KONTRPİYE

REVISED DESIGN REPORT

LINUX EVOLUTION SOCCER

1394865 Ufuk Dallı

1448505 Hakan Çağlar

1502145 Berker Batur

1502202 Uğur Büyükköy

Contents

1. Introduction

1.1 Purpose of the Report

This document is the detailed design report of Linux Evolution Soccer documented by the developer team Kontrpiye, the senior project team of four METU Computer Engineering students. It aims to cover all the aspects of the design concepts and methods implemented in the project Linux Evolution Soccer in regards of definitions and descriptions.

1.2 Problem Definition

“Video games” has been a great industry in the information age with millions of users and big companies behind the scene producing couple of games say each year. Those companies invest huge budgets into those projects and conduct them with their well-organized, qualified teams. The electronic platform of Linux Evolution Soccer is not a game console (PSP, Xbox) but a personal computer (PC). Computers have been the big targets and the instruments as well as the other game consoles since they happened to be able to be bought easily in the last ten (or so) years.

Still, why would someone bother developing games which are big money-makers in these days for Linux operating systems, operating systems which have been even never heard of by many computer game addicts, but not the famous Windows operating systems with millions of users for which plenty of games had been already developed and still being developed as well? The difference in the user group size should be clear and so should be the outcome of two imaginary projects. Having assumed two conditions in which the first condition is a game put on the market for Windows operating systems for a regular price and the second is the same game with the same price is put on which runs on Linux operating systems. Due to the range of your phantom customers, the profit to be gained from the second condition would be pretty low in comparison with the first condition.

Linux users have been suffering from the lack of prevailing games, naturally, since the majority of computer game users have only Windows operating systems on their computers and game companies are aware of that fact. Most of those users are not even wise up to know that Linux is free. This is undeniably a handicap for Linux environments and also Linux operating system users. Linux users are disregarded in this case and they are pushed either not to be able to play cool games running on their Linux operating systems or installing a Windows operating system to get to play them.

1.3 Motivation

Provided our game, no Linux users will need to have any other operating system to be able to play a qualified football game on their computers. Also users will be able to install and run LES on their Windows operating systems. So LES can be run on both Windows and Linux environments. However, the nuance is that LES is free. Linux is free as well, where Windows is a quite expensive operating system. Users are forced to realize that they may run a cool and free game on some free operating system.

Additionally, as Kontrpiye, we believe when more and more qualified games come up available to the Linux environments, which we are on our way to increment the counter one, also the range of Linux operating systems over all the computer users among the entire world will raise accordingly. We assure that the access to LES is free being available on the internet to be downloaded for free. Having accomplished this, the computer game users are supposed to notice that both Linux and LES are free, trying them would not hurt.

We also consider ourselves sort of innovative since our focus is not on developing a game looking realistic as if it is a football simulation but a game which is really fun to play for hours, and also ending up fun for us too while developing it. Because there are already some considerable games which had been developed running on Linux operating systems and they are free. However, none of them would satisfy a computer game addict. In conclusion, LES aims to be free and fun-to-play. The main approach in the development is the fun-to-play approach. LES is free and fun-to-play for both Linux and Windows users who are supposed to

spend hours, hopefully, playing LES with each other.

1.4 Project Definition and Goals

As it is pretty much implied in the name, Linux Evolution Soccer is meant to be a football game for Linux operating systems however it also runs in the Windows environments. It is a 3D real-time football game resembling today's popular football games Pro Evolution Soccer, Sensible World of Soccer etc. Computer game players will get to be able to play a really 'fun to play' football game on their Linux operating systems too, ultimately. LES also allows someone to challenge his/her friends in multiplayer mode by the help of either additional IO devices on the same computer or via a network to play an exhibition match. It also offers a career in single player mode including a tournament and a league which can be saved and loaded as a profile to be played any time by the user. An artificial intelligence which is able to play with the human user (human vs. computer) or playing to itself (computer vs. computer) is the crucial point in the single player mode. It aims to offer solutions for the lack of fun in the games for Linux systems still standing in the market. This will be accomplished via the spiral model to be described in the "Design Methodologies" chapter.
Offline playing support, including Single Player Mode and Multiplayer Mode on the same computer, should be a big attribution in the sense of the design of the artificial intelligence, hopefully ending up in more and more number of potential users obviously. Existing football games do not provide much fun when it is human player versus computer since the artificial intelligence embedded is not good enough.

The users are also provided entertaining challenges in LES. Since the primary goal is keeping the fun aspect at the top level due to the motivation contributing to make the Linux users really enjoy the game. Several challenging achievements are implemented in LES for the users to accomplish in the Single Player -> Career Mode including playing a whole country league, World Cup 2010 and UEFA Cup.

2. Design Methodologies

Since this is a football game and as developers we not being so experienced at developing games, prototyping becomes as the primary option out of all the choices coming up as the process model. Interactive prototypes are to be developed which can be quickly replaced or changed in line with design feedback coming from the users which have been testing at all prototypes and all.

Prototyping helped in resolving uncertainty about how the design of the game could be better in respect of users' needs. This will be accomplished by obtaining information from users about the corresponding functionalities of the game. Since prototyping is actually a form of collecting information on requirements and the adequacy of possible designs, the prototype designed at each stage is to be thrown-away. Figure 1 shows how a spiral process model works and it is the process model used in the development of LES.

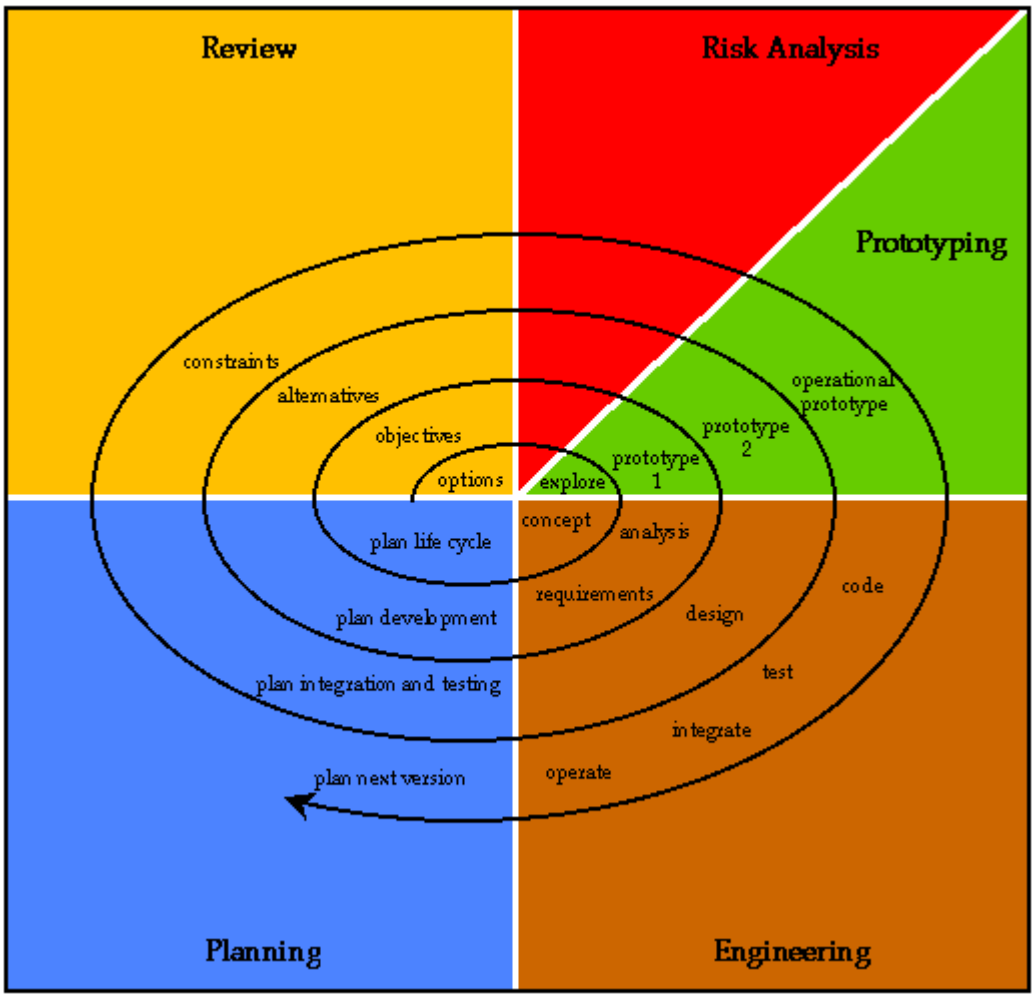


Figure 1 - Spiral-Model: http://www.csse.monash.edu.au/~jonmc/CSE2305/Topics/07.13.SWEng1/html/spiral.GIF

Once a good version of any design aspect of the game comes up, it is most probably to be assumed it is the best solution for that aspect. However at each prototype, every design aspect is to be re-checked because usually the relations between modules also differ depending on any single little change on the design. It is not an easy job when your aim is to keep users playing a game having fun for a considerable amount of time and it is just obligatory to have users' opinion on every single matter. That is simply why the spiral model is the best solution here.

Appropriate and manipulated (by the developers) users will be testing the prototypes to cover all the ranges of the corresponding design. Manipulation happens by supplying users with former information that they necessarily get to try. Opinions and advices of every single

user with their past with football games will be recorded to be considered at the design following the current one obviously.

While using this method, creating too polished and looking already finished prototypes may result in too much satisfied testers also who are not capable of telling what is really missing. To exemplify, too fast responses in the game like starting a match instantaneously or too sophisticated graphics in an initial design will not help. Thus the prototypes will have the response times which are predicted to be in the final product. Spending too much time on the initial prototypes will also be avoided since user evaluations may result in substantial changes in those phases.

Briefly, spiral mode benefits will be exploited mostly in regards of more accurate recognition of user requirements. All in all, this process model will result in lower costs in the long run and also less considerable failures whereas costs and risks increase at each levels of the spiral as easily can be figured from the spiral model in Figure 1, above.

