KONTRPİYE

INITIAL 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 initial design report of Linux Evolution Soccer project documented by the developer team Kontrpiye, the senior project team of four METU Computer Engineering students. It includes initial design concepts and descriptions of the proposed software system design of the project LES and the software development methods & strategies underlying.

1.2 Problem and Motivation

Linux users have been suffering from the nonexistence of a prevailing football game since time immemorial. This is undeniably a handicap for Linux environments and also Linux operating system users. Provided our game, no Linux users will need to have other operating systems to be able to play qualified football games on their computers. As Kontrpiye, we believe when more and more qualified games come up into the Linux environments, also the range of Linux operating systems over all the

computer users among all the world will raise accordingly. The access to the game is also a quality that we assure since LES will be available on the internet for free to be downloaded.

We also consider ourselves sort of innovative since our focus is not on developing a game looking real like 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.

1.3 Project Definition and Goals

As it is pretty much implied in the name, Linux Evolution Soccer is a football game for Linux operating systems. Specifically, it is a 3D real-time football game resembling today's popular football games Pro Evolution Soccer, Sensible Soccer etc. Computer players will get to be able to play a really 'fun to play' football game on their Linux operating systems ultimately. A football game which can be played in single player mode and also in multi-player mode by either multi pads on the same computer (keypad, joy pad, game pad) or through a network. 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 of the project. LES aims to offer solutions

for the lack of fun in the games in Linux systems still standing in the market. This will be accomplished via rapid prototyping to be described in the second chapter.

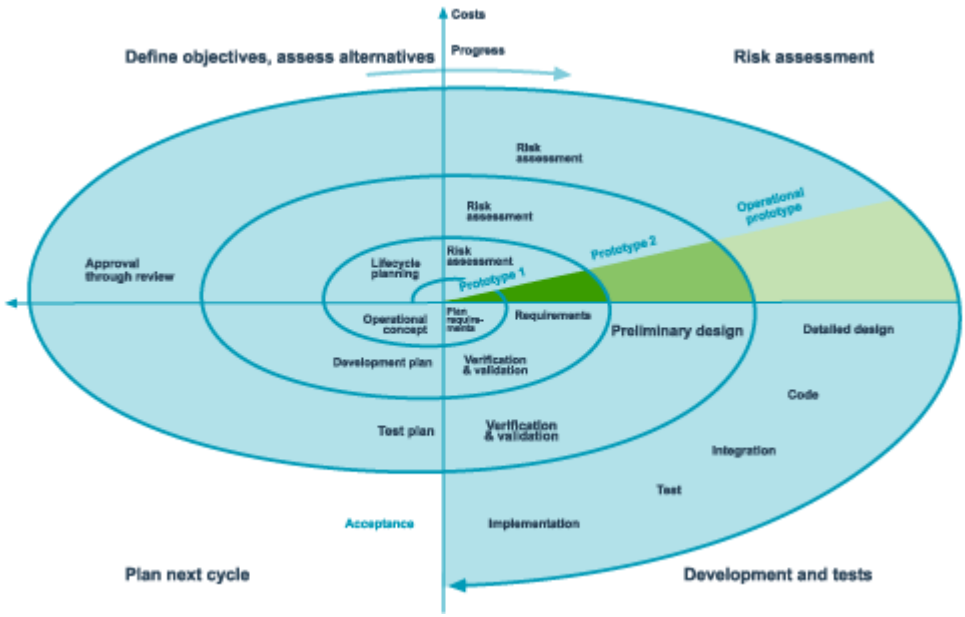
Offline playing support should be a big attribution in the sense of the design of the artificial intelligence, hopefully ending up in more and more number of potential users obviously. Existing football games do not provide much fun when it is human versus computer, because the artificial intelligence embedded is not qualified.

The users will also be provided an entertaining challenge with LES. The primary goal is to keep the fun aspect at the top level since our motivation contributes to keep the Linux users really enjoy the game. The game will provide a challenge to the user to achieve several achievements in the game in the single player mode; also the user will be able to challenge others through the game in the multi-player mode via the network they belong or the internet.

2 Design Methodologies

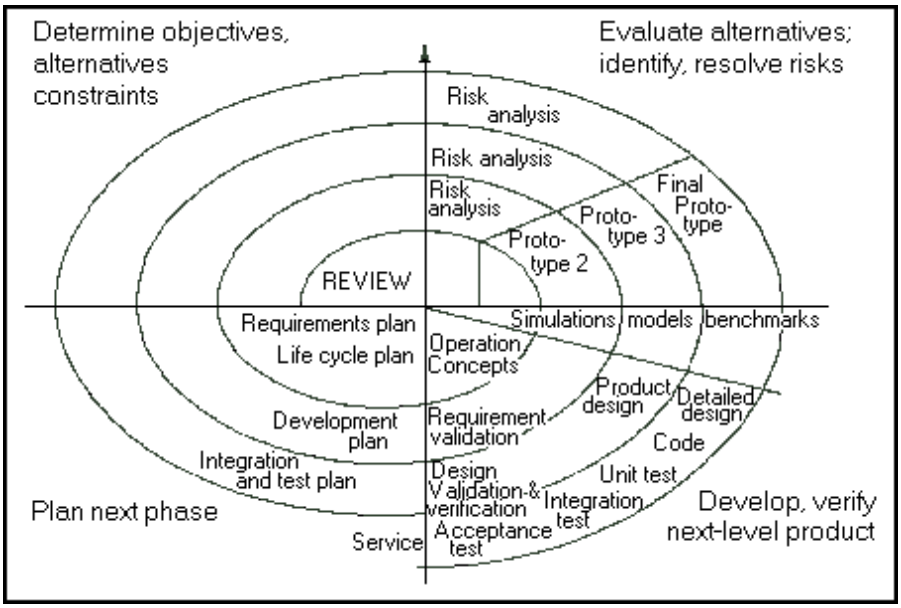
Since this is a football game and as developers we not being so experienced at developing games, prototyping is the first of the choices coming up as the process model. Interactive prototypes are to be developed which can be quickly replaced or changed in line with design feedback coming from the users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

Prototyping will be helping in resolving uncertainty about how the design of the game could be better in respect of user's needs. This will be accomplished by obtaining information from users about the necessary 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.



