PROJECT CONTORIUM

CEngBall Programming Game

Software Design Description

Alper Demir, Doga Uzuncukoglu, Emre Can Kucukoglu, Necati Cevik

Change History

Date	Revision	Comment
25.11.2013	1.0	Created.
28.12.2013	1.1	Preface updated. Design Rationale added. Information Viewpoint updated. State Dynamics Viewpoint updated. Planning Estimation section added.

PREFACE

This document contains the system design information about CengBall project. This document is prepared according to the “IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE Std 1016 – 2009”.

This Software Design Documentation provides a complete description of all the system design and views of the project. The first section of this document includes Project Identification, Stakeholders Identification and requirements, Composition of the developers’ team.

The following sections include document purpose and design viewpoints of the system.

Table of Contents

Table of Figures

1. Overview

This design report includes a complete description of the CEngBall project. This document includes features, functionalities, specifications and explanations about the project which is a design project for the Computer Engineering Design class of the Middle East Technical University.

1.1 Problem Definition

Artificial Intelligence is a huge field in computer science because of its innate glamour. Human brain is one of the most interesting things in the world and programs simulating brainwork is a true fascination. However to participate in this field, interesting as it may be, is quite difficult and also there are not many ways a curious person can satisfy his\her interest in the AI field.

This interesting field could be more popular among the people who are interested in software if there was an easy way a programmer could implement an AI agent for a certain task and then could also test his/her agent against other agents which are programmed by other developers to see whose agent is better. Also if the task at hand is an amusing subject, people will be even more interested in such effort.

There can be a system where people can both do programming and have fun and with this project we aim to provide such platform using video games as a tool.

1.2 Purpose

This document describes the design plan for our project and how the requirements stated in the Software Requirement Specifications will be fulfilled. Requirements in the said document will be translated into structural components of the project. Design concepts and architecture of the project will be explained in detail.

Design issues discussed in this document will be the main guideline for the development team but said guidelines can always change according to circumstances. The intended audience of this document is the software developers who are interested in the design process of such project.

1.3 Scope

The scope of this SDD consists of design patterns of the said project, brief explanation about the goals and objectives, constraints, assumptions, dependencies, system architecture with its components, user interface and actions of objects, libraries and the tools that will be used.

The project itself is actually a platform where people will compete against each other in a football-like video game via code writing. The system will allow the users to implement an AI agent and submit it. Later submitted AIs will compete with each other in simulations of a game much like football. The simulation process will be saved. Saved simulations can later be visualized for the competing parties to enjoy.

1.4 System Overview

“CEngBall” is a video game where players play by implementing a Java™ interface. These implementations are going to be the AI agents of the players. These AI agents are responsible for deciding the next moves of the virtual football players of their respective virtual football teams. The system is going to provide certain utilities to the users so that they can implement an agent in a way they desire.

The “Simulator” component of the system will use two of these AI agents. It will ask for their input periodically. Simulator will affect the playing field according to the input. After a certain amount of turns the simulation will hold. Simulation will be saved on a file for visualization.

The “Visualizer” component of the system will visualize the saved simulation files. These files will contain objects and their places. The visualizer component will read these files and draw a playing field for people to understand what is going on during the simulation.

The game will have a simple user interface that makes main functions to operate. The user will be able to choose the mode of the game in the main screen and also be able to view options and change them. In single player screen, the user will be able to load his/her team and choose an opponent while in multiplayer screen, the user will be able to load 2 teams. Also, the match screen will visualize the game.

1.5 References

- IEEE. IEEE STD 1016-2009 IEEE Standard for Information Technology – System Design – Software Design Descriptions. IEEE Computer Society, 2009.
- StarUML 5.0 User Guide. (2005). Retrieved from http://staruml.sourceforge.net/docs/user-guide(en)/toc.html

2. Definitions, Acronyms and Abbreviations

CEng	Computer Engineering
SRS	Software requirements specification
SDD	Software Design Document
AI	Artificial Intelligence
StarUML	An open source UML tool, modified version of GNU GPL
UML	Unified Modeling Language
IEEE	Institute of Electrical and Electronics Engineers
GUI	Graphical user interface
Dribble	(Sports) Move while controlling the ball.

3. Conceptual Model for Software Design Descriptions

The project is about artificial intelligence and football so to understand this document better, a basic knowledge of both would be nice.

3.1 Software Design in Context

This project is designed in an object-oriented fashion because the concept of football is suitable for this approach. This makes the product more easy to use, because real life objects like football players and balls will be represented by their class forms. This way the convoluted process of AI development will be simpler for the end user.

The project is also modular, meaning the whole system is composed of independent parts. This way the project can move forward in a nonlinear way as the developers work on different parts of the project without affecting each other's work.

3.2 Software Design Descriptions within Life Cycle

3.2.1 Influences on SDD Preparation

This design document is written with respect to the SRS document of the project. The design process tries to satisfy all the requirements specified in a smooth and optimized way.

Since the project is not easily commercializable, there is no real customer involved. That makes the biggest stakeholders, the development team. Therefore, another influencing factor is the development team's needs and as the project goes new design material can surface.

3.2.2 Influences on Software Life Cycle Products

Project consists of modular components which can be developed separately after the communications between them are described in an unambiguous and clean form. That means after defining the components, first thing to do is to define how they communicate. After that implementation can go concurrently.

These components are the "Simulator", "Visualizer" and "AI Agent".

3.2.3 Design Verification and Design Role in Validation

Verification and validation will be tested after preparation of the test cases. Since we will use the spiral model for CEngBall software project, test cases will be prepared with the development simultaneously. All system parts will be tested against these cases. It will be checked for whether the requirements fulfilled or not.

4. Design Description Information Content

4.1 Introduction

Software design description of CengBall identifies how this game will be designed and implemented. Design of CengBall will have a modular and object-oriented structure. There will also be a neatly designed visualization component and a graphical user interface.

4.2 SDD Identification

CEngBall software design development parts are designed at the 28th of November, 2013. CEngBall project's main purpose is to compare player developed AI's on football simulation. After testing for the verification and validation, CEngBall will be available for the public. The game can be used for improving artificial intelligence development skills and gaining experience in the area of software and game development. It will be a relatively simple case of artificial intelligence development for the players.

4.3 Design Stakeholders and Their Concerns

CEngBall's stakeholders are the development team. Stakeholders' possible concerns are user friendly development, simplification of AI development process and neat visualization of the simulation. The users will be able to perform simple tasks like simulating the game via a simple user interface. Simplification of AI development is important for providing a relatively simple development experience.

4.4 Design Views

The project will be implemented in a modular fashion. The stakeholders can add new features or remove the unwanted ones in project. Object oriented paradigm is chosen so that new features can be integrated without much effort. Users will see a basic interface to upload their code when the program is run. After uploading their code, users will be able to simulate the game and watch the visual representation of the simulation. Visualization and simulation will be two different components of the software. They will interact with each other to exchange their data. Each simulation will produce a file to log the details of simulation. This log files will be used to visualize the entire game. This will also enable us to easily save and replay each game without much effort.

Product context is specified and restricted limitations given in the SRS document. A logical view of the product is explained and also it is supported by diagrams. Relationships between classes are easily perceived.

4.5 Design Viewpoints

This SDD document conforms to the view and viewpoint concepts stated in the standard document. The fifth section is divided into subsections under the names of design viewpoints specifying each design concern that the stakeholders had separately. Design views associated with a design viewpoint corresponds to particular diagrams in this document.

Context viewpoint shows what is expected from the user actor in the system. The roles of the user are clearly specified. Design entities will be AI controlled teams created by users and the information will flow between this user and the system. Input-output relations will be explained in context viewpoint-design elements.

4.6 Design Elements

Design choices are made in a way such that we can easily upgrade the project according to the needs of the stakeholders and users. Each component of the implementation like functions, variables and classes will be commented such that, for a further modification on software it will be very easy to understand the code and improve it.

4.7 Design Rationale

The division of the project into three main components (namely Simulator, Visualizer and AI Agents) allowed developers to maintain a modular approach. It also made the design process easier since the team can concentrate on different parts of the project without having to consider the possible harm they could inflict on other parts. Moreover, our decisions make the system portable, since project can be applied to online multiplayer system in the future.

4.8 Design Languages

Unified modeling language (UML) is adopted as the design language. Rather than following a specific version of UML, and all its definitions strictly, our diagrams are characterized by highest conformance to the UML 1.6 standard where necessary additions/deviations are applied henever considered appropriate by the developer team. StarUML v5.0 was used as the modeling tool.

5. Design Viewpoints

5.1 Introduction

In this part of the document, different design viewpoints are going to be explained. Here is a list of the existing viewpoints

- Context Viewpoint
- Logical Viewpoint
- Interface Viewpoint
- Dependency Viewpoint
- Interaction Viewpoint
- State Dynamics Viewpoint
- Information Viewpoint

5.2 Context Viewpoint

This viewpoint describes the functionalities the software provides by the elements that interact with it like users.

