MACERA TÜNELİ

by Gizem

Software Design Descriptions Report

CENG 491
Desing Project
2012 – 2013

Bahri TOKMAK
Gizem BAŞER
Hakan ORAL
Kadir Eray DOĞANLAR

CONTENTS

1. Introduction

Software Design Document (SDD) of Macera Tüneli provides necessary definitions to conceptualize and further formalize design of the software, whose requirements and functionalities were summarized in Software Requirements Specifications (SRS) Report. Aim is to provide guidance to a design which could be easily implemented by any designer reading this report. The approach used in this specification is adapted from IEEE Std 1016-1998[1].

1.1. Problem Definition

Macera Tüneli aims to support self-development of both preschool-age children and disabled children. On one hand, getting familiar with learning in a systematic fashion before school is so important for a child, which will prepare him to go further more easily in education life. On the other hand, disabled children should be encouraged to take part in real life more and to improve their abilities as much as they can.
Macare Tüneli will focus on these facts. It will invite children not only to play a game, but also to be a part of it. Thus, they will think more, act more, create more, learn more and feel more, which will lead them to discover their limits. Learning in each phase and discovering his own limits, his personality in time, one will turn into a healthier and more confident child.

1.2. Purpose

SDD is intended to provide a software system design which will satisfy functional and nonfunctional requirements. It is aimed to serve as a guideline throughout development phase of the project for developers. It also handles how the software requirements should be implemented.
Its audience consists of developers of Gizem Team, giving them a better understanding of the project and preparing them for development phase; project co-workers, Prof. Dr. Kürşat Çağıltay and his team, ensuring them that all the requirements are planned to be satisfied; and CENG491 instructors and teaching assistants, explaining them in details what is to be developed.

1.3. Scope

SDD of Macera Tüneli includes design patterns giving brief explanation about goal of the project, design constraints, assumptions, dependencies, detailed data description, system

architecture with its components, user interface, libraries and tools and time planning. Main objective is to provide detailed design descriptions so that following this document, developed the system will easily satisfy the requirements stated in the SRS.

1.4. Overview

This document has 9 sections in order to clarify the design of the project. In section 2, a brief system overview will be given. In section 3, design constraints, assumptions and dependencies will be handled. In section 4, detailed data descriptions will be provided. In section 5, system architecture with its components will be focused on. In section 6, interface design will be examined. Finally, in section 7, libraries and tools which will be used in the project will be explained.

1.5. Definitions, Acronyms and Abbreviations

SRS: Software Requirements Specifications
SDD: Software Design Descriptions
GUI: Graphical User Interface
IEEE: Institute of Electrical and Electronics Engineers
iOS: iPhone Operating System
OS: Operating System
KISS: Keep It Simple Stupid

1.6. References

[1] IEEE, IEEE Std 1016-1998 IEEE Recommended Practice for Software Design Descriptions. IEEE Computer Society
[2] http://www.utility.com/

2. System Overview

Intending to play Macera Tüneli, a child will be select a cartoon character, firstly. The cartoon character will accompany him throughout the tunnel. It is aimed to make the game more interesting, more fun and more optional with the characters. There will be five different alternatives:

- Ay Savaşçıcı (Sailor Moon)
- Şirinler (Smurfs)
- Heidi
- Mario
- Batman

Then, the child will continue to choose an environment in which he will be playing with a special concept. Any of these environments may be chosen with any characters. There are no restrictions in that case. Five different environment may be chosen:

- Amusement park
- House
- Market
- Garden
- School

After selecting a character and an environment, different tasks will be given to child to complete. All tasks are special or related to the environment. Now, let us examine procedure of the game dependent to the environment.

- Amusement Park:

Selecting "amusement park environment", a child has to pay to enter the park, firstly. Dragging the correct money figure on the screen, he will make the payment. Then, he gets a card to use in the park. However, he needs a photo of himself since this is special to a person. He takes a photo of himself and sees his own photo on the card later. Now, he will be given a certain amount of coins to spend on the toys. Selecting a toy to enjoy, he needs to pay the right coin, which will be appear in different colors, by dragging it on the screen again. Accomplishing this task, now he is on the toy enjoying with his character. Moreover, he is now familiar with colors.

- Home:

Selecting “home environment”, a child may do different activities at home. Thus, he will select an activity now.

- Cleaning:

The child enters a room which is decorated with a certain number of furniture. Then, in the next step, he finds himself in the same, but scattered room this time. He has to place furniture properly by dragging them and clean the room.

- Eating:

The child enters a kitchen where he will be asked to prepare a dinner table with certain kitchenware. He will do so by matching figures.

- Playing a game:

The child enters a living room where he may prefer to play alone or with his friends. If he prefers to play alone, he may draw something, do puzzles or record a video or sound. If he wants to play with his friends, he may tell them a story or sing a song by recording his voice or video. After completing this phase, the child will have been involved in the game actively and used his creation skills. Moreover, his tendency for playing and socialization may be observed.

- Market:

Selecting “market environment”, children, firstly, needs to learn needed three items at home from his mother. These items will be pre-determined and from different sections of the market. For example, one item will be a vegetable or a fruit, the other will be beverage and the last one will be a cleaning material such as a toilet paper. Now, he has to go and find the items on shelves and pay for them. Payment will be done again by dragging the correct money on the screen. At the end of this part, the child is expected to be more familiar with fruits, vegetables and main market products which are highly used in daily life.

- Garden:

Selecting “garden environment”, he goes to a garden. Firstly, he will be asked to choose a song to listen while spending time in the garden. In order to do this activity, he may record his sound singing a song or he may play his favorite from his computer. Then, he may do two different activities in the garden:

- *Animal feeding:*

He should select an animal to feed from certain types of animals by touching the screen. Meanwhile, he will be hearing specific sounds of animals. He may feed the animal by dragging food which will appear on the screen.

- Collecting garbage:
If he wants to collect garbage, he will be given a garbage bag. He will collect the garbage into this bag by dragging them. Then he will be asked to separate them according to certain types. After this part, he will be more familiar with animals. Moreover, he will gain a better comprehension of environmental consciousness.

- School:
Selecting "school environment", firstly, he will be expected to match figures related to certain class materials by dragging them on the screen. Then, he will select a course to take. There will be three options:

- Music: He may record a video or sound.

- Art: He may draw something or do puzzle.

- Mathematics: He may match certain mathematical shapes.