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, because it is not an easy job when your aim is to keep users playing a game having fun for a considerable amount of time. It is just obligatory to have their opinion on every single matter. That is why rapid prototyping is the best solution here.

Appropriate users will be testing the prototypes trying to cover all the ranges of the corresponding design. Users will also be supplied with former information that they are 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.

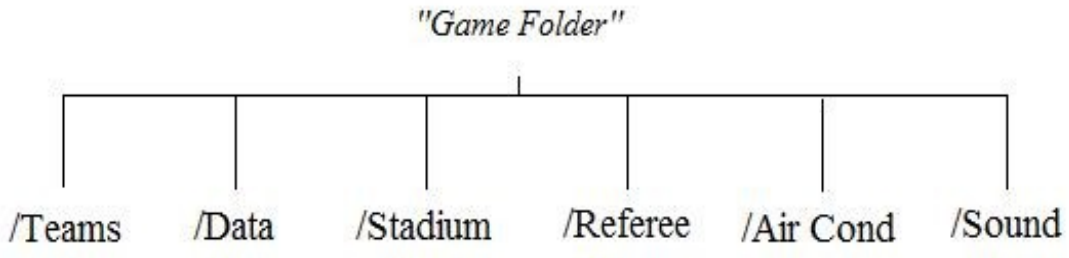


While using this method, creating too polished and looking already finished prototypes may result in too much satisfied testers also who are not capable of telling what is really missing. To exemplify, too fast responses in the game like starting a match instantaneously or too sophisticated graphics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

Briefly, we will be exploiting the benefits of rapid prototyping mostly in respect of more accurate understanding of user requirements. All in all, rapid prototyping 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 models above.

3 Data Design

In the data design of the project our team will not use any database management systems. Instead of database tables we will read data from files. These files will include all required data. There will be six main folders which will be consisting of inner folders and data files inside. Here is the main hierarchy of our folders:

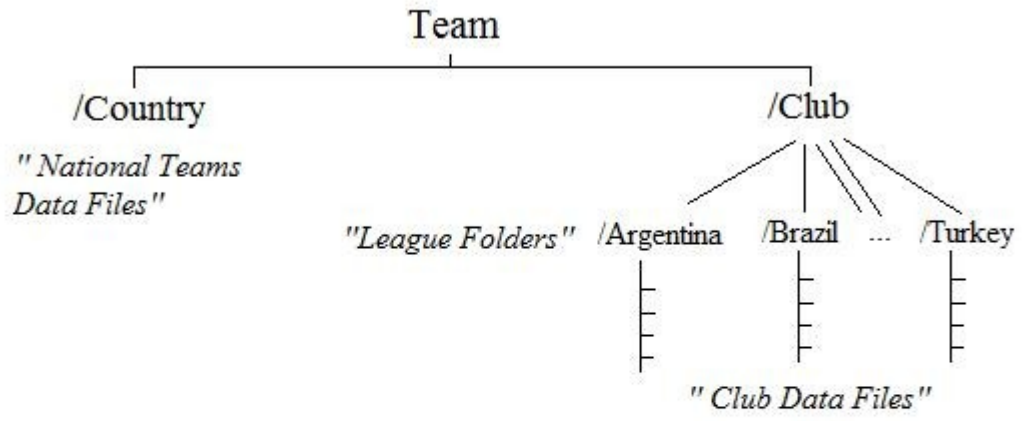


All data files at the same time they are text files, sound files or pictures etc, will be leafs of our system shown above. The data needed in game play will be taken from these files to memory, while loading the game. None of these files will be editable through the game play that if a user want to transfer a football player from a team to his/her team, he/she will be able to transfer player after choosing team and the team's data file is loaded to memory.

In the sub-sections of this part we are going to introduce each of six main folders and its contents.

3.1 Teams

Structures of the text files under the Team folder and its sub-folders will be the same. These text files will include anything about the team such as full name of the team, country of the team, control string whether it is a national team or a football club from a league. All team properties needed in game can be found in files. Below hierarchy of the teams folder can be seen.



At the leaf level of the hierarchy “National Teams Data Files” and “Club Data Files” will take place. The structure of these files will be as following:

Team Properties	Explanations

Team Name	This field corresponds to the name of the team. (i.e. Barcelona, Benfica)
Country	This field corresponds to the country of the team. The team name itself if it is a national team. (i.e. Brazil, Turkey)
Type	This field corresponds to the type of the team. (i.e. Brazil is type of national, Barcelona is type of club)
Home Color	This field corresponds to the home color of the team strips. (i.e. black-white, red-yellow)
Away Color	This field corresponds to the away color of the team strips. (i.e. plain white, red-black)
Formation	This field corresponds to the formation of the team.(i.e. 4-4-2, 3-5-2)
Rating	This field corresponds to the rating of the team among all other teams. (i.e. Relatively high rating for strong teams)