3. Data Design

3.1 ER Diagram

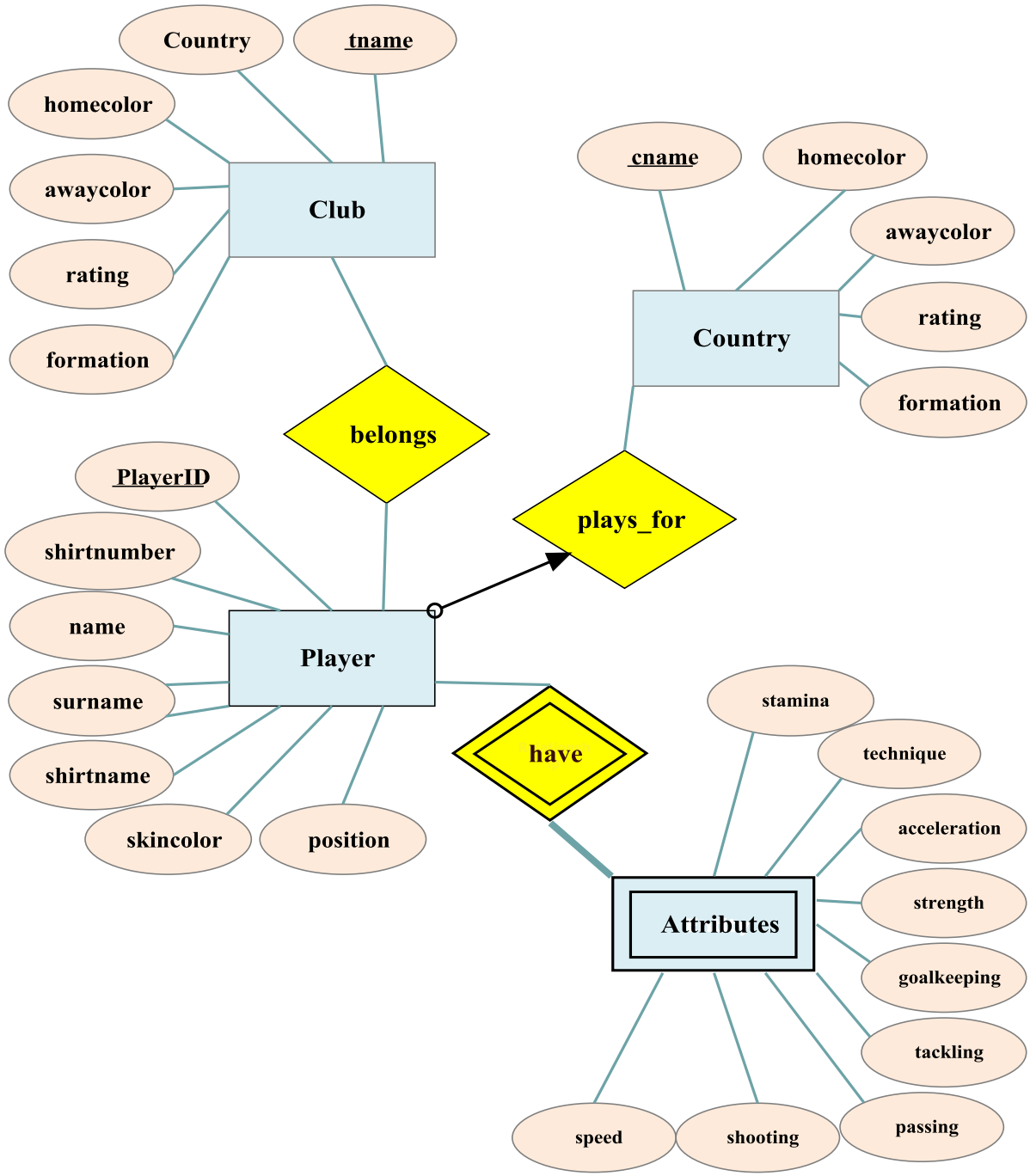


Figure 2 - ER Diagram of Database Design (1)

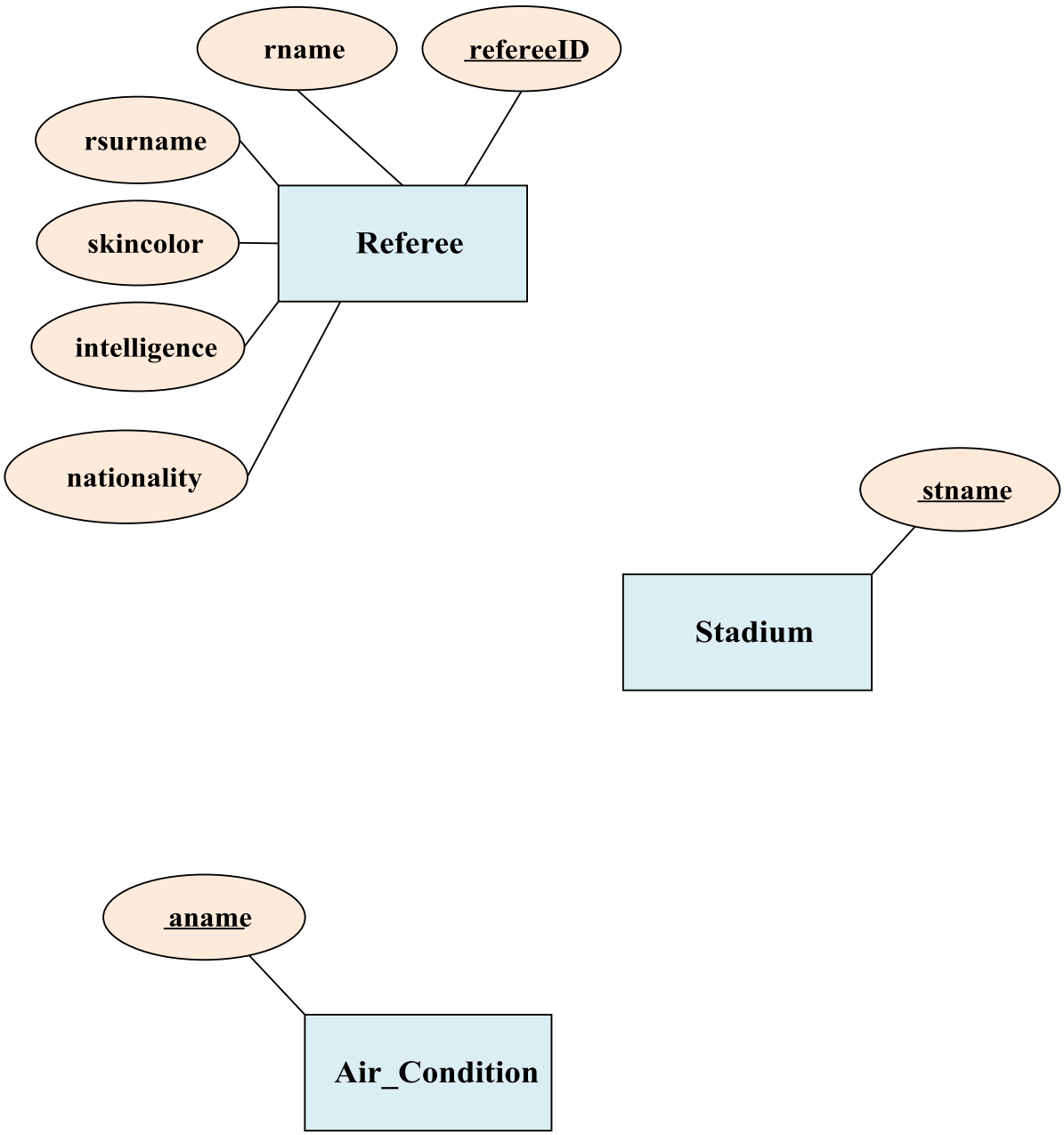


Figure 3 - ER Diagram of Database Design (2)

3.2 Database Schemas

3.2.1 Club Table

Attributes	Type	Null	Unique	Foreign Key	References
tname	Varchar(30)	No	Yes	No	-
country	Varchar(30)	No	No	No	-
homecolor	Varchar(30)	No	No	No	-
awaycolor	Varchar(30)	No	No	No	-
rating	Float	No	No	No	-
formation	Varchar(20)	No	No	No	-

This table holds information about the clubs in terms of name of the club and its main features including colors of the home and away strips and the country the club belongs to. It also contains some basic information about the club. Rating of the club is calculated via the attributes of the player it has. Formation of the players are specific for all clubs at the beginning and it can be altered by the user in the last screen just before the match in the user interface, namely in “Formations”, prior to starting the game for a match only or could be permanently for the future by saving the formation in a profile. This table has no foreign keys and references to the other tables and has the primary key ‘tname’ which stands for the name of the club.

3.2.2 Country Table

Attributes	Type	Null	Unique	Foreign Key	References
cname	Varchar(30)	No	Yes	No	-
homecolor	Varchar(30)	No	No	No	-
awaycolor	Varchar(30)	No	No	No	-
rating	Float	No	No	No	-
formation	Varchar(20)	No	No	No	-

This table holds information about the country teams in terms of name of the country and its main feature as colors of the home and away strips. National teams are referred here by countries such as Turkey National Football Team or England National Football Team. This table also contains some basic information about the country team. Rating of a country team is calculated via the attributes of the football players who plays for that country. Formation of the players are specific for all countries at the beginning and it can be altered by the user in the last screen just before the match in the user interface, namely in “Formations”, prior to starting the game for a match only or could be permanently for the future by saving the formation in a profile. This table has no foreign keys and references to the other tables and has the primary key ‘cname’ which stands for the name of the country. Players playing for a country are determined at the beginning and users cannot select any different players for these national teams.

3.2.3 Player Table

Attributes	Type	Null	Unique	Foreign Key	References
PlayerID	Integer	No	Yes	No	-
shirtnumber	Integer	No	No	No	-
name	Varchar(20)	No	No	No	-
surname	Varchar(20)	No	No	No	-
shirtname	Varchar(20)	Yes	No	No	-
skincolor	Varchar(10)	No	No	No	-
position	Varchar(10)	No	No	No	-

This table holds information about the players. Player properties which are not related to the gameplay are held here. It contains the primary key 'PlayerID', shirt number, name and surname, name of the player on his shirt, position of the player in a match and skin color of the player. PlayerIDs are to be constructed after the table is created completely with all other attributes. They are assigned from one to the number of all players in the table. Skin color is of three types including blonde, brown and dark. A player may play in more than one position and the 'position' key is used such as 'DC-MR-MC' which means "defence center", "midfield right" and "midfield center" respectively. Shirt name of the player may be null and if it is null, then first name of the player is to be used for his shirt name written in the back top of his shirt.

3.2.4 have_Attri Table

Attributes	Type	Null	Unique	Foreign Key	References
PlayerID	Integer	No	Yes	Yes	Player
stamina	Integer	No	No	No	-
technique	Integer	No	No	No	-
acceleration	Integer	No	No	No	-
strength	Integer	No	No	No	-
goalkeeping	Integer	No	No	No	-
tackling	Integer	No	No	No	-
passing	Integer	No	No	No	-
shooting	Integer	No	No	No	-
speed	Integer	No	No	No	-

This table holds the information of a weak entity which is “Attributes” and a relation between this weak entity and Player table. This table is meant to deal with ingame features of the players. There are nine fields in the table which are PlayerID, stamina, technique, acceleration, strength, goalkeeping, tackling, passing, shooting and speed. These fields are filled values out of hundred (100) and they are all used during a match to calculate the actions of the players. For example, a player with a high passing attribute can pass the ball to his teammate smoothly with a high probability. However, a player with low value of passing is not be able to send the ball to the target that easily. In addition, some of the values in these fields may vary during a match, such as stamina which shows that the corresponding player gets exhausted when decreasing.