In order to accomplish this phase, the child has to take these three course in any order. After taking three of them and doing the required activities, he will be elected as class president and will be asked to make a speech about his feeling.
Completing this part, he will have used his creativity and made a significant effort. Moreover, his opinions about leadership, school, friendship, authority, etc. may be examined from his speech.

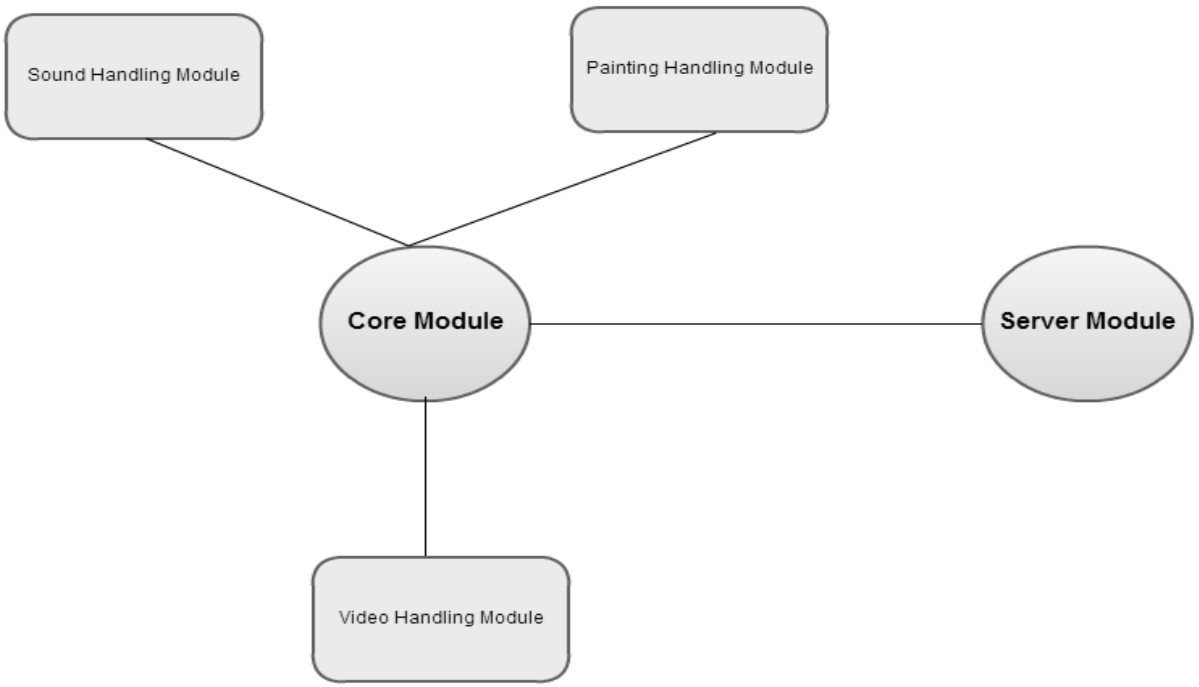


Figure 1 : System Overview Diagram

All these above functionalities of the game will be done by modules of our design. As illustrated in Figure 1, the system will have a core module as leading center of data flow; handler modules (sound handling module, painting handling module, video handling module) which will carry out drawing, taking pictures, recording sound, recording video, etc. operations and get needed data; a server module which will let us to record all activities on a web platform.

3. Design Considerations

3.1. Design Assumptions, Dependencies and Constraints

3.1.1. Software Constraints

Macera Tüneli will be developed by using Utility which will provide needed functionality and connection to the web platform.

3.1.2. Hardware Constraints

Macera Tüneli is aimed to operate on a mobile device having Android OS or iOS. All user interfaces, font and image sizes will be arranged according to mobile device screens. Moreover, depending on the web platform or memory capacity of the device, memory constraints may arise.

3.1.3. Performance Constraints

Depending on hardware quality and hardware performance of the device, performance problems may arise.

3.1.4. User Constraints

Since target users may does not have any education; evenmore, they may have some disabilities, there is always a risk that they may not be interested in Macera Tüneli or they may not finish it as expectedly.

3.2. Design Goals and Guidelines

3.2.1 Usability

Since target users are children, it is highly important that Macera Tüneli should be easy going. Interfaces should be as user-friendly as possible and all parts should be expressed clearly.

3.2.2. Maintainability

It allows a product to be maintained in order to correct malfunctions easily, provide new requirements, cope with a changed environment and make future maintenance easier.

3.2.3. Portability

Macera Tüneli is aimed to function platform independently. It will be played on iOS and Android devices.

3.2.4. KISS Principle

It will be a guideline to help us maintain the design as simple as possible during the development phase. Moreover, it avoids unnecessary complexity of the game.

4. Data Design

4.1. Data Description

Since our game will be played mainly between player and one of the characters of the game, the main data objects in the project are Player and Character objects. All other objects which is mentioned below in second part and including these two objects will be associated with Game objects. The story will be shaped according to the actions of the player and character on items in specific tasks in specific environment. Following parts will introduce the data objects and their attributes in detail.

4.1.1. Data Objects

Player : This data object represents the player.

Figure 2 : Player Object

playerId: holds the player information.

playerName: holds the name that is entered by the player at the beginning of the game if it is not entered a default name will be used during the game.

playerImage: holds the player's picture which is took by the camera.

Character : This data object is representing the character choice of the leading character of the game that is the center element.

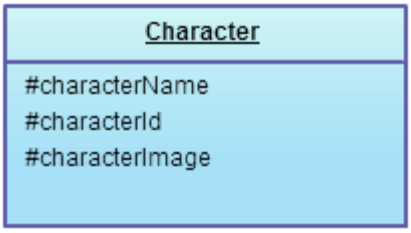


Figure 3 : Character Object

characterId: holds the character information of the leading character that is selected by the player at the beginning of the game.

characterName: holds the name of the leading character.

characterImage: holds the character's graphics designed by the graphics designer which represents
the image of the character.

Environment : This data object is representing the environment choice of the game.

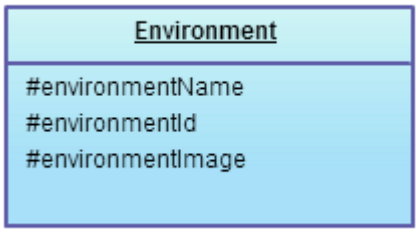


Figure 4 : Environment Object

environmentId: holds the environment information that is selected by the player at the beginning of the game.

environmentName: holds the name of the environment.

environmentImage: holds the initial environment graphics designed by the graphics designer which represents the picture of the environment.