Player Properties	Explanation
ID	This field corresponds to the unique ID of a player. The player ID has two characters due to the team he plays. This means already existing players in club teams have different records in national teams, so different

	attributes to be able to become better/worse when playing in his national team.(i.e. BR032, GS417)
Shirt Number	This field corresponds to the player number which is unique among a specific team. (i.e. 9,5 and 1 for a GK)
Name	This field corresponds to the name of the player. (i.e. Lionel, Arda)
Surname	This field corresponds to the surname of the player. (i.e. Messi, Turan)
Shirt Name	This field corresponds to the name written on the back of the player's shirt. This field is equal to the name or the surname field of the player most of the time. But for some players shirt name will differ from both (i.e. Messi, Balta)
Skin Color	This field corresponds to the skin color of the player with enumeration. (i.e. 3 for African, 5 for blonde)
Position	This field corresponds to the main position of the player among the field. (i.e. AMC, GK)

Player Attributes	Explanation
Technique	This field corresponds to the technique attribute of the player.
Acceleration	This field corresponds to the acceleration attribute of the player.
Strength	This field corresponds to the strength attribute of the player.
Goalkeeping	This field corresponds to the goalkeeping attribute of the player (Pretty high on goalkeepers rather than field players).
Tackling	This field corresponds to the tackling attribute of the player.
Passing	This field corresponds to the passing attribute of the player.

Shooting	This field corresponds to the shooting attribute of the player.
Speed	This field corresponds to the speed attribute of the player.
Stamina	This field corresponds to the stamina attribute of the player. Every player has a default stamina which varies during the game. Default is kept here.

3.2Stadium

There exists a text file that consists of names of stadiums and names of the corresponding folder s.

Stadium Properties	Explanation
Stadium Name	Name of the stadium
Folder Name	Name of the folder which includes stadium files, images and sprites.

3.3 Referee

Referee folder includes only text files.

Referee Properties	Explanation
ID	This field corresponds to the unique ID of a referee. The referee ID has two characters due to the country he is from. (i.e. IT05,TR42)
Name	This field corresponds to the name of the referee. (i.e. Selçuk)
Surname	This field corresponds to the surname of the referee. (i.e. Dereli)
Skin Color	This field corresponds to the skin color of the referee with enumeration. (i.e. 3 for African, 5 for blonde)
Intelligence	This field corresponds to the percentage of the right decisions out of all decisions of the referee.

3.4 Air Conditions

Referee folder includes only text files. Sound and image files about air condition will be in another folder.

Air Condition	Explanation
Sunny	There will be a sunny air file holding the values for sunny air conditions. (i.e. friction constant)
Rainy	There will be a rainy air file holding the values for rainy air conditions. (i.e. friction constant)
Snowing	There will be a snowing air file holding the values for snowing air conditions. (i.e. friction constant)

3.5 Sound/Music

This folder contains all sound files to be played during the game.

- Supporter Cheers
- Ball Sounds
- Whistle Sounds
- Player Sounds
- Menu Tracks

3.6 Other Data Files

Miscellaneous data files are kept in this folder such as interface background pictures, button images, saved profile files etc.