This project is actually a video game and video games usually have very definitive use cases with a lot of action involved. But it is not played like a regular game, therefore it has a very unique set of user use cases.

5.2.1 User Use Cases

User interacts with the system via submitting his/her code (AI agent) and the user use cases are related to that topic. These use cases can be seen in Figure 1 below.

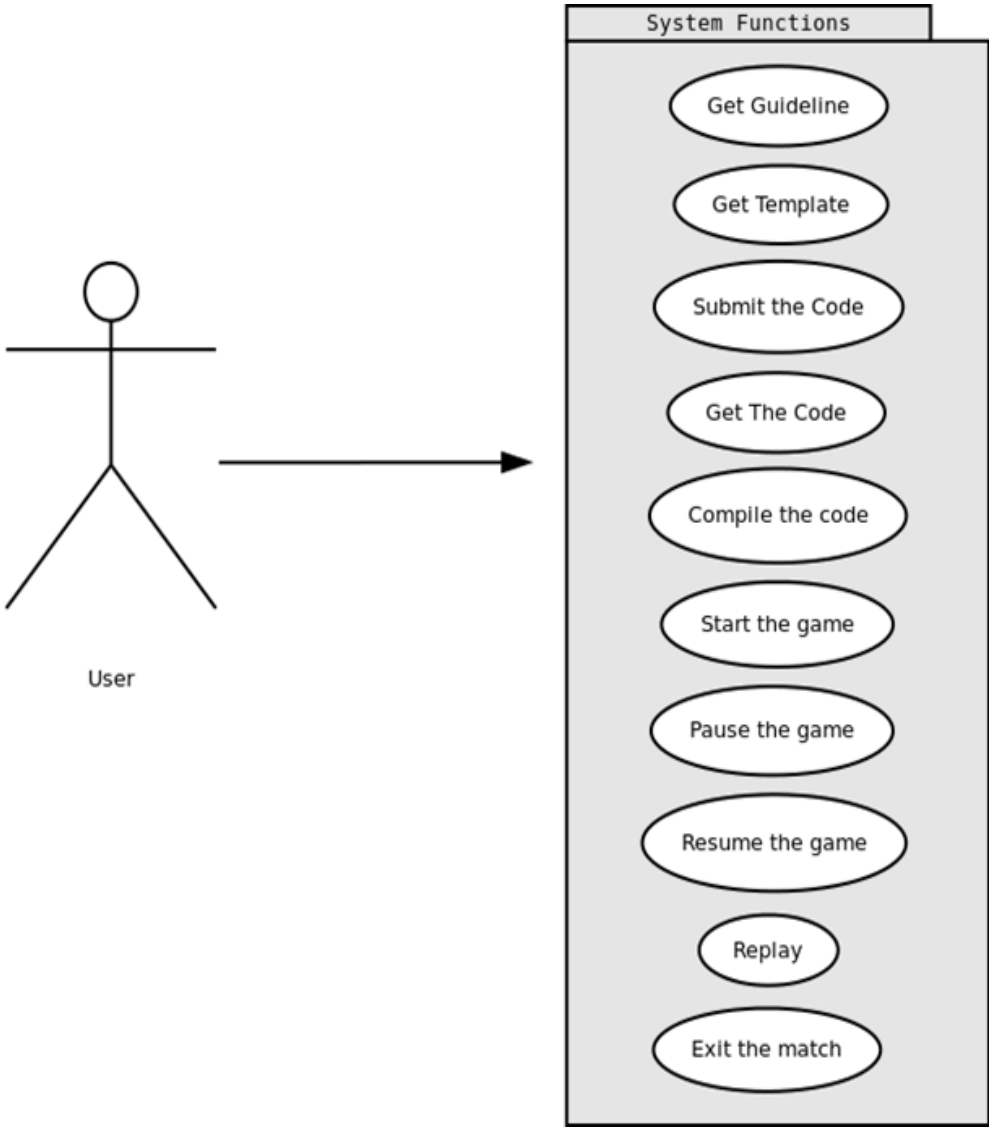

Figure 1 User Use Cases

5.2.1.1 Get Guideline Use Case

Use Case ID	use_case_1
Use Case Name	Get Guideline
Trigger	The user selects to get the guideline
Precondition	-
Basic Path	1. The user selects to download the guideline to a directory in the computer 2. The user selects the directory to download in browse screen. 3. The user confirms the directory. 4. The game downloads the guideline to the directory.
Alternative Paths	-
Postcondition	The guideline is downloaded to the selected directory.
Exception Paths	The user may cancel downloading.
Other	The guideline includes the API information. It has definition of functions, variables and classes. Also, it has the information about how the game works and how a submitted code should be.

5.2.1.2 *Get Template Use Case*

Use Case ID	use_case_2
Use Case Name	Get Template
Trigger	The user selects to get the template
Precondition	-
Basic Path	1. The user selects to download the template to a

	directory in the computer 2. The user selects the directory to download in browse screen. 3. The user confirms the directory. 4. The game downloads the template to the directory.
Alternative Paths	-
Postcondition	The template is downloaded to the selected directory.
Exception Paths	The user may cancel downloading.
Other	The template is an example code for a player AI. It includes the must functions for the game to be played. Also, it has information about the functions in comments.

5.2.1.3 *Submit the Code Use Case*

Use Case ID	use_case_3
Use Case Name	Submit the code
Trigger	The user selects to submit the code written
Precondition	-
Basic Path	1. The user selects to submit the code to the game. 2. The user selects the directory of the code. 3. The user confirms the directory. 4. The game gets the code.
Alternative Paths	-
Postcondition	The code is submitted to the game.
Exception Paths	The user may cancel submitting.
Other	-

5.2.1.4 Get the Code Use Case

Use Case ID	use_case_4
Use Case Name	Get The Code
Trigger	The user selects to get the code of his/her
Precondition	The code must be submitted to the game.
Basic Path	1. The user selects to download the code to a directory in the computer 2. The user selects the directory to download in browse screen. 3. The user confirms the directory. 4. The game downloads the code to the directory.
Alternative Paths	-
Postcondition	The code is downloaded to the selected directory.
Exception Paths	The user may cancel downloading.
Other	-

5.2.1.5 Compile the Code Use Case

Use Case ID	use_case_5
Use Case Name	Compile the code
Trigger	The user selects to compile the code submitted
Precondition	The code must be submitted to the game.
Basic Path	1. The user selects to compile the code. 2. The game compiles the code, checks whether it obeys the rules of the game and returns back the result of compile operation.

Alternative Paths	-
Postcondition	The code is compiled, checked and the result of the compile operation is returned.
Exception Paths	-
Other	-

5.2.1.6 Start the Game Use Case

Use Case ID	use_case_6
Use Case Name	Start the game
Trigger	The user selects to start the match.
Precondition	In single player mode, the code of the user must be submitted to the game. In two player mode, the codes of the both users must be submitted to the game.
Basic Path	1. The user selects to start the match. 2. The game opens a new screen and starts to simulate and visualize the match.
Alternative Paths	-
Post condition	The match is started and getting visualized.
Exception Paths	-
Other	

5.2.1.7 Pause the Game Use Case

Use Case ID	use_case_7
Use Case Name	Pause the game

Trigger	The user selects to pause the game.
Precondition	The match must be started and being played.
Basic Path	1. The user selects to pause the match. 2. The game freezes and pauses the match. Also, a pause screen will be shown.
Alternative Paths	-
Postcondition	The match is paused.
Exception Paths	-
Other	The pause panel will have resume, replay and exit option.

5.2.1.8 Resume the Game Use Case

Use Case ID	use_case_8
Use Case Name	Resume the game
Trigger	The user selects to resume the game.
Precondition	The match must be paused.
Basic Path	1. The user selects to resume the match. 2. The game continues to simulate and visualize the match.
Alternative Paths	-
Postcondition	The match is resumed.
Exception Paths	-
Other	-

5.2.1.9 Replay Use Case

Use Case ID	use_case_9
Use Case Name	Replay
Trigger	The user selects to replay.
Precondition	The match must be paused.
Basic Path	1. The user selects to replay a scene. 2. The replay screen opens. 3. The user selects to play the scene. 4. The game plays the scene.
Alternative Paths	-
Post condition	The scene is played.
Exception Paths	The user may exit the replay screen.
Other	-

5.2.1.10 Exit the Game Use Case

Use Case ID	use_case_10
Use Case Name	Exit the match
Trigger	The user selects to exit the match.
Precondition	The match must be paused.
Basic Path	1. The user selects to exit the match. 2. The game shows a popup screen to confirm exit operation. 3. The user confirms the operation. 4. The game finishes the simulation, closes the match screen and returns to the main screen.
Alternative Paths	In step 3, the user cancels the exit operation and the game returns back to the pause screen.

Post condition	The match is exited.
Exception Paths	-
Other	-

5.3 Logical Viewpoint

The classes and the relations amongst them will be explained here. All classes involved in the project and their relations can be seen in Figure 2. There will be tables explaining the fields and methods of classes in detail.

