CANICULA CAMPUSUS

DESIGN SPECIFICATION

Contents

1. INTRODUCTION

The aim of this report is to present a detailed design description of the Cannicula Campusus. Game's name coming from latin word Canicula which means in English "Dog", and fabrication word Campuses. In this document, we will mention about data model, software model and user interface model. In this way, we will explain project's goal, content and organization.

1.1 Goals

The goal of the project is to develop Android game application with the aid of Unity 3D Platform. This project is game for android users. Our game encapsulate 6 different modes which are chasing cats, daily routine of a dog, running from cars in a traffic in the campus, fastest arrive of a point, daily challenge and multiplayer area wars. One of the most important aim of this project is to provide fun to our game users and have good times with their friends.

The goals of the project team can be listed as follows:

- To have created a fun game that keeps player playing.
- To have created a game that will target users who enjoy playing casual styles mobile gamers.
- To establish game mechanics that will improve the gameplay in a way that will be appealing to the player.
- To provide balance between game difficulty level and user experience.

1.2 Contents and Organization of the Document

Organization of this document is as the following;

We will provide general data model, important data considerations and data flow diagram in terms of data model. Futhermore, we will prepare system architecture, component diagram, class diagram and sequence diagram in the software model part. Finally, we will show important screen mock-ups in user interface model part.

Explanation of the content is below:

- **Data Model:** It is the process of creating a data model by applying formal data model descriptions using data modeling techniques. The data model is eventually implemented in a database.
    - **General Data Model:** This part, we will show entity relationship diagram of our project.

    - **Important Data Considerations**: In this stage,we will give some information about data format preferences which are related to Unity 3D, SQL and Remote Procedure Calls.
    - **Data Flow**: In this phase, we will indicate data flow diagram and mention its explanation.
- **Software Model:** The development models are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives.
    - **System Architecture:** This part, we will explain our project's architecture with the aid of diagram.
    - **Component Diagram:** In this phase, inputs and outputs of each component and services should be included.
    - **Class Diagram:** Class diagrams are displayed for each component in this stage.
    - **Sequence Diagam:** In the last stage, we will indicate sequence diagram for each component in terms of user stories via our requirements specification.
- **User Interface Model:** Finally, we will display some screen mocks-up of our game project. Our screen are listed as below:
    - Menu
    - Load
    - Options
    - Credits
    - High Scores
    - Register
    - Login

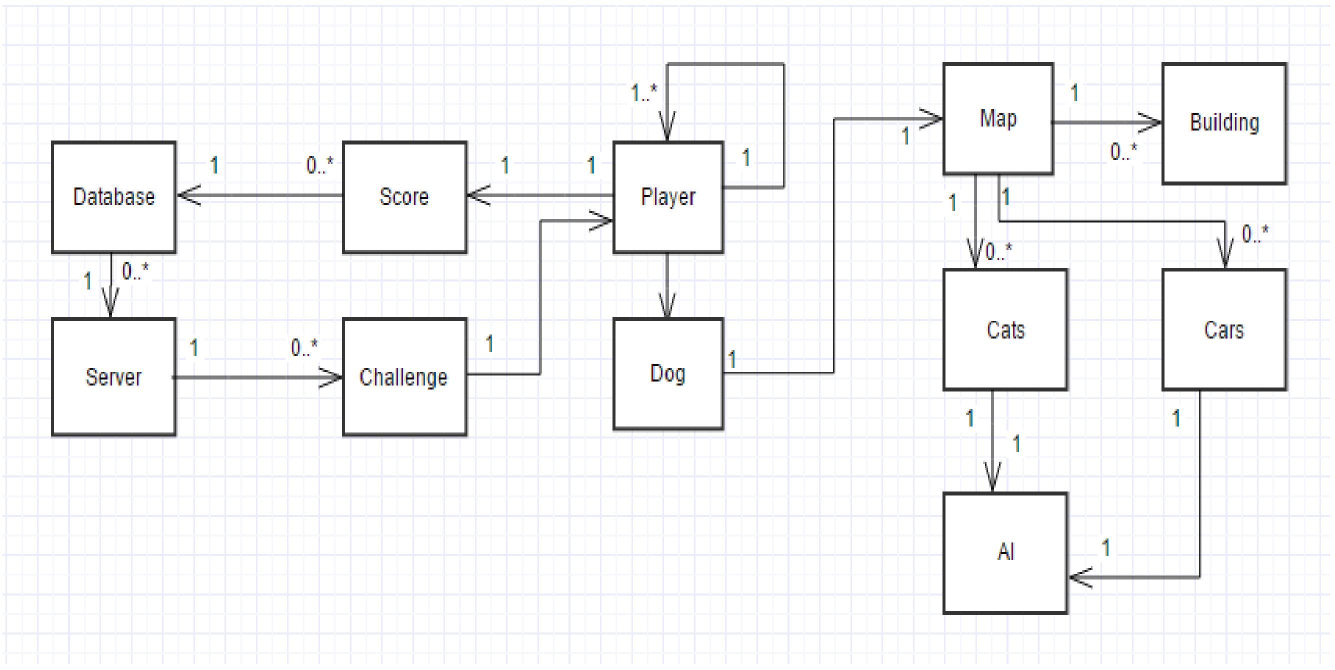
2. DATA MODEL

Data modeling is a process used to define and analyze data requirements needed to support the business processes within the scope of corresponding information systems in organizations.

2.1 General Data Model

This part, we will show conceptual model from our requirements specifications report and we will create ER Diagram which is based on this model.

Our conceptual model can be found below:



Conceptual Model

Our database consists of four database tables. These tables are:

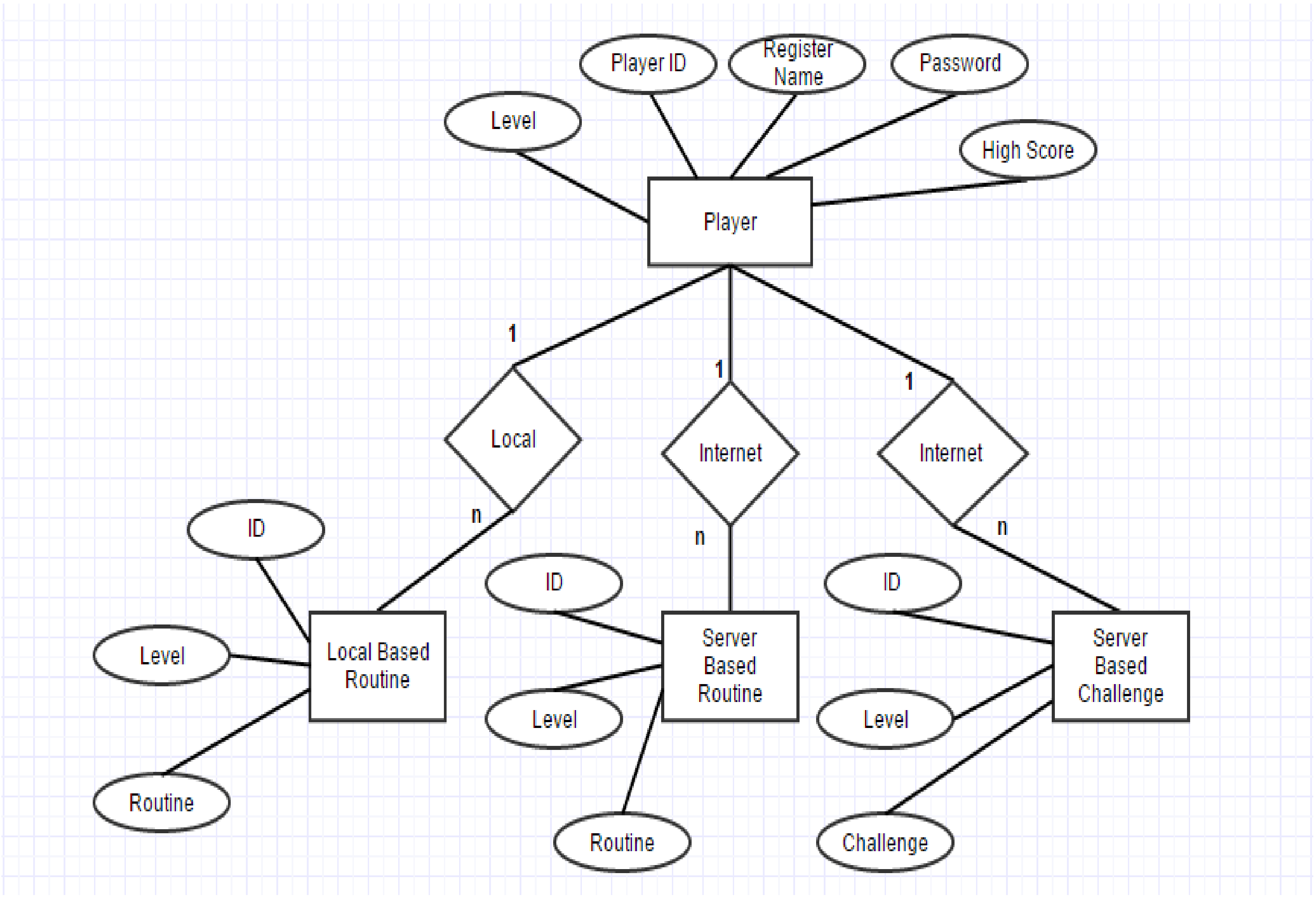
- Player: This table keeps the necessary player informations.
  - Player ID
  - Register Name
  - Password
  - Level
  - High Score
- Challenge: This table keeps the informations of new challenges.
  - ID
  - Challenge
  - Level
- Server Based Routine: This table is kept in the server side that keeps server based routine informations.
  - ID
  - Routine
  - Level
- Local Based Routine This table is kept in the local side that keeps local based routine informations.
  - ID
  - Routine
  - Level

There is no need for different account and character tables since one account can only have one character. Users can log in to different accounts which means different characters. These informations are kept in one table which is our player table.