4 Architectural Design

4.1 Design Overview

4.1.1 Game Design

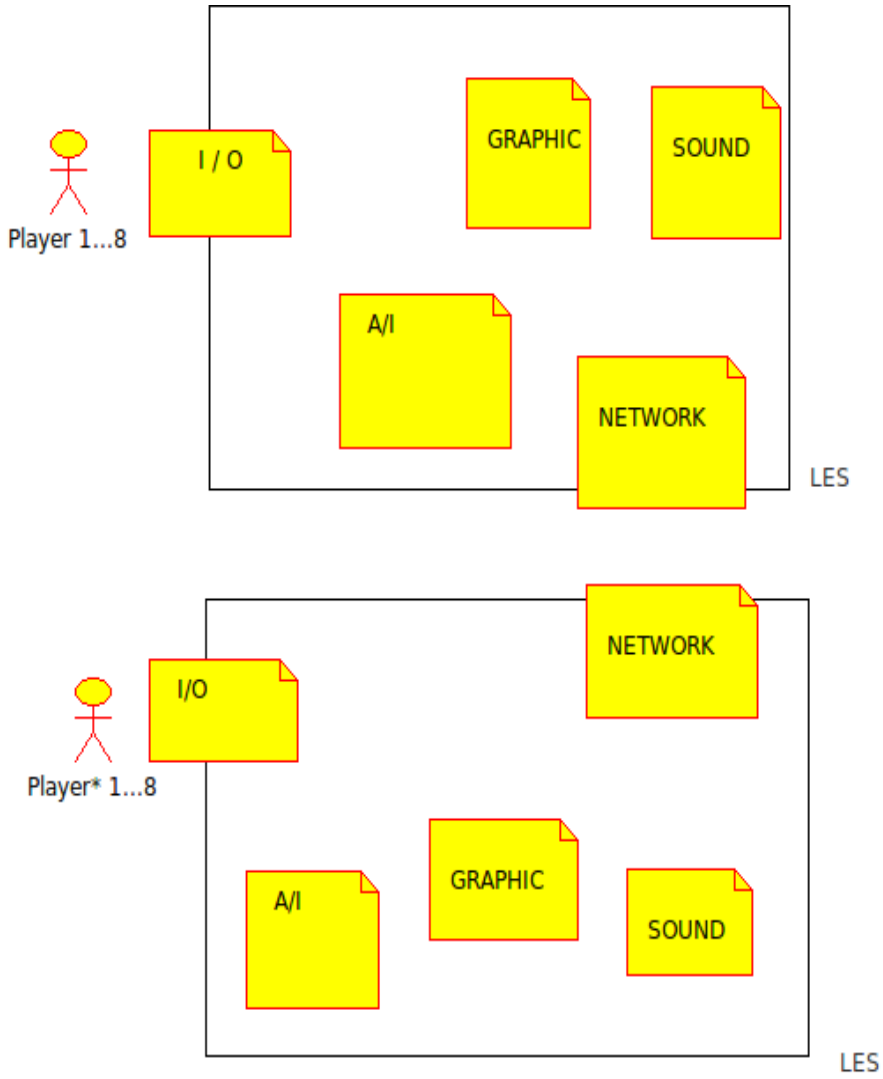


Figure 4.1

LES high level game design is planned to be able to correspond to the needs of the users who will play the game. As it can be seen in the diagram, there are modules that do appropriate jobs in the game. I/O module meets the need of playing the game with keyboard or joystick. Changing control buttons also be allowed in our game. A/I module make user be able to play against computer rather than a multiplayer game. Network module makes our game be able to play online with other users at different places with TCP/IP network.

4.1.2 LES Environment and Rules Design

The rules of the game are uncomplicated. There are two teams; each team contains ten field player and one goalkeeper. A goal is scored by kicking the ball over the opposing team's goal line.

There are throw-ins, corners, goal kicks, penalties, faults and off sides in our game. There is also one referee in the pitch. Users of the game could change the formations of teams, decide line-up players, make substitutions, choose stadium, choose team, choose ball type, and choose air condition. Game environment basically consists of:

- A pitch
- Stadium
- Two goals
- One ball
- Two teams
- Twenty Players
- Two Goal Keepers
- One Referee

4.2 Class Diagrams

4.2.1 High Level Object Hierarchy

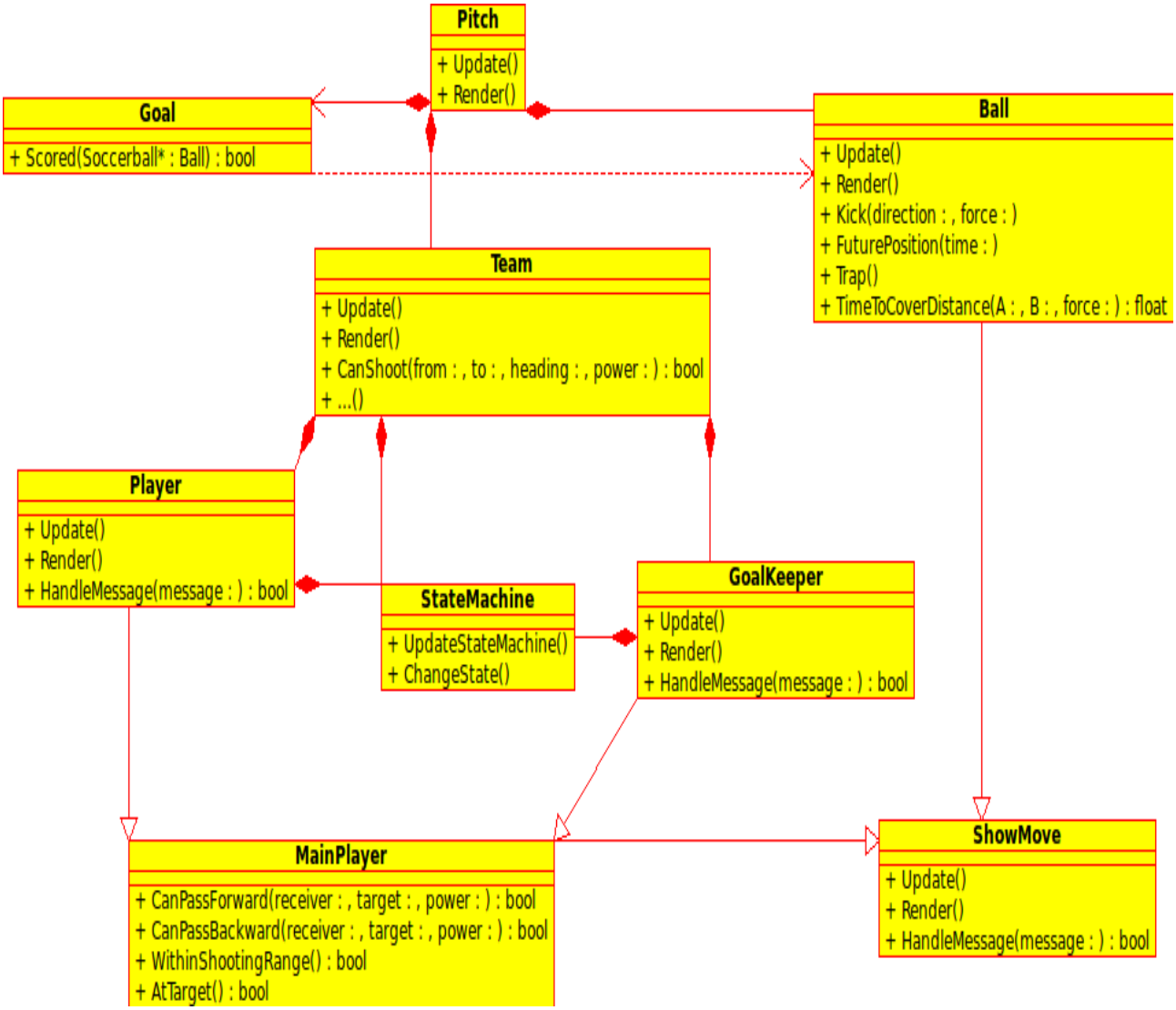


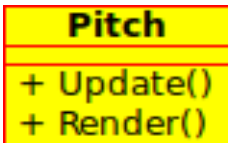
Figure 4.2

Each item type in our game encapsulated as an object. You can see how they are related to each other in the UML class diagram shown in figure 4.2.