3.2.5 Belongs Table

Attributes	Type	Null	Unique	Foreign Key	References
tname	Varchar(30)	No	No	Yes	Club
PlayerID	Integer	No	No	Yes	Player

This is a relation table, which shows the relation between Club table and Player table. This table includes the players which belong to a club. There are two primary and foreign keys which are ‘tname’ and ‘PlayerID’ referencing to the tables Club and Player respectively. In other words, this is a list of active players playing for a club. Therefore, through the PlayerIDs, attributes of a player can be reached from other tables.

3.2.6 plays_for Table

Attributes	Type	Null	Unique	Foreign Key	References
cname	Varchar(30)	No	No	Yes	Country
PlayerID	Integer	No	No	Yes	Player

This table holds information just like belongs table. It is also a relation table, which shows the relation between Country table and Player table. It includes the players which plays for a country team. There are two primary and foreign key as ‘cname’ and ‘PlayerID’ which are referenced to the tables Country and Player respectively. In other words, this table can provide the list of active players playing for a national team. Like the belongs table, through the PlayerIDs, attributes of a player can be reached from other tables.

3.2.7 Referee Table

Attributes	Type	Null	Unique	Foreign Key	References
refereeID	Integer	No	Yes	No	-
rname	Varchar(20)	No	No	No	-
rsurname	Varchar(20)	No	No	No	-
skincolor	Varchar(10)	No	No	No	-
intelligence	Integer	No	No	No	-
nationality	Varchar(20)	Yes	No	No	-

This table holds information about the referees in the game. It contains name of the referee, skin color, nationality and intellegence of the referee. The fields being used during a match are ‘skincolor’ and ‘intelligence’. Skin color is for just the rendering the referee model with different skin colors. Intelligence is a value out of 100 just like player attributes. A referee with high intelligence value is able to make the right ingame decisions (corners, penalties) which could be resulting in pretty many mistakes during a match with a referee having low intelligence value.

3.2.8 Stadium Table

Attributes	Type	Null	Unique	Foreign Key	References
stname	Varchar(30)	No	Yes	No	-

This table holds information about the stadiums. The only field is the name of the stadium and the name is the primary key also. In the 'Stadium' menu in the user interface just prior to the match start, the list of the stadiums are sent to the interface menu from this table to be shown in the menu with a small photograph of each stadium as well for the user to choose one.

3.2.9 Air_condition Table

Attributes	Type	Null	Unique	Foreign Key	References
aname	Varchar(30)	No	Yes	No	-

This table holds information about weather conditions. The user is able to pick up an air condition from the available three in the 'Weather' menu prior to the match start. Those are sunny, rainy and snowy. The time of the match can also be chosen here which are day and night times. When the match starts the chosen air condition and time of day result in different textures and lightning of the stadium .

4. Architectural Design

4.1 Design Overview

LES system needs a complex architecture because many modules work cooperatively. Good separation of the modules and the tasks lead up to easier development in regards of implementations of each module and their integration.

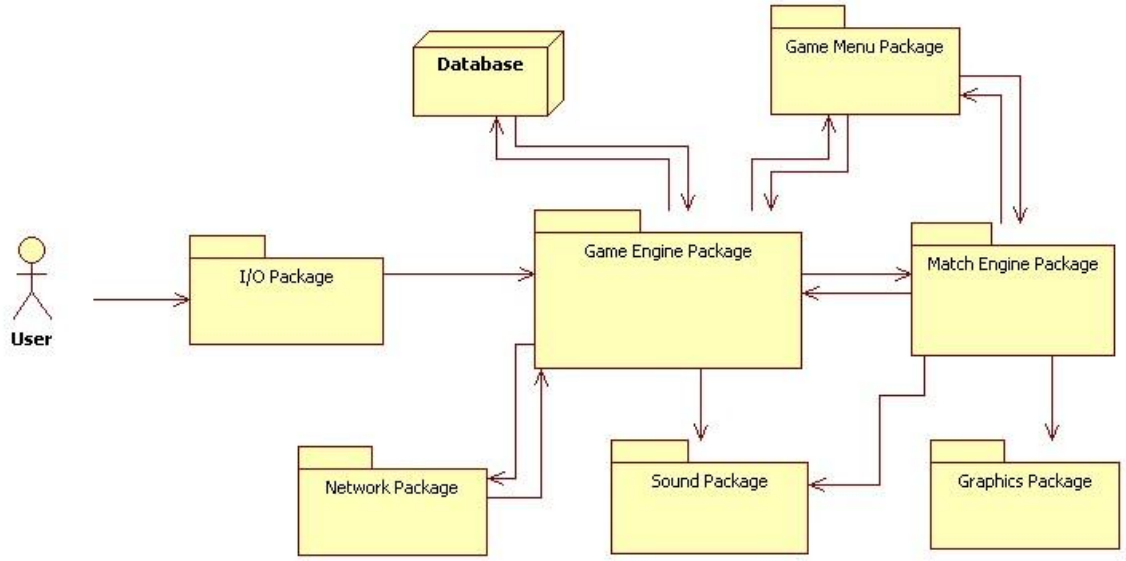


Figure 4 - Design Overview

As shown in Figure 4, there are seven modules in LES which are independent from each other but all dependent to the Game Engine Module. This provides independent implementation of these modules and rapid development of the project. Moreover, when some libraries are to be replaced with another libraries during the development for some reasons in one or more modules, only re-implementation of the related modules are enough.

The core module of the project is the Game Engine Package. It has bidirectional relations with Network, Game Menu, and Match Engine Modules and also the Database of the system. This module takes the input coming from the user(s). It also has one way relation with the Sound Module. When the game (not the match) starts, Game Engine's main class initializes all the other classes in it, as shown in figure 8. Game Engine has the control over the other classes while they perform their specific duties.

When the user is arranging settings before the match, the Game Menu goes to the pre-match menu. Changes which the user makes on that menu are applied also on the database of the user profile if necessary.

Game Engine sends all required data to the Match Engine when the user starts a match. Operations during the match are all conducted by the Match Engine Module. AI of the non-user controlled players and objects is implemented in the Match Engine Module. This module also has relations with Sound and Game Menu Modules. Sounds during a match are performed just due to the calls coming from the Match Engine Module. The menu sound/music is handled by the interface i.e. Game Menu Module. Also there exists a pause menu, available only during a match, is shown when the user wants to alter some settings or manage his team by pressing the ESC key or the corresponding button from the game pad. The changes saved during a match are all handled by the Match Engine Module as well.

4.2 LES Environment and Rules Design

The rules of the game are not complicated. There are two teams; each team contains ten field players and a goalkeeper. A goal is scored by kicking the ball over the opponent team's goal line and the three goalposts.

Throw-ins, corners, goal kicks, penalties, faults and off sides all happen just like in a real football game in LES. Also a chief referee and two linesmen are rendered on the pitch. A user could choose a team (club or country team), change the formations of his/her team, alter line-up players, make substitutions, choose stadium, choose ball type, and choose air condition.

The game environment basically consists of:

- Pitch
- Stadium
- Two goals
- Ball
- Two teams
- Twenty Players
- Two Goal Keepers
- Referee and Two Linesmen

4.3 Data Flow Diagrams

The data flow within the system is illustrated in the data flow diagrams below. As it can be seen from the figure 5, the game consists of complex data flow between modules. Game Engine class, always gets user input from I/O devices. Also Game Engine class gets related data of players, clubs, and countries from the Database Module. The operations that the Game Engine performs on the other classes are database changes, initialization of game menu variables, sending sound type, and sending team, stadium, and pitch data to the Match Engine class. Finally, Match Engine sends the updated player, goalkeeper and ball positions to Graphics Module to be rendered.

