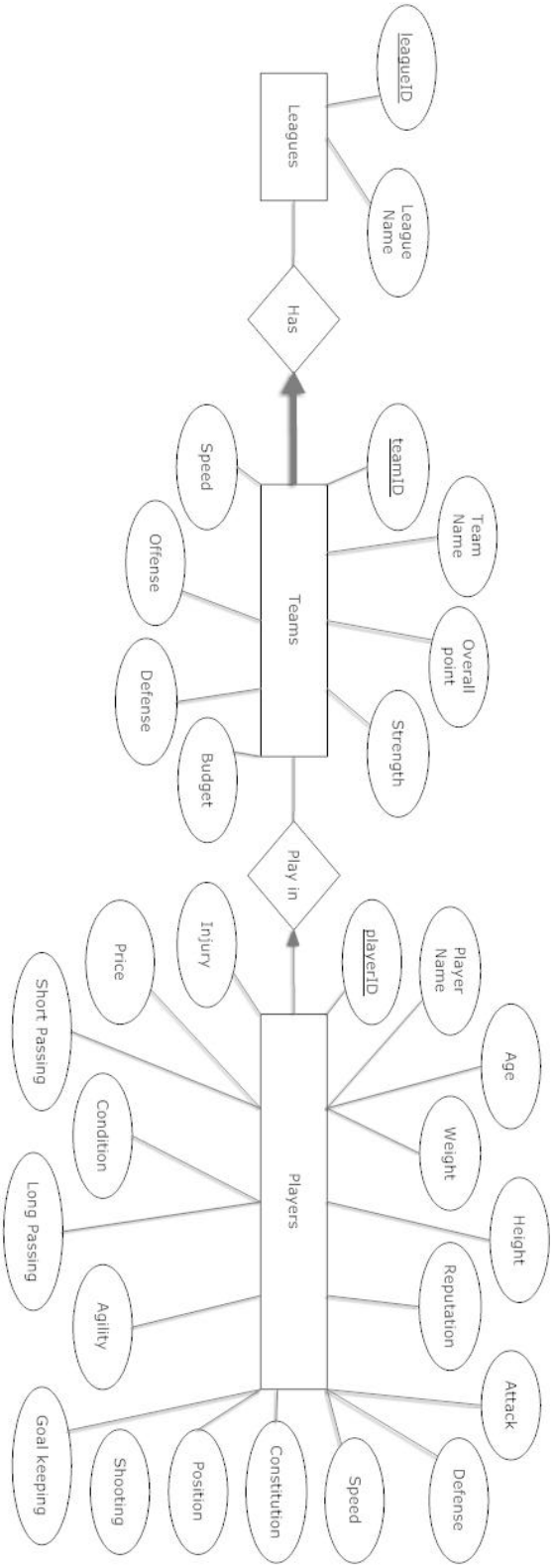


Figure 19: ER Diagram

5.4 Sequence Diagram

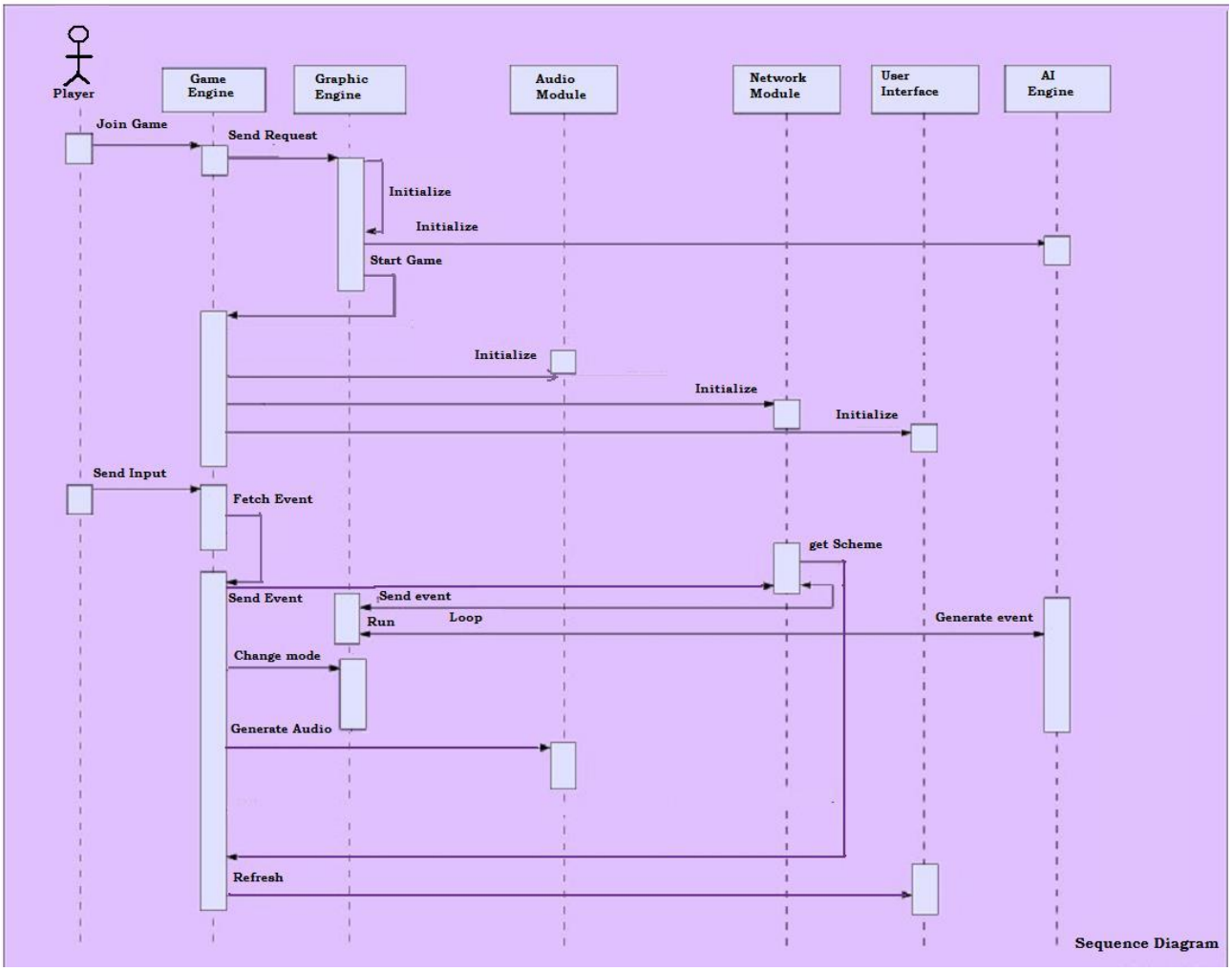


Figure 20: Sequence Diagram

6 Object Oriented Modeling

6.1 Class Definitions

6.1.1 Footballer Class

Footballer class attributes and methods are described below.

- **Attributes:**

status: It is an integer indicating footballer's current status which can be "Run", "Sprint", and "Stop"

direction: It is an integer indicating the direction of footballer's movement on the field.

age: It is an integer, representing footballer's age.

condition: It is an integer representing footballer's condition.

injury: It is an integer representing whether footballer is injured or not.

shooting: It is an integer representing footballer's shooting capability.

current_team: It is a string representing footballer's current team.

height: It is an integer representing footballer's height.

constitution: It is an integer representing footballer's constitution.

agility: It is an integer representing footballer's agility.

mentality: It is an integer representing footballer's mentality.

reputation: It is an integer representing footballer's reputation.

 Methods:

run(): This method takes an input from user and makes footballer to run along its direction.

sprint(): This method takes an input from user and makes footballer to sprint along its direction.

shoot(): This method takes an input from user and makes footballer to shoot the ball.

passfriend(): This method takes an input from user and makes footballer to pass the ball to its team friend.

tackle(): This method takes an input from user and makes footballer to tackle another footballer which has ball possession.

setdirection(): This method takes an input from user and assigns the direction of the footballer according to that input.

6.1.2 Environment Class

Environment class attributes and methods are described below.