The classes are Simulator, Visualizer, Team, Player, Percept, SaveFile, Pitch, Ball, and Position.

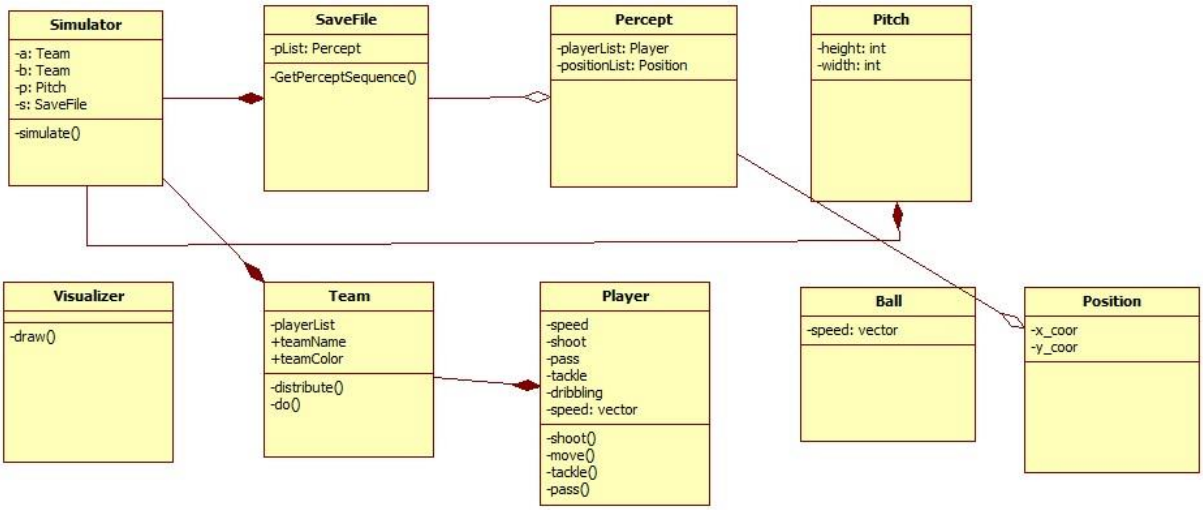

Figure 2 Classes and their relations

5.3.1 Simulator Class

This class will simulate the game according to the input it takes from AI agents and creates a save file for visualization.

Name	Type	Visibility	Definition
team1	Team	Private	The first AI agent. The simulator will ask for input periodically from this field.
team2	Team	Private	The second AI agent. The simulator will ask for input periodically from this field
pitch	Pitch	Private	The field that the game is going to played on. Places of the goals and lines are going to be decided according to this field.

saveFile	Save File	Private	The field representing the save file of the simulation. Simulation will fill this field every turn and at the end of the simulation it will hard save it as a file.
simulate()	void	Public	This method will periodically ask for input from the Team fields, update the SaveFile field accordingly and before it returns it saves the saveFile entity as a real file.

5.3.2 Visualizer Class

This class will read from a saved simulation file and visualize the simulation in a way that the audience can enjoy. Since this is a football game, it is expected to be fun to watch.

Name	Type	Visibility	Definition
draw()	void	Public	Reads the file and draws the objects on a field

5.3.3 Team Class

This class will be the AI agent of the user. The users will implement a Team interface. The simulator will call methods of this class to move the simulation forward.

Name	Type	Visibility	Definition
playerList	List<Player>	Private	A list of the player objects. User will have access to virtual football players via this list.
teamName	String	Private	Name of the team. It will be used for visual purposes.
teamColor	String	Private	Color of the team. It will be used for visual purposes.

distribute()	void	Public	This method will distribute the limited amount of skill points available for the user among the virtual players.
do()	Percept	Public	The simulator instance will call this method periodically. Users will use this method to manipulate their virtual football players.

5.3.4 Player Class

This class's instances will be the virtual football players the user controls via the agent he/she implemented (an instance of a Team class). Player class will loosely resemble a real life football player in an abstract way.

Name	Type	Visibility	Definition
speed	int	Private	The stat that will determine how fast this player instance can move.
shoot	int	Private	The stat that will determine how well this player instance can shoot the ball.
pass	int	Private	The stat that will determine how well this player instance can pass the ball.
tackle	int	Private	The stat that will determine how well this player instance can tackle the opponent player in order to take control of the ball.
dribbling	int	Private	The stat that will determine how well this player instance can control and dribble the ball.
speedVector	Vector	Private	A 2d vector which is the current speed vector of the player instance. With help of this field, the simulator will determine the next position of the players.

shoot()	void	Public	The method users will use to shoot the ball. It will change the speed vector of the ball according to player's stats and the parameters AI agent provides.
move()	void	Public	The method users will use to move the players. It will change the speed vector of the player. A move operation will only change the speed vector of the player and the player will start moving in the designated direction as turns pass.
tackle()	void	Public	The method users will use to challenge the player instance controlling the ball for the control of the ball.
pass()	void	Public	The method users will use to pass the ball to another player object. This method will change the speed vector of the current ball instance.

5.3.4 Percept Class

This class will be a record of the positions of the objects in the simulation. This class will be used to save and visualize the simulation.

Name	Type	Visibility	Definition
playerList	List	Private	A list containing all the player instances in the active simulation.
positionsList	List	Private	A list containing all the positions of the objects in the active simulation.

5.3.5 SaveFile Class

This class will be the representation of the real save files of our simulations. After each turn simulator will add more percepts into the active instance of this class. After the simulation terminates it will create a real file using this class.

Name	Type	Visibility	Definition
pList	List	Private	A percept list. It will contain the complete set of percepts of a simulation.
getPerceptSequence()	List	Public	Returns a specific set of percepts. Users will be able to use this method to inspect past turns.

5.3.6 Pitch Class

This class will determine the dimensions of the playing field including goal and line positions.

Name	Type	Visibility	Definition
height	int	Private	Height of the playing field.
width	int	Private	Width of the playing field.

5.3.7 Ball Class

This class will represent a football ball.

Name	Type	Visibility	Definition
speed	Vector	Private	Speed vector of the ball. This field will determine where the ball will move in coming turns.

5.3.8 Position Class

This class will be used to store position information.

Name	Type	Visibility	Definition
x_coor	int	Private	X coordinate information
y_coor	int	Private	Y coordinate information

5.3.9 Relationships between Classes

All of the relations can be observed from Figure 2.

- Simulator class has elements from Team, Pitch and SaveFile classes.
- SaveFile class has elements from Percept class.
- Team class has elements from Player class.

5.4 Interface Viewpoint

5.4.1 User Interface

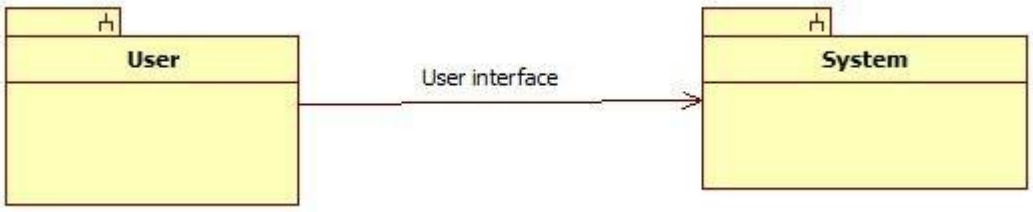

Figure 3 User Interface Diagram

The user interface will be held by screens. The user will be able to use the functionalities of the game through these screens. Any response from the game will be shown to the user.

There will be 5 screens for user to view. These screens are Main Screen, Single Player Screen, Multiplayer Screen, Options Screen and Match Screen.

5.4.1.1 User Interface Elements

5.4.1.1.1 Main Screen

The main screen is the initial screen of the game. When the user starts the game, this screen appears first.

It has the links for Single Player, Multiplayer, Options screens and it is possible to exit the game in this screen. When one of the button is pressed relative screen will be shown to the user.