Moreover, our another tables which are called Server Based Routine and Local Based Routine. Usage of these tables differs on the mode the user plays. Server Based Routine is used for network connected routine gameplay and Local Based Routine is used for local routine gameplay.

In addition, high score informations are also being kept in the player tablet hat is stored locally.

Our ER Diagram is shown as the following:



ER Diagram

2.2 Important Data Considerations

In Unity 3D, neccassary informations will be stored in the phone by using PlayerPrefs class and its proper methods. Methods of PlayerPrefs class can be found below:

- **DeleteAll**: Removes all keys and values from the preferences.
- **DeleteKey:** Removes key and its corresponding value from the preferences.
- **GetFloat:** Returns the value corresponding to key in the preference file if it exists.
- **GetInt:** Returns the value corresponding to key in the preference file if it exists.
- **GetString:** Returns the value corresponding to key in the preference file if it exists**.**
- **HasKey:** Returns true if key exists in the preferences.
- **Save:** Writes all modified preferences to disk.
- **SetFloat**: Sets the value of the preference identified by key.
- **SetInt:** Sets the value of the preference identified by key.
- **SetString**: Sets the value of the preference identified by key.

In this project, database informations on the web are handled via SQL, connection to the SQL database will be created through the Unity 3D network capabilities.

Unity 3D makes extensive use of the Network View component to share data over networks. One of the functions provided by Network Views is Remote Procedure Calls (RPC). This enables us to call a function on one or more remote machines which may be clients or servers. Remote Procedure Calls will be used to synchorinize locations of the players and projectiles in multiplayer gameplays.

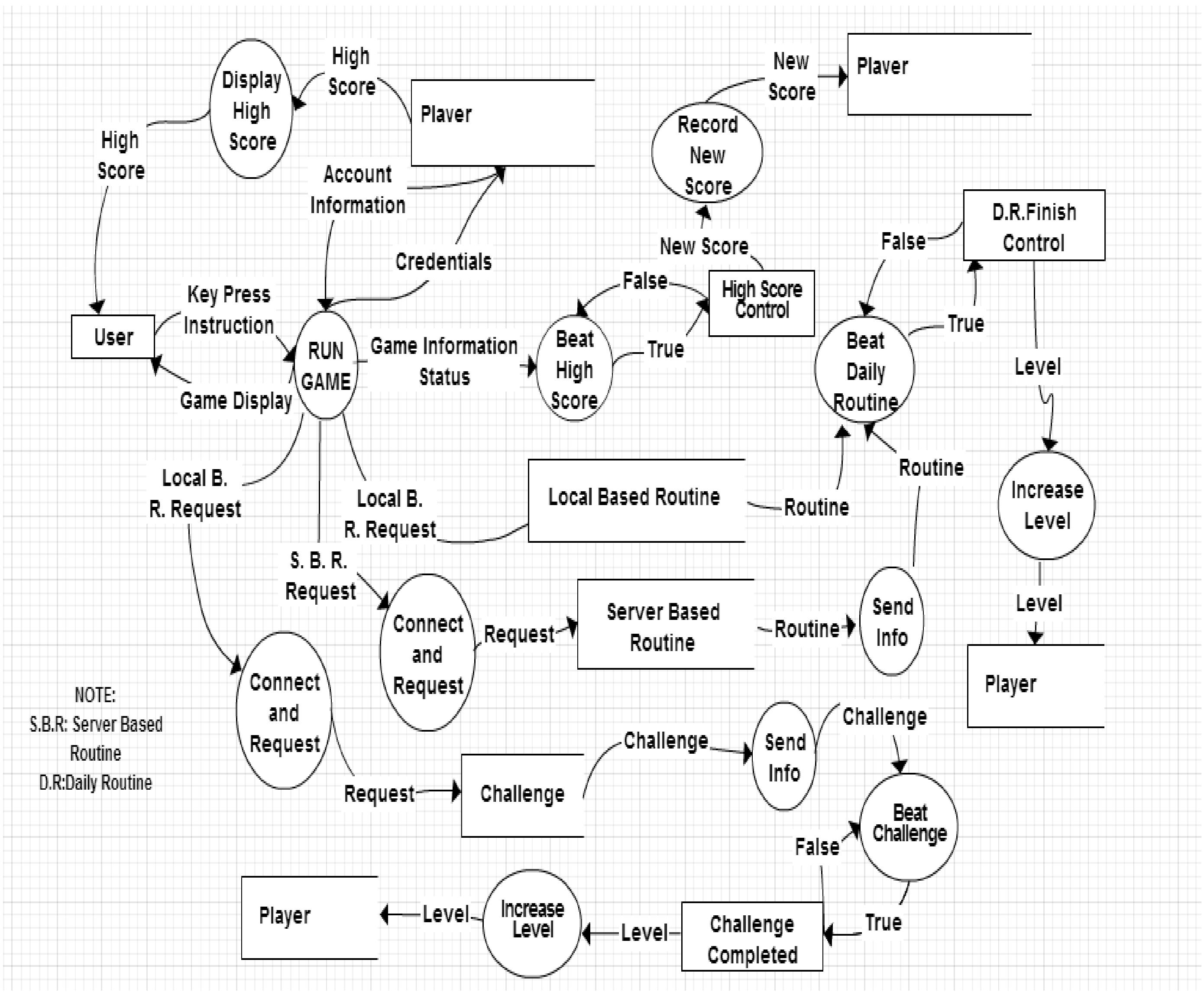
2.3 Data Flow Diagram

A Data Flow Diagram shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. . A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

Explanation of Data Flow Diagram

In our data flow diagram, User will run the game through instructions and key presses. Output of the run game process will be displayed to the user. From the run game process with the correct data and appropriate processes three database tables (Local Based Routine, Server Based Routine, Challegenge) will be reached and with proper flags and dataflows, update processes will be invoked and suitable informations on the player table will be updated. Users can also see high scores with the aid of high scores data flow from the player table to the user through the display high score process.

Our Data Flow Diagram is shown as the following:



Data Flow Diagram

3. SOFTWARE MODEL

3.1 System Architecture

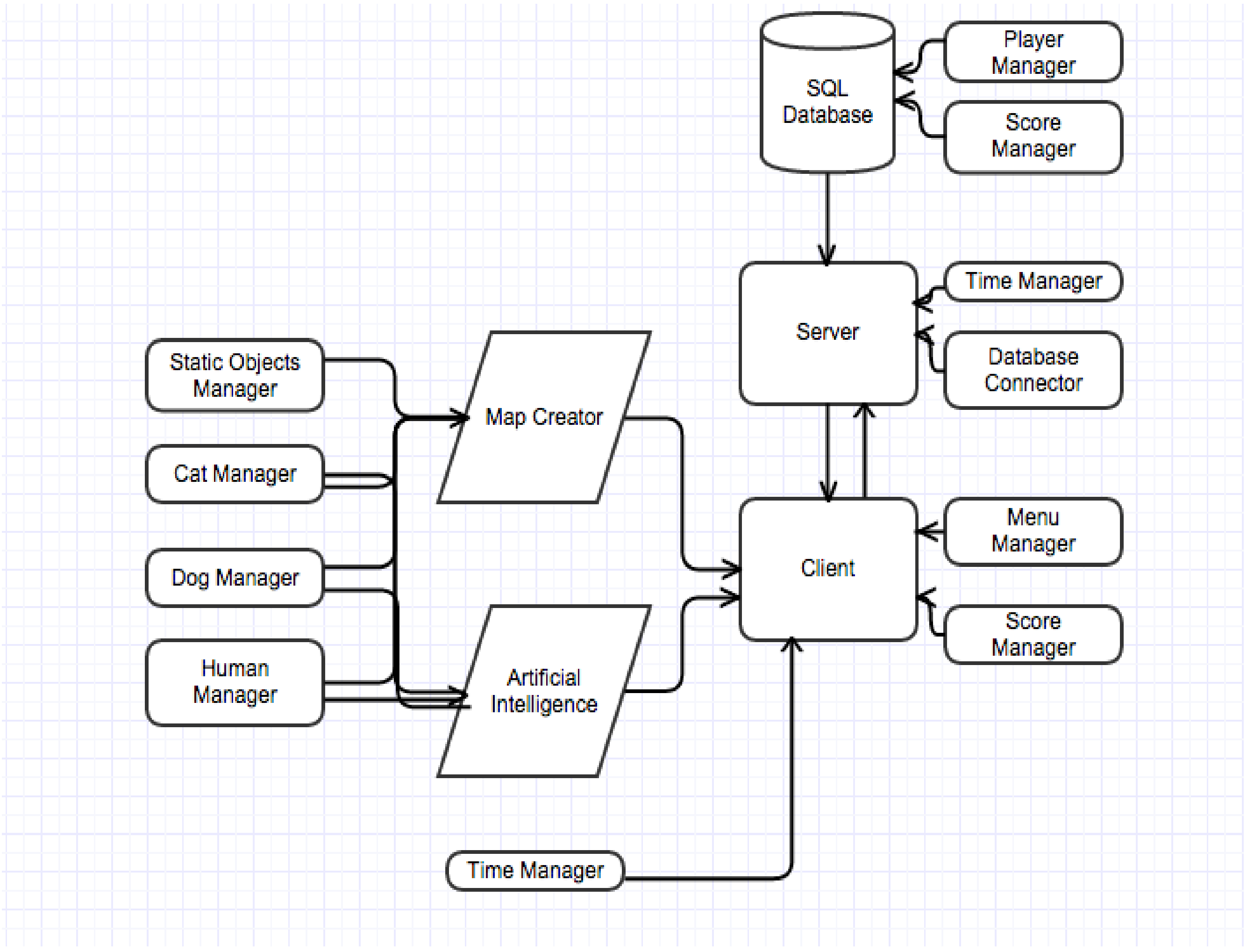
Our architecture uses database for two main reasons; one of them is keeping the player information and the other one is keeping the scores of the players.

To connect between server and database, server needs to database connector. Server also needs a timer to carry out multi player game mode.