Scene : This data object is used for graphics of the game. They will be loaded
according to their ids as the game processed.

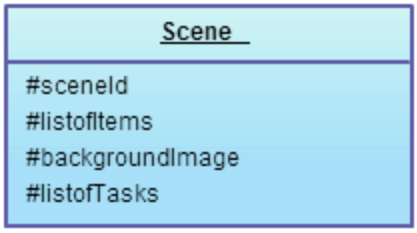


Figure 5 : Scene Object

sceneId: holds the id of the graphics to be loaded when necessary in order to compose the visuals of the game.

listofItems: this data object is representing all the items that can be seen in the game which are going to be helpful for user to solve very simple problems and also the items which are going to be
displayed in the current scene of the game.

backgroundImage: holds the graphics designed by the graphics designer for the non-altering background of the game. They are going to reflect the characteristical view of the current time which the character traveled through.

listofTasks: holds the task items that the player picked until he/she reaches the task. These items will be displayed on the screen and said by the character in high volume and the player will be able to read or listen these tasks whenever he/she wants.

Item: This data object is holding information of each item which is in the listofItems of scene object.

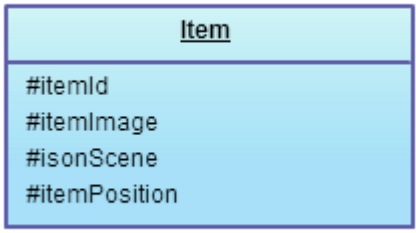


Figure 6 : Item Object

isonScene: represents whether an item is on the scene or not.

itemPosition: holds the position information of the item in the scene.

Game: This data objeitemImage: holds the graphics of the items in the current scene designed by graphics
designer.

itemId: holds the id of the graphics of the item to load when necessary.

ct is representing the whole game and includes all the data
objects stated in this section.

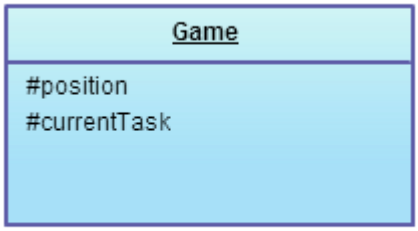


Figure 7 : Game Object

position: holds the position information of the player in the game.

currentTask: holds the task information which the player is at in the game currently

Task: This data object is representing each quest in the game.

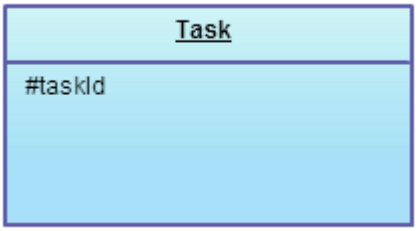


Figure 8 : Task Object

taskId: holds the id information of the task to load proper quest.

When new tasks are added to the project, it will be easy to adopt them to the game simply

by creating new task object with new task id.

Sound: This data object is representing the sounds that will be used to inform the player about the situation. Also this object will hold the general music which will be played during the game.

Figure 9 : Sound Object

musicId: holds the id information of the music which will be played when necessary in the scene.

musicFile: holds the music to play according to the id when necessary.

Transition: This data object is representing the transition between tasks in order to proceed in the game.

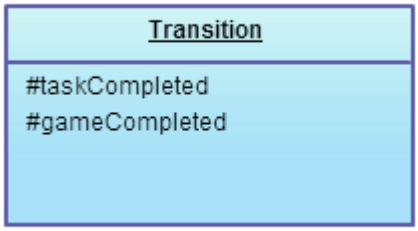


Figure 10 : Transition Object

4.1.2. Relationships

Character-Game: Game object has only one Player object in the game.

Item-Game: Game object may have one or more Item objects representing that there can be one or more items available in the game environment.

Task-Game: Game object may have one or more Task

Player-Task: Player object In this section we will introduce the relationships between all the data objects described above.

Player-Game: Game object has only one Player object in the game.will be interactive with a Task objects to pass in order to proceed.

Player-Item: Player object may have one or more Item objects to use in order to solve problems.

Scene-Game: Game object may have one or more Scene objects to use in order to preceed the story of the game.

Environment- Game: Game object may have one Environment object.

Sound-Game: Game object has one or more Sound object that is dynamically changed.

Transition-Game: Game object has one or more Transition object
that is dynamically changed.