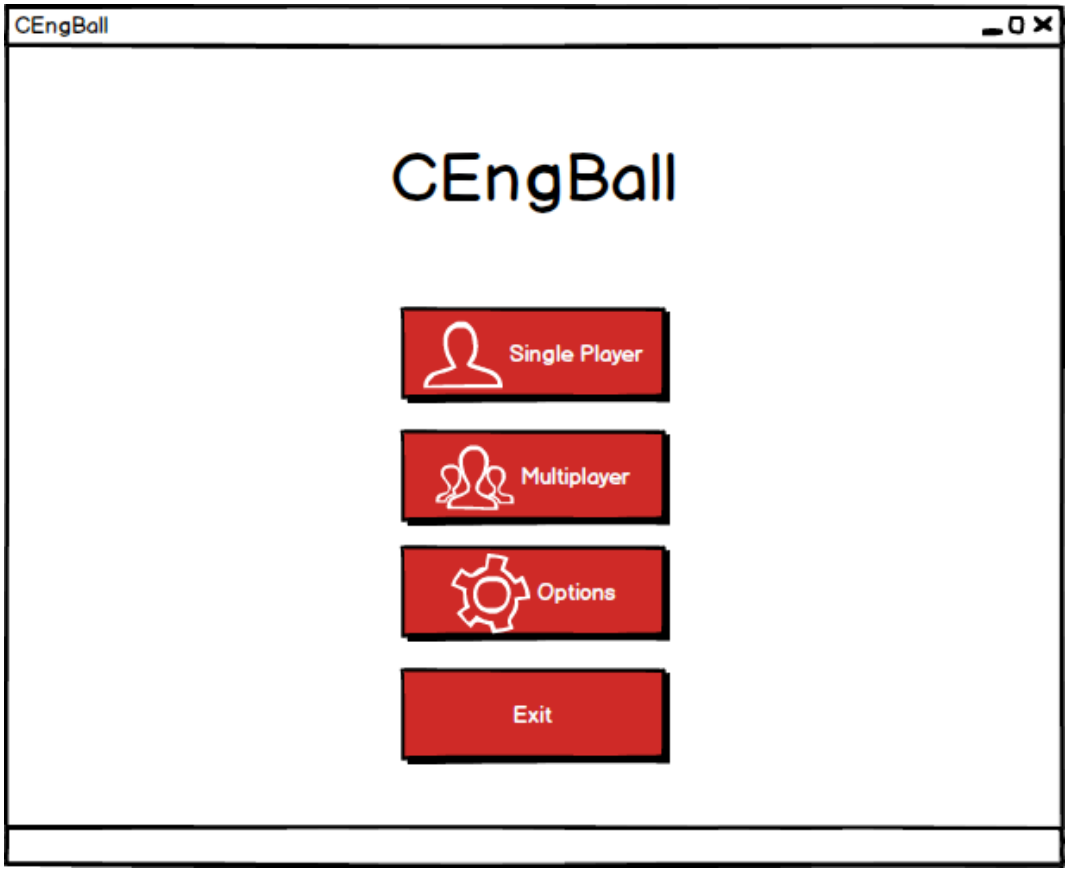

Figure 4 Main Screen

5.4.1.1.2 Options Screen

This screen is related the options that the user can change. This options are about visualization and simulation.

The screen includes options for match length, commentary and player kits. The match length setting will have options 5 minutes to 15 minutes. This setting will affect both simulation and visualization time. The commentary setting will have ON and OFF options. If it is set to ON, then there will be a commentary on the match, otherwise the game will be shown without a commentary. Moreover, the player kits setting will also have ON and OFF options. When it is set to ON, the player kits will be shown in the match.

When the options are set, the user will be able to apply them with a button. Also, the user will be able to return back to the main screen.

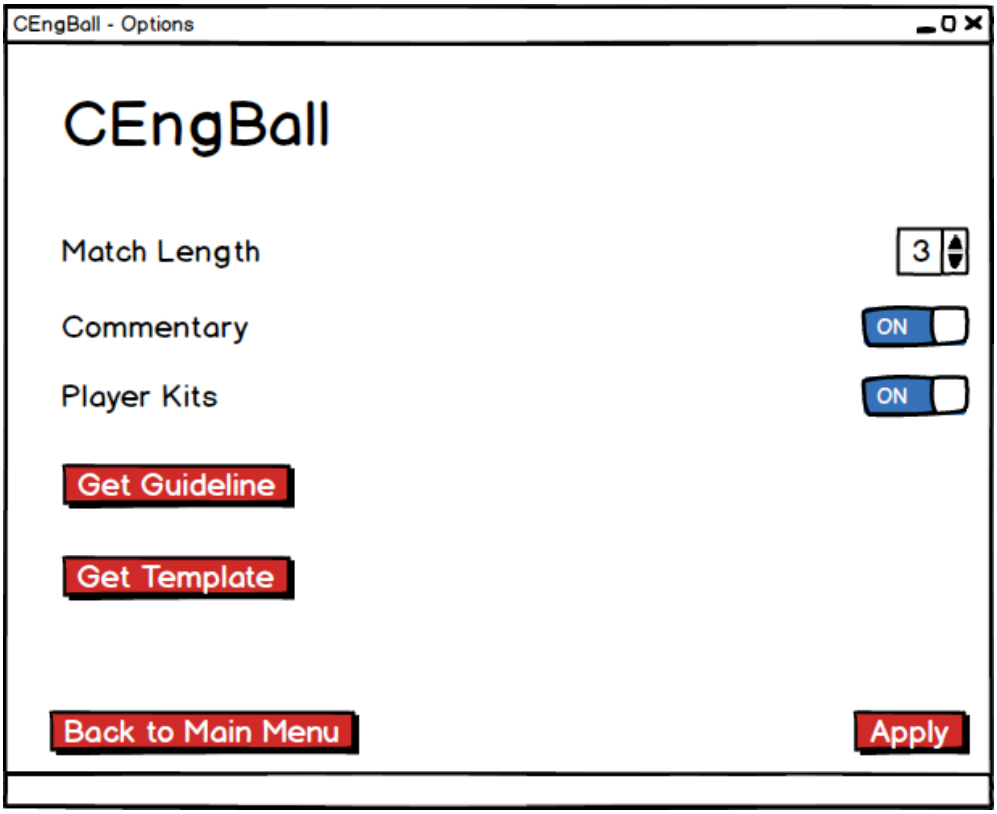

Figure 5 Options Screen

5.4.1.1.3 Single Player Screen

This screen is for the single player mode. When the user wants to challenge the built-in agents with his/her own agent, he/she will use this screen to proceed.

In the screen, it will be possible to load the user code, compile and check it and choose the opponent difficulty. After both load and compile operations, the result will be shown to the user. Also, the user will be able to return back to the main screen or start the game. The “Start the Game” button will be enable after the user loads the code, compiles it and chooses a difficulty for the opponent, otherwise, it is disabled.

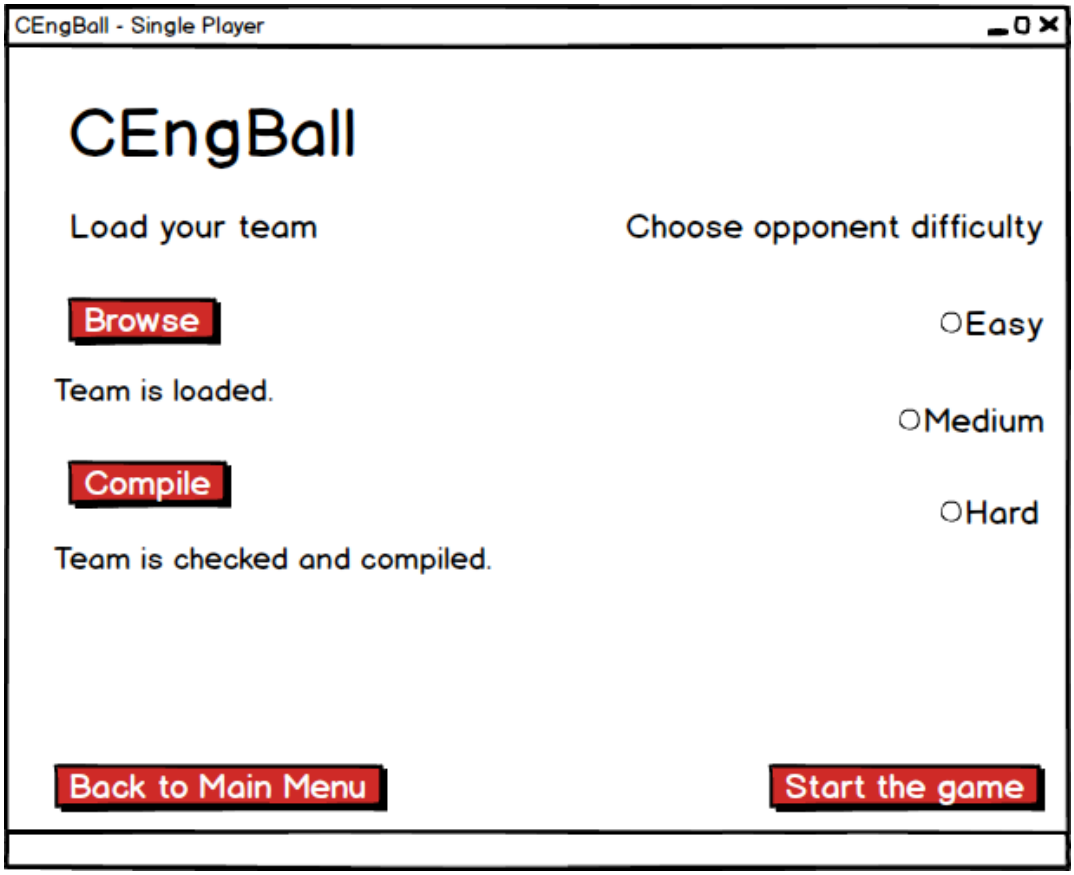

Figure 6 Single Player Screen

5.4.1.1.4 Multiplayer Screen

This screen is for the multiplayer mode. When the user wants to challenge two user codes, he/she will chooses this mode.