4.3.1 Level 0

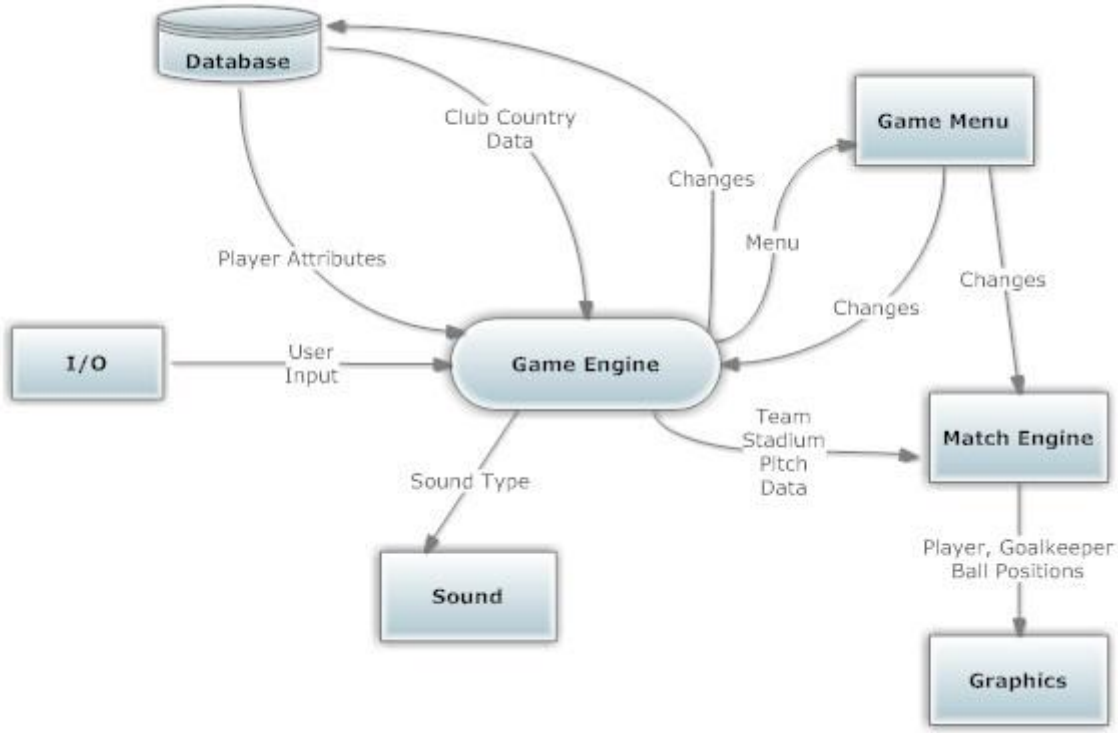


Figure 5 - Level 0 Data Flow

4.3.2 Level 1: Game Engine

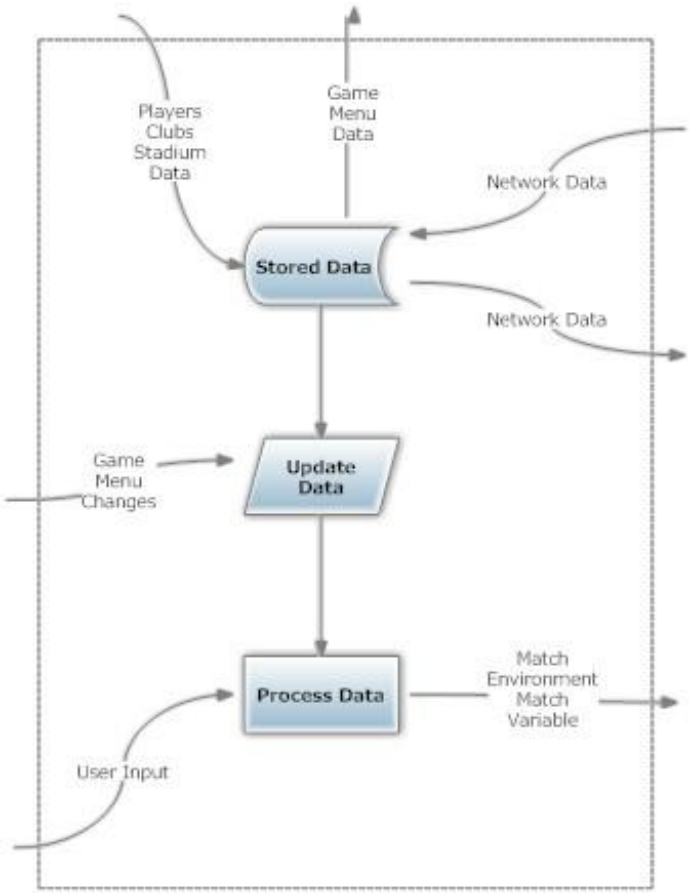


Figure 6 - Level 1 Data Flow

,

4.3.3 Level 2: Match Engine

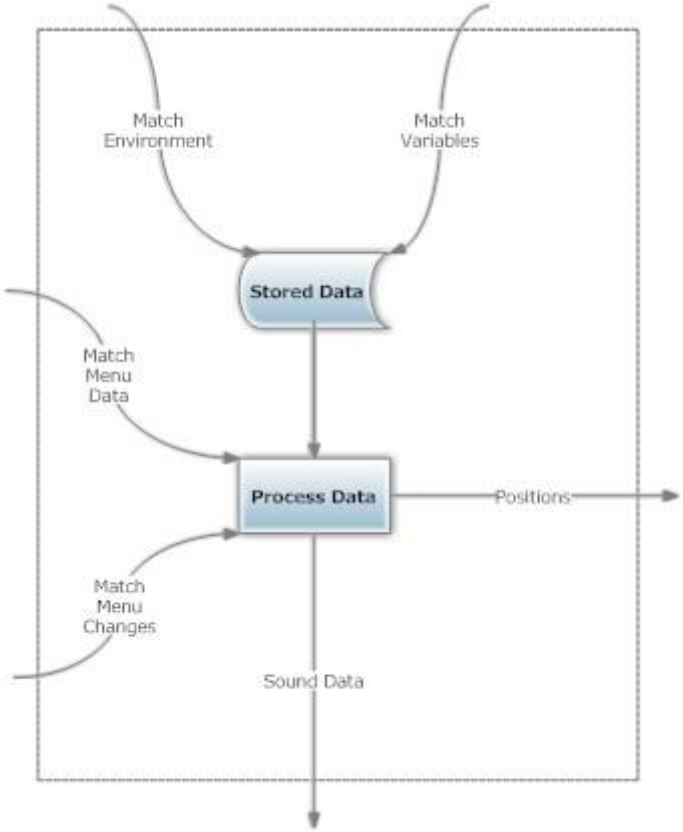


Figure 7 - Level 2 Data Flow

4.4 Class Diagrams

The class hierarchy of the LES is illustrated as class diagrams in each package. The detailed explanations about these classes and their functions are shown in subheadings below.

4.4.1 Game Engine Package

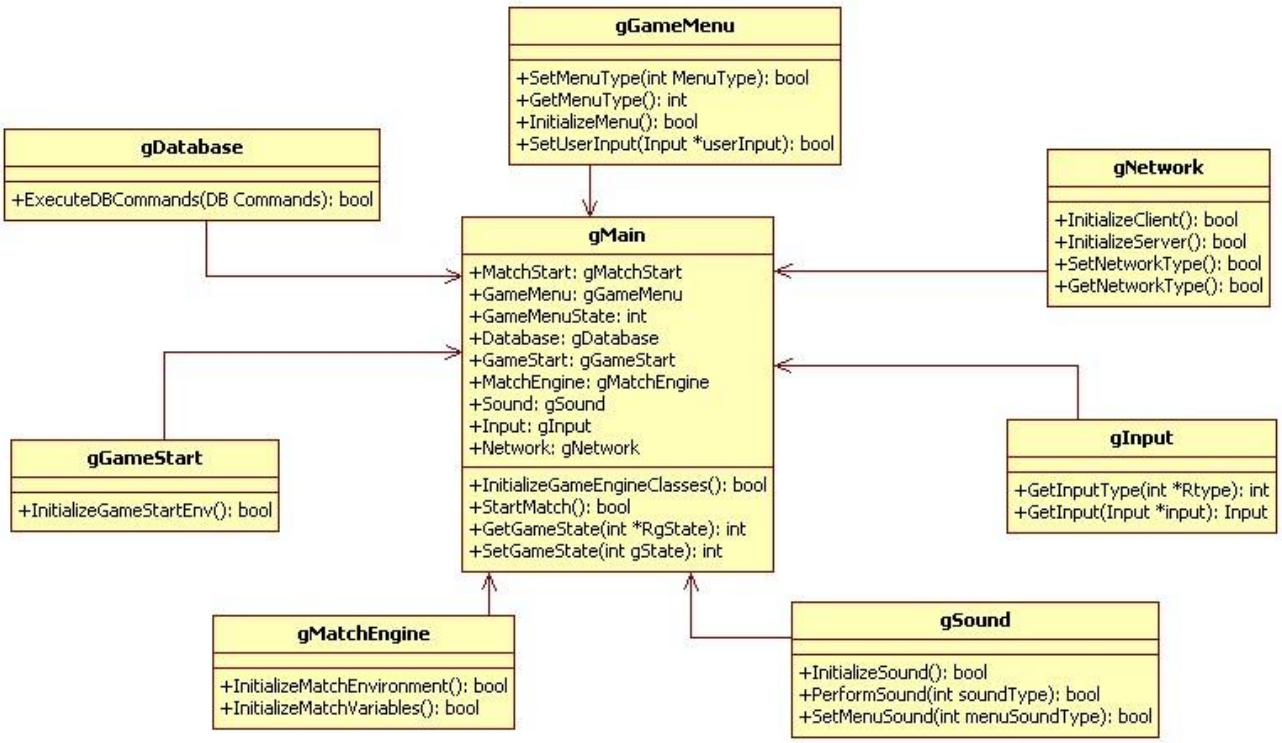


Figure 8 - Game Engine Package

Game Engine classes are at the top of the class hierarchy. Except AI-based jobs (Match Engine Classes), all other things are done by their game engine classes. The main (gMain) class of this package has attributes which are other classes' instances. This main class controls the flow of the game. When the game starts, it makes related initializations. Before the match starts, main class works cooperatively with Game Menu, I/O, Sound, and Network packages. It also executes database commands while transferring data from the database or

updating some data on it. The other classes in this package, which are related with other packages with respect to their names, basically initialize those packages' classes according to the game variables.

4.4.2 Match Engine Package

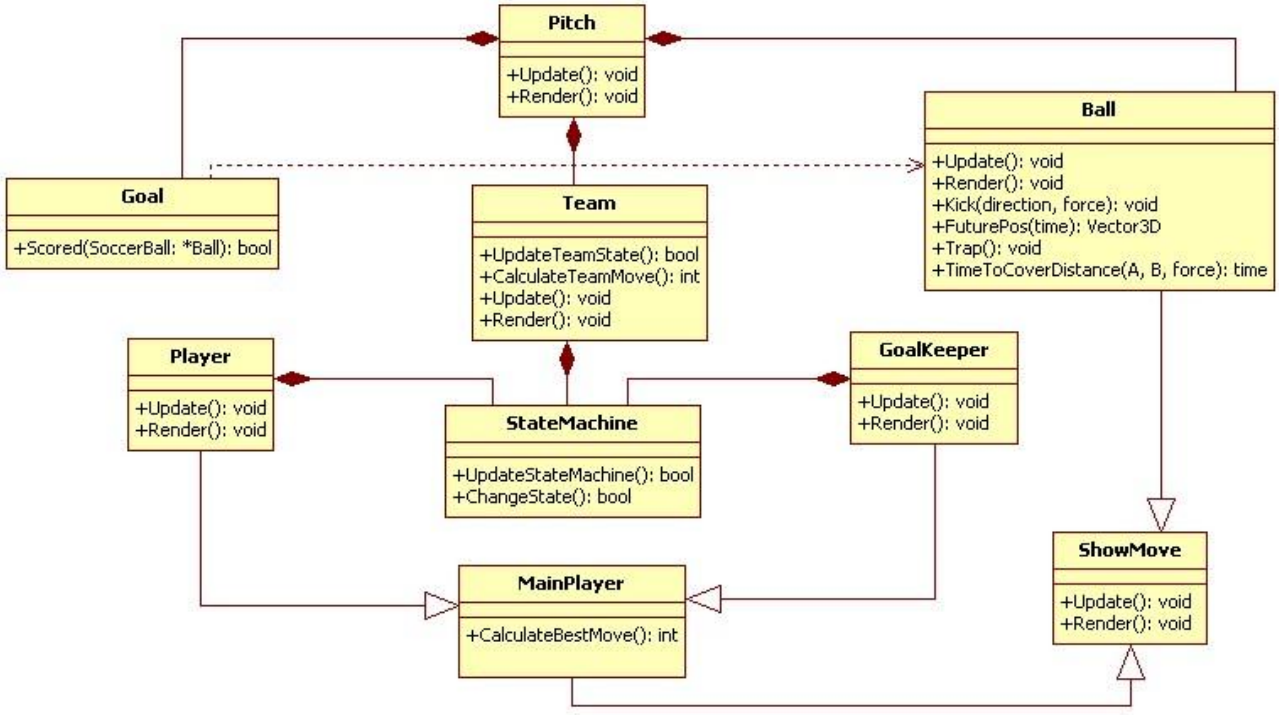


Figure 9 - Match Engine Package

As shown in figure 9, soccer players are separated as players and goalkeepers. Both types are derived from the same base class, MainPlayer. Both own finite state machines, with their own states. Teams also own a StateMachine, supplying a team, the ability to change its behavior depending on the current state of play. Detailed explanations of all classes and their methods are below.

