CENG 491
Computer Engineering
Design 1

Initial Design
Report

Geeks In Action
Contents
1 Introduction .................................................................................................................. 5
  1.1 Motivation ............................................................................................................. 5
  1.2 Project Description .............................................................................................. 5
  1.3 Purpose of Document .......................................................................................... 6
    1.3.1 Game Play ..................................................................................................... 6
    1.3.2 Game User Interface ....................................................................................... 6
    1.3.3 Game Concept ............................................................................................... 6
  1.4 Design Constraints ............................................................................................... 7
    1.4.1 Project Schedule ............................................................................................ 7
    1.4.2 Language Constraints .................................................................................... 7
    1.4.3 Data Constraints ........................................................................................... 7
    1.4.4 User Interface ................................................................................................ 7
  1.5 Project Goals and Scope .................................................................................... 7
  1.6 Team organization ............................................................................................... 8
  1.7 Tools .................................................................................................................... 8
  2 Constraints of Development Process ................................................................... 9
    2.1 Constraints Related to Members of the Project Team ....................................... 9
    2.2 Constraints Related to Implementation ............................................................ 9
    2.3 Constraints Related to Licensing and Environment .......................................... 9
  3 Game Description and Mechanics ...................................................................... 10
    3.1 Game modes: ..................................................................................................... 10
      3.1.1 Traditional Mode: ....................................................................................... 10
      3.1.2 Japanese goal mode .................................................................................. 11
      3.1.3 Street football mode .................................................................................. 12
      3.1.4 German goal mode ................................................................................... 13
      3.1.5 Indoor Mode (Saloon Football) .................................................................. 13
      3.1.6 League Mode ............................................................................................ 13
      3.1.7 Training Mode ........................................................................................... 14
    3.2 Edit Mode ......................................................................................................... 15
      3.2.1 Footballer Properties ................................................................................ 16
      3.2.2 Team Properties ......................................................................................... 17
    3.3 Weather Condition ............................................................................................ 18
    3.4 Environment Properties .................................................................................... 19
    3.5 Sound Properties ............................................................................................... 19
  4 Project Requirements ............................................................................................ 19
    4.1 Functional Requirements ................................................................................ 19
    4.2 Operational and Structural Requirements ....................................................... 20
      4.2.1 Graphic Engine ........................................................................................... 20
      4.2.1.1 Irrlicht Engine ......................................................................................... 20
      4.2.1.2 Torque 3D .............................................................................................. 20
      4.2.1.3 OGRE ..................................................................................................... 21
      4.2.1.4 Id Tech 3 ............................................................................................... 21
Geeks In Action
Initial Design Report