In the screen, it will be possible to load the codes and compile them. The result of both the load and compile operations will be shown to the user. Moreover, the user will be able to start the game or return back to the main screen. Starting the game operation will be enabled when the both load and compile operations are completed.

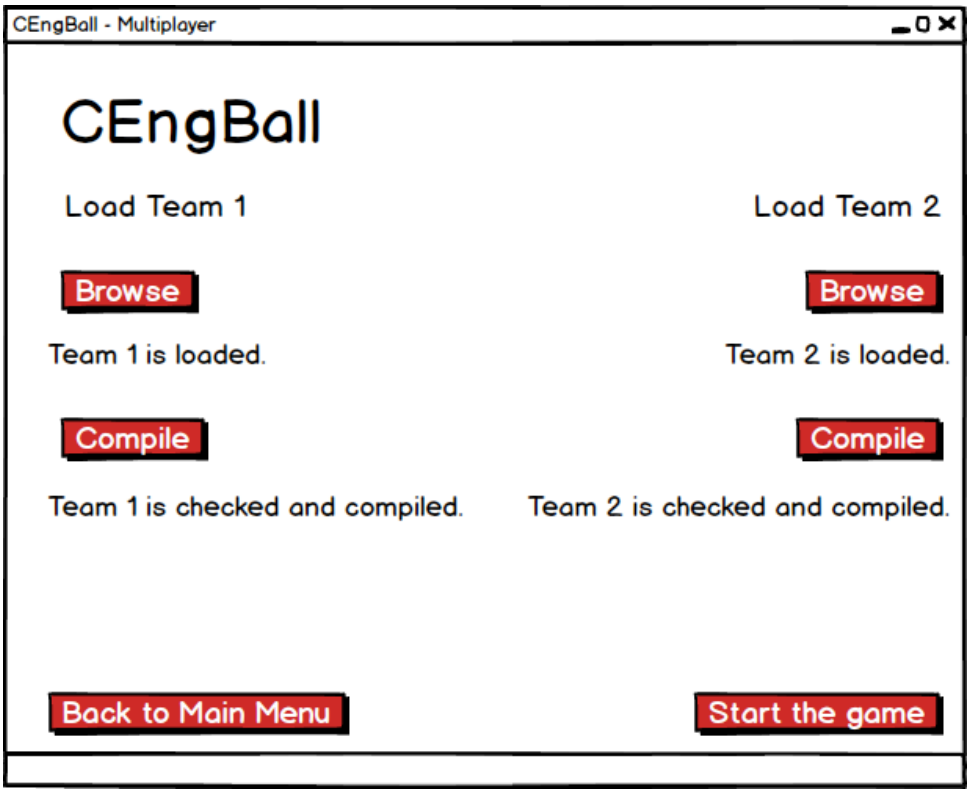

Figure 7 Multiplayer Screen

5.4.1.1.5 Match Screen

This screen is for the visualization of the match. In both modes, this screen will show the match to the user.

In the screen, there will be a pitch to play the game and a commentary panel if it is enabled in options screen. The user will be able to watch the game or skip the game. If skipping operation is selected, the result of the match will be shown to the user. At the end of the match, the user will be able to finish the game and return back to the previous screen. Also, the user will be able to exit the match any time.