 Attributes:

gamemode: It is a string that contains game mode information.

weather: It is a string describing the weather condition.

field: It is a string that contains football field information, which can be a stadium, a street or a saloon.

- **Methods:**

setweather(): This method applies the weather condition, which is set by user, to in game screen.

setgamemode(): This method applies the game mode, which is set by user, to in game screen.

setfield(): This method applies the field information, which is set by user, to in game screen.

renderpeople(): This method places people to their initial positions.

6.1.3 Network Class

Network class attributes and methods are described below.

- **Attributes:**

gamemode: It is a string, that contains game mode information.

game_id: It is an integer assigned uniquely to each created game.

- **Methods:**

listenports(): This method listens the defined ports for created game.

hostgame(): This method opens a new game and sends game_id to the network.

joingame(): This method provides connection to a created game.

startgame(): This method sends game start signal to client(s).

sendgamedata(): This method sends the game data from host to client(s) or from client(s) to host.

recievegamedata(): This method receives the game data from host to client(s) or from client(s) to host.

6.1.4 Input Class

Input class attributes and methods are described as below.

- **Attributes:**

  keyboard_input : It is a value depending on which keyboard button is pressed.

- **Methods**

  GetMouseInput() : This method gets inputs from mouse.

  GetKeyboardInput(): This method gets inputs from keyboard.

  SendMouseInput() : This method sends mouse input to appropriate classes

  SendKeyboardInput(): This method sends keyboard input to appropriate classes

6.1.5 Audio Class

Audio class attributes and methods are described as below.

- **Attributes:**

  audio_id : It is a unique integer assigned to each audio data.

  volume : It is a float value used for audio volume level.

- **Methods:**

  LoadAudioFile() : This function loads the audio file which's id is audio_id.

  PlayAudioFile() : This function plays the audio file described with audio_id attribute.

  StopAudioFile() : This function stops the audio file described with audio_id attribute.

  SetVolume() : This function sets the volume attribute.

  LoadVolume() : This function loads the volume attribute.

6.1.6 Referee Class

Referee class attributes and methods are described below.

- **Attributes:**

  offside: It is a Boolean value, indicating that movement of attacking team is offside or not.

  goalkick: It is a Boolean value, indicating position is a goal kick or not.

  throwin: It is a Boolean value, indicating position is a throw in or not.

cornerkick: It is a Boolean value, indicating position is a corner kick or not.

subject: It is a string value, which is passed as a parameter showyellowcard() and showredcard() methods.

- **Methods:**

showyellowcard(subject): This method decides "subject " is punished with a yellow card by referee

showredcard(subject): This method decides "subject " is punished with a red card by referee

decidefreekick(): This method decides that a movement is free kick.

decidethrowin(): This method decides that a movement is throw in.

decidegoalkick(): This method decides that a movement is goal kick.

decidecornerkick(): This method decides that a movement is corner kick.

endgame(): This method ends the match.

startgame(): This method starts the match.

6.1.6 Ball Class

Ball class attributes and methods are described below.

- **Attributes:**

x_position: It is a float, defining x position of ball.

y_position: It is a float, defining y position of ball.

z_position: It is a float, defining z position of ball.

velocity: It is a float, defining speed position of ball.

angle: It is a float, defining the angle between initial velocity and the xz plane.

- **Methods:**

getpositionx(): returns the x position of ball.

getpositiony(): returns the y position of ball.

getpositionz(): returns the z position o ball.

ballvelocity(): returns the velocity of ball.

6.2 Class Diagrams

The UML class diagram of our initial design is given below.

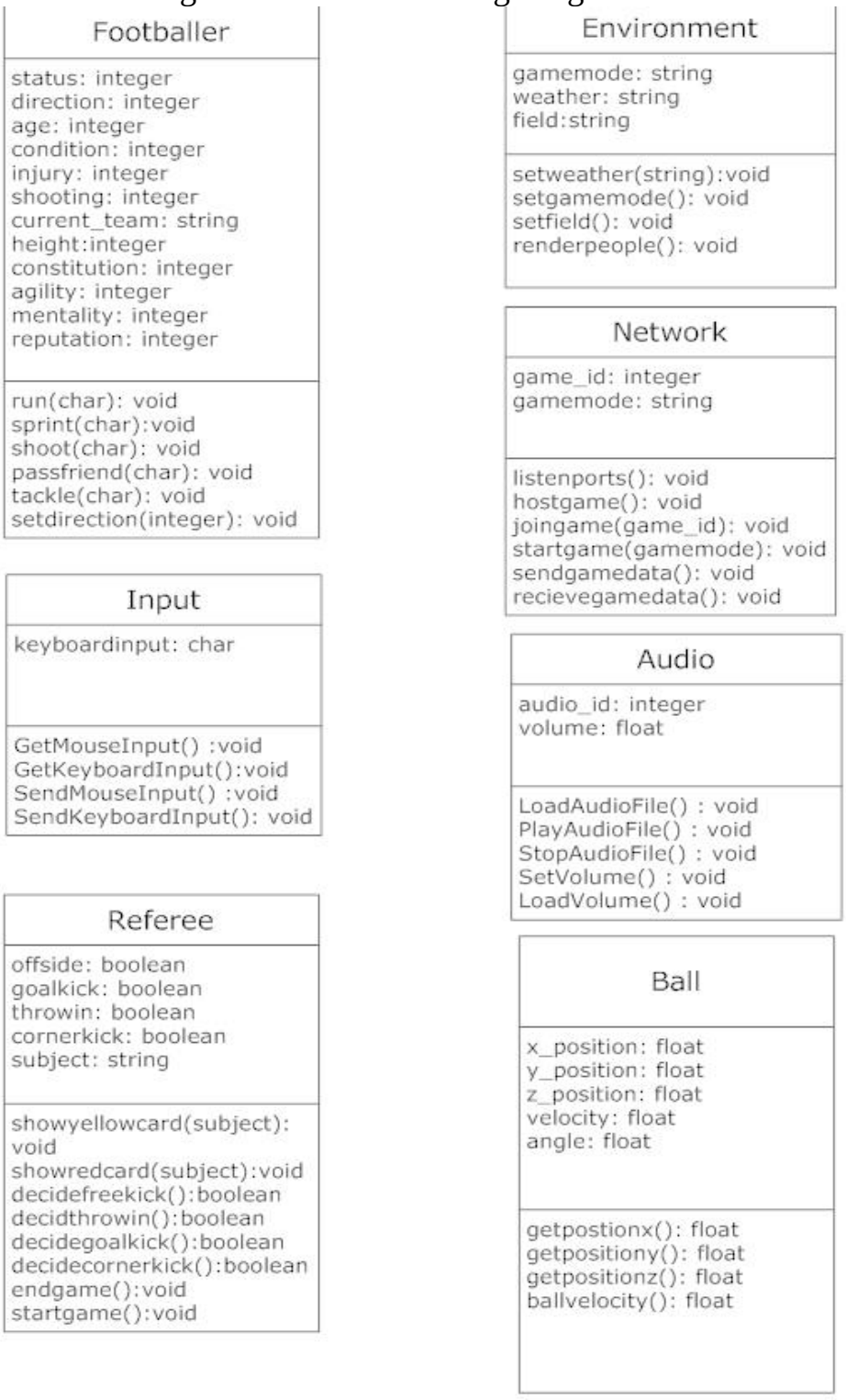


Figure 21: Class Diagrams

7 User Interface Design

7.1 Start Menu

Figure 22: Start Menu

Friendly Game: Opens "Friendly Game" menu (will be mentioned in detail in 7.1.1) where you can start a friendly game against computer.

Start League: Opens "Start League" menu (will be mentioned in detail in 7.1.2) where you can start league.

Continue League: Opens "Continue League" menu (will be mentioned in detail in 7.1.3) where you can continue a previously created league.

Multiplayer Game: Opens "Multiplayer Game" menu (will be mentioned in detail in 7.1.4) where you can open a LAN Game or join a LAN game.

Fun Modes: Opens "Fun Modes" menu (will be mentioned in detail in 7.1.5) where you can start a fun mode game.