Client, which is smart phone or tablet in this case, can navigate via menu manager. Score manager keeps the local scores for a player.

System needs static, dynamic objects and timer to construct a map. Any dynamic object in the map, needs an artificial intelligence to move in the map.

Our System Architecture Diagram is indicated as below:



System Architecture Diagram

3.2 Class Diagram

In our project, there are 10 classes. These are Menu, Register, Game, Score, Map, Statics, Dynamics, Dog, Car and Cat.

Menu: This class is first activated class to play game, login, save, view the credits, see the highest score and register. This class has directed association with Score class, aggregation with Register class and composition with Game class.

Register: This class is for registering the user to the system and it has aggregation with Menu class.

Game: This class sets the game environment and starts the selected game mode. It has composition with Menu class and Map class, has dependency with Score class.

Score: This class is about changing the current and highest score of the user. It has composition with Menu classes and dependency with Game class.

Map: This class is for changing and setting the map with objects in synchronized or not synchronized mode. It has composition with Game, Statics and Dynamics classes.

Statics: This class is for the static objects in the map. It has composition with Map.

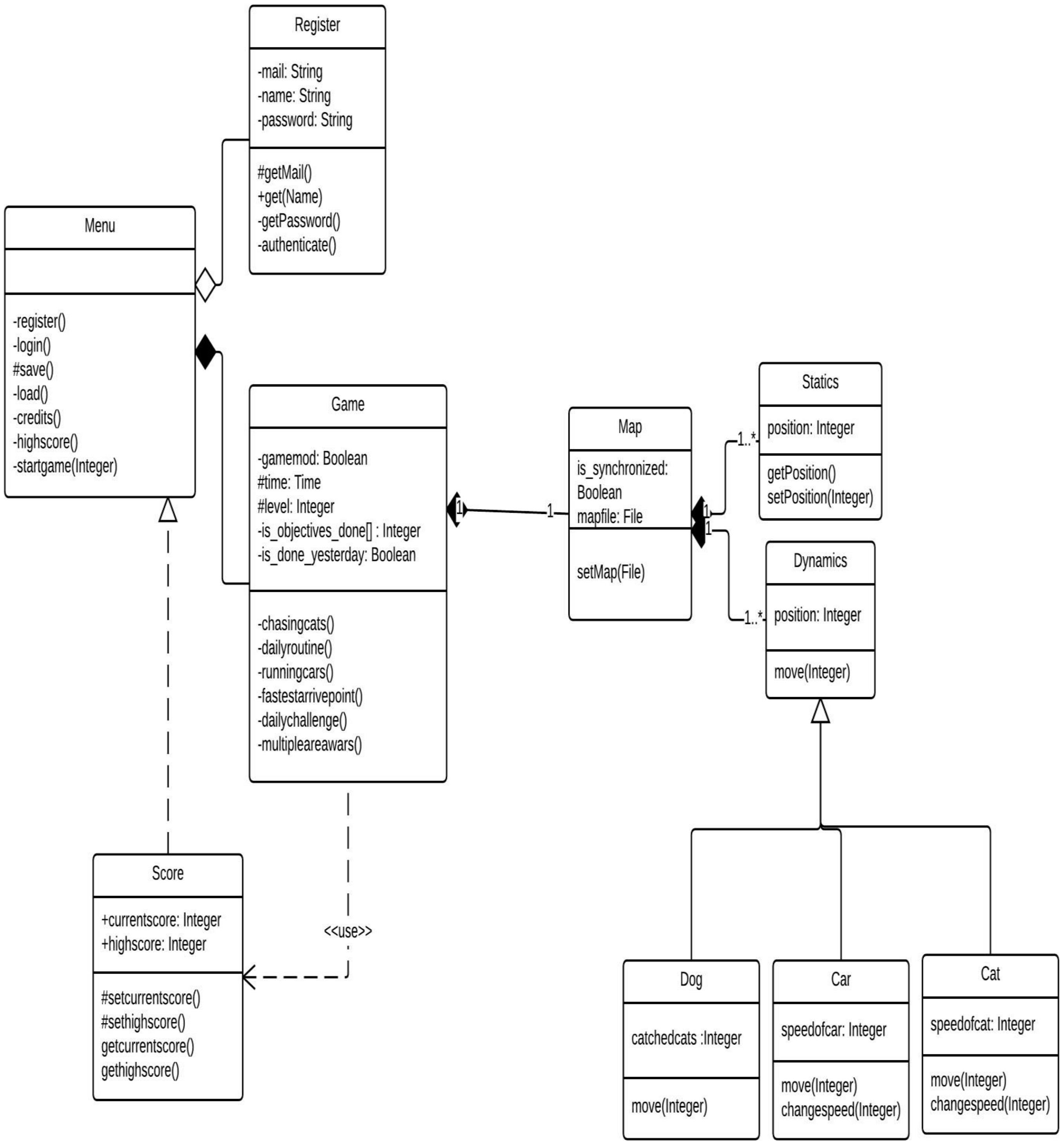
Dynamics: This class is for the dynamic objects in the map. It has composition with Map, generalization with Cat, Car, Dog classes.

Dog: This class is for the dog which is the main character of the game and it has generalization with Dynamics class.

Car: This class is for moving cars in the game environment and kills the dog. It has generalization with Dynamics class.

Cat: Cat is implemented in this class which runs from dog and it has generalization with Dynamics class.

Our Class Diagram is indicated as below:



Class Diagram

3.3 Component (Package) Diagram

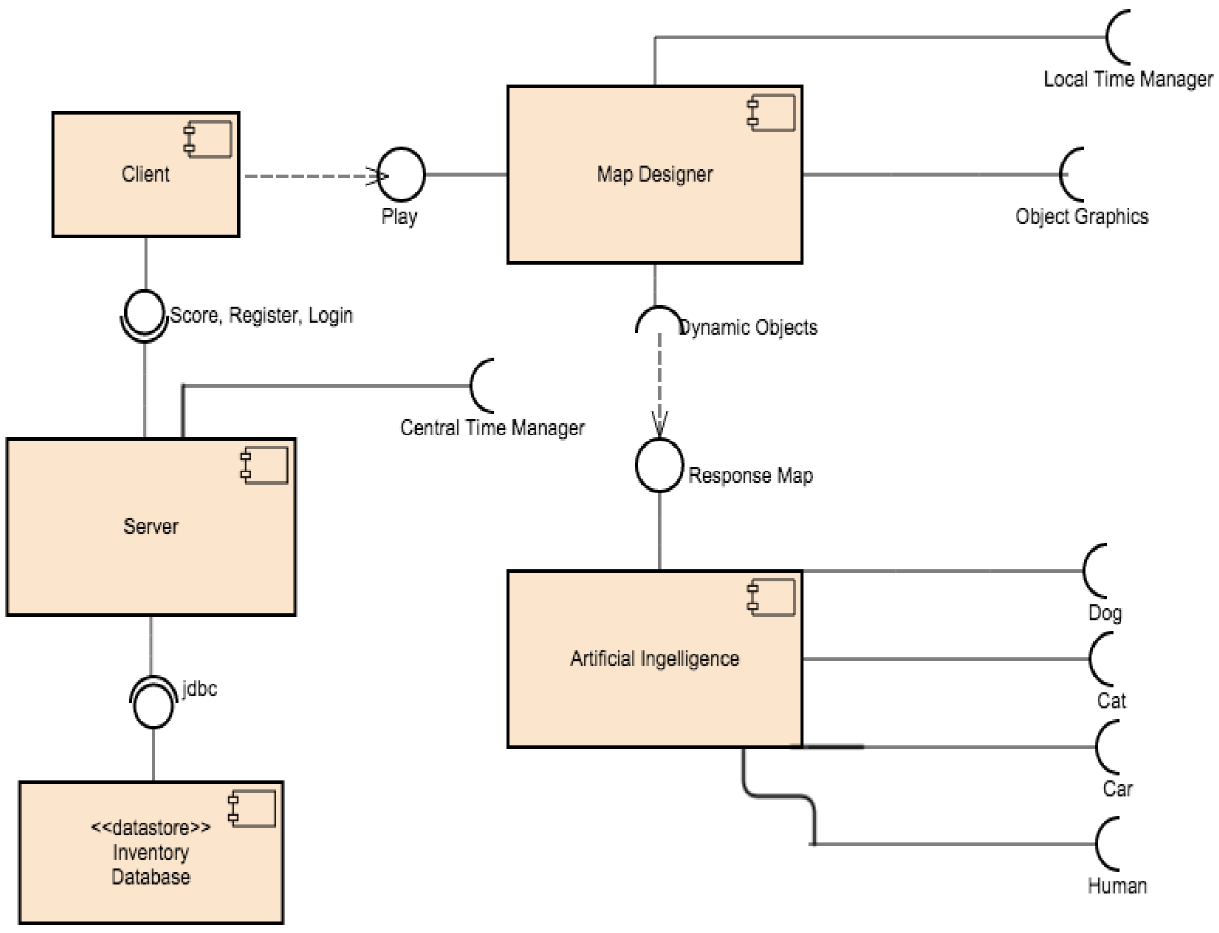
System basically exists from two component which are 'Server' and 'Client'. Both of them has Timer Manager because, either single player game modes or multi player game modes needs protected time information to create game map, and use artificial intelligence.

Server component connects a database component which keeps information about both players and their scores.

All of the games are operated on Client side. If game mode includes multi player attribute, than it also should get time information from central time manager which is connected from server component to synchronize both clients. In other game modes, since synchronization is not a issue, client uses own time manager.

Once the game started, Client first calls Map Constructor to build the map. Map Constructor needs graphics and static/dynamic objects to fulfil the creation of map. After all, the map was created successfully, artificial intelligence plays with the dynamic objects.