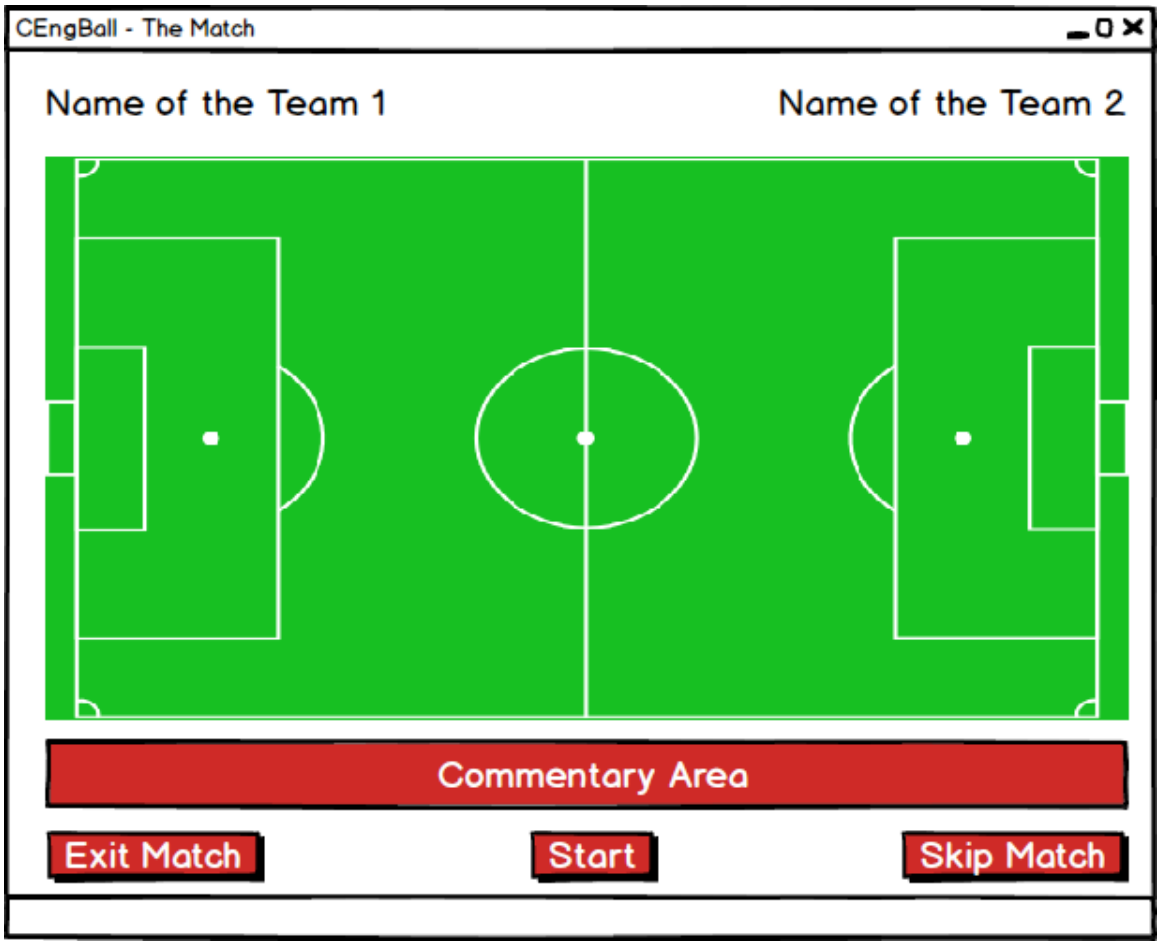

Figure 8 Match Screen

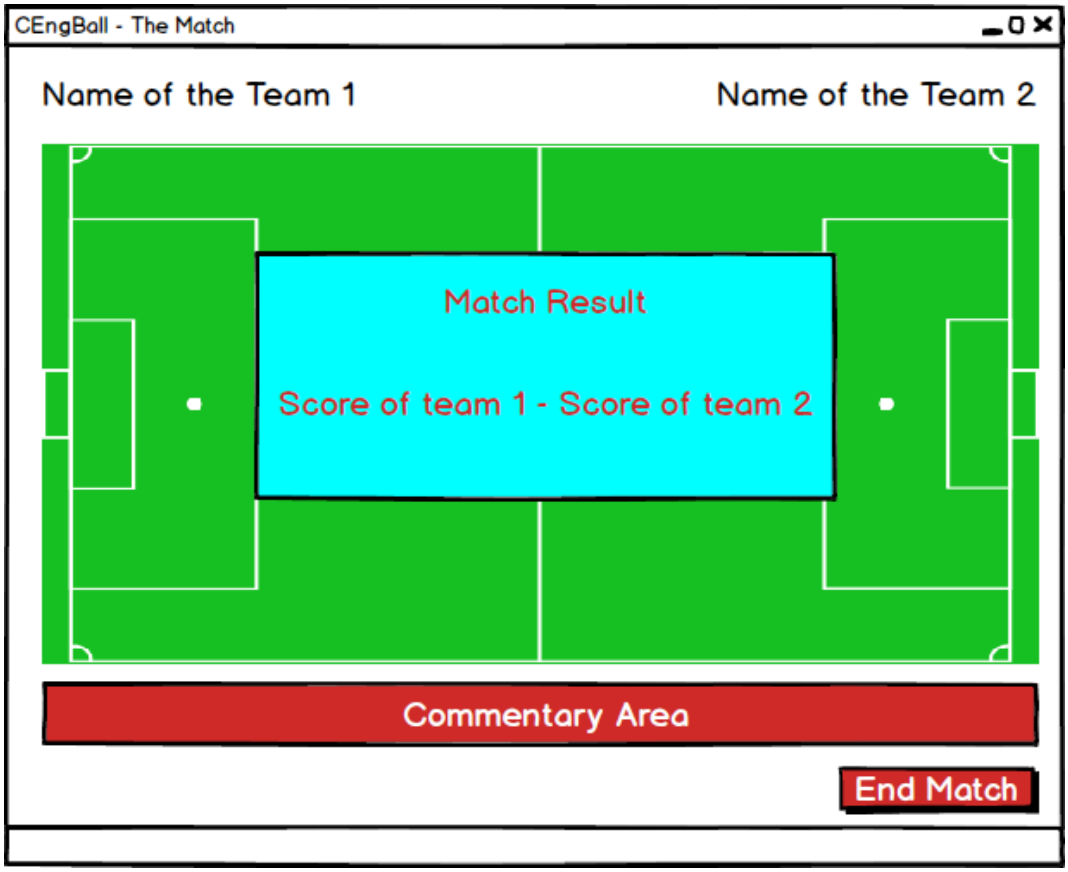

Figure 9 End Game Screen

5.4.2 AI Agent - Simulator Interface

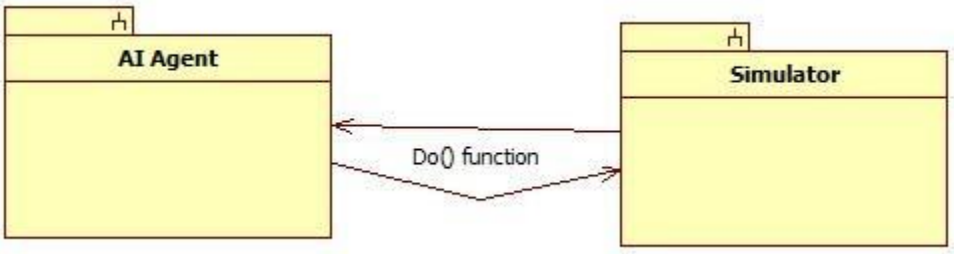

Figure 10 AI Agent - Simulator Interface Diagram

Via Do() function these two subsystems interact. The simulator asks for a percept instants and the AI Agent provides it via Do() function.

In each turn, the simulator will get the last percept that is played by a user agent and update the simulation. After updating, it will wait the other user agent to make a move. This Do() function is the interface for the user agent to make the move. There will be a timeout for the new move.

After the time runs out, the simulator will take the latest changes in the objects and uses them as the move of the agent.

This communication will continue turn by turn. The simulator will provide the current percept to one agent and asks for a new move in one turn while it does these for the other agent in the following turn.

5.4.3 Visualizer - Simulator Interface

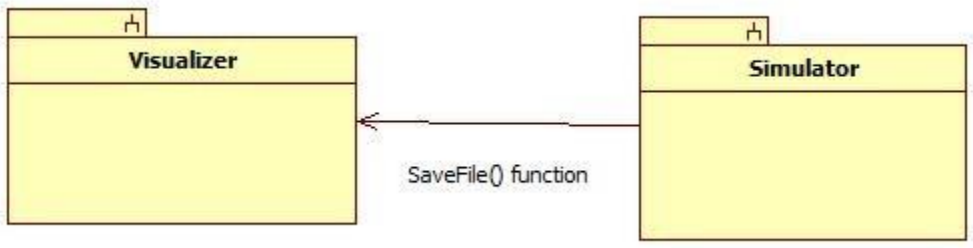

Figure 11 Visualizer Simulator Interface Diagram

Via SaveFile() function these two subsystems interact. The visualizer asks for a log file and the simulator provides it via SaveFile() function.

The save file is necessary for visualizer to display the game. It will have the percepts in each turn as logs. The visualizer will display the game according to these logs. When the user wants to watch the game, the visualizer will ask for the save file of the match and the simulator will return an instance of save file with a percept list. Then, the visualizer will play the game on the screen.

This communication is a one way communication which will happen only once. When the visualizer gets the percept list, it will just play the game.

5.5 Dependency Viewpoint

This viewpoint will list the subsystems and explain the interconnections amongst them in detail.

Whole system consists of three subsystems namely, the simulator and the visualizer and the AI agents.

5.5.1 Simulator Subsystem

This subsystem is responsible for running the simulation according to the input from the AI agents. It contains two AI Agents and a virtual playing field. The way this subsystem operates can be explained as:

- The simulator will load two agents and a field, then it will start the simulation.
- The subsystem will ask AI Agents periodically for input.
- Agents will move their players and the ball.
- The simulator will record this movements as percepts. It will add this percepts to a saveFile object.
- After the simulation terminates, it will dump that saveFile instance into a file on the hard disk.

5.5.2 Visualizer Subsystem

This subsystem is responsible for visualizing a saved simulation. It contains a saveFile object. Simulations are saved as a list of percepts. Each percept is a turn passed. Since percepts does not contain any data related with time, visualizer will decide the length of the visualization. The way this subsystem operates can be explained as:

- The visualizer will load a saveFile and it will create percepts according to it.
- It will recognize the objects (Players and Ball) and will draw these objects according to the position information provided by the percepts.

5.5.3 AI Agent Subsystem

This subsystem is responsible for providing input to the simulator subsystem. It contains the user implemented AI so the user interacts with the whole system via this subsystem. The way this subsystem operates can be explained as:

- User will implement an AI Agent and load it into the system using the user interface.
- User will start the simulation and then this subsystem will be asked for input by the simulator subsystem.
- Agent will decide what each of the player instances will do using Player Class's methods. (Shoot, move etc.)
- Agent will return a new percept instance to the simulator and will wait for the next turn.

5.5.4 Subsystem Connections

There are several connections in the system. The simulator and AI Agents are connected via software, while the simulator and the visualizer communicate via a save file.

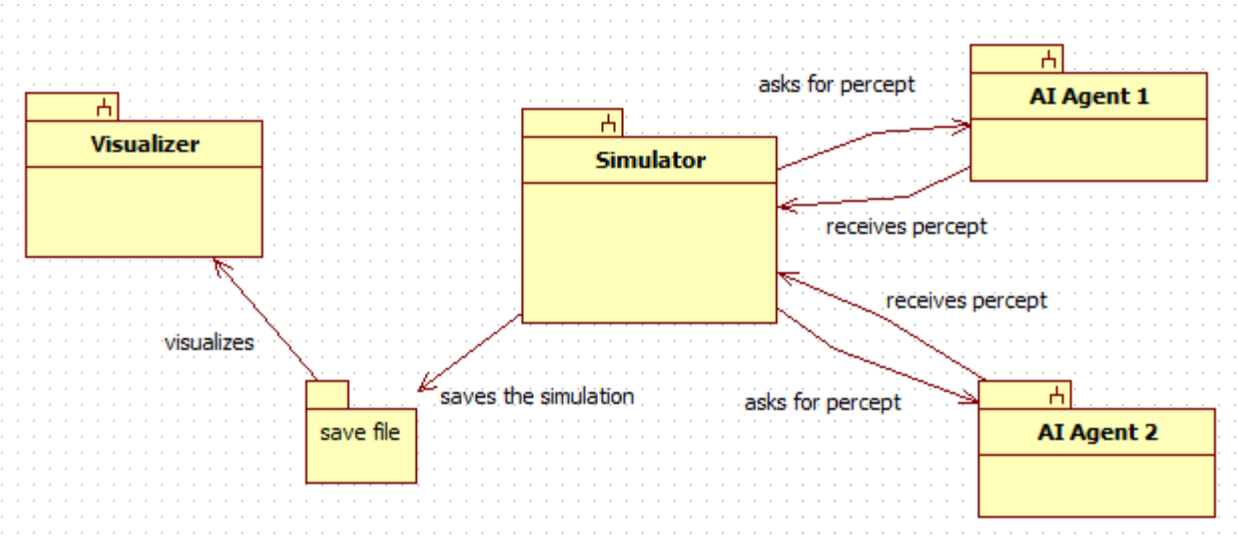

Figure 12 Subsystem Connections

5.6 Interaction Viewpoint

In this viewpoint, the interaction and the connection between user interface elements will be explained screen by screen in detail. Also, main variable will be explained.