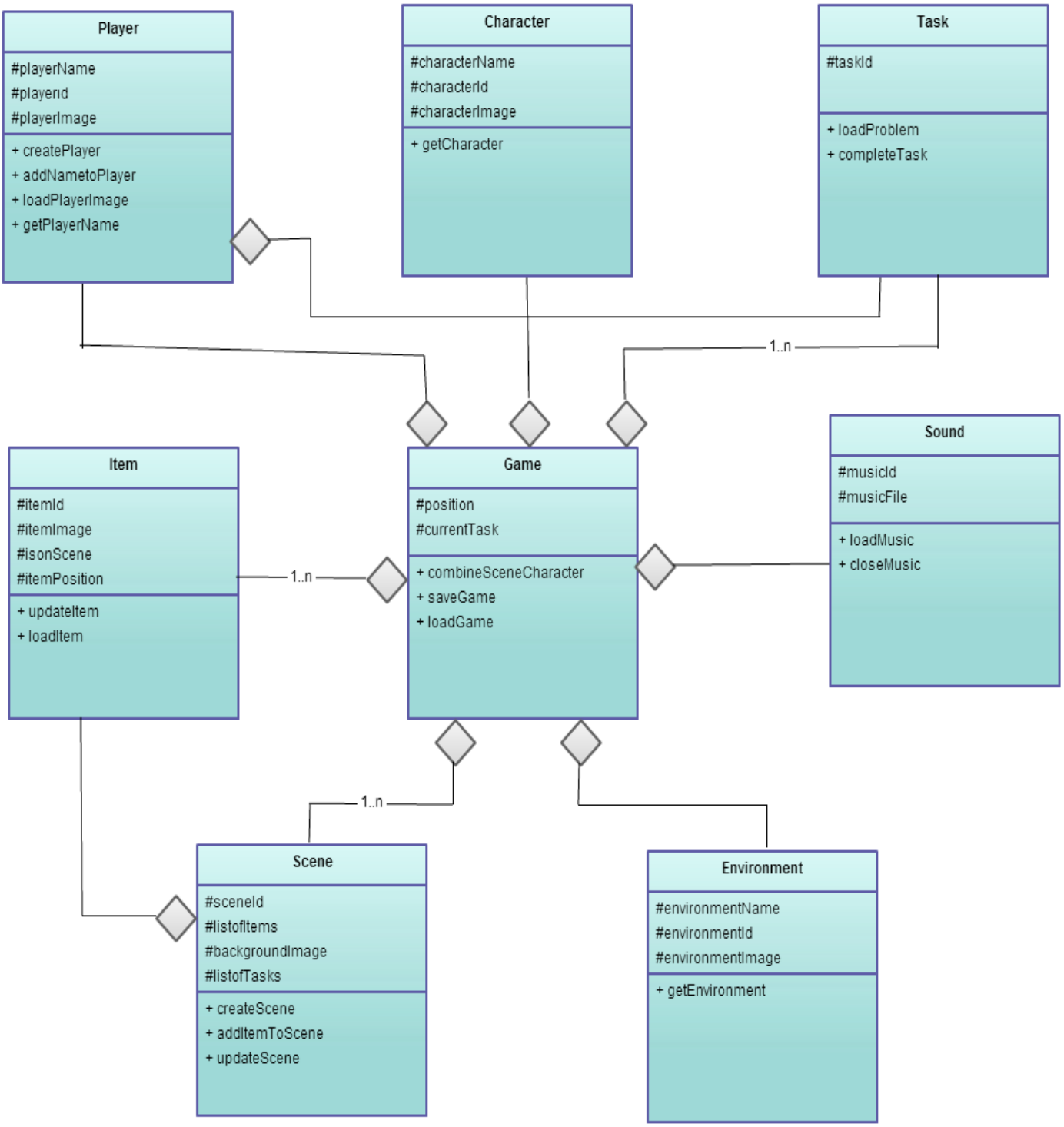


Figure 11 : Class Diagram

4.2. Data Dictionary

player: is the user in the game that communicates with the character and tries to complete the tasks.

character: means the player in the game environment, the hero/heroin of the game.

task or quest: is the each problem need to be solved by the user in game.

Item: means all the objects in the game environment available for player to use in solving the problems, for example; fruits,animals,garbages etc.

background: can be thought as contiguous pictures to compose the story of the game as the character walks.

5. System Architecture

The system has 1 main module and 4 sub modules. The definitions and functionalities of the modules are explained in the following sub sections of the document.

5.1. Architectural Design

Design of the whole system is done by flow based programming concept. Operating aim of the components of the system is sequentially which provides to handle by related components as well. Macera Tüneli consists of 1 main and 4 sub modules.

The Core Module is the first module of the system.The Core Module has central importance for the whole system. It interacts with the all sub modules which are Video Handling Module, Image Handling Module, Painting Handling Module and Server Module. The Core Module behaves like the main function of a program. It receives video data from the Video Handling Module , image data from Image Handling Module , painting data from Painting Handling Module which are submitted by the user then, sends query to the Server Module to save or to use the saved data.

The second module is Video Handling Module,which is used by the user to submit video file. And the module delivers data to the Core Module.

Another part of the system is Image Handling Module which has a user-friendly GUI to provide user to take photos. This photos are used in the following part of the game and one of them is used in the ID card of the user. Pre-saved photos also can be selected by the user. This module transmits the image to the Core Module and only interacts with the Core Module.

Third part of the system is Painting Handling Module.It gets the data which are recognized by the device and transmits them to the Core Module. With this feature user can use his/her imagination. User draws pictures by using his hand or a kit. It depends on the device whether capacitive or not.

The last and very important module is the Server Module which provides to store the videos, photos and the data of the game on a common server after re-evaluation of disk space constraint of the system.Server gets some requests and reply them .