Our Component Diagram is shown as the following:



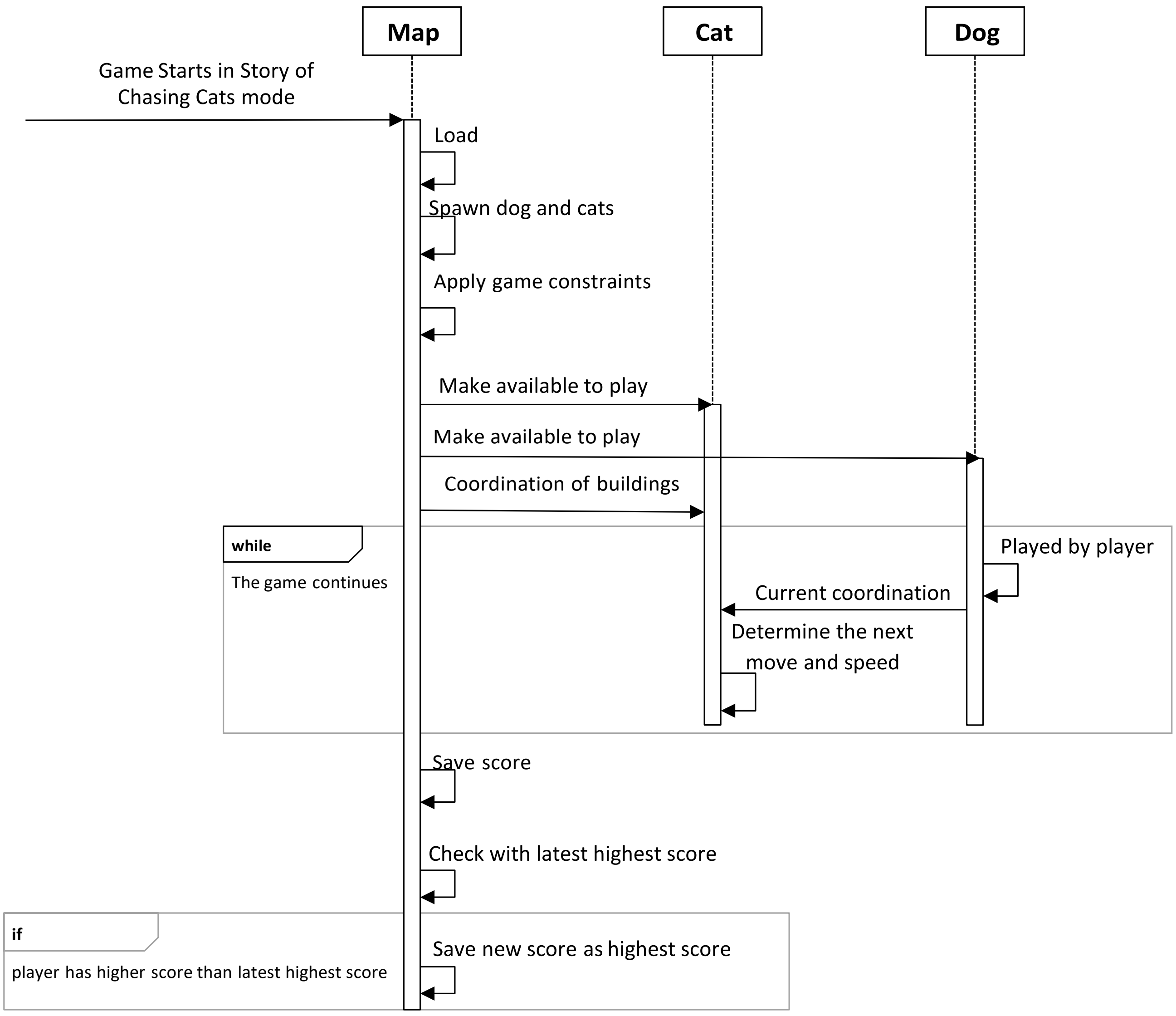
Component Diagram

3.4 Sequence Diagram

Our Sequence Diagram is created according to our user stories.

Story of Chasing Cats

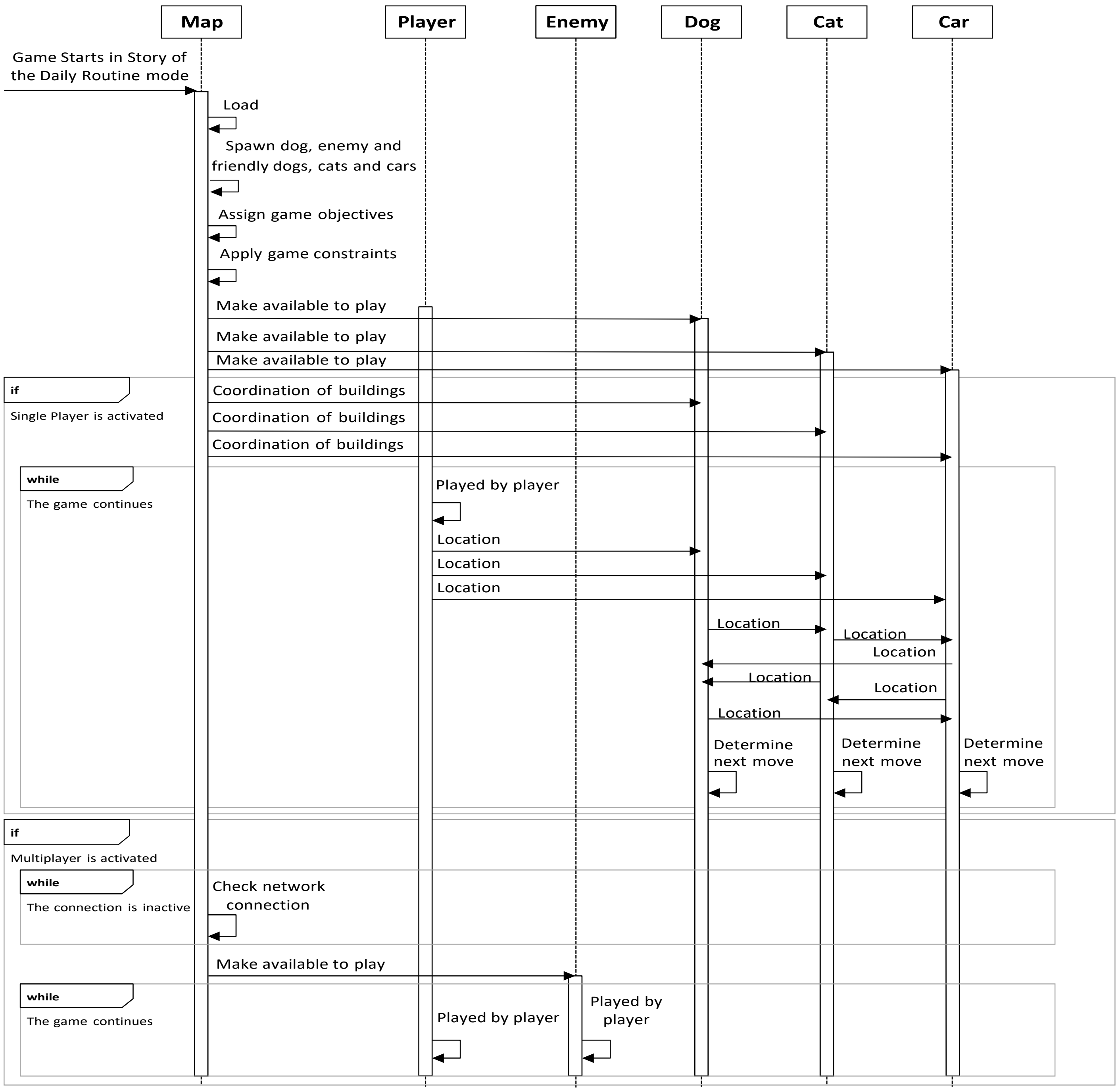
The Story of Chasing Cats mode contains three objects, which are the map, cats and dogs. When the game starts in the Story of Chasing Cats mode, the map object becomes active. Then, the map is loaded, the dog and cats are spawned and the game constraints are applied. Then the map object activates the dog and cat objects by making them available to play. At the beginning of the game, the map object sends information about the coordination of buildings to the cats, which are controlled by artificial intelligence. Then in the game, as the player manages the dog, the coordination of the dog is sent to the cat object and according to the coordination of buildings and the dog, the artificial intelligence determines the next move and speed of cats while the game continues. When the game finishes, score of the player is saved and compared with the latest highest score to check whether the player has higher score than the latest highest score and new score is saved as the highest score if new score is higher than the existing highest score.


Map
Cat
Dog
Game Starts in Story of Chasing Cats mode
Load
Spawn dog and cats
Apply game constraints
Make available to play
Make available to play
Coordination of buildings
while
The game continues
Played by player
Current coordination
Determine the next move and speed
Save score
Check with latest highest score
if
player has higher score than latest highest score
Save new score as highest score