4.4.2.1 Pitch Class

Figure 10 – Pitch Class

Pitch is a rectangular playing area enclosed by throw-ins and goal kick lines. Playing area is encapsulated by the class Pitch. A single instance of this class is to be instantiated when the match begins. The Pitch object owns instances of Team, Ball, and Goal objects as shown in figure 9. The Pitch::Update and Pitch::Render functions are at the top of the update and render hierarchy. At each update step, these methods are called from within the main game loop and, in turn, the appropriate Render and Update methods of every other game entity are called.

4.4.2.2 Goal Class

Figure 11 – Goal Class

Goals are defined by left, right, and top goalposts. A goal is scored if 100% of the ball crosses the goal line. At each run of the match engine loop, the Scored method of each team's goal is called from within Pitch::Update to know if a goal is detected or not.

4.4.2.3 Ball Class

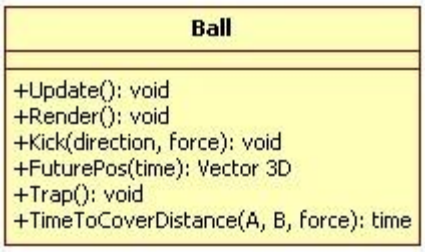


Figure 12 – Ball Class

The data and methods to encapsulate a soccer ball are encoded in the Ball class. Ball also has data members for recording the ball's last updated position and methods for kicking the ball, testing for collision with goalposts and with the players, and calculating the next position of the ball.

4.4.2.4 MainPlayer Class

Figure 13 – MainPlayer Class

Players, goalkeepers, and teams classes inherit from MainPlayer class. Thanks to the home region attribute, all players in the game have their home regions. Basically AI of the players is implemented in the MainPlayer class. CalculateBestMove of the MainPlayer class allows a player to choose what to do in the next time step and updates its state according to that decision. In the calculation of the best move, a player is checked if he:

- can pass forward,
- can pass backward,
- is threatened,
- is in the kicking range,
- is in the support spot range,
- is at the target,
- is controlled player,
- is the closest team member to the ball,
- is the closest player on pitch to the ball,
- is in the home region.

According to results of these queries, the player's best move is determined in that particular state.

4.4.2.5 ShowMove Class

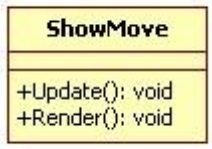


Figure 14-ShowMove Class

Each object which can move in LES derives from ShowMove class and its updated status is graphically rendered thanks to the ShowMove class. ShowMove class keeps the positions and the states of the objects and sends them to Graphics Package.

4.4.2.6 Team Class

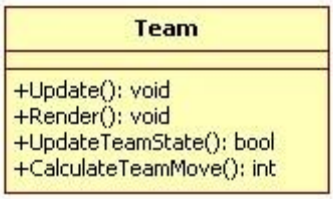


Figure 15 – Team Class

The Team class owns instances of the players that comprise the team. It has pointers to the pitch, the opponent team, the team's home goal, and its opponent's goal. Additionally, it has pointers to the key players on the pitch. Individual players can query their soccer team and use this information in their state machine logic. Team class updates team state as attacking, defending or normal states. Also, team class gets players' best moves, and then decide best move of the team.

4.4.2.7 Player Class

Figure 16 – Player Class

The players who run around the field, passing the ball and taking shots at their opponent's goal are instantiated as objects of the Player class.

4.4.2.8 GoalKeeper Class

Figure 17 – GoalKeeper Class

Goalkeepers in two teams are instantiated as objects of the GoalKeeper class. At first, the two goalkeepers are assigned to the region that overlaps its team's goal.

4.4.2.9 StateMachine Class

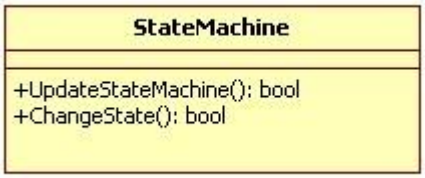


Figure 18 – StateMachine Class

StateMachine class updates and changes the state machine. States are for determination of the time when players do their next moves. When state machine changes state, next positions and moves of the objects in the match are determined for that particular state.

4.4.3 Game Menu Package

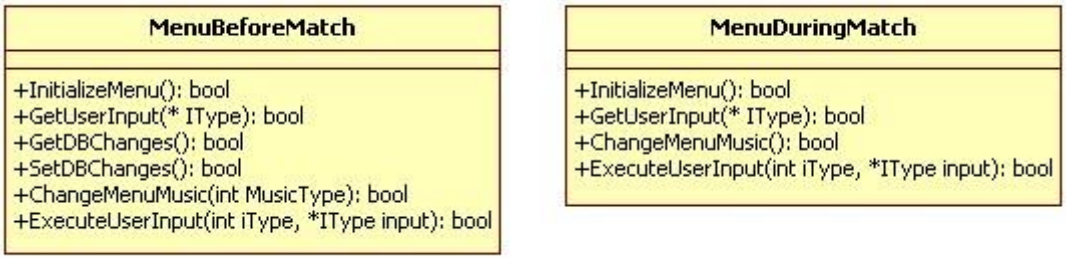


Figure 19 – Game Menu Package

Game Menu package has two classes named MenuBeforeMatch and MenuDuringMatch. MenuBeforeMatch class is in charge of the creation of the menu which shows up when the game (Attention: not the match.) starts. According to the actions that user do on this menu, this class makes related database changes, menu changes, or changing menu music. MenuDuringMatch class is in charge of the creation of the menu which shows up when user stops the match. In this menu, user can make changes in the line-up (substitutions), menu music, and game settings changes (sound, controllers, graphics).

4.4.4 Graphics Package

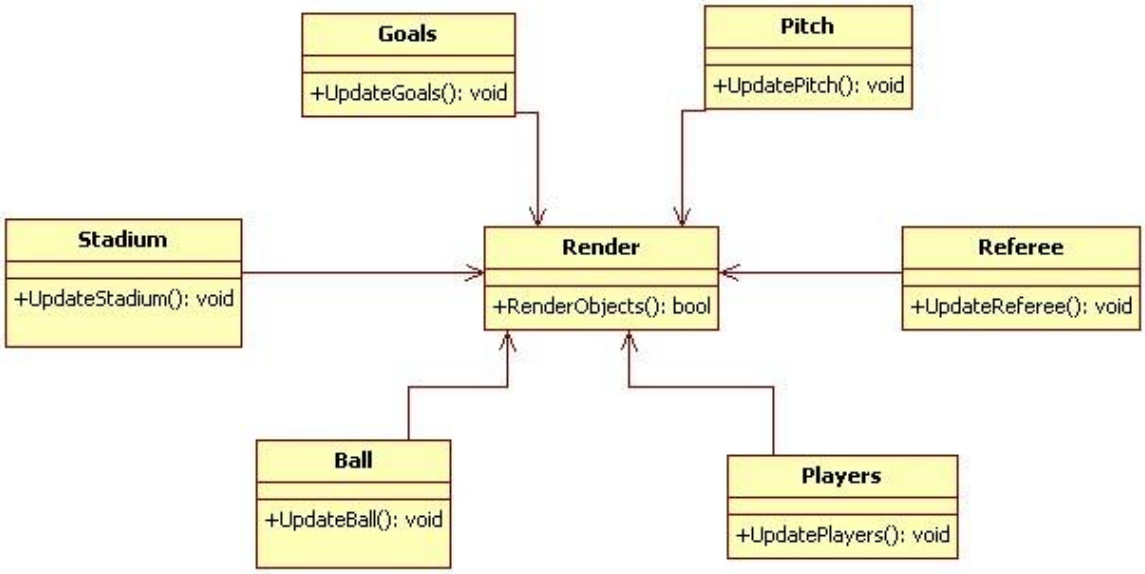


Figure 20 – Graphics Package

Graphics package is in charge of rendering the objects which are in the pitch environment. Classes except Render, update the positions of the objects they represent and then Render class renders all these objects. Render class consists of an implementation of Irrlicht Graphics Engine[1]. The reason of choosing especially Irrlicht Graphics Engine and its features are explained in **Libraries and Tools** section.

4.4.5 Sound Package

Figure 21 – gSound Class

[1] http://www.irrlicht3d.org/

Sound package is in charge of sound works in the game. It acquires the type of the sound from two different modules, as shown in figure 4. Game Engine and Match Engine classes send which sound should be played in that particular state, and then PlaySound method carries out it. This package consists of the implementation of IrrKlang[2] library which is an extension library of Irrlicht and it is quite consistent with it. The reason of choosing especially IrrKlang and its features are explained in **Libraries and Tools** section.

4.4.6 I/O Package

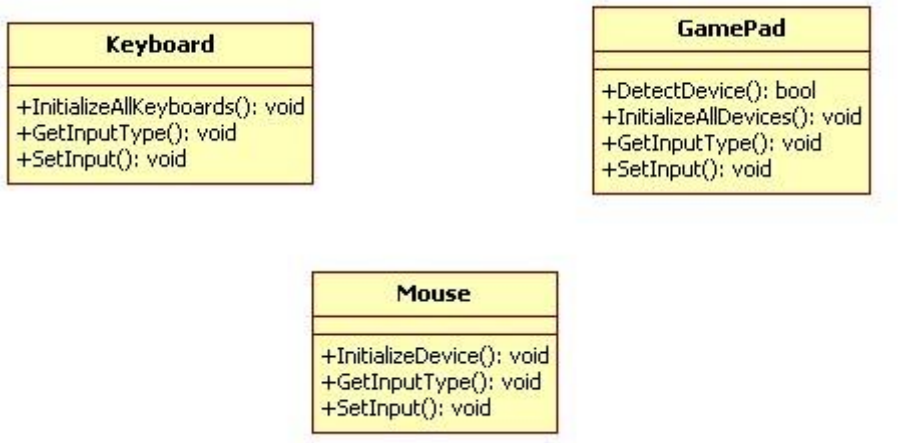


Figure 22 – I/O Package

I/O package is in charge of getting user inputs and making them ready to be used by the Game Engine and the Match Engine packages. The processes of getting user inputs are held by methods of Irrlicht Engine. The user(s) can use the mouse, keyboard and also the game pad while roaming the menus of the game. They can use the keyboard and/or the game pad during a match.

[2] http://www.ambiera.com/irrklang/

4.4.7 Network Package

Figure 23 – Network Package

Network package is in charge of being able to play the game via network. A specific implementation of the RakNet Networking Engine[3] is used in these classes. The reason of choosing especially RakNet Networking Engine for network support and features of it are explained in **Libraries and Tools** section.