4.2.2 Game Engine Classes

Game engine classes at top of the class hierarchy and they are mostly integrated with visible objects. Except from A/I based jobs, all other things will be done by there game engine classes.

4.2.2.1 Pitch Class

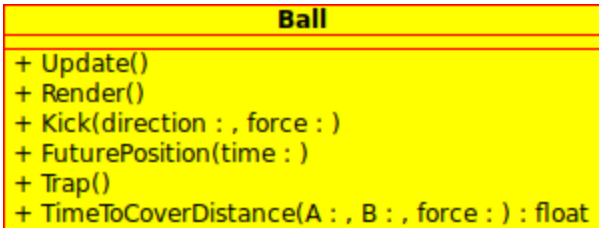


Pitch is a rectangular playing area enclosed by throw-ins and goal kick lines. Playing area is encapsulated by the class Pitch. A single instance of this class will be instantiated when the match begins. The Pitch object owns instances of Team, Ball, and Goal objects. The Pitch::Update and Pitch::Render 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.

4.2.2.2 Goal Class

Goals are defined by left, right, and top goal posts. A goal is scored if any part of the ball crosses the goal line. Each time step, the Scored method of each team's goal is called from within Pitch::Update to know if a goal is detected or not.

4.2.2.3 Ball Class



The data and methods to encapsulate a soccer ball are encoded in the Ball class. A soccer ball moves so its class inherits from the ShowMove and PerformMove classes as shown in figure 4.2. Ball also has data members for recording the ball's last updated position and methods for kicking the ball, testing for collision (only with goals' posts not with players), and calculating the future position of the ball.