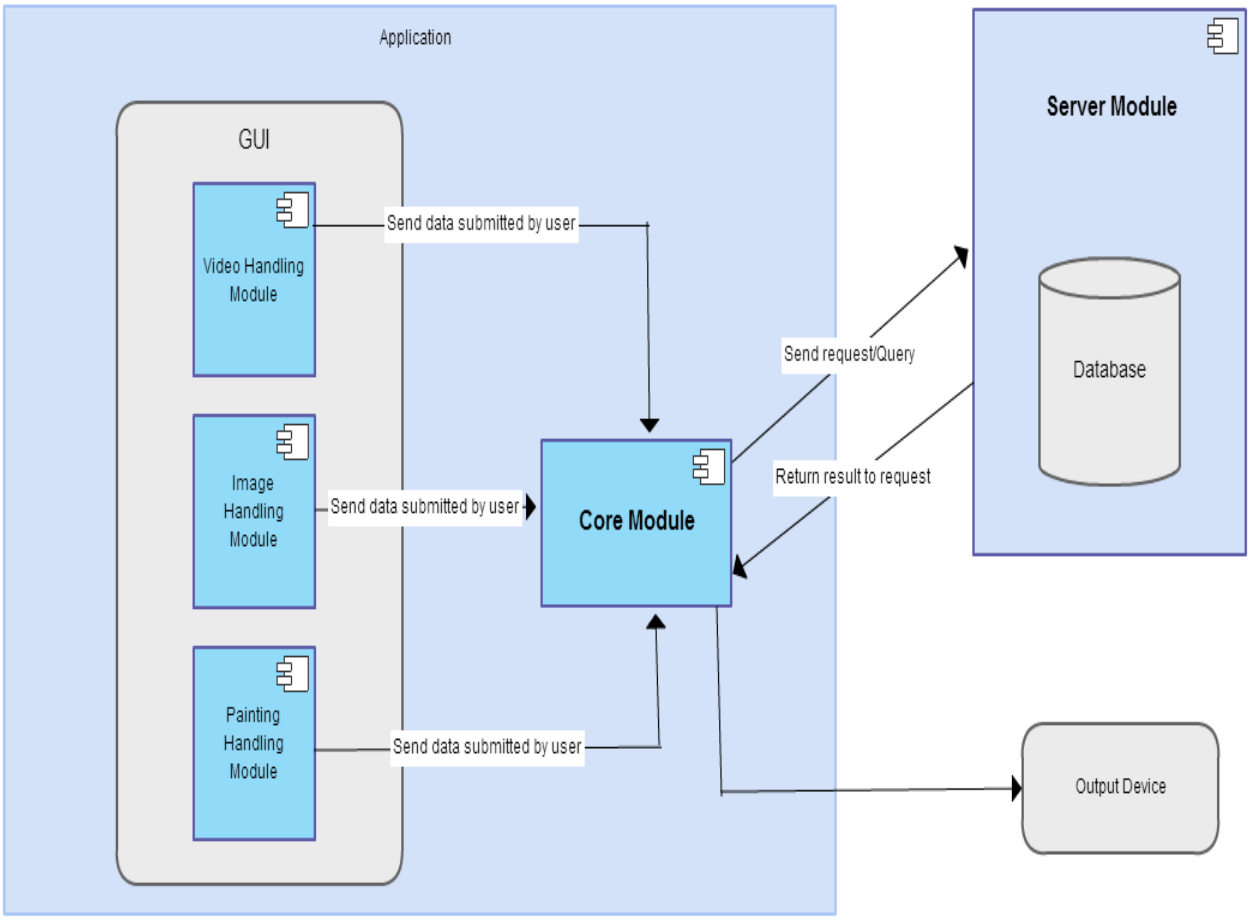


Figure 12 : System Architecture

5.2. Description of Composition

5.2.1.PreGame Component

5.2.1.1.Processing narrative for PreGame component

This is the sub system which is responsible from taking all the configurations that the player will give to the system. After passing this system, player will never be able to change any configuration regarding the character. To be able to make such changes, he/she will have to come back to that stage, make the desired changes and start the game all over again. This component will also be responsible from some user actions, such as load Game, start New Game, select Character. Besides those player-oriented actions, some systems actions will take place,such as start Game, draw Scene.

5.2.1.2.PreGame Component Interface Description

For this subsystem, user will be supplied a menu which he/she can configure some settings and take some actions mentioned in the previous sections. The components' input interfaces are the events that player triggers. Such events are mouse clicks, keyboard entries etc.

In this stage, user will also be able to take a photo that will be used through out the game. For this purpose output of such action will be the photo choices that we are going to offer to the players.

5.2.1.3. PreGame Component Processing Detail

This component itself does not have an important algorithmic implementation. The important points of the implementation are handling the player choices and make the necessary adjustments accordingly. Also, if a previously saved game is going to be loaded, the correct flow of the game is crucial. Correct parsing of the saved game file is important.

5.2.1.4. Dynamic Behavior of PreGame Component

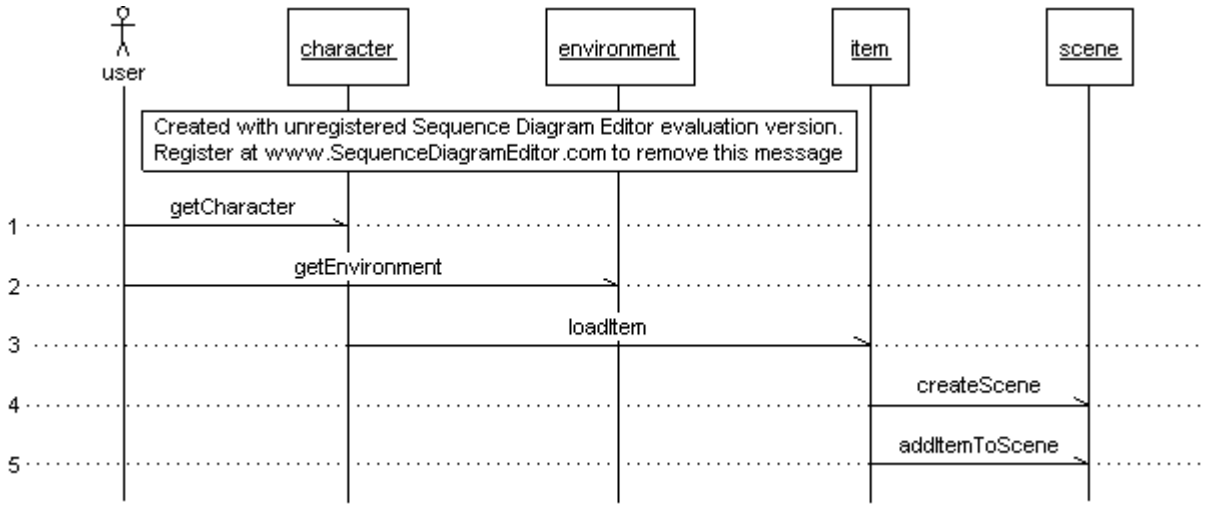


Figure-13 Sequence diagram of "Create a New Game"

5.2.2.Game Component

This is the top most layer of the system. All components will be combined into this component to make users play the game we created.

5.2.2.1.Processing narrative for Game component

This subsystem is where the actual game playing takes place; however we find it useful to divide this subsystem into two subcategories, which are namely level Transition and game Playing. In the level Transition, transition between parts within thegame will be done smoothly, related actions will be taken, and system will be prepared from the new part. On the other hand, in the game Playing stage, the player will be more effective on deciding which actions are going to be applied as he/she decides what to do while playing game.

5.2.2.2.Game Component Interface Description

When the player completes a part, a video will be shown to him/her. That way, the transition between the parts will be applied and the system will be able to be prepared for the next one. Also, by this video shown, player will be able to follow the story be in the game.

In the game Playing stage, user will have controller buttons,, and user will be able to complete the game.

5.2.2.4. Dynamic Behavior of DuringGame Component

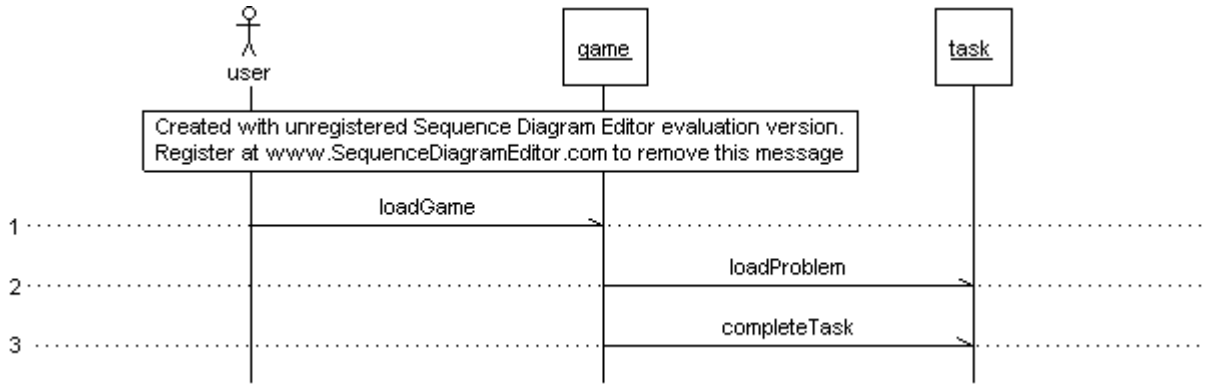


Figure-14 Sequence diagram of "Play Game"

5.2.3.Post Game Component

5.2.3.1.Processing narrative for Post Game component

This is the subsystem which is responsible from the actions that player wants to take after finishing the whole game or save the game for a future time. This will be enabled by some methods, such save Game, finish Game, show Results etc.