4.2.3 Sounds............................................................................................................ 22
4.2.3.1 The Open Audio Library, OpenAL................................................................. 22
4.2.3.2 FMOD ........................................................................................................... 22
4.2.4 Networking....................................................................................................... 23
4.2.4.1 UDP (User Datagram Protocol)................................................................. 23
4.2.4.2 RakNET......................................................................................................... 23
4.2.4.3 Zoidcom ...................................................................................................... 24
4.2.5 Physics ............................................................................................................ 24
4.2.5.1 ODE (Open Dynamics Engine) ................................................................... 24
4.2.6 Artificial Intelligence .................................................................................... 25
4.3 Non-functional Requirements ........................................................................... 25
4.3.1 Usability and Playability .............................................................................. 25
4.3.2 Reliability and Security ................................................................................ 25
4.3.3 Portability ........................................................................................................ 25
5 Functional Modeling ............................................................................................ 26
5.1 Level 0 of Data Flow Diagram ......................................................................... 26
5.2 Level 1 of Data Flow Diagram ......................................................................... 27
5.3 Entity Relationship Diagram ............................................................................ 28
6 Object Oriented Modeling .................................................................................... 29
6.1 Class Definitions ............................................................................................... 29
6.1.1 Footballer Class ........................................................................................... 29
6.1.2 Environment Class ....................................................................................... 30
6.1.3 Network Class .............................................................................................. 30
6.1.4 Input Class .................................................................................................... 31
6.1.5 Audio Class .................................................................................................. 32
6.2 Class Diagrams ................................................................................................. 33
7 User Interface Design .......................................................................................... 34
7.1 Start Menu ......................................................................................................... 34
7.1.1 Friendly Game .............................................................................................. 35
7.1.3 Continue League: ....................................................................................... 37
7.1.4 Multiplayer Game ......................................................................................... 38
7.1.4.1 Open LAN Game ....................................................................................... 39
7.1.5 Fun Modes: .................................................................................................. 40
7.1.6 Training Modes ............................................................................................ 41
7.1.7 Settings ......................................................................................................... 42
7.1.7.1 General Settings ..................................................................................... 43
7.1.7.2 Display Settings ....................................................................................... 44
7.1.7.3 Sound Settings ........................................................................................ 45
7.1.7.4 Controller Settings ................................................................................ 46
7.2 League Menu ..................................................................................................... 47
7.2.1 Fixtures .......................................................................................................... 48
7.2.2 Tactics & Players .......................................................................................... 49
7.2.3 Transfer ......................................................................................................... 50
7.3. In Game Menu .................................................................................................. 51
8 Usage Scenarios ...................................................................................................... 52
8.1 Start Menu Usage .............................................................................................. 52
8.2 League Menu Usage .......................................................................................... 53
8.3 In Game Menu Usage ................................................................. 54
8 Project Modules ........................................................................ 55
  8.1 Graphical User Interface Module ........................................... 55
  8.2 Game Engine Module ............................................................. 56
  8.3 Input Module ......................................................................... 57
  8.4 Menu Module ......................................................................... 57
  8.5 Artificial Intelligence Engine Module ...................................... 57
  8.6 Graphic Engine Module ........................................................... 57
  8.7 Network Module .................................................................... 57
  8.8 Audio Module ......................................................................... 58
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 game is to develop well and entertaining a football game for Linux users and to reduce 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. 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 games will 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 own rules. Number of players will be changed between 1 and 4 players. All players can use only keyboard. The game can be played both single player and multiplayer via LAN. The
game will be played in a 3D atmosphere. Atmosphere will be rendered as realistic as possible. Stadiums, outdoor game areas and 3D characters will 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 will control unselected footballers. AI will satisfy basic conditions.

1.3 Purpose of Document
Purpose of this document is to give information about the initial design steps of the game. These topics are introduced and explained briefly 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 will be both understandable and functional, complex graphic components are used in modern games. In our project, we will design not so much complex but easy to understandable and functional graphical interfaces.

1.3.3 Game Concept
Our game will be both similar and different from classical football games. In other words game will 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 have chosen 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 will be easy and simple in order to making it user friendly. It will be easy to reach games. In other words users do not need to enter a lot of menu to reach 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
- technical support

1.6 Team organization

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

Our team is going to have democratic decentralized structure.

1.7 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 will use C++ for implementation and Irrlicht as graphics library and game engine.

- In terms of graphics, OpenGL will 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 said department. Here are some manners about project:

- The process entered on October 2009 and will end in June 2010. Now we have 7 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 limiting 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 for noncommercial purposes. We are going to license our game with GPL so our game should be designed in a way that will not 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 on 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 will be two halves; each half length will be equal. Match length will be between 5 and 30 minutes. There will be a referee. There will be rules like penalty, throw in, corner, goal kick.
3.1.2 Japanese goal mode

Another mode is Japanese goal mode. Japanese mode has different rules from classical mode. The rules of this mode are below. In this mode, there will be no goal keeper. This mode is based on the skills of the player. The more talented team can win the game easily. Size of a goal will become smaller than traditional mode. Each team will have 2 or 3 players. There will not be a referee. If a team performed 3 corner kicks, there will be a penalty shoot out. There will not be throw in or goal kick and match will be finished when a team scores a defined amount of goals.
3.1.3 Street football mode
Another mode is the street football mode. The rules of this mode are described below. The game is going to include 1 vs 1 or 2 vs 2 playing. There will be only one goal and one goalkeeper. There won`t be a match time. There will be a goal limit to win the match, for example; 3 goals win.
3.1.4 German goal mode

The first “for fun” mode is “German goal” mode. The rules of German goal mode are that; each team will have three players and the game will not be played in a stadium. There will be different place choices such as streets and suburb areas.

3.1.5 Indoor Mode (Saloon Football)

Another “for fun” mode is indoor mode. As expected, the game area is an astro pitch and rules will be same as an astro pitch football game. The number of players is limited as 6 for each team. Matches will be played on 2 halves. There will be no substitution. There are no rules such as throw in, goal kick and corner kick. There won’t be a referee. There are different indoor stadium choices.

3.1.6 League Mode

Different league choices are going to be involved such as Serie A, La Liga etc. Each league choice is going to have specific league cups and each league is going to
have certain number of matches.