4.2.3 Artificial Intelligence Design Class Diagram

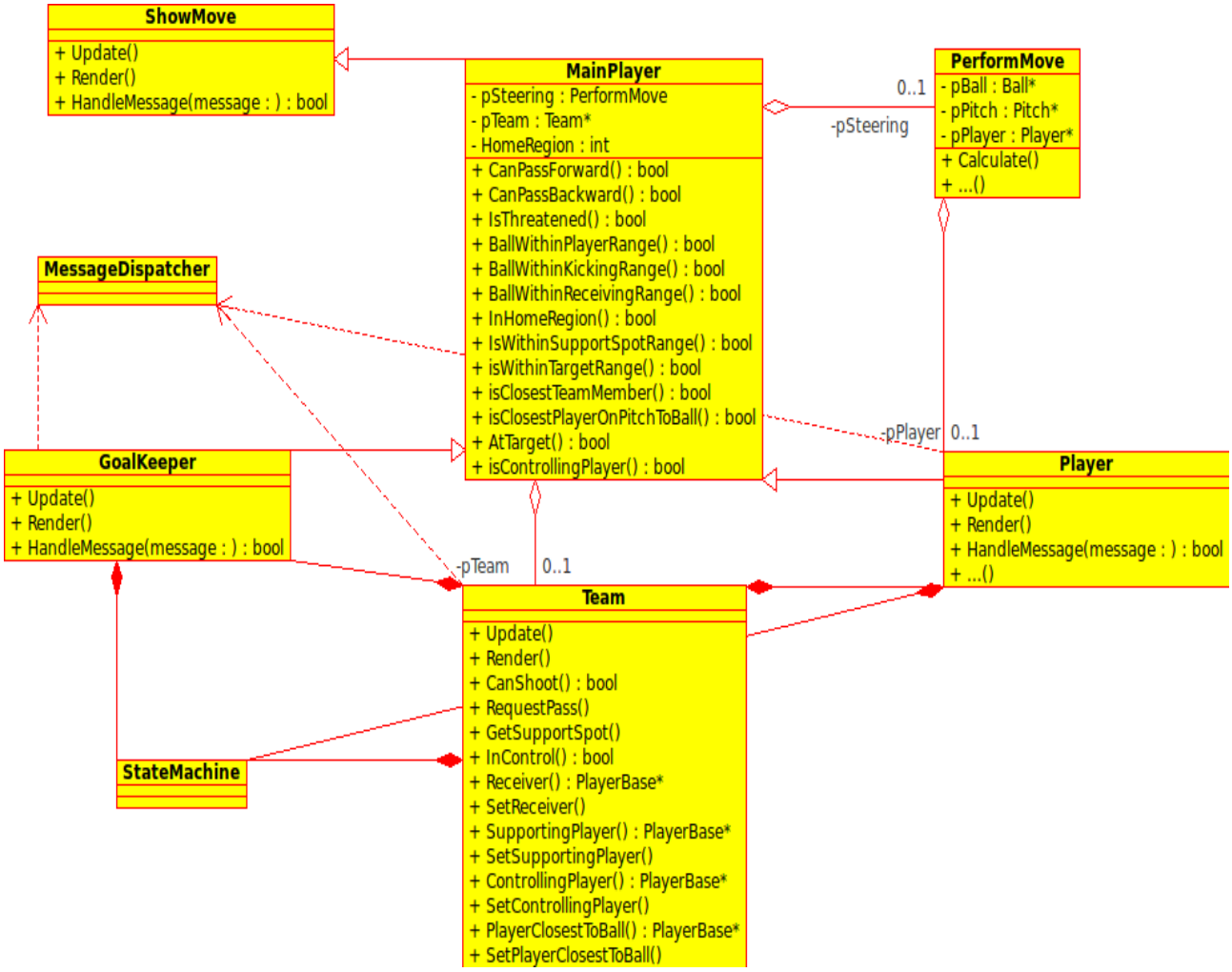


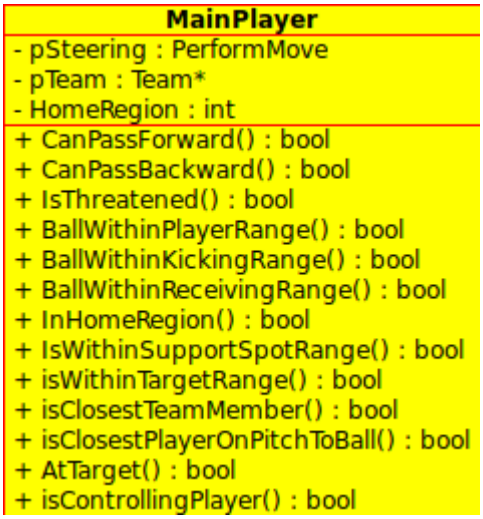
Figure 4.3

As mentioned earlier, soccer players are separated as players and goalkeepers. Both types are derived from the same base class, MainPlayer. Both make use the PerformMove class and both own finite state machines, with their own states.

Teams also own a State Machine, supplying a team, the ability to change its

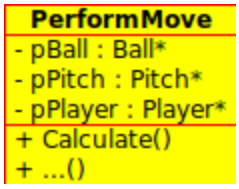
behavior depending on the current state of play. Players and their teams have ability to send messages. Messages may be passed from player to player or from soccer team to player.

4.2.3.1 MainPlayer Class



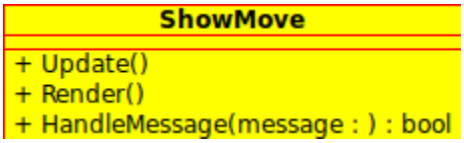
Players, goalkeepers, and teams classes inherit from MainPlayer class. Thanks to the home region attribute, all players in the game have their home regions. Other methods of the MainPlayer class allow a player to choose what to do in the next time step and update its state according to that decision. Detailed explanations that what these methods do, will be discussed at “Detailed Design Phase”.

4.2.3.2 PerformMove Class



PerformMove class's methods perform the appropriate behavior that players choose at that time step. More methods will be added to PerformMove class and they will be discussed at “Detailed Design Phase”.

4.2.3.3 ShowMove Class

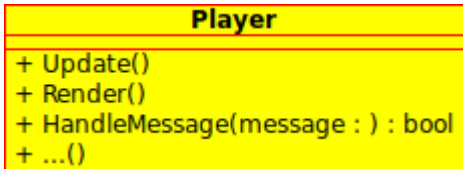


Every object which has a movement in our game derives from ShowMove class and their updated status will be graphically shown thanks to the ShowMove class.

4.2.3.4 Team Class

The Team class owns instances of the players that comprise the soccer teams. It has pointers to the pitch, the opposing 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. Detailed explanations of key players and methods of the Team class will discussed at “Detailed Design Phase”.

4.2.3.5 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. Their motions, roles, and states will be discussed at “Detailed Design Phase”.

4.2.3.6 GoalKeeper Class

Goalkeepers in two teams are instantiated as objects of the GoalKeeper class. In first place goalkeepers assigned to the region that overlaps its team's goal. Their motions and states will be discussed at “Detailed Design Phase”.