5.2.3.2. Post Game Component Interface Description

If the player interrupts the game and wants to save the game for a future use, he/she will have to click a button which is placed on the main game screen. If he/she does so, he/she will be redirected to a menu similar to the one in the beginning. By the touch on the screen, game will be saved. If he/she want store turn the game and continue, this will be possible as well If the player completes the whole game, after watching the ending video, he/she will be supplied an interface which he/she can see his/her point and how well he/she did during the game. After that, whether he/she wants to start a new game will be asked on the screen. Later on, necessary actions will be taken accordingly.

5.2.1.3. Post Game Component Processing Detail

All the performance of the player will be kept in some format throughout the game, so when he/she finishes the game, his/her point will be calculated based on those information.

5.2.1.4. Dynamic Behavior of PostGame Component

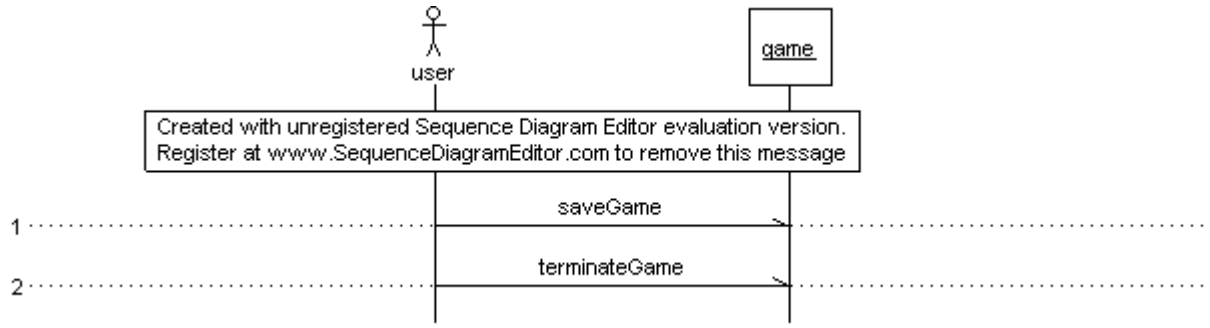


Figure-15 Sequence diagram of “Exit Game”

5.3. Design Rationale

We chose to divide our system into those subsystems, because we wanted separate the actions within and out of the games. That way, the implementation of the whole system will work in a more organized way since different group members can deal with different stages without depending one another, since the actions in the game will have no effect the ones which are not. Although the actions which are taken before the game effects how the game and the character are created, after completing that task, there is not a dependency or relation between those actions and stages and vice versa.