[3] http://www.jenkinssoftware.com/raknet/index.html

4.5 Activity Diagrams

The user activities before the match begins, in other words interactions with the GUI, are shown within an activity diagram in figure 24.

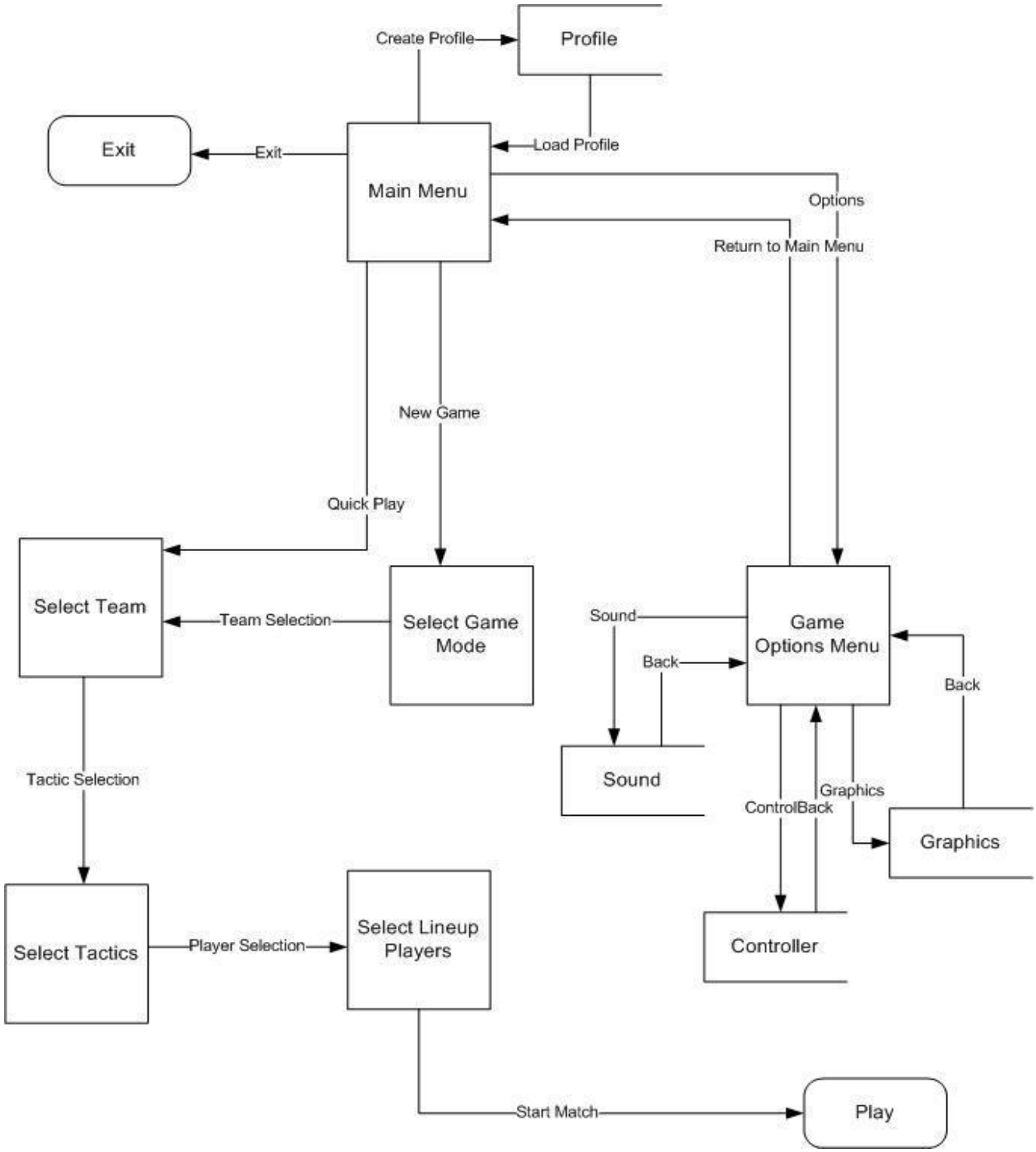


Figure 24 – Activity Diagram

4.6 Sequence Diagrams

In this part of the report, the logic and flow of operations are explained using class methods.

4.6.1 Game Menu Diagram

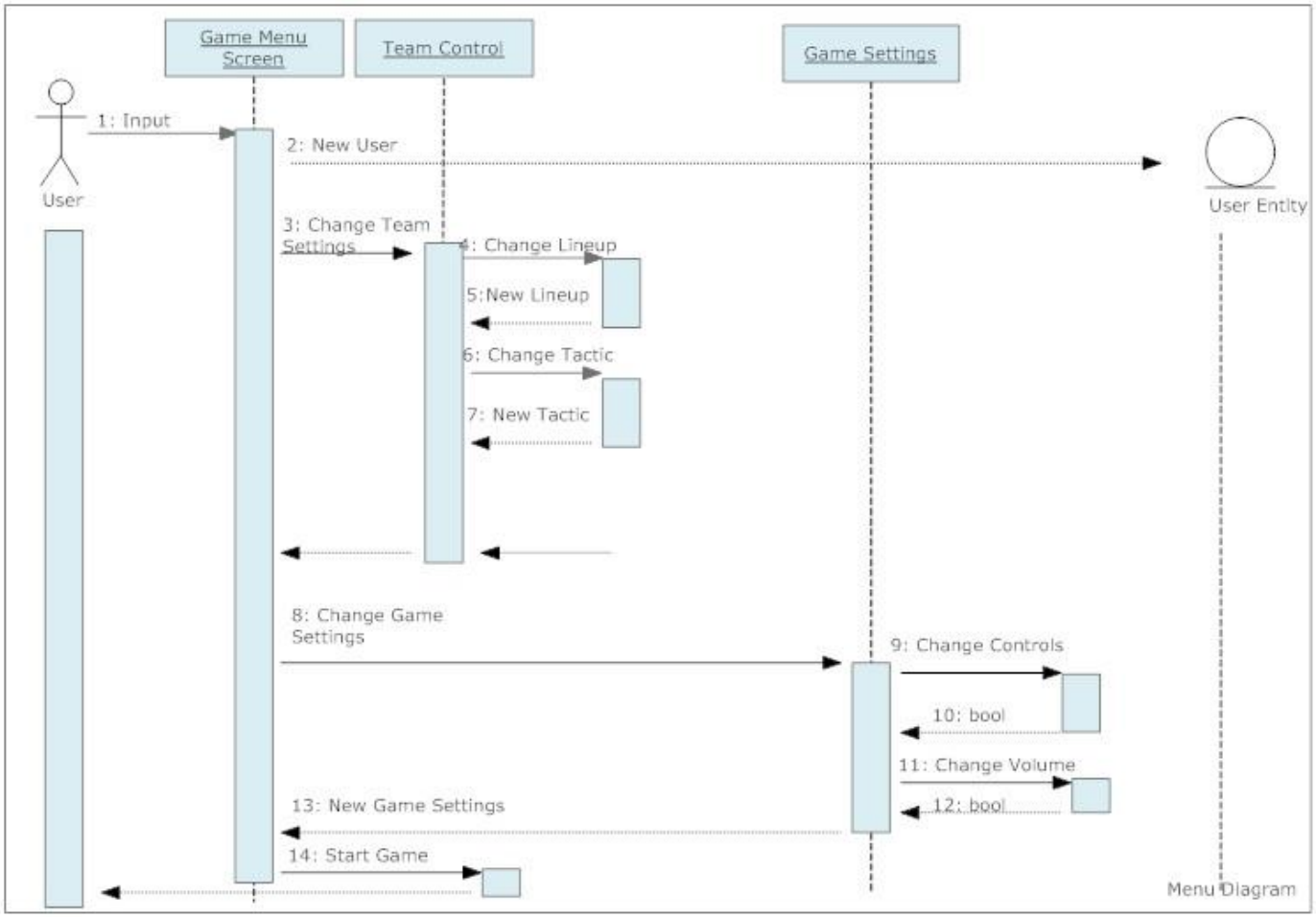


Figure 25 – Game Menu Diagram

4.6.2 Match Diagram

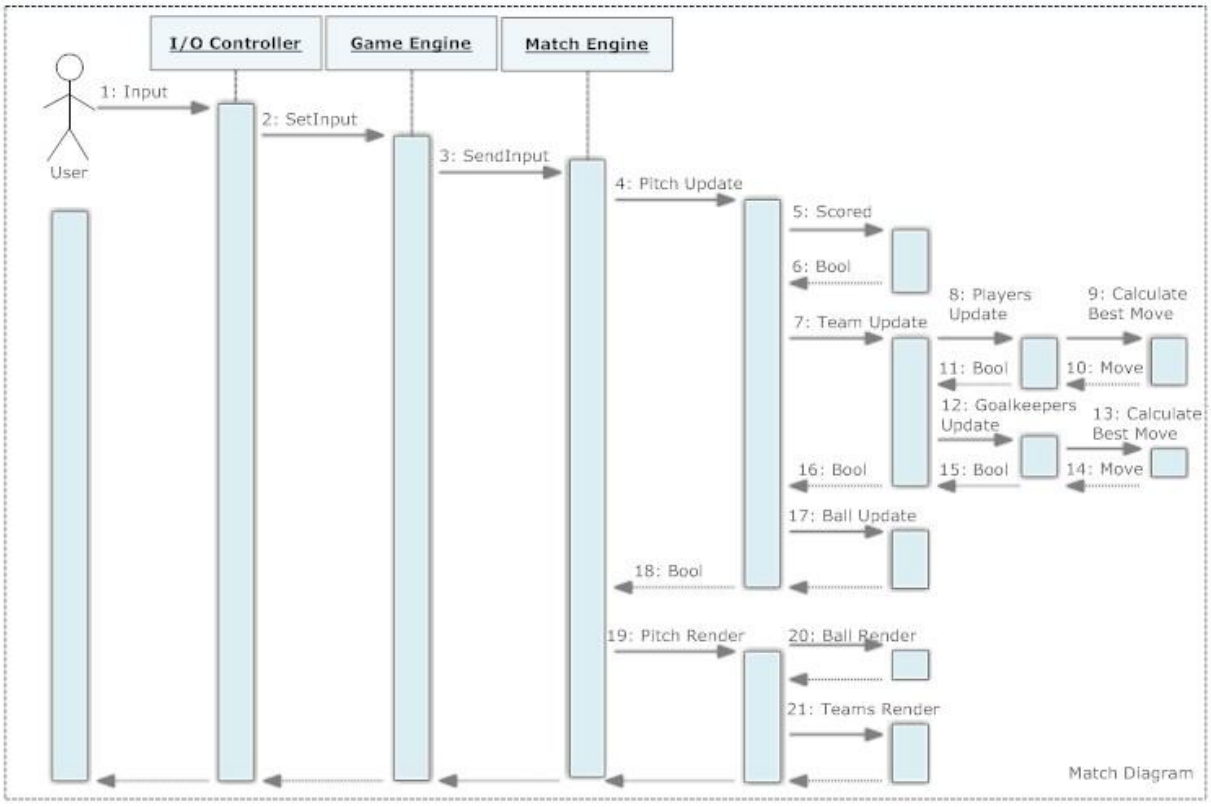


Figure 26 – Match Diagram

5. Interface Design

5.1 User Interface

Figure 27 - User Interface (Main)

This is the screen appearing when LES first starts. The interface with the user in LES is provided via two different tools one of which is a simple but a fast toolbar. The second tool for the interface is a menu made up from click-down buttons.