Story of Chasing Cats

Story of the Daily Routine

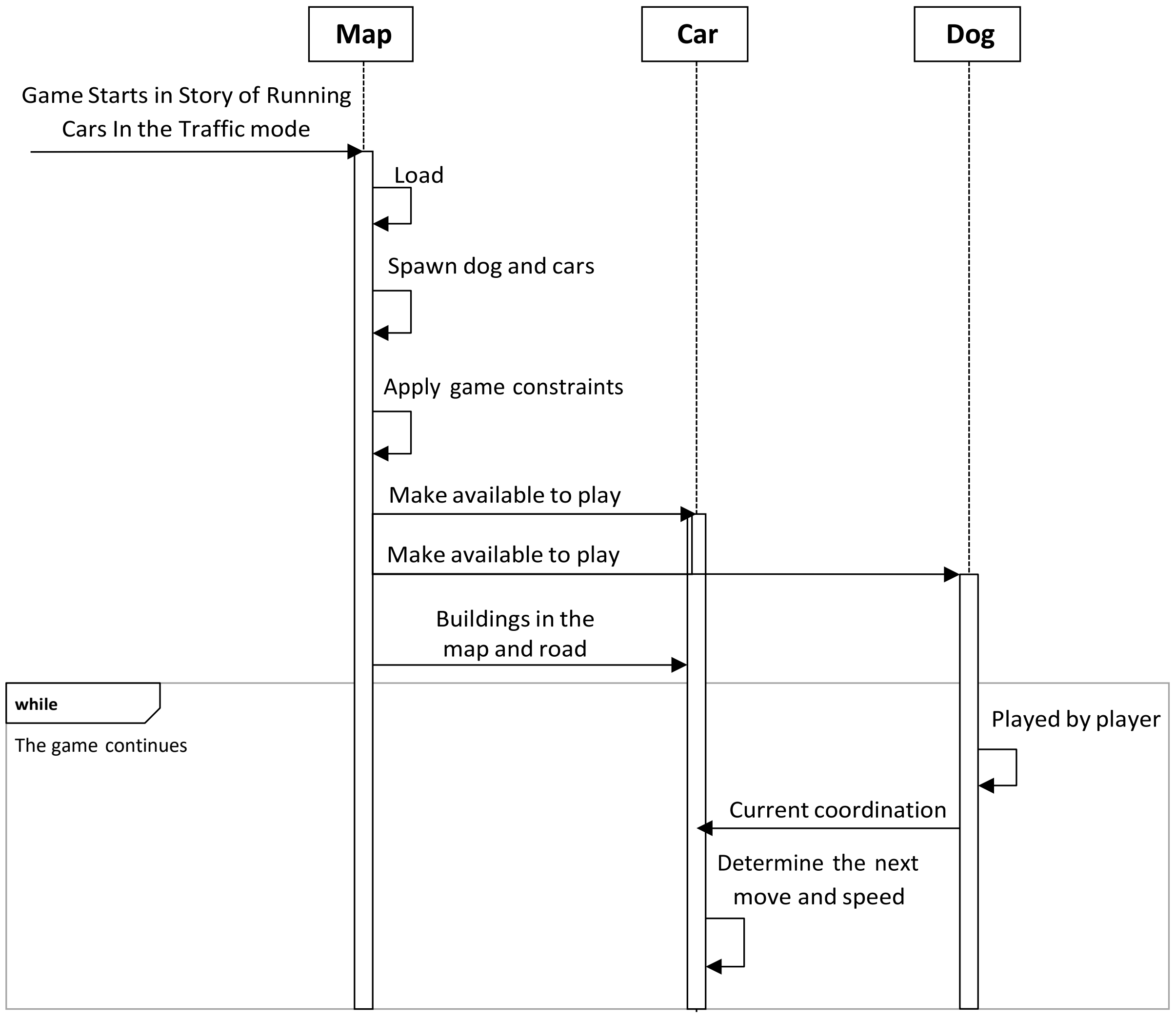
The Story of the Daily Routine mode contains six objects, which are the player, the enemy player, dogs, cats and cars. When the game starts in Story of the Daily Routine mode, the map object becomes active. Then, enemy and friendly dogs, cats and cars are spawned, game objectives are assigned and game constraints are applied. Then the map object activates the dog, cat and car objects by making them available to play. Then, the game mode is checked to see whether game is started in Single Player or Multiplayer Mode. If game is started in Multiplayer mode, network connection is checked to see whether an opponent player is available. Then, the game starts when an opponent player is available.



Story of Daily Routine

Story of Running Cars In The Traffic

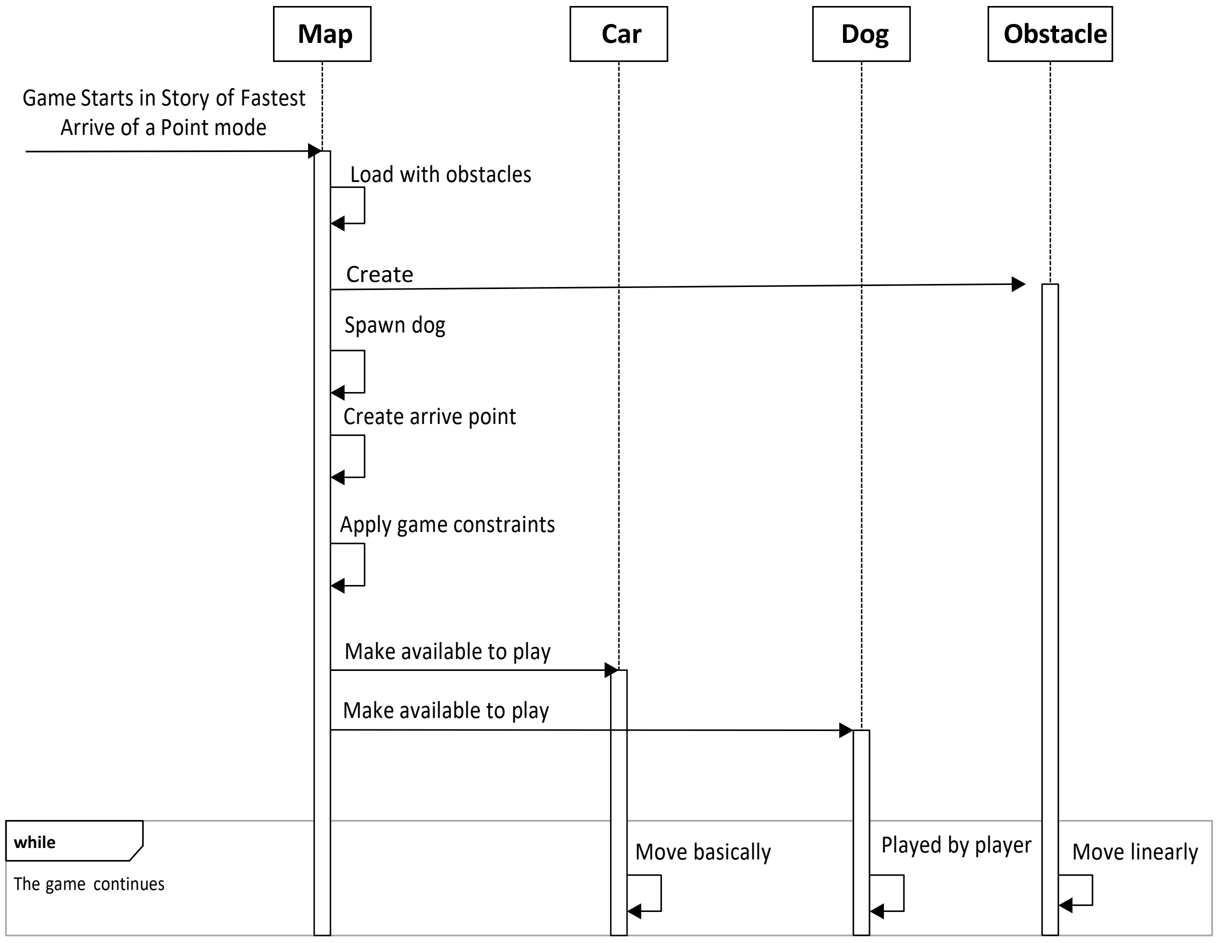
The Story of Running Cars in the Traffic mode contains three objects, which are the map, the dog and the cars. When the game starts in Story of Running Cars in the Traffic mode, the map object becomes active. Then, the map is loaded, the dog and cars are spawned and the game constraints are applied. Then the map object activates the dog and car objects by making them available to play. At the beginning of the game, the map object sends information about the buildings in the map and road to the cars, which are controlled by artificial intelligence. Then in the game, as the player manages the dog, the coordination of the dog is sent to the car object and according to the coordination of buildings and the dog and road, the artificial intelligence determines the next move and speed of cars while the game continues.



Story of Running Cars In The Traffic

Story of Fastest Arrive of a Point

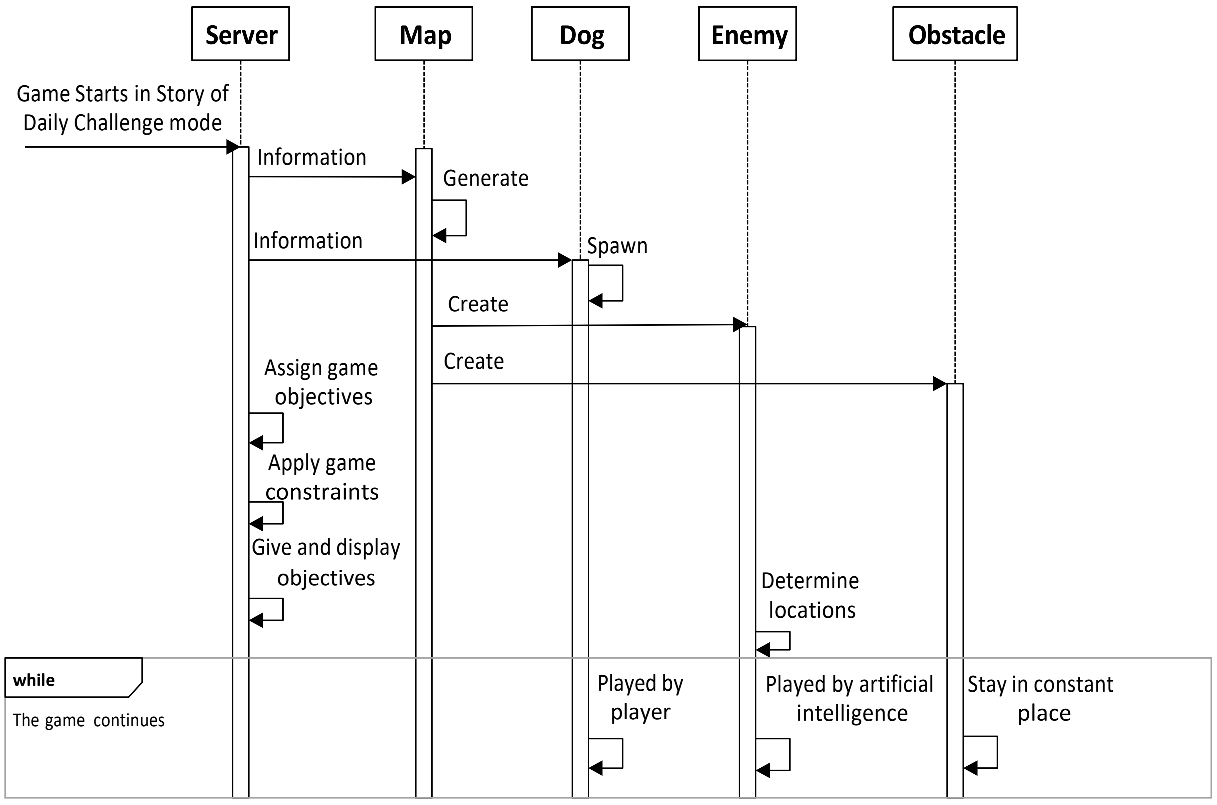
The Story of Fastest Arrive of a Point mode contains four objects, which are the map, the dog, cars and obstacles. When the game starts in Story of Fastest Arrive of a Point mode, the map object becomes active. Then, the map is loaded with obstacles and the obstacles object become active when the map creates obstacles. Then, the dog is spawned, the arrive point is created and the game constraints are applied. Then the map object activates the dog and car objects by making them available to play. In the game, as the player manages the dog, the cars move basically and the obstacles move linearly.