Training Modes: Opens "Training Modes" menu (will be mentioned in detail in 7.1.6) where you can enter training modes.

Settings: Opens "Settings" menu (will be mentioned in detail in 7.1.7) where you can change game settings.

Edit Menu: Opens "Edit Mode" menu (will be mentioned in detail in 7.1.8) where you can edit or create players.

Exit: Closes the program and returns to Linux.

7.1.1 Friendly Game

Figure 23: Friendly Game Mode

In this menu, player firstly chooses a league and a team in it for herself. By doing this strength, speed, offense, defense and overall ratings of the team will be shown. Secondly, player chooses a league and a team in it for computer. By doing this strength, speed, offense, defense and overall ratings of the team will be shown.

Start Game: Starts the game if teams are chosen for both player and computer.

Back: Returns to Main Menu

7.1.2 Start League

Figure 24: Start League Mode

In this menu player chooses a league and a team to play with it in whole season. After selecting the team strength, speed, offense, defense and overall points are shown. If the stats of the team satisfy the player, he/she can start the league.

Start League: Starts selected league with selected team.

Back: Returns to Main Menu.

7.1.3 Continue League:

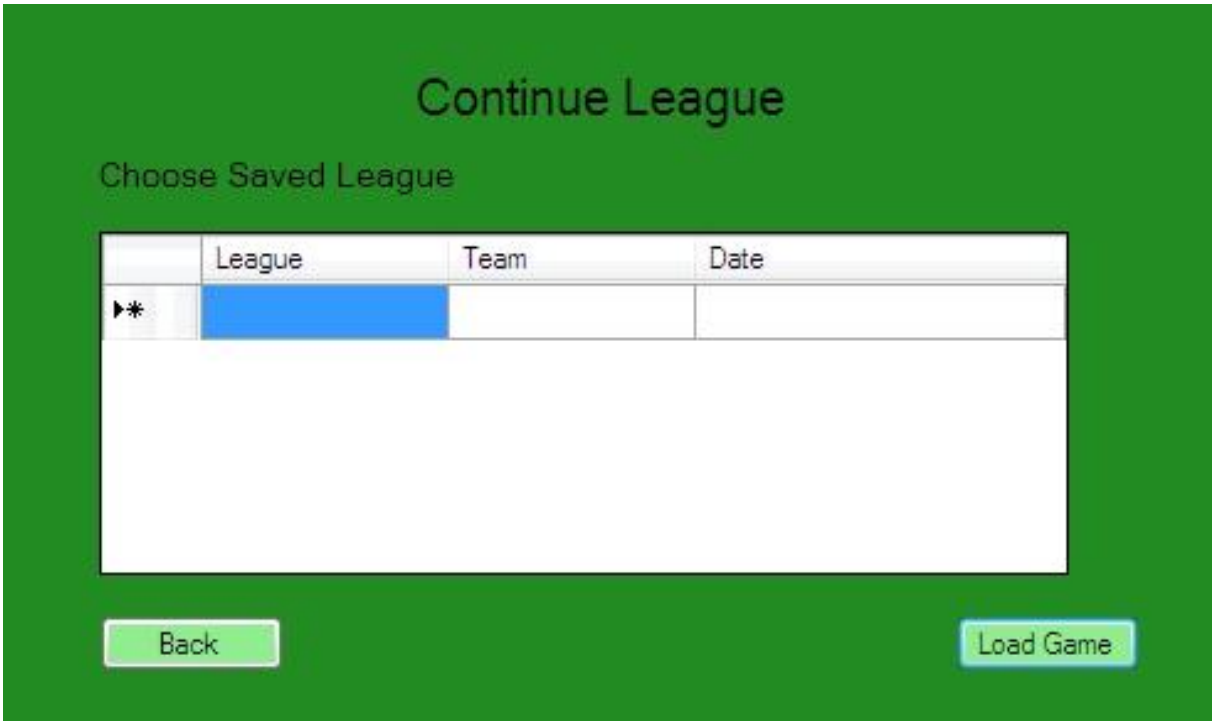


Figure 25: Continue League Mode

In this menu; player chooses a previously saved league game record that contains league, team and save date info.

Load Game: Loads the selected league game record.

Back: Returns to Main Menu.

7.1.4 Multiplayer Game

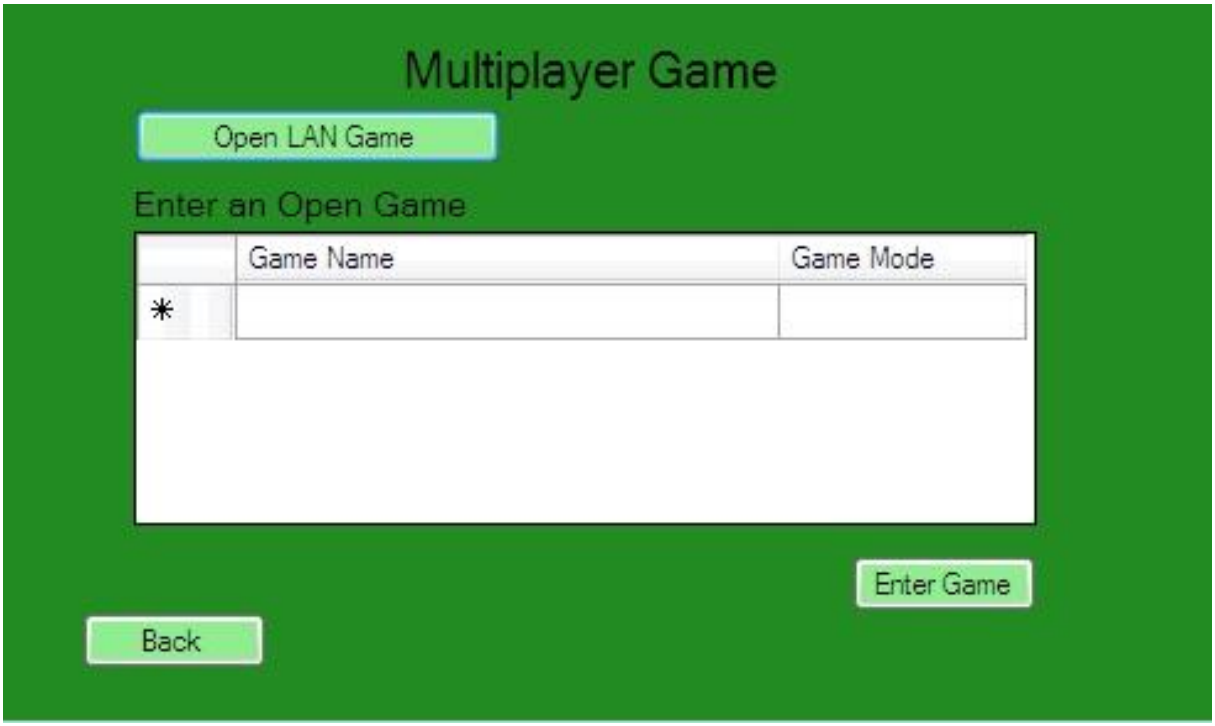


Figure 26: Multiplayer Game

In this menu, player has two options: Creating a game or joining a previously opened game selected from table. Table includes Game Name and Game Mode information which is either normal (traditional) mode or nPvP (Player vs. Player) mode.

Enter Game: Joins the selected game.

Back: Returns to Main Menu.

7.1.4.1 Open LAN Game

Figure 27: Open LAN Game

In this menu player enters a game name (that is not open at that time) and selects one of the multiplayer game modes which are either Normal Mode or nPvP.

Open Game: Starts a multiplayer game with selected mode.

Back: Returns to Multiplayer Game Menu.

7.1.5 Fun Modes:

Figure 28: Fun Modes

In this menu player chooses from Fun Modes.

Japanese Goal Match: Starts a Japanese Goal Match game.

Street Football: Starts a Street Football game.

German Goal Match: Starts a German Goal Match game.

nPvP: Because this mode can only be played as multiplayer, clicking this will redirect user to Multiplayer Game Menu.