Variable Identification	Type	Explanation	Domain
GAME_MODE	ENUM	The mode of the game	{ SINGLE, MULTI }
MATCH_LENGTH	INTEGER	The length of the game in minutes	[5-15]
COMMENTARY_SELECTION	ENUM	The selection about whether there will be a commentary on the match or not	{ ON, OFF }
PLAYERKIT_SELECTION	ENUM	The selection about whether the player kit	{ ON, OFF }

		numbers will be shown on the match or not	
SPEED_VALUE	INTEGER	The speed skill value for a player	[0-20]
PASS_VALUE	INTEGER	The pass skill value for a player	[0-20]
SHOOT_VALUE	INTEGER	The shoot skill value for a player	[0-20]
TACKLE_VALUE	INTEGER	The tackle skill value for a player	[0-20]
DRIBBLING_VALUE	INTEGER	The dribbling skill value for a player	[0-20]
SKILL_POINTS	INTEGER	The number of skill points that a team will get	100
MAX_PERCEPT_NUMBER	INTEGER	The maximum number for percept that a team can get	5
OPPONENT_DIFFICULTY	ENUM	The difficulty of the built-in AI.	{ EASY, MEDIUM, HARD }
CURRENT_SCREEN	ENUM	The current screen of the game	{ MAIN, SINGLEPLAYER, MULTIPLAYER, OPTIONS, MATCH }

MATCH_STATE	ENUM	The state of the match	{ PLAYING, STOPPED, READY, FINISHED }

Entity Identification	**Type**	**Use Case**	**Screen**	**Function**
Single Player	Button	-	MAIN	When pressed; 1. GAME_MODE must be set to SINGLE 2. CURRENT_SCREEN must be set to SINGLEPLAYER
Multi Player	Button	-	MAIN	When pressed; 1. GAME_MODE must be set to MULTI 2. CURRENT_SCREEN must be set to MULTIPLAYER
Options	Button	-	MAIN	When pressed; 1. CURRENT_SCREEN must be set to OPTIONS
Exit	Button	-	MAIN	When pressed; 1. The game must terminate

Entity Identification	Type	Use Case	Screen	Function
Match Length	Spinner	-	OPTIONS	When pressed; 1. The domain of the MATCH_LENGTH variable must be shown to the user When a value is selected in the domain MATCH_LENGTH; 1. MATCH_LENGTH must be set to the selected value
Commentary	Toggle Button	-	OPTIONS	When pressed; If COMMENTARY_SELECTION is set to ON; 1. COMMENTARY_SELECTION must be set to OFF If COMMENTARY_SELECTION is set to OFF; 1. COMMENTARY_SELECTION must be set to ON
Player Kits	Toggle Button	-	OPTIONS	When pressed; If PLAYERKIT_SELECTION is set to ON; 1. PLAYERKIT_SELECTION must be set to OFF If PLAYERKIT_SELECTION is set to OFF; 1. PLAYERKIT_SELECTION must be set to ON
Get Guideline	Button	use_case_1	OPTIONS	When pressed; 1. A browse screen must appear to select the directory to save the guideline
Get Template	Button	use_case_2	OPTIONS	When pressed; 1. A browse screen must appear to select the directory to save the template
Back to main menu	Button	-	OPTIONS	When pressed; 1. CURRENT_SCREEN must be set to MAIN
Apply	Button	-	OPTIONS	When pressed;

				1. The values of the MATCH_LENGTH, COMMENTARY_SELECTION, PLAYERKIT_SELECTION must be saved.

Entity Identification	Type	Use Case	Screen	Function
Browse	Button	use_case_3	SINGLEPLAYER	When pressed; 1. A browse screen must appear to select the directory to load the user code 2. The system must show the result of load operation
Compile	Button	use_case_5	SINGLEPLAYER	When pressed; 1. The system must compile and check the user code 2. The system must show the result of the compile operation
Easy Difficulty	Radio Button	-	SINGLEPLAYER	When pressed; 1. OPPONENT_DIFFICULTY must be set to EASY
Medium Difficulty	Radio Button	-	SINGLEPLAYER	When pressed; 1. OPPONENT_DIFFICULTY must be set to MEDIUM
Hard Difficulty	Radio Button	-	SINGLEPLAYER	When pressed; 1. OPPONENT_DIFFICULTY must be set to HARD
Back to main menu	Button	-	SINGLEPLAYER	When pressed; 1. CURRENT_SCREEN must be set to MAIN

Start the game	Button	-	SINGLEPLAYER	When pressed; 1. CURRENT_SCREEN must be set to MATCH 2. MATCH_STATE must be set to READY

Entity Identification	**Type**	**Use Case**	**Screen**	**Function**
Browse 1	Button	use_case_3	MULTIPLAYER	When pressed; 1. A browse screen must appear to select the directory to load the code of the team 1 2. The system must show the result of the load operation of team 1
Browse 2	Button	use_case_3	MULTIPLAYER	When pressed; 1. A browse screen must appear to select the directory to load the code of the team 2 2. The system must show the result of the load operation of team 2
Compile 1	Button	use_case_5	MULTIPLAYER	When pressed; 1. The system must compile and the check the code of team 1 2. The system must show the result of the compile operation of team 1
Compile 2	Button	use_case_5	MULTIPLAYER	When pressed; 1. The system must compile and the check the code of team 2 2. The system must show the result of the compile operation of team 2

Back to main menu	Button	-	MULTIPLAYER	When pressed; 1. CURRENT_SCREEN must be set to MAIN
Start the game	Button	-	MULTIPLAYER	When pressed; 1. CURRENT_SCREEN must be set to MATCH 2. MATCH_STATE must be set to READY

Entity Identification	**Type**	**Use Case**	**Screen**	**Function**
Skip the match	Button	-	MATCH	When pressed; 1. The system must show the final match result 2. End match button must be visible
Exit the match	Button	use_case_10	MATCH	When pressed; 1. The system must stop the visualization of the game 2. CURRENT_SCREEN must be set to MAIN
End Match	Button	-	MATCH	When pressed; If MATCH_STATE is FINISHED 1. CURRENT_SCREEN must be set to MAIN
Start Match	Button	use_case_6	MATCH	When pressed; If MATCH_STATE is READY; 1. MATCH_STATE must be set to PLAYING 2. The system must start to visualize the game up to MATCH_LENGTH. If MATCH_LENGTH value is reached, MATCH_STATE must be set to FINISHED 3. Start Match button must become the Pause

				Match Button
Pause Match	Button	use_case _7	MATCH	When pressed; If MATCH_STATE is PLAYING; 1. MATCH_STATE must be set to STOPPED 2. The system must stop to visualize the game 3. Pause Match button must become the Resume Match button
Resume Match	Button	use_case _8	MATCH	When pressed; If MATCH_STATE is STOPPED; 1. MATCH_STATE must be set to PLAYING 2. The system must continue to visualize the game 3. Resume Match button must become the Pause Match button

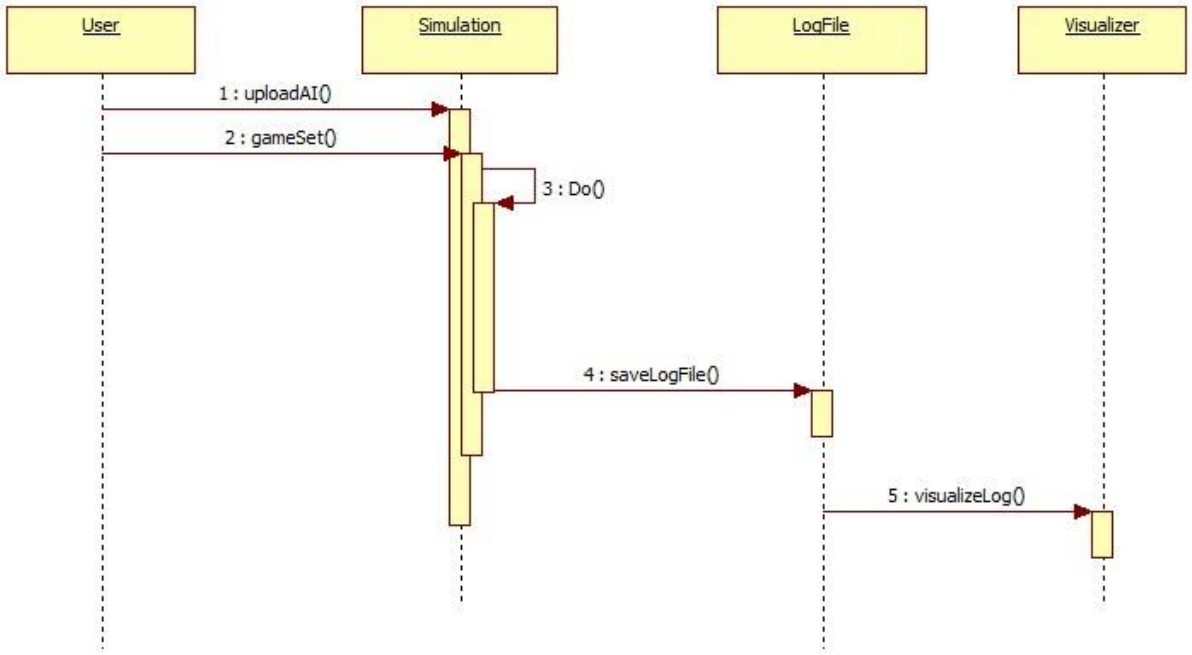

Figure 13 Sequence Diagram

5.7 State Dynamics Viewpoint

In this section, an illustration of our simulation steps is shown. At the beginning of program execution, user can select former imported AI agents to play game, or directly can go to upload state. Moreover user has an opportunity to view saved log file with visualizer. If user uploads his/her AI, system will check validity of this file and then system goes to “ready state”. After that user should select game preferences and according to these settings, simulation will start. Later user can save log file and visualize it.