Story of Fastest Arrive of a Point

Story of Daily Challenge

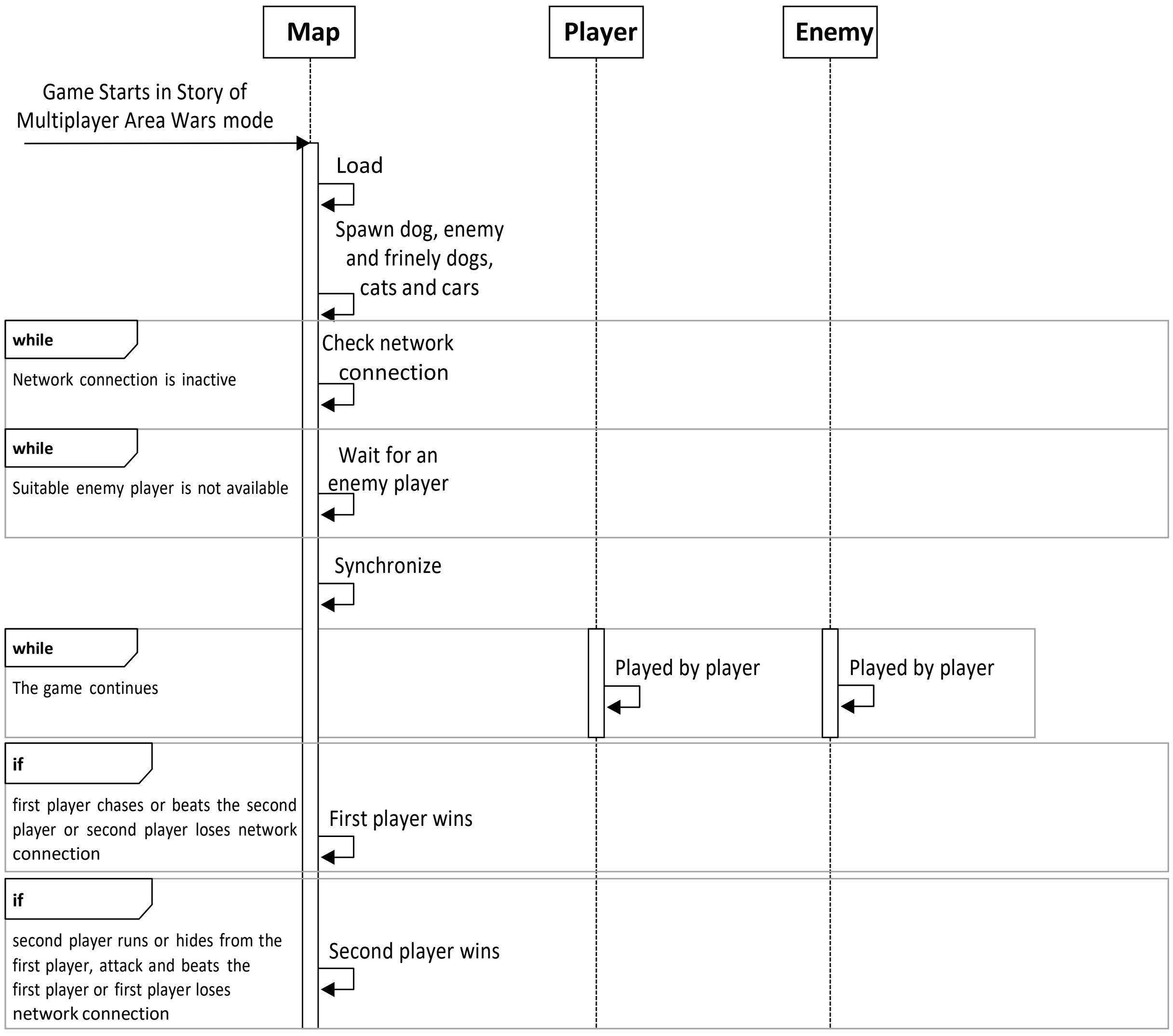
The Story of Daily Challenge mode contains five objects, which are the server, the map, the dog, the enemy and the obstacles. When the game starts in Story of Daily Challenge mode, the server and the map objects become active and the map is generated according to the server based information. Then, the dog object becomes active when the server sends information to the dog and the dog is spawned according to the information given by the server. Then, the map object activates the enemy and obstacles by creating them. After that, game objectives are assigned, game constraints are applied and the objectives are given and displayed to the player. At the beginning of the game, locations of enemies are determined by artificial intelligence. In the game, the dog is managed by player, enemies are played by artificial intelligence and the obstacles stay in constant place.



Story of Daily Challenge

Story of Multiplayer Area Wars

The Story of Multiplayer Area Wars mode contains three objects, which are the map, the player and the enemy. When the game starts in the Story of Multiplayer Area Wars mode, the map object becomes active. Then, enemy and friendly dogs, cats and cars are spawned and network connection is checked until it becomes active. After that, an opponent player is waited to connect to the game for the game to begin. When, a player is connected to the game, game map is synchronized between two players and the game begins. In the game, first player who is in the Story of Multiplayer Area Wars mode, should beat the second player who is in the daily routine mode, in a certain time in order to win the game. If second player can beat the first player or if the second player can run or hide from the first player, second player will win. And if one of the player exits the game or loses the network connection, enemy of this player is accepted as winner.

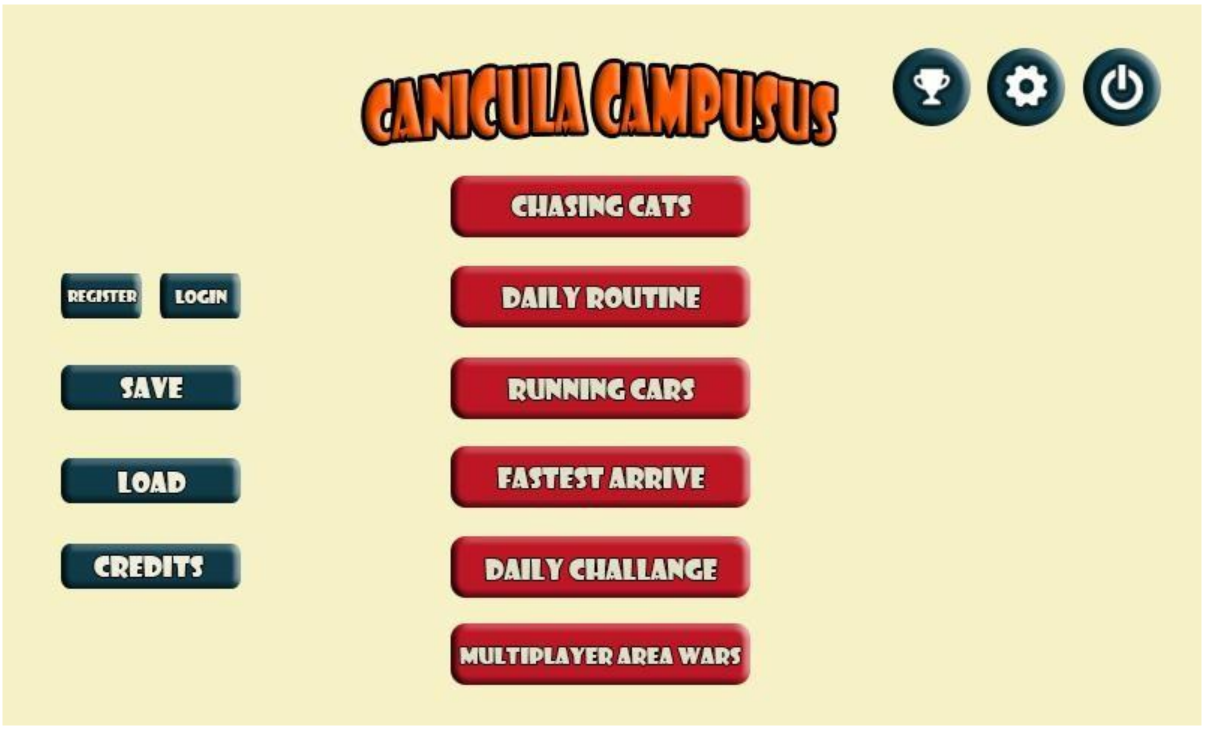

Map
Player
Enemy
Game Starts in Story of Multiplayer Area Wars mode
Load
Spawn dog, enemy and frinely dogs, cats and cars
while
Network connection is inactive
Check network connection
while
Suitable enemy player is not available
Wait for an enemy player
Synchronize
while
The game continues
Played by player
Played by player
if
first player chases or beats the second player or second player loses network connection
First player wins
if
second player runs or hides from the first player, attack and beats the first player or first player loses network connection
Second player wins