7.1.6 Training Modes

Figure 29: Training Menu Mode

In this menu player chooses from Training Modes.

Shooting: Starts a shooting training.

Free Kick: Starts a free kick training.

Penalty: Starts a penalty shootout training.

Corner Kick: Starts a corner kick training.

7.1.7 Settings

Figure 30: Settings

In this menu, player can access to different settings.

General Settings: Opens General Settings Menu (which will be mentioned in 7.1.7.1)

Display Settings: Opens Display Settings Menu (which will be mentioned in 7.1.7.2)

Sound Settings: Opens Sound Settings Menu (which will be mentioned in 7.1.7.3)

Controller Settings: Opens Controller Settings Menu (which will be mentioned in 7.1.7.4)

Back: Returns to Main Menu.

7.1.7.1 General Settings

Figure 31: General Settings

In this menu player can change referee setting, match length or either offside is enabled or not.

Save: Saves selected general settings.

Back: Returns to Settings Menu.

7.1.7.2 Display Settings

Figure 32: Display Settings

In this menu player can change resolution, lights and texture quality.

Save: Saves selected display settings.

Back: Returns to Settings Menu.

7.1.7.3 Sound Settings

Figure 33: Sound Settings

In this menu player can change music volume, sound volume, audience sound or mute all the sounds.

Save: Saves selected sound settings.

Back: Returns to Settings Menu.

7.1.7.4 Controller Settings

Figure 34: Controller Settings

In this menu player can change user controller keys of shoot, pass, sprint, tackle and heading actions.

Save: Saves selected controller settings.

Back: Returns to Settings Menu.

7.1.8 Edit Menu

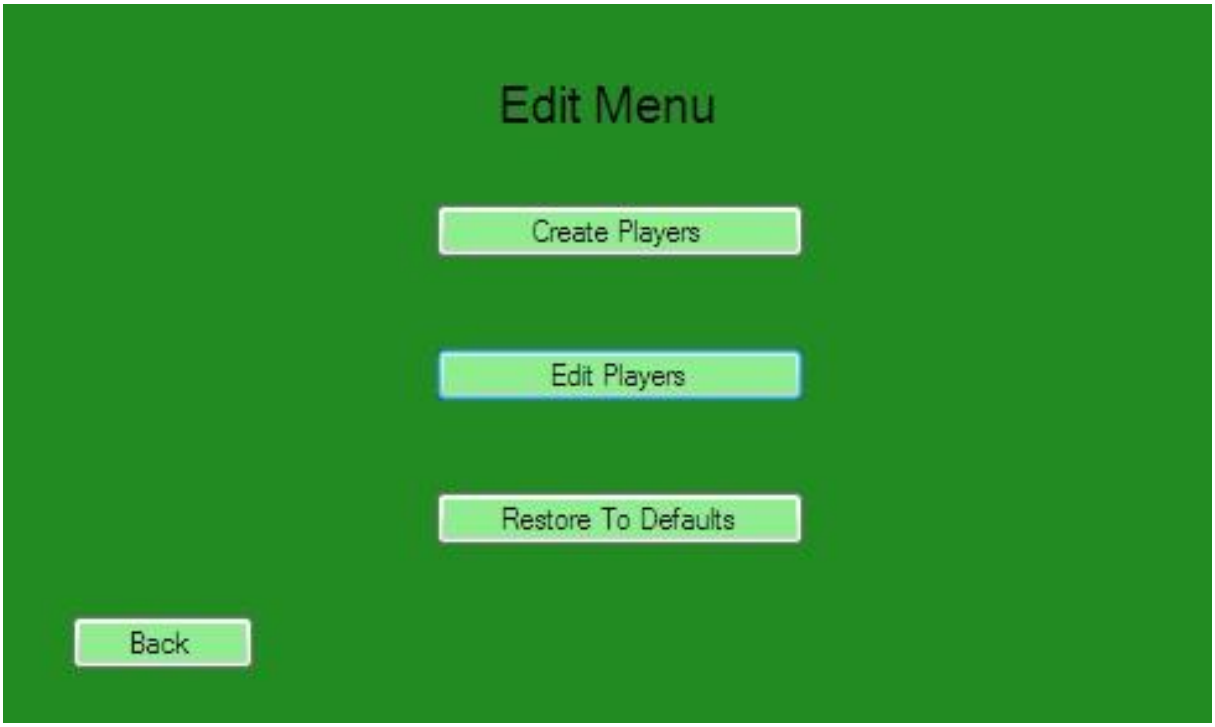


Figure 35: Edit Menu

In this menu user can edit current players, create new players or restore the player database to defaults.

Create Players: Opens Create Player Menu (which will be mentioned in 7.1.8.1)

Edit Players: Opens Choose Player Menu (which will be mentioned in 7.1.8.2)

Restore To Defaults: Restores the player databse to defaults

Back: Returns to Main Menu

7.1.8.1 Create Player Menu

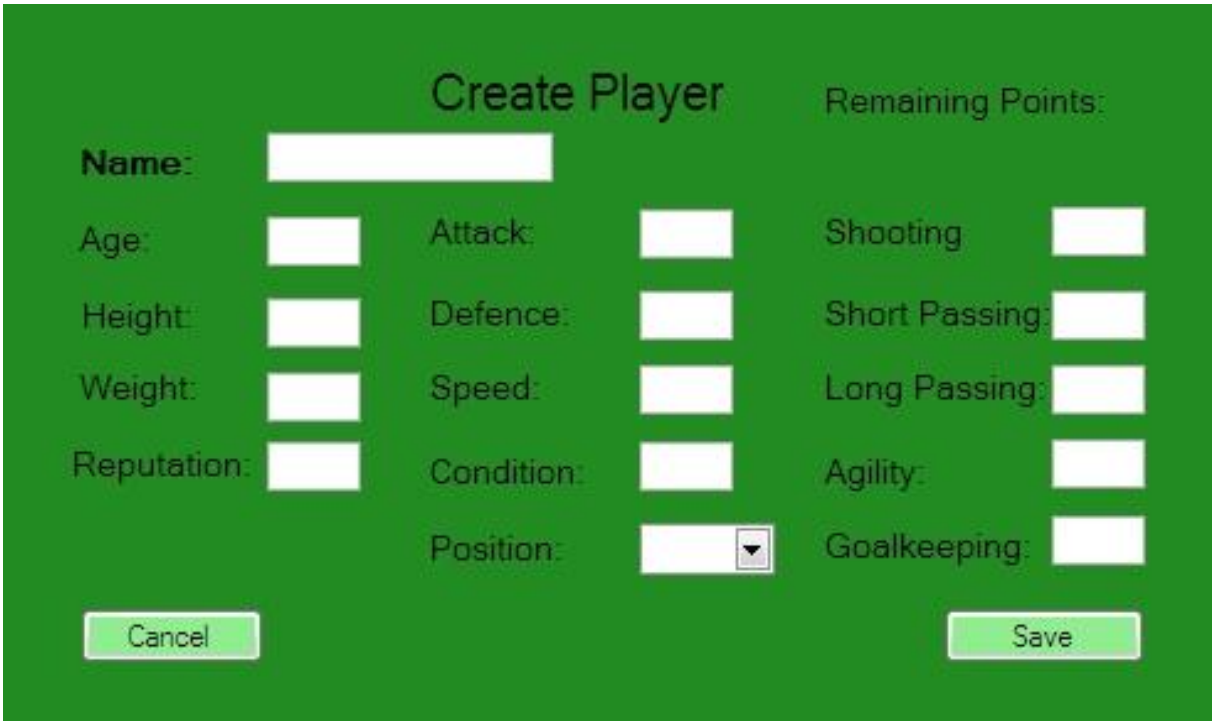


Figure 36: Create Player Menu

In this menu user can create a new player with given attributes. There will be total attribute points that can be distrubeted among attributes and remaining points are going to seen at upper right corner.

Save: Opens a confirmation window to save player with given data.