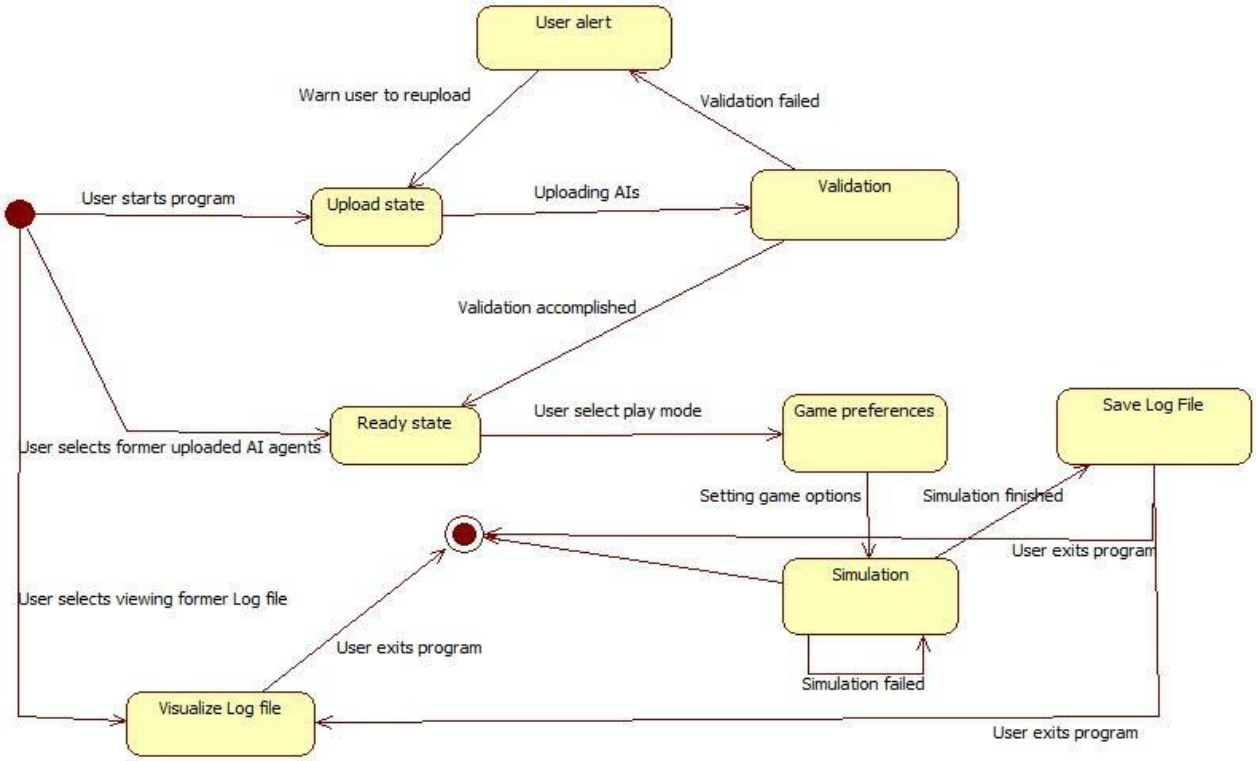

Figure 14 State Diagram

5.8 Information Viewpoint

In the project, there will be only one type of persistent data storage. This data file will be saveFile. The save file will store the percepts in each turn. It will also have a metadata section which will contain general information. For each turn, not to store whole player entities, metadata will store team id – player ids relation table. The simulator will write to these files. They will be used by the visualizer to display the game after the simulation.

Structure of the saveFile:

- It will be in the format of JSON.

- Each JSON object will have the information about a percept with the positions of the players and the ball, the score and the turn number.
- The number of JSON objects will be equal to the number of turns.

Operations on the file:

The file will be stored in a directory that both the simulator and the visualizer can access to it. The simulator will be responsible for the creation of the file. The save file of each match will have a different name formatted as timestamp.

The simulator will have the right to write to the file. In each turn, it will add a new JSON object at the end of the file.

The visualizer can only read from the file. It cannot change its content. It will read the file, parse it and then visualize the percepts.

Example of saveFile


```
{ "metadata": {
  "team1": {"id": 0,"name": "Carlos", "colorR": 0, "colorG": 0,"colorB": 0, "players": [
 {"id": 0,"name": "I aint got no name!","kitNumber": -1},
 {"id": 0,"name": "I aint got no name!","kitNumber": -1},
 {"id": 0,"name": "I aint got no name!","kitNumber": -1} ]},
  "team2": {"id": 0,"name": "Roberto","colorR": 0,"colorG": 0,"colorB": 0,"players": [
 {"id": 0,"name": "I aint got no name!","kitNumber": -1},
 {"id": 0,"name": "I aint got no name!","kitNumber": -1},
 {"id": 0,"name": "I aint got no name!","kitNumber": -1}]},
  "pitchWidth": 0,"pitchHeight": 0},
  "perceptList": [{"ball": {"position": {"x": 64.0,"y": 48.0},"speedX": 0.0,"speedY": 0.0},
  "playerInfoList": [{"id": "0","position": {"x": 12.0,"y": 61.0}},
  "team1Score": 0,"team2Score": 0}]}
```

6. Planning

6.1 Estimation

This is our Gantt chart representation of weekly planning schedule.

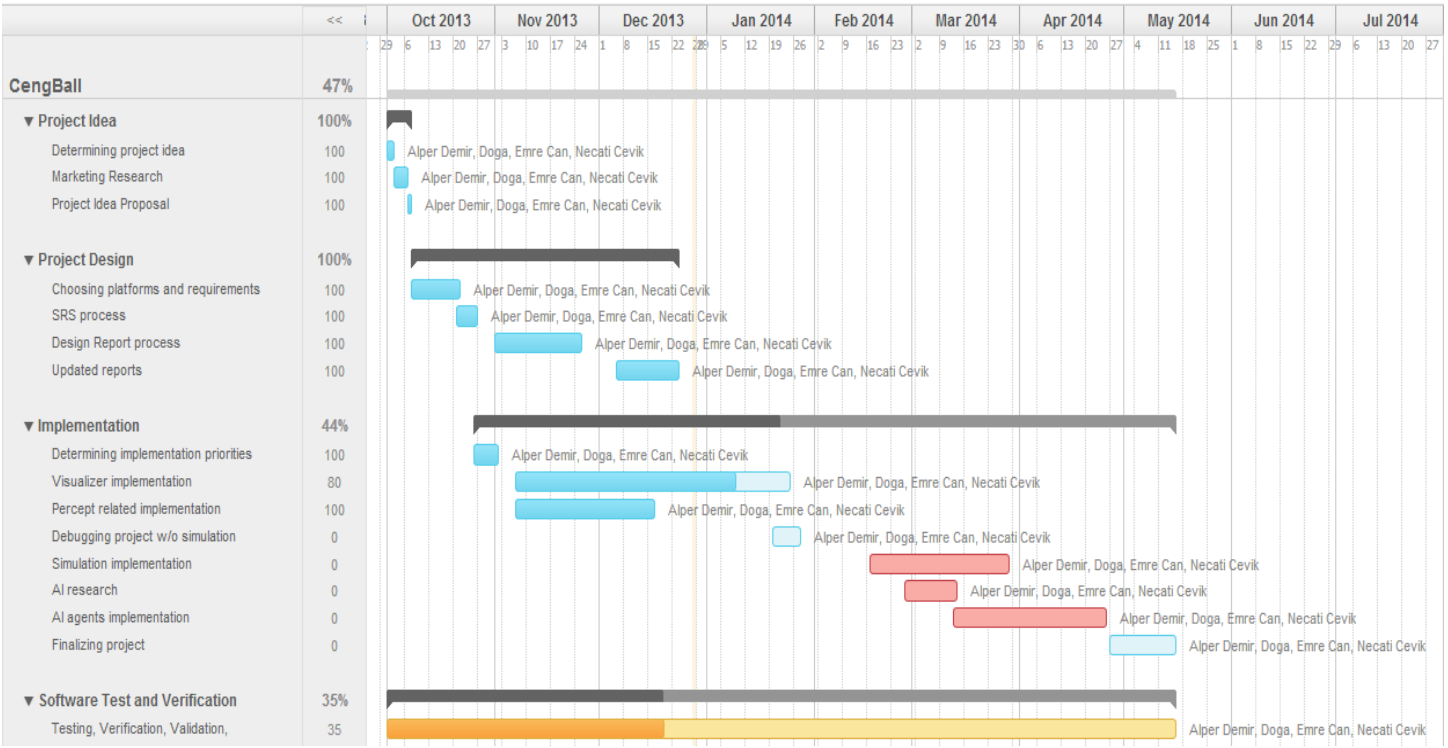

Figure 15 Project Estimation Gantt Chart

7. Conclusion

This SDD is prepared to give detailed information for all design patterns of CEngBall Project. First, general overview and definitions of project were given. Then, relative information about design's concerns is mentioned. After that, system architecture is provided with all of its components. In the following sections, user interfaces and actions of objects are stated in design viewpoints section.