Story of Multiplayer Area Wars

4. USER INTERFACE MODEL

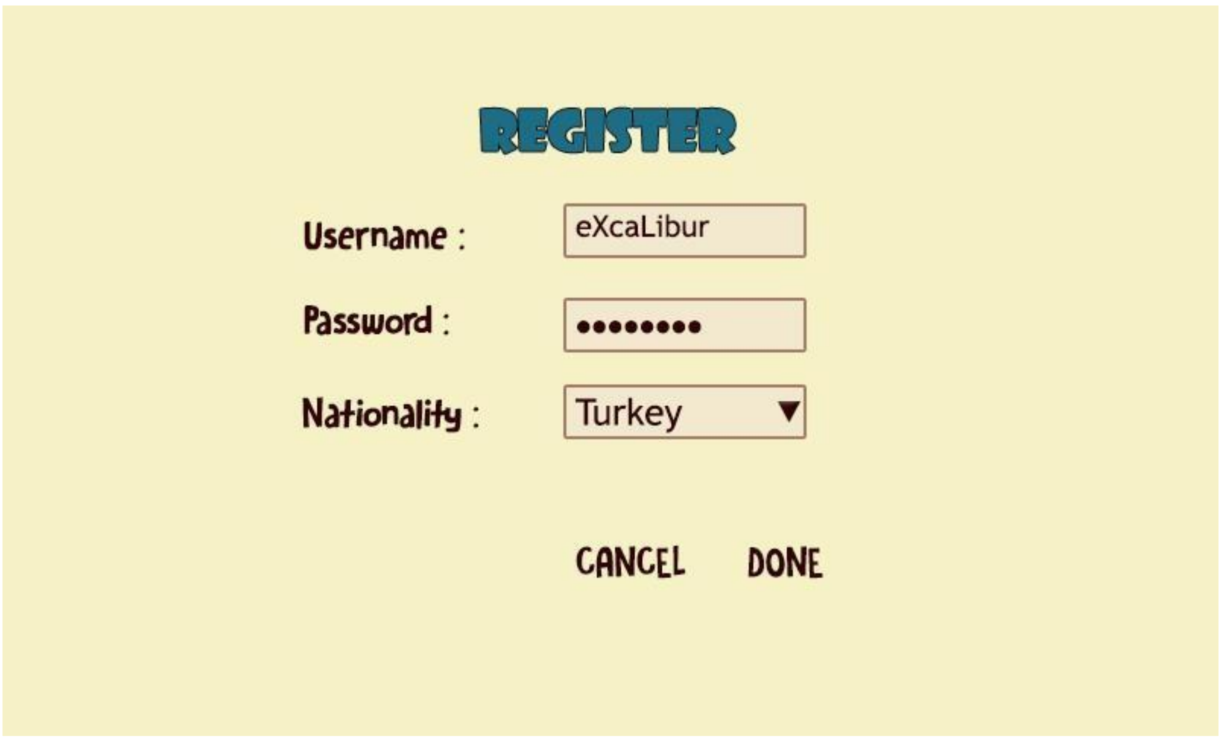
4.1 Main Menu

The name of the game placed on the top of the screen. Game mode buttons are commonly used piece of the main menu, so all of them are placed on the center of the screen. “Exit”, “Options” and “High Scores” buttons are put on the top right-hand corner to access quickly. “Register”, “Login”, ”Load”, ”Save” and “Credits” buttons are put on the center left-hand side to be seen easily. Colors of buttons and background are specified according to increase ease of use.



4.2 Register

The name of the sub menu is placed on the top of the screen. There is three text box in this page. “Username”, “Password” and “Nationality” entries can be given from proper fields. “Cancel” and “Done” buttons are located on the bottom of the page.



REGISTER

Username : eXcaLibur

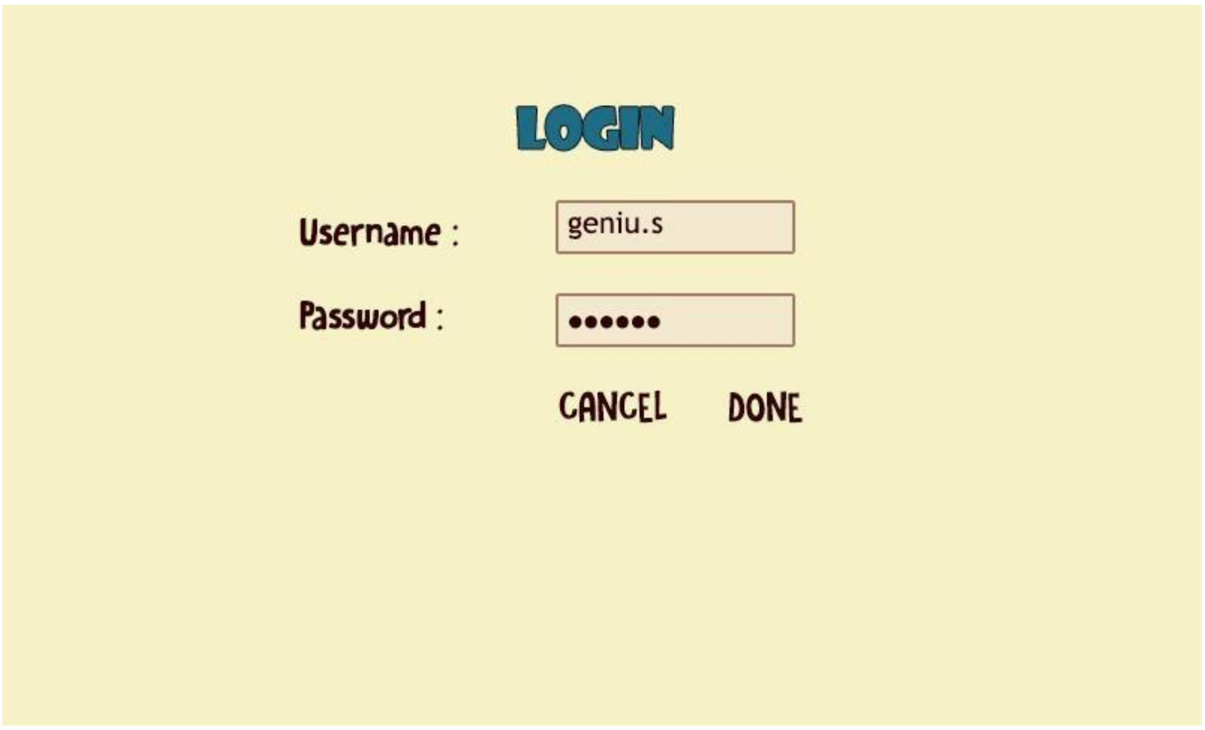
Password : ••••••••

Nationality : Turkey ▼

CANCEL DONE

4.3 Login Page

The name of the sub menu is placed on the top of the screen. “Username” and “Password” entries can be given from proper fields. “Cancel” and “Done” buttons are located on the bottom of the page.



LOGIN

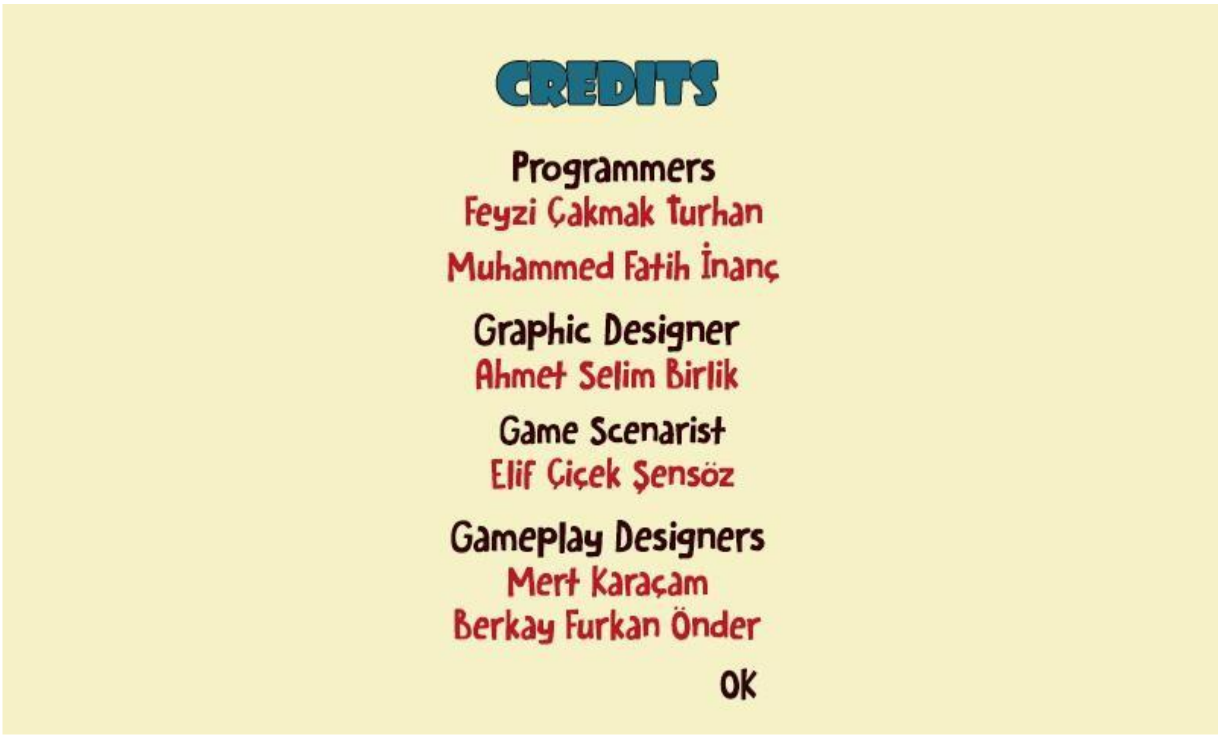
Username : geniu.s

Password : ••••••

CANCEL DONE

4.4 Credits

The name of the sub menu is placed on the top of the screen. Names of the contributors and their duties are written to credits page. “OK” button is located on the bottom of the page which is supposed to take the user back to the main menu. .