Cancel: Disposes all entered data and returns to Edit Menu

7.1.8.2 Choose Player Menu

Figure 37: Choose Player Menu

In this menu user selects a league and a team in it to load all players in selected team to the table. Then by choosing a player in the table, user can proceed to editing.

Edit: Opens Edit Player Menu (which will be mentioned in 7.1.8.2.1)

Back: Returns to Main Menu.

7.1.8.2.1 Edit Player Menu

Figure 38: Edit Player Menu

In this menu user can edit a player's attributes. There will be total attribute points that can be distrubeted among attributes and remaining points are going to seen at upper right corner.

Save: Opens a confirmation window to save player with given data.

Cancel: Disposes all entered data and returns to Edit Menu

7.2 League Menu

Figure 39 : League Menu

In this menu player can access fixtures of the league, change tactics & players of his/her own team or transfer players from or to her team. Moreover, he/she can see next match to be played; current tactics and players of the team and current ranking of his/her own team in league table.

Fixtures: Opens Fixtures Menu (which will be mentioned in 7.2.1)

Tactics – Players: Opens Tactics & Players Menu (which will be mentioned in 7.2.2)

Transfer: Opens Transfer Menu (which will be mentioned in 7.2.3)

Start Match: Starts the next match.

7.2.1 Fixtures

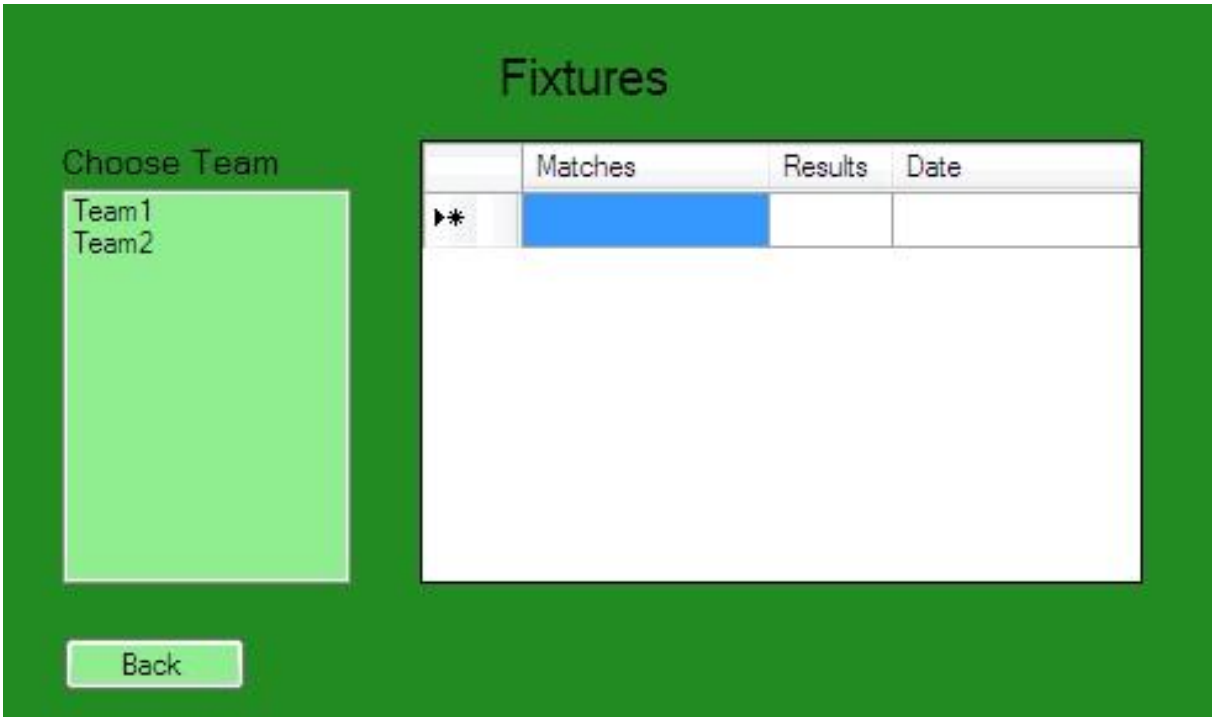


Figure 40: Fixtures

In this menu player chooses a team from left table and sees all the matches that team has played and will play this league with their dates. If match has already been played results are shown, too.

Back: Returns to League Menu.

7.2.2 Tactics & Players

Figure 41: Tactics & Players

In this menu player chooses with which tactic he/she will be playing, if this tactic will be offensive or defensive. Moreover he/she can change team players with substitutes or reserves.

Next: Saves the current tactics and players and returns to League Menu.

Back: Returns to League Menu without saving.

7.2.3 Transfer

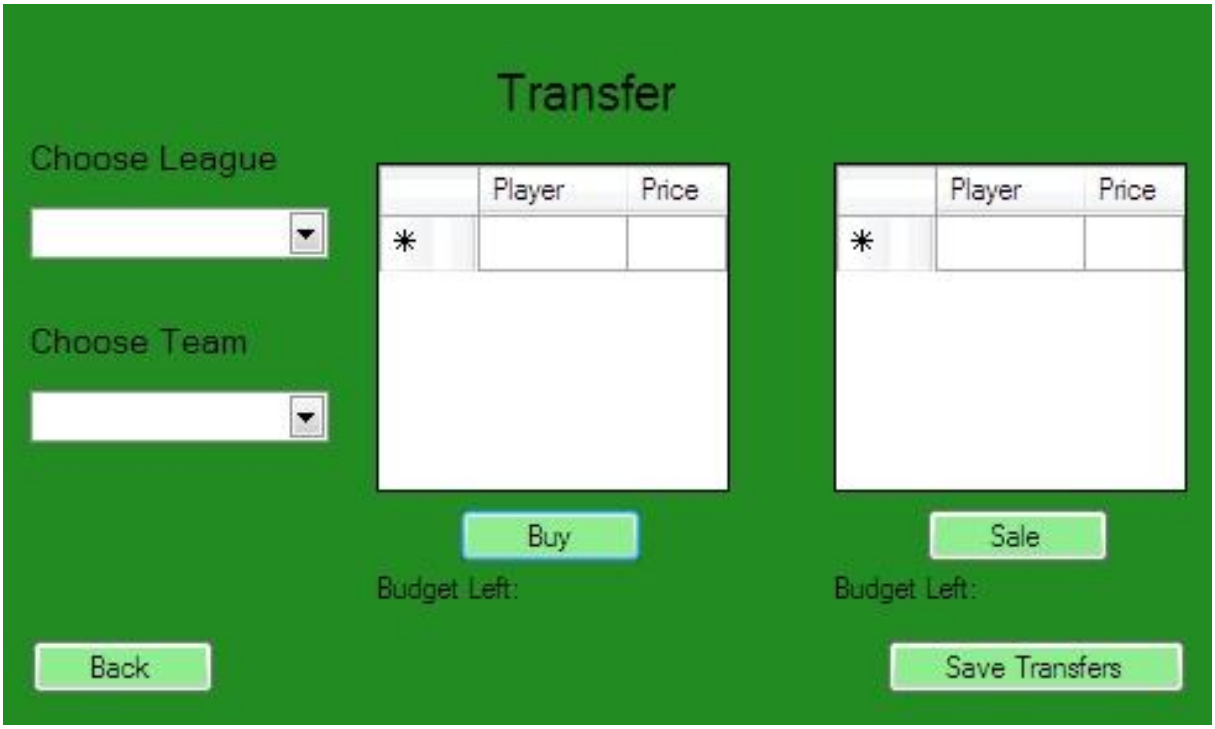


Figure 42: Transfer

In this menu, player chooses the league and a team which he/she wants to make player transfers. After selecting a team, all players in that team will be shown in left table and all of his/her own team's players will be shown in right table. Since transfers will be only budget based, left budgets of the teams will be shown too.

Buy: Transfers the selected player from selected team to own team if own budget is enough.

Sale: Transfers the selected player from own team to selected team if its budget is enough.

Save Transfers: Saves the applied transfers and returns to League Menu.

Back: Returns to League Menu without saving applied transfers.