4.3 Activity Diagram

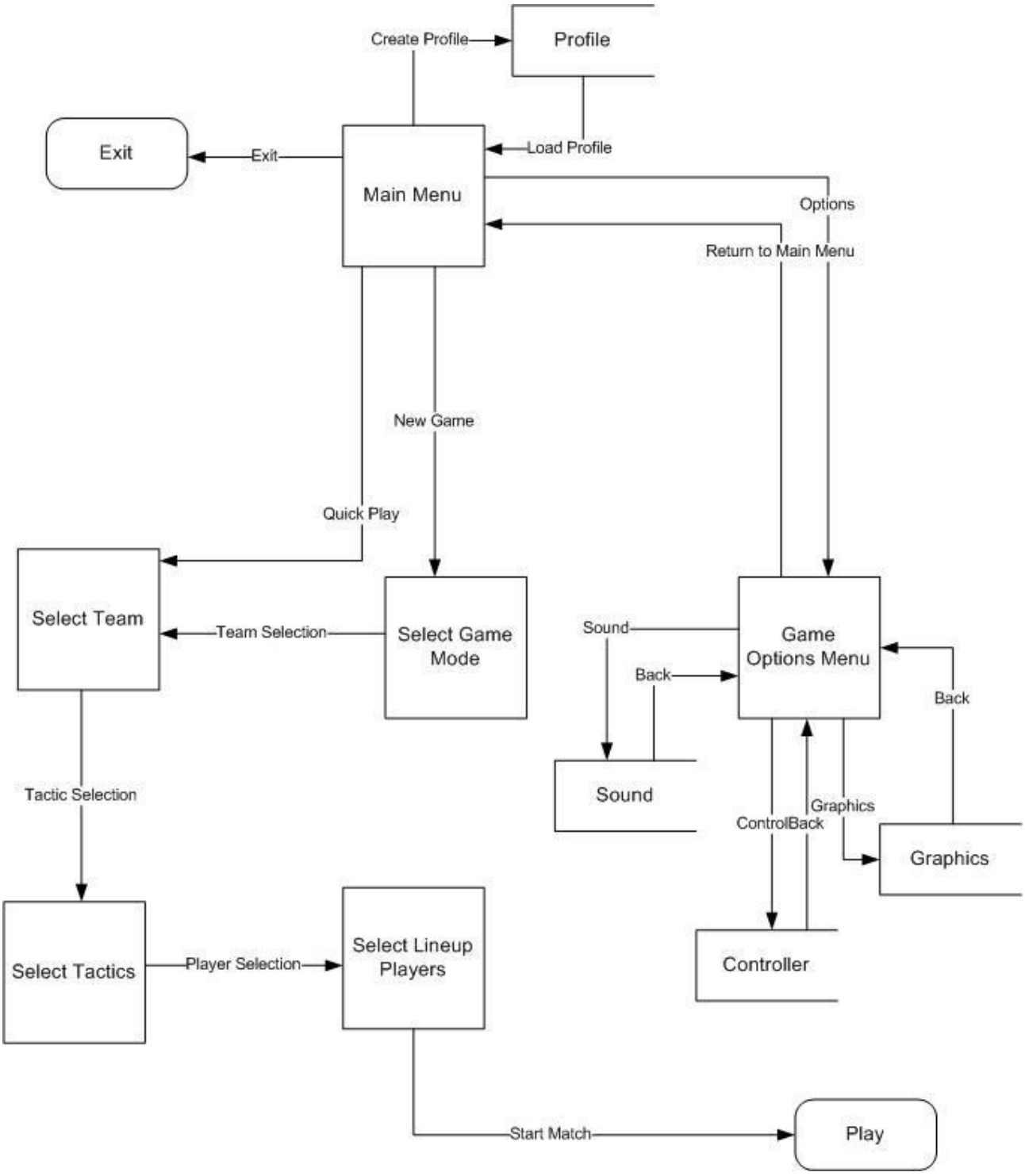


Figure4.4

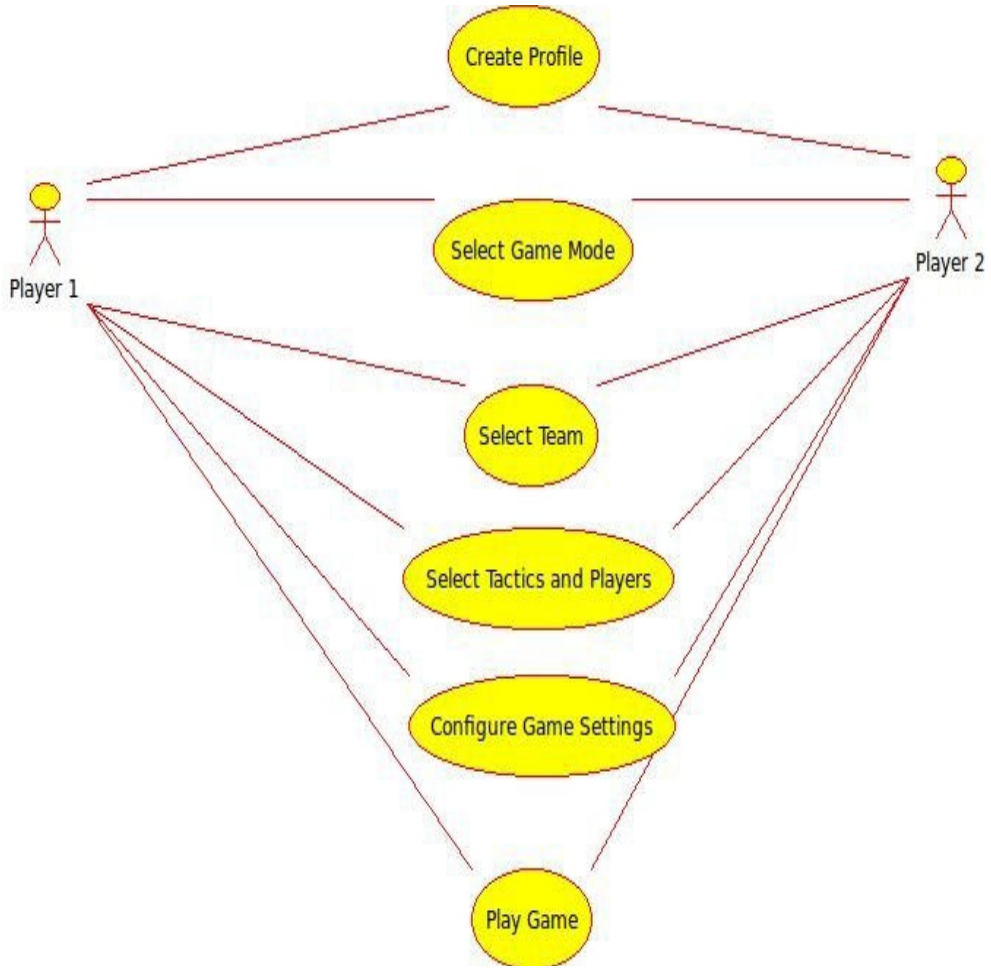
4.4 Sequence Diagrams

Sequence Diagrams supposed to show the logic and flow of the operations are to be discussed further at “Detailed Design Report”. Logic behind the A/I, sequence of the methods, beginning from game menu up to actions of the players are all to be explained.

5 Interface Design

5.1 Use Case Diagram

According to the use case diagram here which we had constructed during the requirement analysis stage, we designed the interface of LES.



5.2 Interface Design

Main Menu:

- Select Game Mode:
  - ✓ Single Player Career
  - ✓ Multiplayer
    - ➢ Same computer
    - ➢ Network
- Exit
- Profiles:
  - ✓ Create
  - ✓ Load
- Game Options Menu
  - ✓ Sound
  - ✓ Graphics
  - ✓ Controller
- Quick Play

From the main menu, one can go to the game options menu. In the options menu there are three option titles. Graphics settings can be altered between 640x480, 800x600 and 1024x768 resolutions. In sound settings the volume of the game can be arranged, muted totally. Also the tracks playing within the menus are to be arranged in here. In the controller settings, button configuration of keyboard or the game pad can be arranged.