CREDITS

Programmers
Feyzi Çakmak Turhan
Muhammed Fatih İnanç

Graphic Designer
Ahmet Selim Birlik

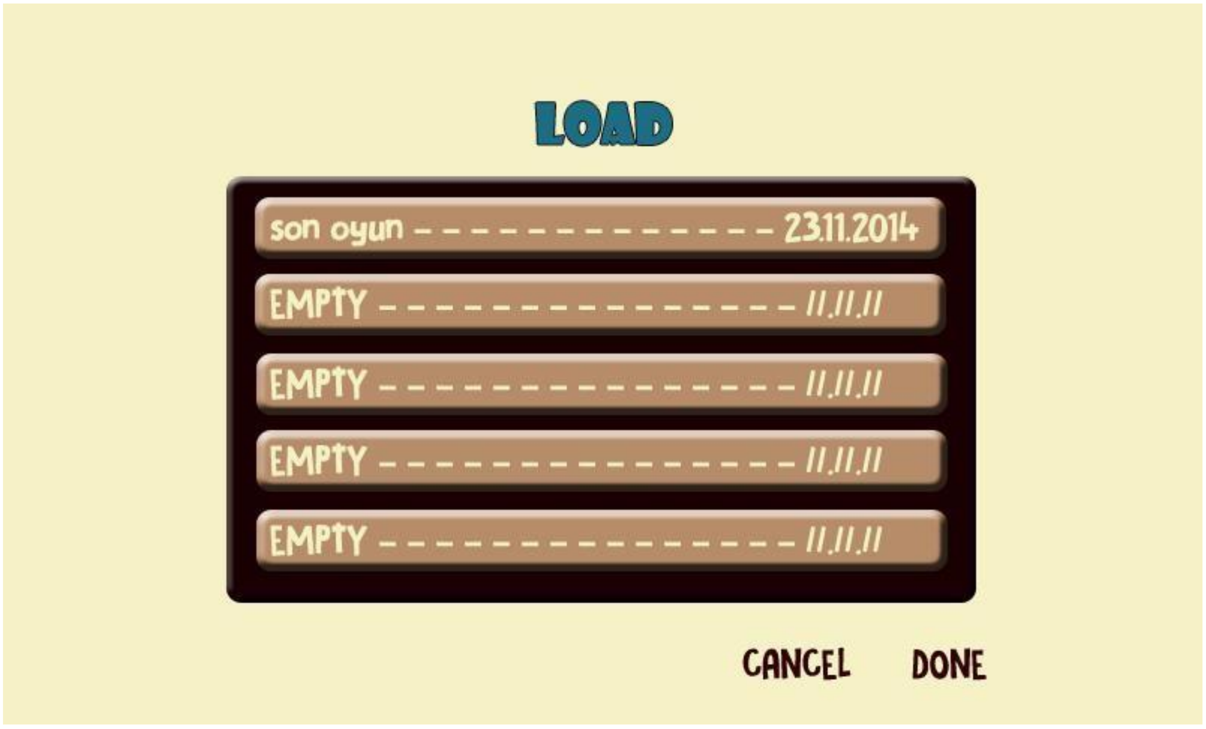
Game Scenarist
Elif Çiçek Şensöz

Gameplay Designers
Mert Karaçam
Berkay Furkan Önder

OK

4.5 Load

The name of the sub menu is placed on the top of the screen. This page include names of saved sessions and also their dates. “Cancel” and “Done” buttons are located on the bottom of the page.



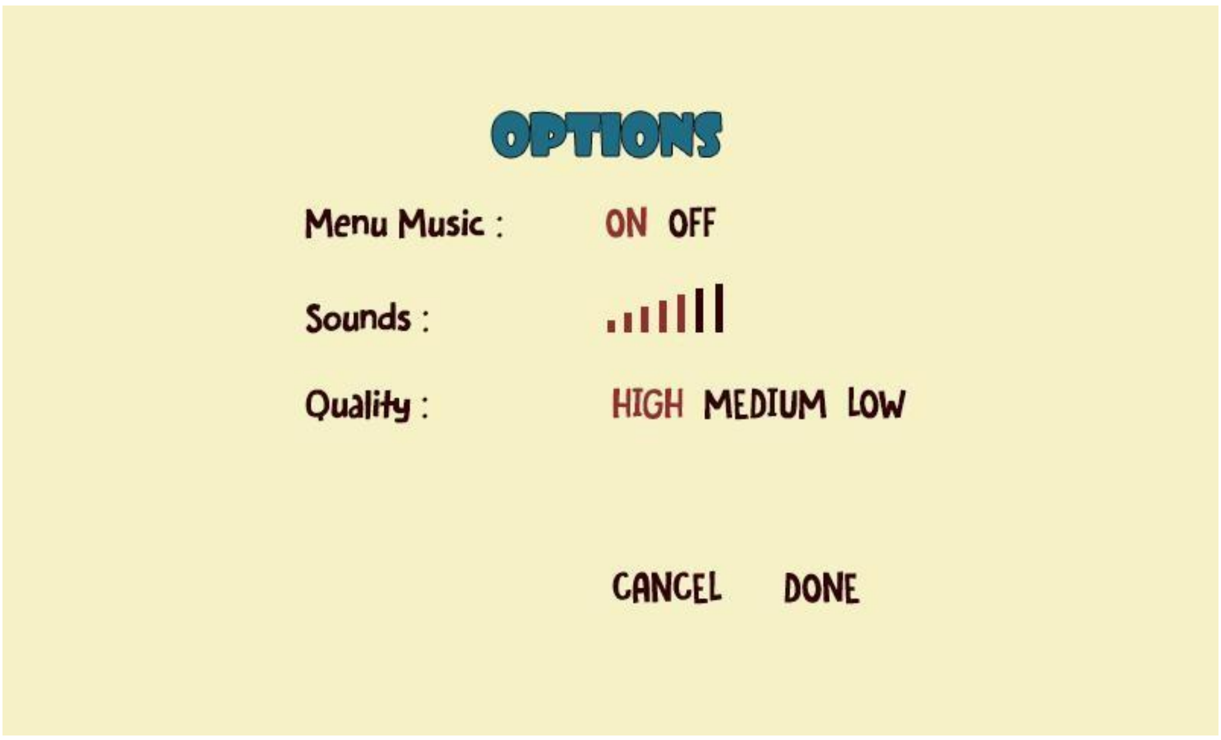
LOAD

son oyun – – – – – – – – – – – – – – 23.11.2014
EMPTY – – – – – – – – – – – – – – – – – //.//.//
EMPTY – – – – – – – – – – – – – – – – – //.//.//
EMPTY – – – – – – – – – – – – – – – – – //.//.//
EMPTY – – – – – – – – – – – – – – – – – //.//.//

CANCEL DONE

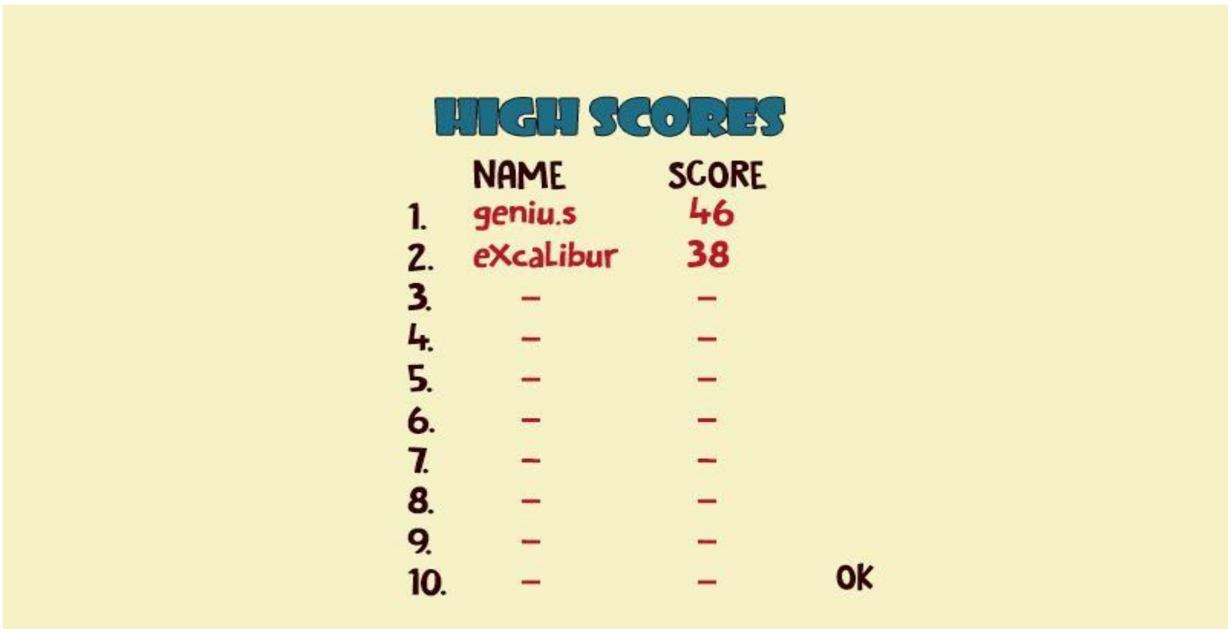
4.6 Options

The name of the sub menu is placed on the top of the screen. This page includes three basic game settings. The user can put the menu music on or off and can also adjust the sound level in this page. Furthermore user can change the screen resolution quality of the game with three choices “High”, “Medium” and “Low”. “Cancel” and “Done” buttons are located on the bottom of the page.



4.7 High Scores

The name of the sub menu is placed on the top of the screen. This page shows the top ten scores and their user names in decreasing order. “OK” button is located on the bottom of the page which is supposed to take the user back to the main menu.



	NAME	SCORE
1.	geniu.s	46
2.	eXcaLibur	38
3.	–	–
4.	–	–
5.	–	–
6.	–	–
7.	–	–
8.	–	–
9.	–	–
10.	–	–