7.3. In Game Menu

Figure 43: In Game Menu

This menu is shown when match is paused during game play.

Resume: Returns to the match.

Tactics – Substitutions: Opens Tactics & Substitutions Menu which mentioned in 2.2. In this menu only difference will be that reserve players will not be listed.

Exit: Cancels current match and returns to League Menu.

Exit to Linux: Cancels current match, closes the game and returns to Linux.

8 Usage Scenarios

8.1 Start Menu Usage

Start menu usage scenario is illustrated with this use case diagram.

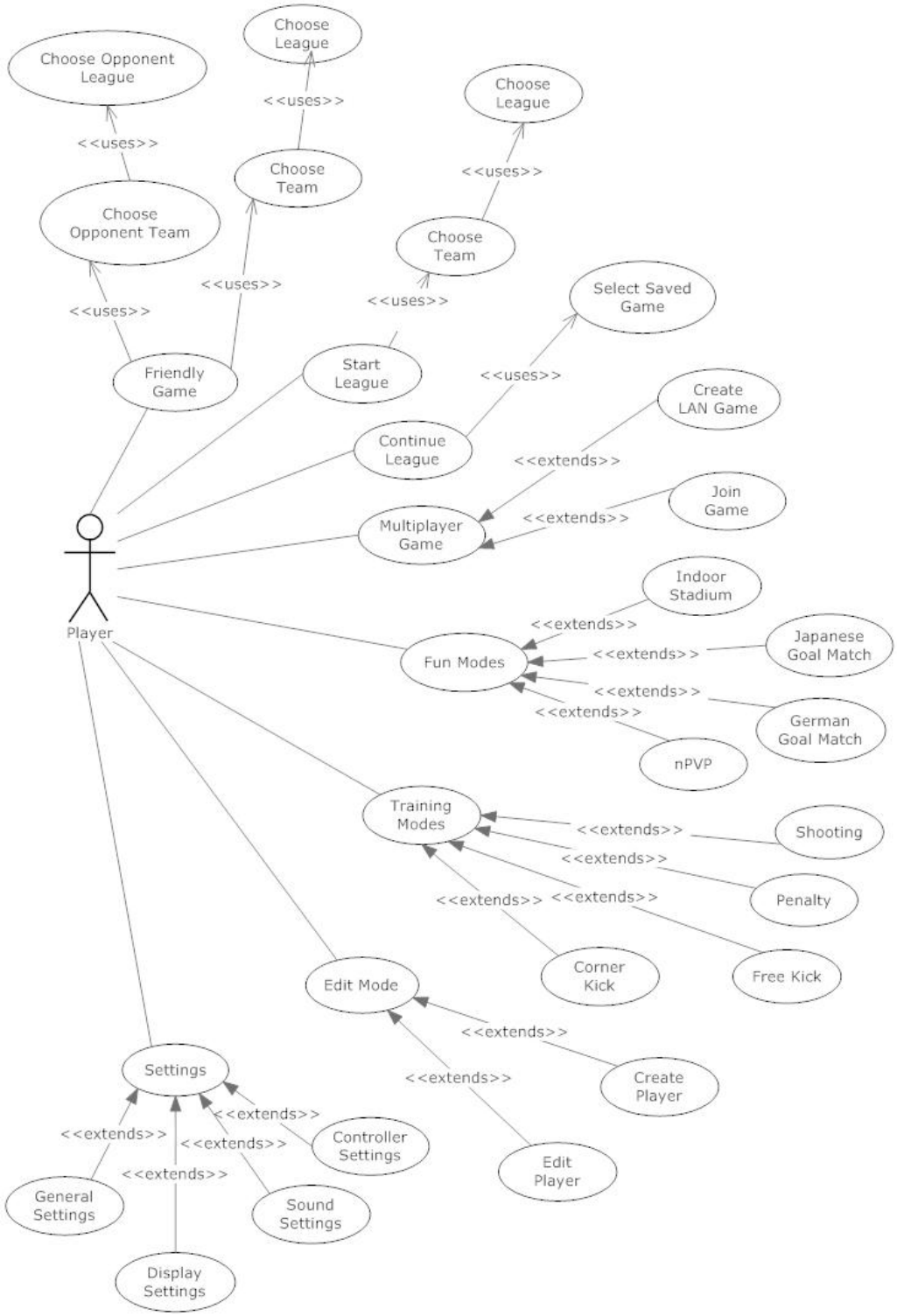


Figure 44: Start Menu Usage Use Case Diagram

8.2 League Menu Usage

League menu usage scenario is illustrated with this use case diagram.

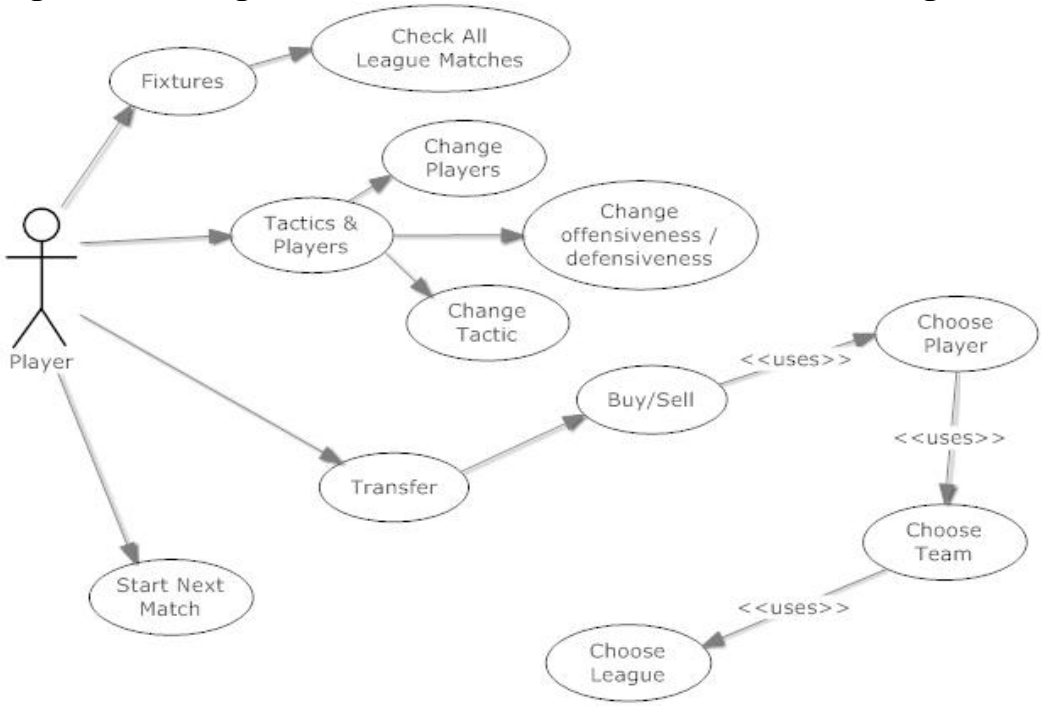


Figure 45: Use Case Diagram of League Menu

8.3 In Game Menu Usage

In game menu usage scenario is illustrated with this use case diagram.

Figure 46: Use Case Diagram of In Game Menu

9 Project Modules

9.1 Graphical User Interface Module

Graphical user interface is going to perform interactions between human players and game engine. Irrlicht game engine is going to be used to implement graphical user interface module.

As an example about graphical user interface of 3D football games, some screenshots from graphical user interface of Pro Evolution Soccer 2010 game are shown below.