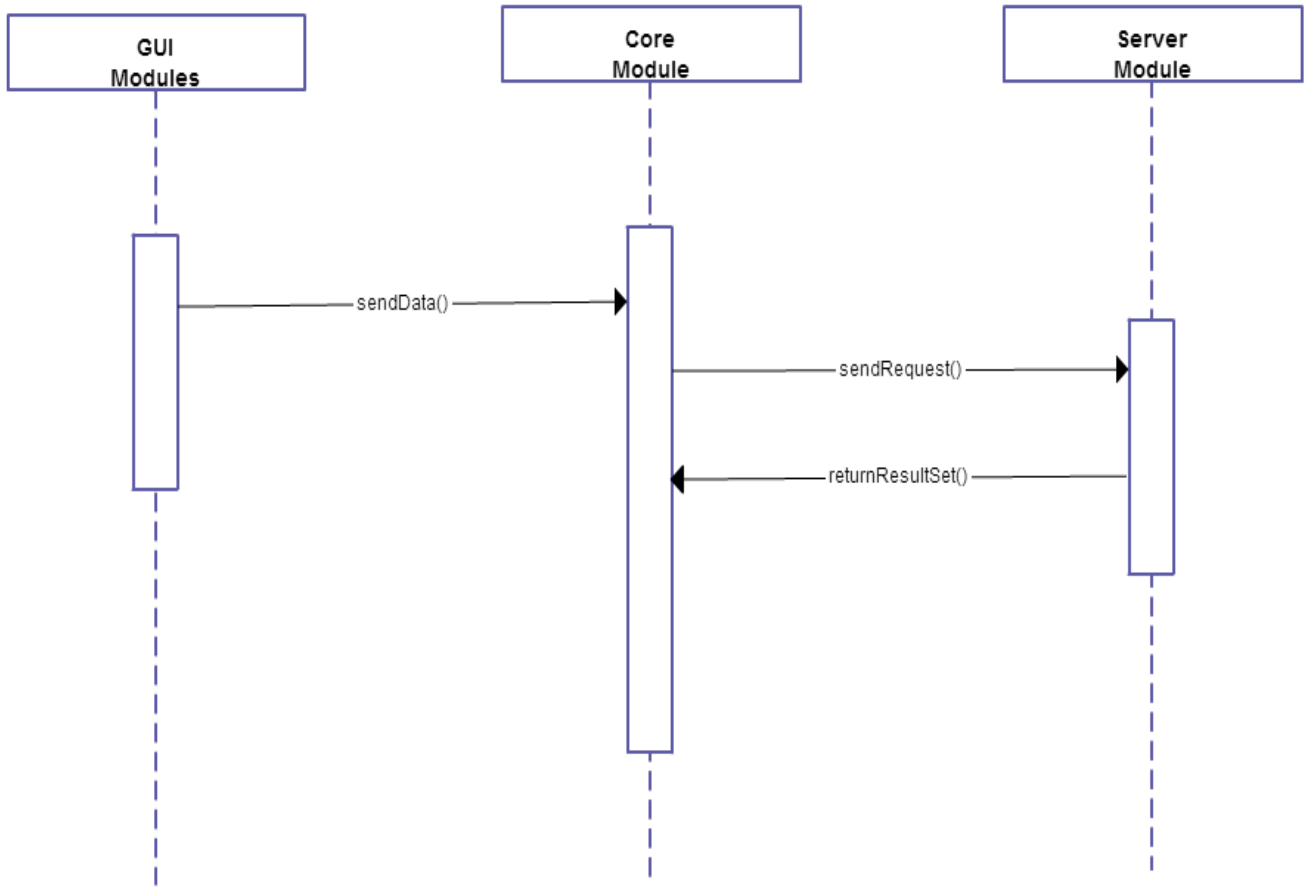


Figure 16 : Sequence Diagram of Core Module

5.3.1. Video Handling Module

5.3.3.1. Dynamic Behaviour

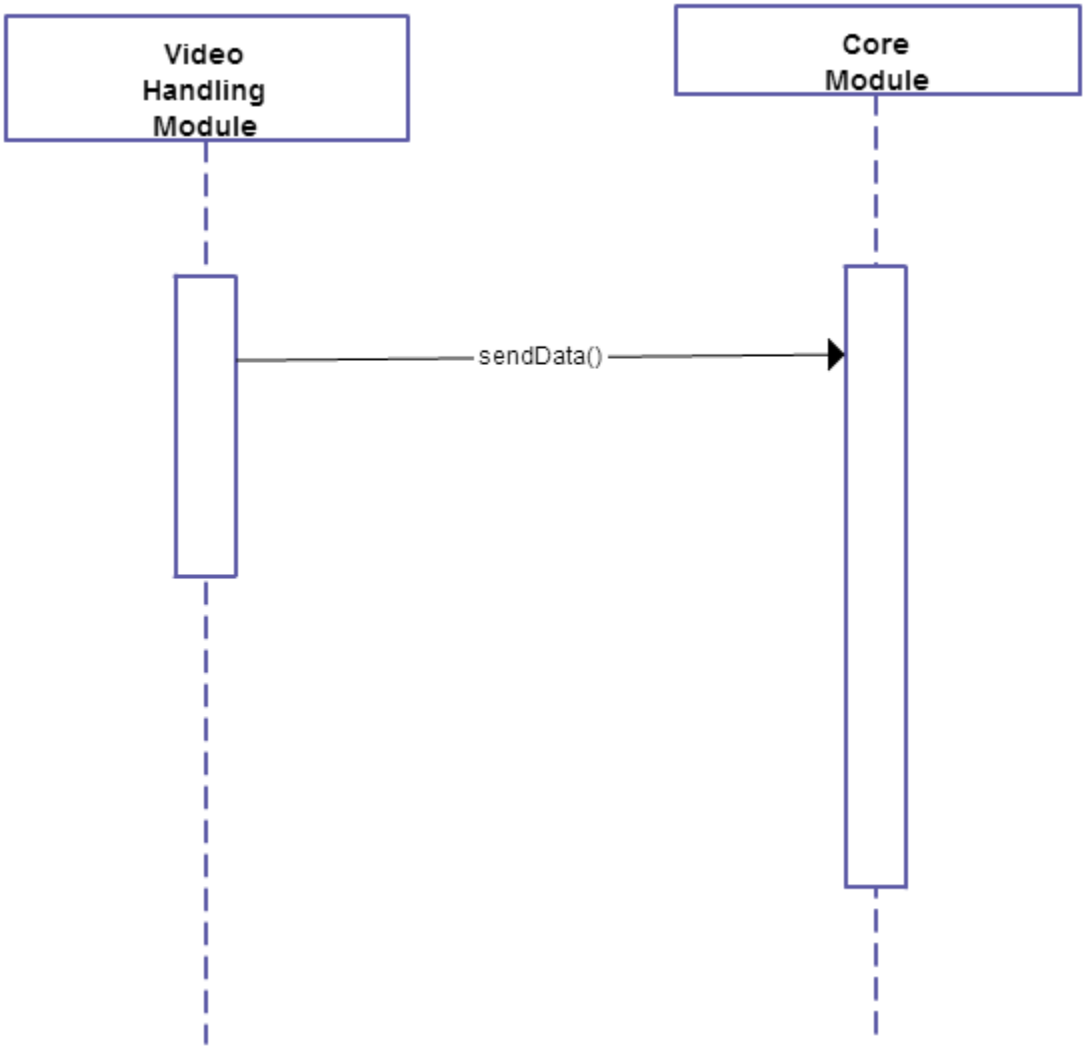


Figure 17 : Sequence Diagrams of Video Handler Module

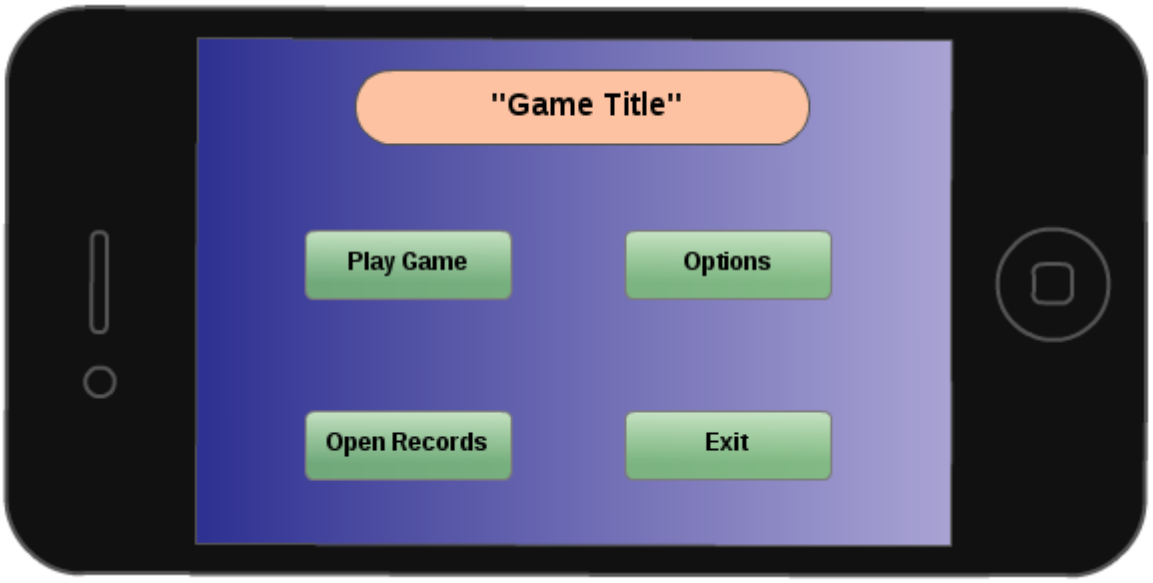
6. User Interface Design

6.1. Overview of User Interface

The user starts to use the game by first clicking to the game icon. After that a main menu appears on the screen that user has four choices whether playing a new game or changing the options or checking past record or exiting from the game. If user selects play game button he/she will be directed to character selection screen and He/she will have five choices. After choosing one of the character,user will pass to another page which calls environment selection screen. In this page user again has five choices and He/she will be waited to choose one of those. With the decision of environment,user can start to play the game. User can also click the option button in the main menu to be able to change the sound (on-off) or the language (english-turkish) of the game.

6.2. Screen Images

6.2.1 Title Screen



Screen 1: Game Title Screen

The Game Title Screen will be the first screen introduced to the player after the application has been loaded.