5.1.1 Toolbar

Figure 28 – Toolbar

Toolbar in a football game, actually in most games, is not a common approach for the interface. However, it happens to be way faster than a regular menu consisting of buttons through which the user is supposed to go to a leaf level when the whole menu is considered as a tree. The toolbar has four tabs: 'File', 'Settings', 'Game Modes' and 'Help' each of which are to be explained in detail in regards of their submenus and functions.

5.1.1.1 File

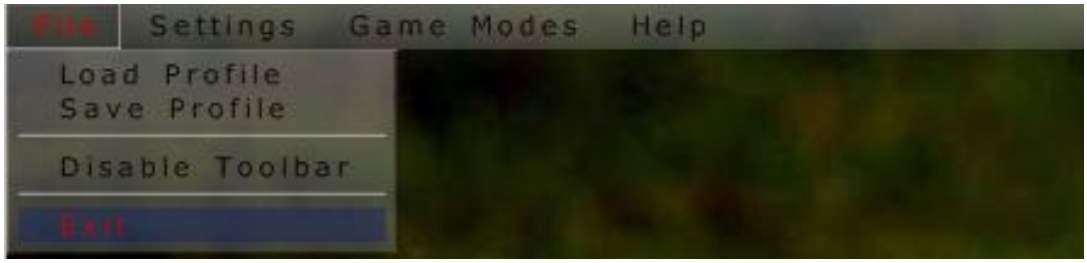


Figure 29 – File

File tab is typically where the user can load a profile (a profile is the set of graphics, sound, controller settings) which he/she saved earlier. There is a submenu in File tab which can close this very tool bar, namely 'Disable Toolbar, which closes the toolbar in case the user wishes. Finally, 'Exit' obviously terminates the execution of LES returning to the running operating system behind.

5.1.1.2 Settings

Figure 30 – Settings

'Settings' is the tab in which the user can alter the game preferences. Those include either using a window or full screen when running LES. There are three more submenus which are to be used to alter the graphical, sound and controller (keyboard and/or game pad) configurations. The default is the full screen which means when LES is first run, which means no profile exists including the option of 'Windowed' choice, it starts in full screen and may be altered later. Next subsections explain the four submenus functions and their submenus in detail.

5.1.1.2.1 Resolution

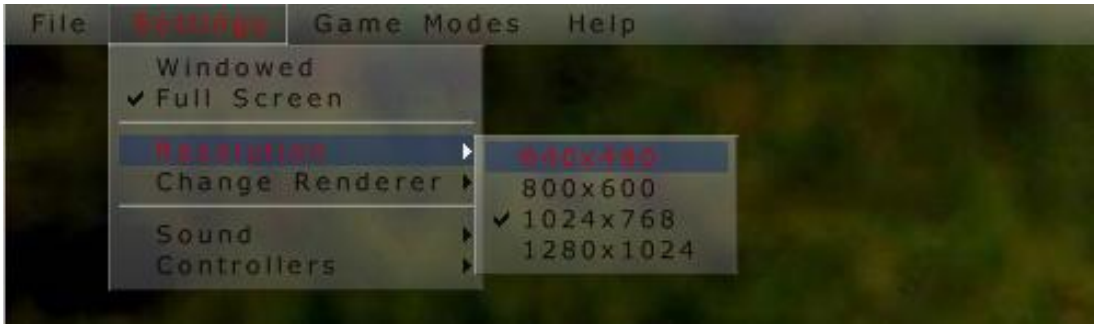


Figure 31 – Resolution

'Resolution' submenu of 'Settings' is where the user can alter the resolution LES runs in. There are four of them: 640x480, 800x600, 1024x768, and 1280x1024. The default one is 1024x768.

5.1.1.2.2 Change Renderer

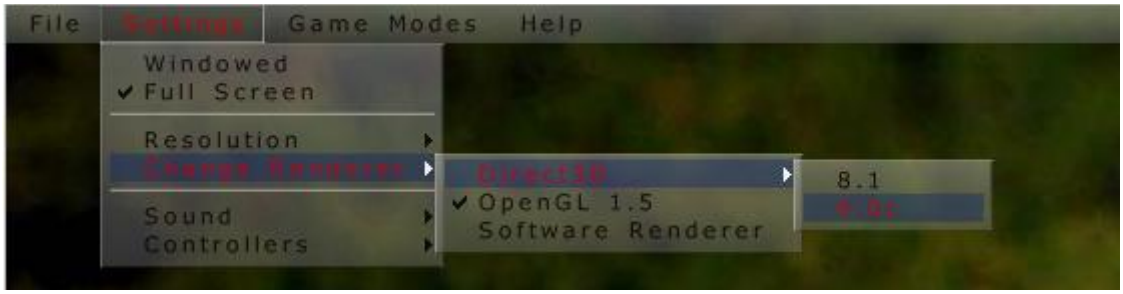


Figure 32 - Change Renderer

Irrlicht Engine provides couple of renderer options, three of them: Direct3D in either 8.1 or 9.0c version, OpenGL 1.5 and also Software Renderer for computers lacking a qualifying video card. Renderer is to be altered in case the user wishes so before a match starts or the user is to be forced to restart his/her match since during a match is being played the rendering options of Irrlicht may not be altered.

5.1.1.2.3 Sound

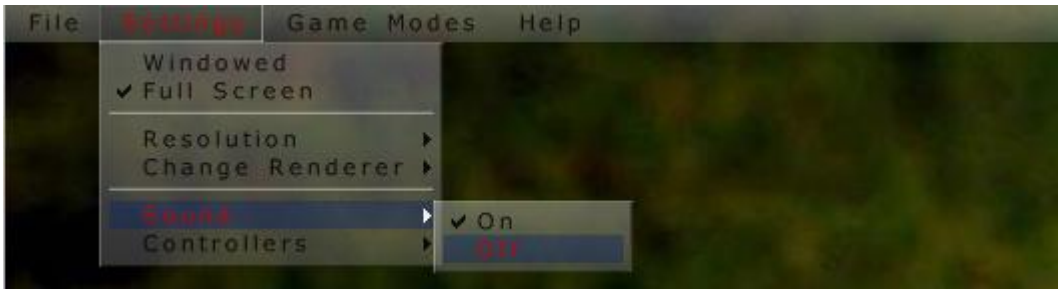


Figure 33 – Sound

Another submenu in the Settings tab is ‘Sound’ which may either set to ‘On’ or ‘Off’. When it is on, the menu tracks and button sounds, and all in game sounds are played. When it is off, no sound is to be heard from LES. The default option of sound is on.

5.1.1.2.4 Controllers

Figure 34 – Controllers

This submenu of Settings tab is where the input device(s) from the user side is/are to be chosen. There are two of them, namely game pad and keyboard. When LES is first installed the default option is only the keyboard chosen as user I/O. Obviously to be able to control a team in a match at least one of those devices are to be marked to be used in the game. Choosing one is done by first clicking on it which is followed by a small window opened including the button configurations for the game controls.

5.1.1.3 Game Modes

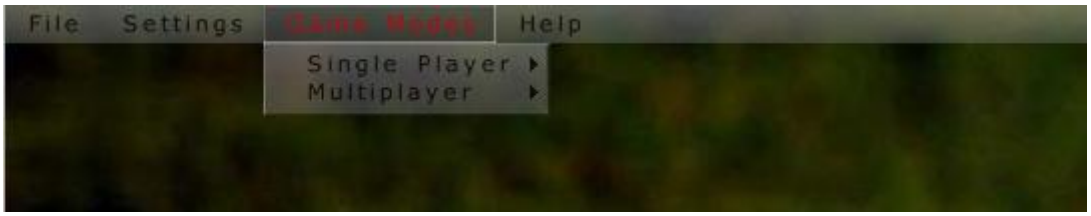


Figure 35 - Game Modes

The 'Game Modes' tab is where the user can choose either a single player game (played by one user) or a multiplayer game (two users). Both have submenus which are to be described below.

5.1.1.3.1 Single Player

Figure 36 - Single Player

Single player option includes three submenus which are ‘Career’, ‘Exhibition’, and ‘Quick Match’. The career mode is where the user can play a tournament or league scenario in the long run by loading a career file which is saved automatically after every match is over in one of the three career modes. Those career modes are ‘League’ in which the user chooses a club team among several countries and play for some seasons with his particular team, most likely. Also World Cup and UEFA Champions League careers are available. Exhibition mode is the mode where the user is directed to a buttons menu below to choose a team of all the teams including national teams and clubs and formations menu afterwards to play just a single friendly match. Lastly, the ‘Quick Match’ directly asks the user to choose a team to and when it is chosen the match starts immediately. This option is for the users wishing to play a game and then switch back to whatever he/she is doing.

5.1.1.3.2 Multiplayer

Figure 37 – Multiplayer

The multiplayer option presents the user two options to play someone else. The user either plays another one on the same computer by a game pad which corresponds to the '2 Players' option or both users connect to a network through a local area network or the internet to play LES each other.

5.1.1.4 Help

Figure 38 – Help

The 'Help' tab includes five submenus which are 'Help!', "Les vX.X", "About Kontrpiye", "Hints and Tricks", and "Check for Updates". Choosing help opens the FAQ file which includes some usual problems faced while running LES and their solutions for the users having trouble. The second one shows the version information of LES including the major upgrades and changes since the last version. About Kontrpiye shows some brief information of the developer team, Kontrpiye. Hints and Tricks part shows the user some points provided by the developers to exploit LES in couple of in game aspects. Check For Updates automatically goes to the website of LES and checks if a newer version is released and tells user the result. In case there is a newer version the user is asked to whether download the newer one or not.

5.1.2 Menu

5.1.2.1 Main Menu

Figure 39 - Main Menu

The main menu is the one when LES starts up. The main menu basically consists of five buttons each of whose submenus and functions are to be explained below. Those are 'Quick Match', 'Game Modes', 'Settings', 'Credits', and 'Exit' buttons.