Figure 47: A Screenshot from PES 2010

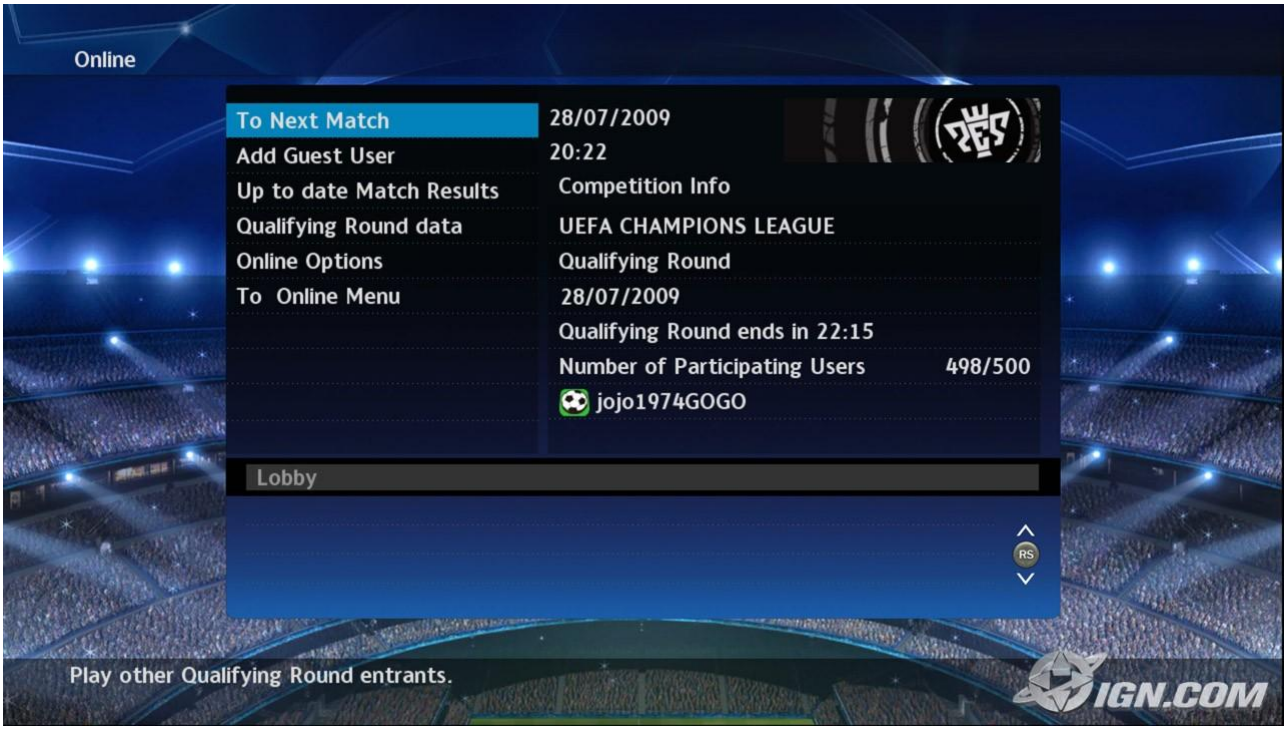


Figure 48: A Screenshot from PES 2010

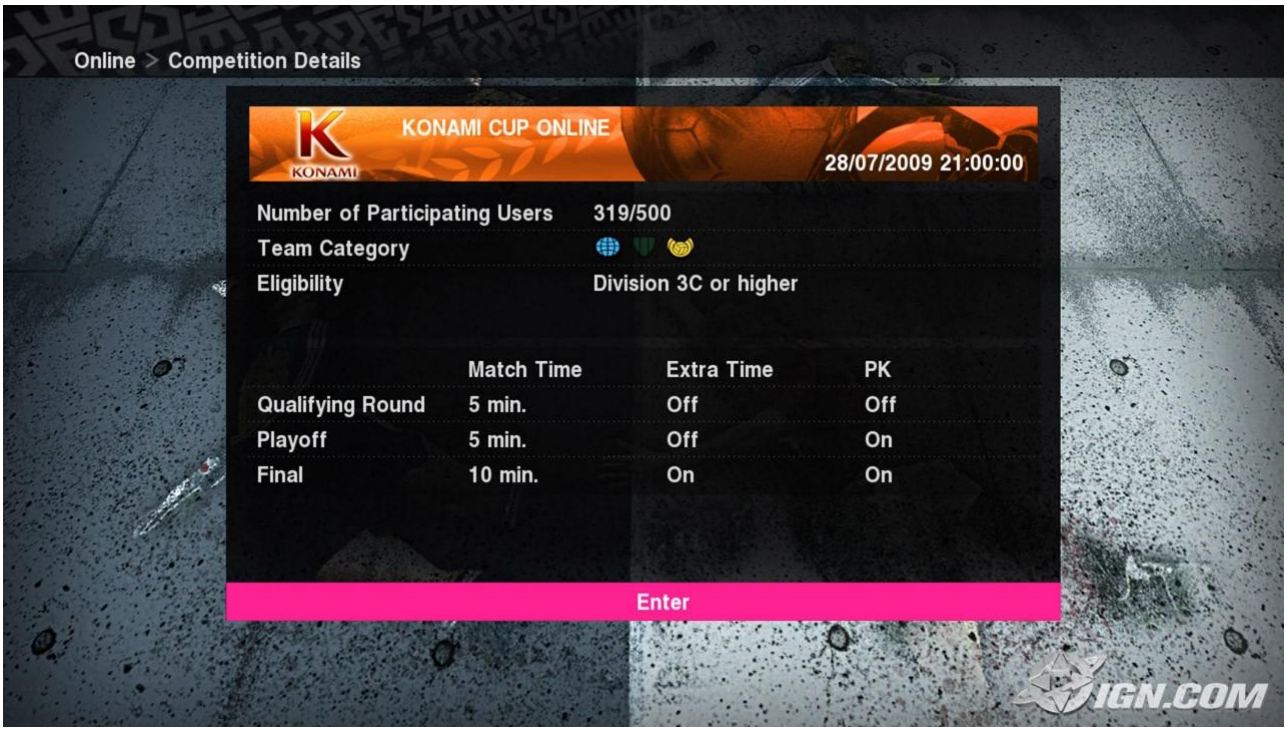


Figure 49: Another Screenshot from PES 2010

9.2 Game Core Module

Since Irrlicht is going to be used in this project, game engine is not going to be implemented from scratch. However certain properties of a 3D football game should be implemented manually, since Irrlicht is not developed specially for 3D football games. Fundamental components of the game core are described in level 2 data flow diagram. Game core is going to be interacting with Irrlicht for 3D graphics, FMOD for sound effects and RakNet for networking.

9.3 Input Module

Mouse and keyboard inputs are going to be used in both main and in-game menus. But only keyboard controller is going to be used in play screen. Our input module is going to be implemented for input data regarding these restrictions.

9.4 Menu Module

A menu module is going to be implemented for all user interface screens described in “Interface Design” chapter of this document.

9.5 Artificial Intelligence Engine Module

Artificial intelligence requirement analysis of the game is introduced in requirement analysis report and is described in “Project Requirements” chapter of this document. Since a good AI is not a primary objective of our project, initially we are going to implement a primitive AI which is going to be enough for simulating a football match. After satisfying our goals, we are going to work on implementing more realistic AI methods for further improvement of the project.

9.6 Graphic Engine Module

As a game engine Irrlicht also includes graphical modeling and rendering, a separate graphics engine module is not going to be implemented.

9.7 Network Module

Since implementation of network module is not restricted with using only low

level libraries such as “socket.h”; Raknet networking engine is going to be used for implementing the network module.

9.8 Audio Module

Audio module is going to be implemented for adding sound property to the game. It is going to be divided into two parts.

Music: In main menu screen, game soundtracks are going to be played in the background. Users are going to be able to adjust its volume level in “Audio Options” screen.

Sound Effects: Several sound effects are going to be used in game actions such as kicking, running, audience cheer etc. Users are going to be able to adjust its volume level in “Audio Options” screen.

Since FMOD is going to be used as a sound library, the sound module is not going to be implemented from scratch.