3.1.7 Training Mode
By the help of trainings users can improve themselves on game play. There will be a training menu for user. In this menu following items will exist.

**Shooting:** User 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 will be goal. So; by the help of free kick training, users become a fatal free kicker. Different training places can be chosen; stadiums, streets and indoor places.
3.2 Edit Mode

In these mode users can create their own players and use these created players in 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.
3.2.1 Footballer Properties

Every football player has unique attributes such as:

- Age
- Condition
- Injury
- Shooting
- Current team
- Height
- Constitution
- Agility
- Mentality
- Reputation

The values of these properties are between 0 and 20. (except current team, injury, age and height)
According to these properties, each football player will have a power between 0 and 20.

### 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 will be decided according to average of other properties.
3.3 Weather Condition

Game will have three main weather conditions: fine, rainy and snow. Each weather condition will have different effects on outdoor football environment. Each weather condition will have different graphic properties.

Snow covered pitch

Fine weather condition
3.4 Environment Properties
The game will contain a lot of game areas. In classical mode; game areas will be stadiums. For Japanese goal mode, German goal mode and street football mode; game areas will be outdoor places. For the indoor mode, matches will be played on an astro pitch. For classical modes there will be various stadium options.

3.5 Sound Properties
Below sound properties will be in the game
- Goal celebrations
- Tackle sound
- Whistle sound
- Ball hit sound
- Audience sound

4 Project Requirements
Software requirements is the documenting completely describes the required attitude of the software.

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.
- Replaying in-game videos.
- 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. Another job of engine is to combine the subcomponents of the game. 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++. 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, OSX, 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. 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 LODs 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. 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. Our game is going to be open source, so we intend that it can be developed by anyone who wants. 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.

4.2.3 Sounds

Our game will have different sounds during game and waiting for the next session. The sounds will change between the steps. Audience, referee, announcer and environment sound is going to be used during games. Sound options will be controlled by user via settings. We looked 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. 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. The
development of FMOD is based on the years of experienced 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). 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. 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. 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 will be “street football” mode, and in this mode there are some calculations needed according to the ball's movement. We considered 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. 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.

Although we considered ODE as a physics engine; we did not choose it for a physics engine, since the necessity of a physics engine is still unclear for our game.
Decisions on this issue will be finalized in detailed design report.

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 inspired some other games on market. We will 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 will not be a platform independent game. There is no much
entertaining football games on Linux operating systems, so our aim to make a funny and entertaining soccer game for Linux.

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.
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.
5.3 Entity Relationship 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”, “Stop”
- **direction**: It is a string indicating the direction of footballer's movement on the field, which can be “North”, “South”, “East”, “West”
- **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.
Geeks In Action
Initial Design Report

**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:

- **game_mode**: 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.
- **join_game()**: 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.
- **receiveGameData()**: 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.
6.2 Class Diagrams

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

![Class Diagram](image-url)
7 User Interface Design

7.1 Start Menu

**Friendly Game**: Opens “Friendly Game” menu (will be mentioned detailly in 7.1.1) where you can start a friendly game against computer.

**Start League**: Opens “Start League” menu (will be mentioned detailly in 7.1.2) where you can start league.

**Continue League**: Opens “Continue League” menu (will be mentioned detailly in 7.1.3) where you can continue a previously created league.

**Multiplayer Game**: Opens “Multiplayer Game” menu (will be mentioned detailly 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 detailly in 7.1.5) where you can start a fun mode game.

**Training Modes**: Opens “Training Modes” menu (will be mentioned detailly in 7.1.6) where you can enter training modes.

**Settings**: Opens “Settings” menu (will be mentioned detailly in 7.1.7) where you can
change game settings.

**Exit**: Closes the program and returns to Linux.

### 7.1.1 Friendly Game

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

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:

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

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 PvP (Player vs. Player) mode.

**Enter Game:** Joins the selected game.

**Back:** Returns to Main Menu.
7.1.4.1 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:

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

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

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

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.
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

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

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.
7.2 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

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

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.
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

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.
8.2 League Menu Usage
League menu usage scenario is illustrated with this use case diagram.
8.3 In Game Menu Usage

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

8.1 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.
8.2 Game Engine Module

Since Irrlicht is going to be used in this project, game engine is not going to be implemented from scratch.
8.3 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.

8.4 Menu Module

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

8.5 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.

8.6 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.

8.7 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.
8.8 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.