5.1.2.1.1 Quick Match

This 'Quick Match' is the same of the one in the toolbar menu which is under 'Game Modes'->'Single Player'. It brings a team selection menu in after which the match is started directly, no waste of time by making formations etc.

5.1.2.1.2 Game Modes

Figure 40 - Game Modes

There are two game modes (actually three with quick match). Notice that 'Quick Match' option is rather in the main menu unlike it was in the 'Game Modes' tab in the toolbar. Since it is supposed to be quick it appears in the highest menu level here but not in game modes. Single player is offline playing in couple of options whereas multiplayer is either offline or online. Those are explained in subsequent subsections. Back button just returns to the upper menu which is Game Modes here.

5.1.2.1.2.1 Single Player Menu

Figure 41 - Single Player Menu

Single player menu has 'Quick Play' again. It is here in single player menu although it was also in the main menu. This is just for the consideration of quick play may be needed here as well after a menu without it (game modes menu) while traversing the 'menu graph'. Others are 'Exhibition' which is a just a friendly game with the user and the computer. Lastly,

'Career Mode' is another option that the user's game play after every match is saved automatically and the very same scenario can be loaded to be continued.

Figure 42 - Career Mode

Career mode has a couple of options in itself too: League, UEFA Cup, and World Cup. League button releases a scroll bar from which a league is selected, and afterwards a team is selected from the corresponding league leading to the match menu.

5.1.2.1.2.1' Last Screen

Figure 43 - Last Screen

After the team(s) is/are selected in quick match, exhibition or multiplayer the last options for the match and the pitch are to be altered here. There are some of weather types (e.g. sunny, rainy etc) which affect the ball physics and the performance of the players actually. Different referee selections result in referees looking different than each other actually, and also referees differing in the proportion of the wrong decisions out of the all decisions they take

during the match. So the referee has a quality too. Different ball selections result in different ball physics again also with different colors. Formations menu is for the organization of the team and has its own submenus to be explained in the next section. ‘Start Match’ tells the referee to whistle the kick off. Back returns to the upper menu.

5.1.2.1.2.1’’ Last Formations

Figure 44 - Last Formations

This is the menu in which the user may substitute some players with the reserves (up to three in default). Tactics is the arrangement of the players on the pitch such as 4-4-2, 3-5-2, 4-5-1 etc. Free kicks button lets the user assign a player for the free kicks to be turned into the game during the match. Lastly Go To Match starts the match.

5.1.3 Settings

Figure 45 – Settings

This menu includes ‘Sound’, ‘Graphics’, and ‘Controllers’ configurations. The volume of the sound is arranged with the scroll bar followed by the sound button. Graphics preferences are resolutions and rendering options. Altering rendering options during a match (in the pause menu or half time) needs LES to be reloaded because of the Irrlicht Engine’s behavior. Controllers are again keyboard and game pad(s) if there any. The button configurations are to be altered in the following screen which is not shown here.

5.1.4 Credits

This is just a button results in some contact information and some quotes from developer team, Kontrpiye. Other than that it has no uses.

5.1.5 Exit

‘Exit’ terminates LES. Any unsaved setting (to a profile) results in a query before returning to operating system asking if the user is sure about quitting before saving the last settings of his/her to a profile.

6. Division of Labor

In LES there exist eight modules to be implemented. These modules are distributed among team members in such a way that each member takes a module he is interested in and really enjoys while developing it. Also, our online communication channel[4] makes a team member able to be aware of all modules' development stages even if he is not in charge of those modules' implementations. Figure 46 clearly shows in which modules a team member works.

	Berker Batur	Uğur Büyükköy	Ufuk Dallı	Hakan Çağlar
Game Engine	X	X		
Match Engine	X		X	
Sound	X	X	X	
Graphics	X	X		X
Game Menu	X	X	X	
Network	X	X	X	X
I/O	X	X	X	X
Database			X	

Figure 46 - Division of Labor

[4] https://projects.zoho.com/portal/kontrpiye/

7. Libraries and Tools

Graphics: In graphics module, we use Irrlicht Graphics Engine[5]. Irrlicht Graphics Engine supports high performance real time 3D rendering using Direct3D and OpenGL. The reasons why this engine was chosen to be used are mainly: it is platform independent, written in C++ (the PL in which LES is purely implemented), easy to understand, has well-documented API, and has direct import of many mesh formats. Moreover, Irrlicht is widely used among game developers and has active forums.

I/O: As mentioned earlier, getting user input from keyboard, game pad and mouse is accomplished by using Irrlicht Graphics Engine's support in LES. Initializing an object which is driven from irr::IEventReceiver[6] is what is actually needed for I/O. Overriding irr::IEventReceiver::OnEvent() method of this object enables one to able to get input from I/O devices connected. This method is called by the engine once when an event happens. The reasons of choosing Irrlicht Support for I/O are that there are many manuals and examples on the internet available and it cannot be much complex to implement since implementation with Irrlicht in Graphics Module is already there. Irrlicht does support gamepads as well and for this reason no external library is necessary other than Irrlicht itself.

Sound: In sound implementation, Irrklang[7] library is used. Reasons it is preferred are that it was written in C and works in C++ naturally, it is a cross-platform library, there are many examples on internet that implements Irrklang's sound methods.

[5] http://www.irrlicht3d.org/
[6] http://irrlicht.sourceforge.net/docu/classirr_1_1_i_event_receiver.html/
[7] http://www.ambiera.com/irrklang/

Network: RakNet[8] is the network support of this project. RakNet is a cross-platform C++ game networking engine. It is designed to be a high performance, easy to integrate, and supporting complete solutions for games, and these are the reasons why RakNet is the one to be implemented in the network module. Moreover there are useful examples on the internet, which implements Irrlicht with RakNet to develop network games which help a lot.

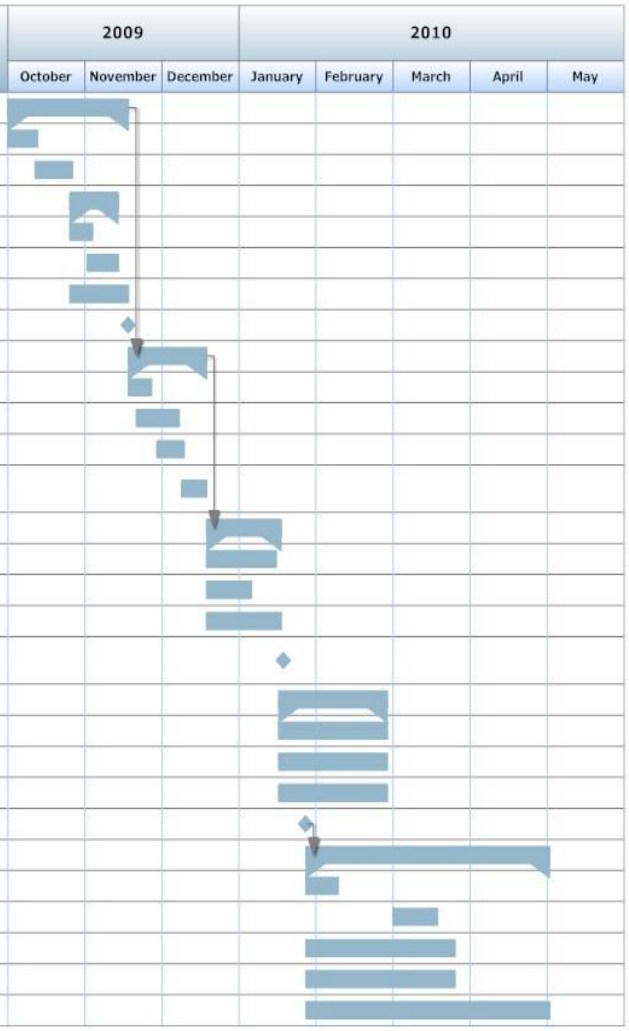
Database: SQLite[9] is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is an embedded SQL database engine and unlike most other SQL databases, it also does not have a separate server process. It just directly reads from or writes into ordinary disk files. Since LES database is pretty small and it is cross-platform (Linux, Windows and more), using SQLite as the SQL database is the best choice. It is written in C itself and has a good C/C++ API with good documentation. The biggest factor is that SQLite is light.

[8] http://www.jenkinssoftware.com/raknet/index.html/
[9] http://www.sqlite.org/

8. Project Schedule

The roadmap of the LES project is illustrated as a Gantt chart below.

Number	Task	Start	End	Duration (days)
1	Analysis	1/10/2009	17/11/2009	34
1.1	Topic Selection	1/10/2009	12/10/2009	8
1.2	Proposal	12/10/2009	26/10/2009	11
1.3	Field Research	26/10/2009	13/11/2009	15
1.3.1	Market Research	26/10/2009	3/11/2009	7
1.3.2	Technology Research	2/11/2009	13/11/2009	10
1.4	Requirement Analysis	26/10/2009	17/11/2009	17
1.5	Milestone (SRS)	17/11/2009	17/11/2009	1
2	Initial Design	18/11/2009	18/12/2009	23
2.1	Components	18/11/2009	26/11/2009	7
2.2	Interfaces	21/11/2009	7/12/2009	11
2.3	Data Specifications	29/11/2009	9/12/2009	8
2.4	Milestone (Initial Design Report)	9/12/2009	18/12/2009	8
3	Detailed Design	19/12/2009	17/1/2010	20
3.1	Database Design	19/12/2009	15/1/2010	20
3.2	Class Hierarchy	19/12/2009	5/1/2010	12
3.3	AI Design	19/12/2009	17/1/2010	20
3.4	Milestone (Detailed Design Report)	18/1/2010	18/1/2010	
4	Implementation	17/1/2010	26/2/2010	30
4.1	Graphic Implementation	17/1/2010	26/2/2010	30
4.2	AI Implementation	17/1/2010	26/2/2010	30
4.3	I/O Implementation	17/1/2010	26/2/2010	30
4.4	Milestone (Prototype)	27/1/2010	27/1/2010	1
5	Further Implementation	28/1/2010	1/5/2010	67
5.1	Graphics Improvement	28/1/2010	8/2/2010	8
5.2	Network Implementation	1/3/2010	18/3/2010	14
5.3	AI Improvement	28/1/2010	25/3/2010	41
5.4	Database Improvement	28/1/2010	25/3/2010	41
5.5	Sound Integration	28/1/2010	1/5/2010	67



9. References

- Software Engineering a Practitioner's Approach, 5th Edition, Roger S. Pressman
- Component Oriented Software Engineering, Ali H. Doğru
- http://www.agilemodeling.com/essays/umlDiagrams.htm
- http://irrlicht3d.org/
- http://www.ambiera.com/irrklang
- http://www.jenkinssoftware.com/raknet/index.html
- http://www.sqlite.org