9.9 Physics Module

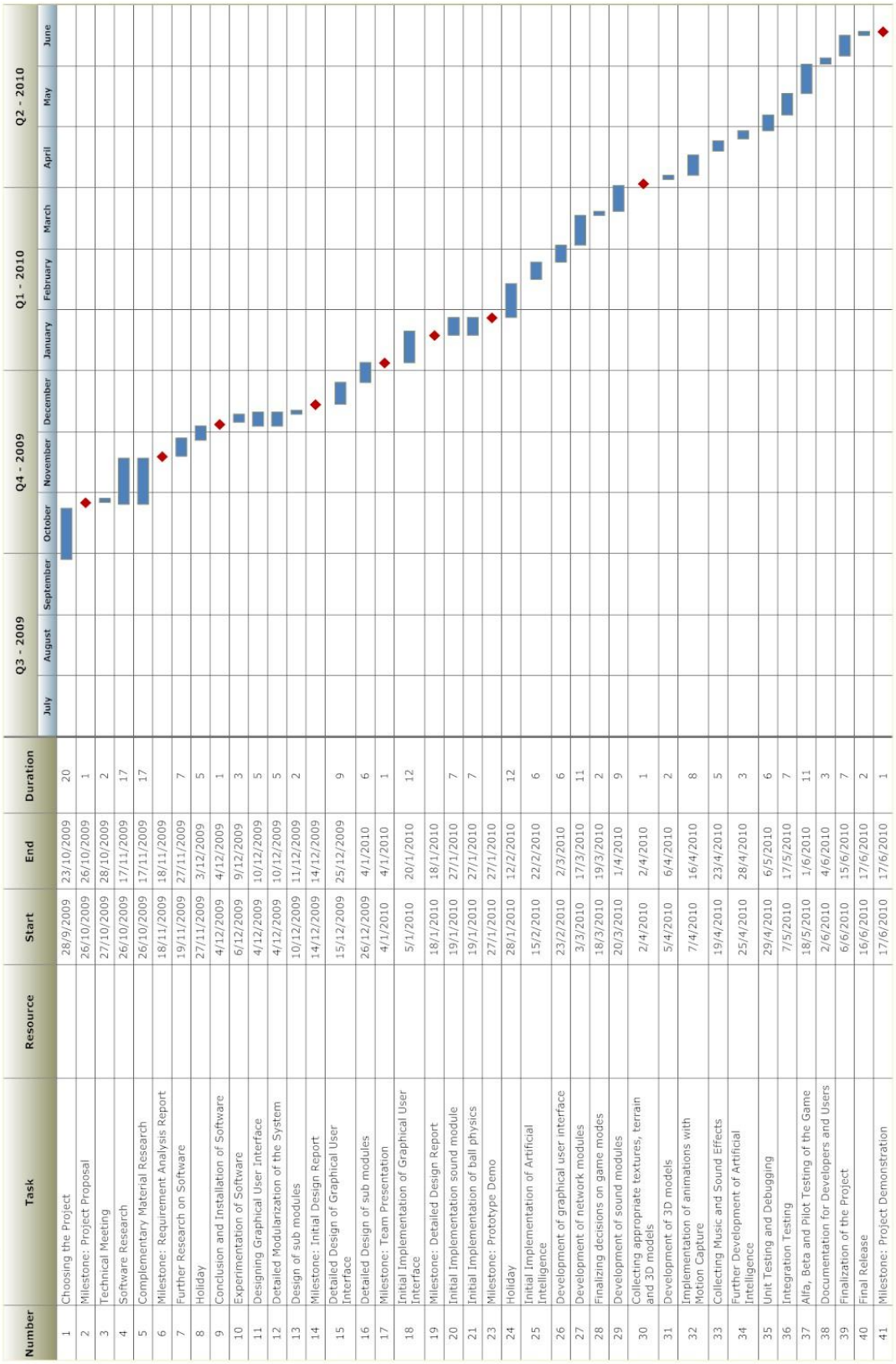
In the game play screen, the movement of the ball and some animations of footballers are simulated considering real life physics. Although real life physics is too much detailed for a perfect simulation, good approximations are made by programmers in the development of physics engines. In this project; we are mainly focused on ball motion to provide a realistic looking game play screen. We are planning to use motion capture system in MODSİMMER building in METU campus in order to deal with footballer animations but this is going to be just an additional feature that we do not include in the main aspects of this project.

The definition of Ball class, which defines ball motion, is given in “Object Oriented Modeling” chapter of this document.

10 Project Schedule

Design Project

Number	Task	Resource	Start	End	Duration
1	Choosing the Project		28/9/2009	23/10/2009	20
2	Milestone: Project Proposal		26/10/2009	26/10/2009	1
3	Technical Meeting		27/10/2009	28/10/2009	2
4	Software Research		26/10/2009	17/11/2009	17
5	Complementary Material Research		26/10/2009	17/11/2009	17
6	Milestone: Requirement Analysis Report		18/11/2009	18/11/2009	
7	Further Research on Software		19/11/2009	27/11/2009	7
8	Holiday		27/11/2009	3/12/2009	5
9	Conclusion and Installation of Software		4/12/2009	4/12/2009	1
10	Experimentation of Software		6/12/2009	9/12/2009	3
11	Designing Graphical User Interface		4/12/2009	10/12/2009	5
12	Detailed Modularization of the System		4/12/2009	10/12/2009	5
13	Design of sub modules		10/12/2009	11/12/2009	2
14	Milestone: Initial Design Report		14/12/2009	14/12/2009	
15	Detailed Design of Graphical User Interface		15/12/2009	25/12/2009	9
16	Detailed Design of sub modules		26/12/2009	4/1/2010	6
17	Milestone: Team Presentation		4/1/2010	4/1/2010	1
18	Initial Implementation of Graphical User Interface		5/1/2010	20/1/2010	12
19	Milestone: Detailed Design Report		18/1/2010	18/1/2010	
20	Initial Implementation sound module		19/1/2010	27/1/2010	7
21	Initial Implementation of ball physics		19/1/2010	27/1/2010	7
23	Milestone: Prototype Demo		27/1/2010	27/1/2010	
24	Holiday		28/1/2010	12/2/2010	12
25	Initial Implementation of Artificial Intelligence		15/2/2010	22/2/2010	6
26	Development of graphical user interface		23/2/2010	2/3/2010	6
27	Development of network modules		3/3/2010	17/3/2010	11
28	Finalizing decisions on game modes		18/3/2010	19/3/2010	2
29	Development of sound modules		20/3/2010	1/4/2010	9
30	Collecting appropriate textures, terrain and 3D models		2/4/2010	2/4/2010	1
31	Development of 3D models		5/4/2010	6/4/2010	2
32	Implementation of animations with Motion Capture		7/4/2010	16/4/2010	8
33	Collecting Music and Sound Effects		19/4/2010	23/4/2010	5
34	Further Development of Artificial Intelligence		25/4/2010	28/4/2010	3
35	Unit Testing and Debugging		29/4/2010	6/5/2010	6
36	Integration Testing		7/5/2010	17/5/2010	7
37	Alfa, Beta and Pilot Testing of the Game		18/5/2010	1/6/2010	11
38	Documentation for Developers and Users		2/6/2010	4/6/2010	3
39	Finalization of the Project		6/6/2010	15/6/2010	7
40	Final Release		16/6/2010	17/6/2010	2
41	Milestone: Project Demonstration		17/6/2010	17/6/2010	1



11 Reference List

These references listed below are for the figures that are used throughout the document. References related with tools and libraries are given in their text.

http://www.konami-pes2010.com/

http://www.ea.com/games/fifa-soccer

http://www.frmtr.com/pes-serisi/3079748-pes-2010-rooney-juventus-become-legend-resimleri.html

http://www.mcpsp.com/oyun-incelemeleri/19144-pes-2010psp-inceleme-herosaint.html

http://www.pespatchs.com/2009/11/5-training-stadium-turfs-by-frenkie.html

http://kitanamedia.com/forums/viewtopic.php?f=32&t=697&start=60

http://forum.donanimhaber.com/m_34615783/mpage_6/f_/key_//tm.htm#35530346

http://www.bluegartrls.com/forum/showthread.php?t=81876

http://www.soccergenclix.net/forum/pes-2010-genel-tartisma/19319-pes-2010-inceleme-mistiq.html

http://yeni.soccercenter.net/forum-konu-goster/-17-Agustos-Pes-2010-Incelemesi-Berker949494_137808/

http://www.pesgaming.com/showthread.php?t=63398