Another submenu of the select game mode menu is the single player career mode. Here the mode of the single player career is to be selected such as UEFA Cup, World Cup etc.

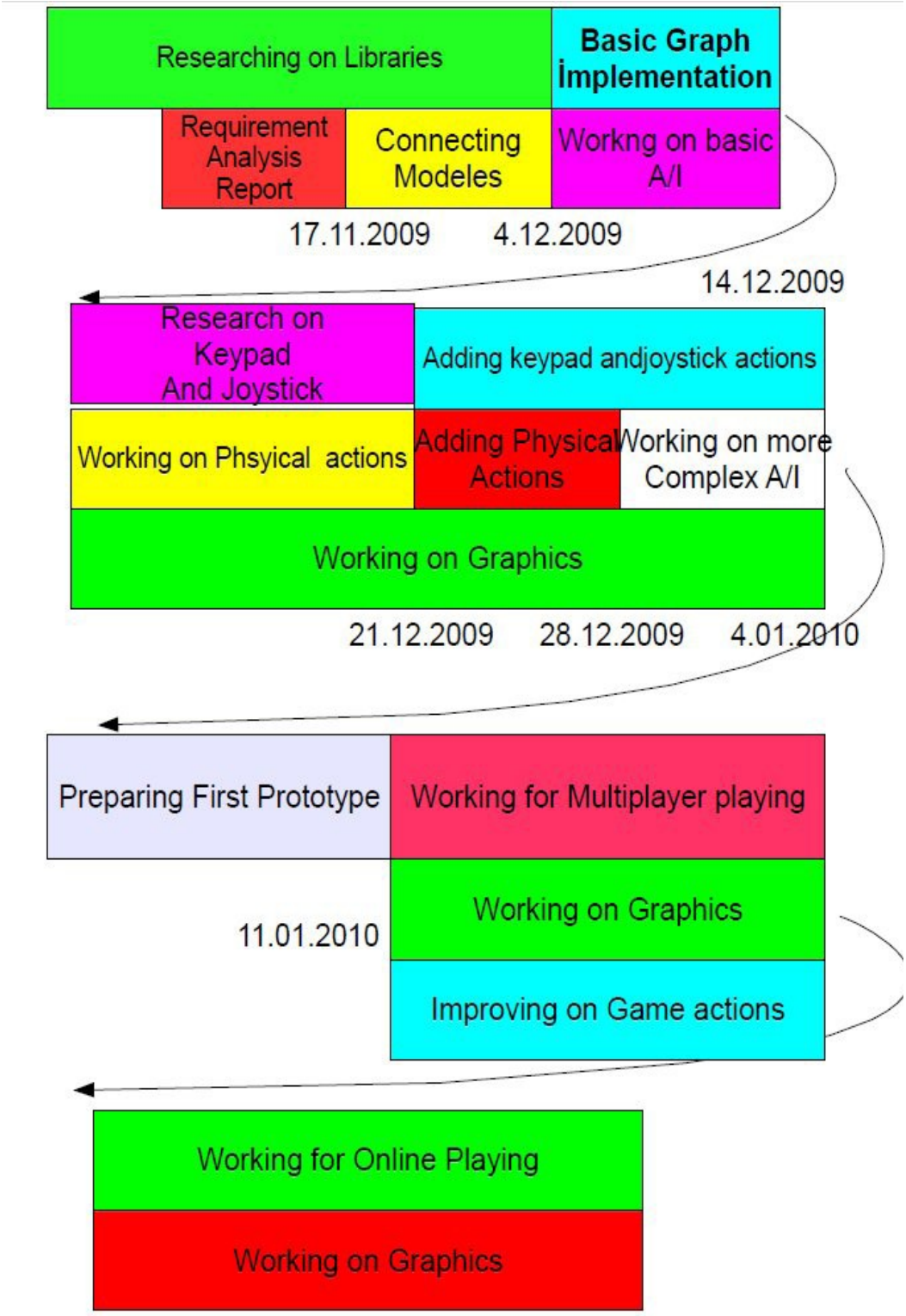
Multiplayer option is also here. Multiplayer mode either means two players on the same computer with two different input pads such as a keyboard and a game pad or could be through the internet.

In the main menu there is also a quick play option which lets the user to go to the match directly by only choosing his/her team not bothering with any career modes.

The profiles menu keeps the profiles already saved such as a career mode saved to be resumed later or personal sound, graphics or controller settings.

Lastly exit option lets user to return to the desktop shutting down LES.

6 Project Schedule



7 References

Graphics Libraries:
http://www.raydium.org/
http://www.ogre3d.org/
http://irrlicht.sourceforge.net/
http://www.crystalspace3d.org/main/Main_Page
http://www.panda3d.org/
http://www.grinninglizard.com/kyra/
http://alleg.sourceforge.net/
http://www.clanlib.org/

AI Libraries:
http://connect.creativelabs.com/openal/default.aspx
http://icculus.org/SDL_sound/
http://www.gamedev.net/reference/list.asp?categoryid=18

Joystick Libraries:
http://javajoystick.sourceforge.net/
http://goldenstudios.or.id/forum/showthread.php?tid=1377

Keypad Libraries:
www.ericsbinaryworld.com/GPL_code/

Sound Libraries:
http://icculus.org/SDL_sound/