It will display the game title, as well as three buttons with which the player can navigate to multiple screens.

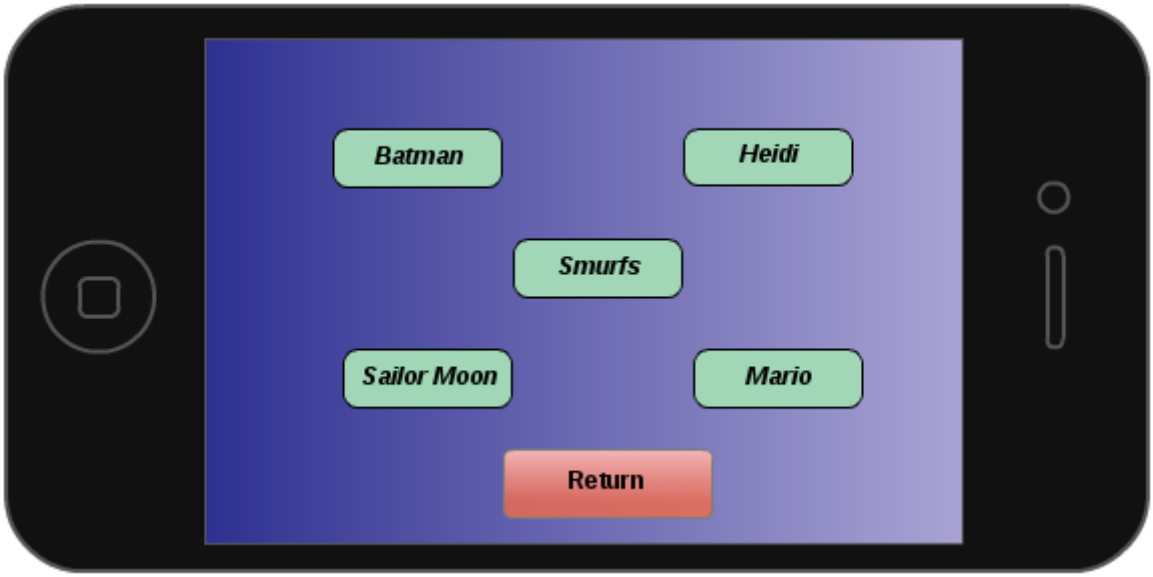
o Play Game Button: When selected, this button will open a new screen containing a graphical list of famous characters.

o Options Button: When selected, this button will open a new screen containing options that are user-modifiable enabling the player to personalize the game.

o Open Records Button: When selected, this button will open a new screen containing last 10 performances of the player.

o Exit Button: When selected, this button will activate a verification window, insuring the player intended to exit the game, if the player selects “Yes” the game will exit, if the player selects “No” the player will be returned to the Title Screen.

6.2.2 Character Selection Screen



Screen 2: Character Selection Screen

The Character Selection Screen will enable the player to select the desired character of the play.

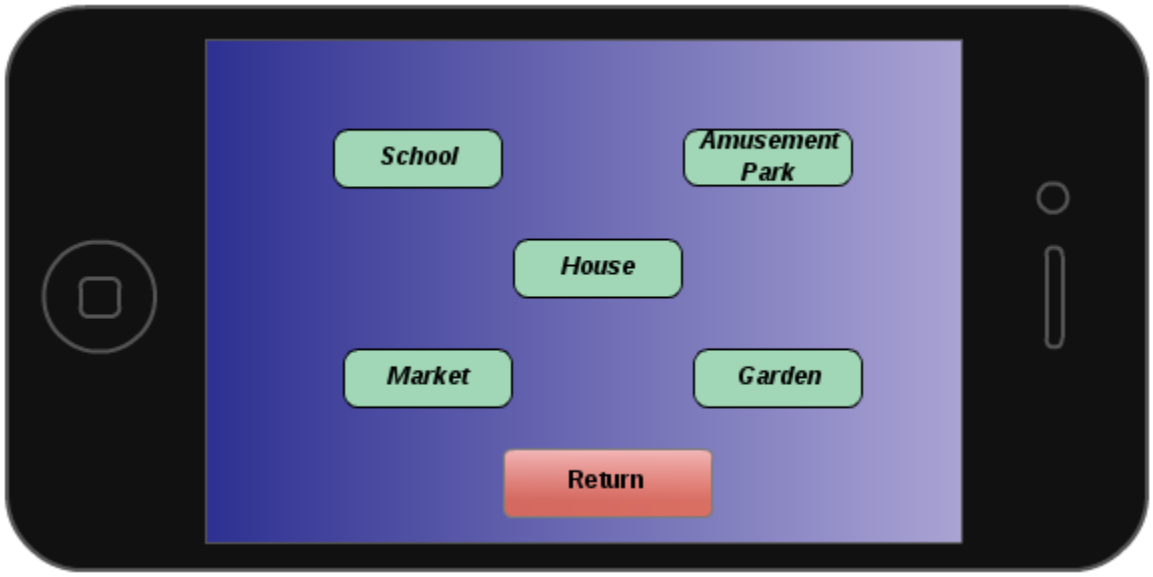
Icon selection:

o Character Icon: When selected, the player will immediately be transferred to the Environment Selection Screen to choose the environment in which he want to play the game.

o Return Button: When selected, the player will be transferred to the Title Screen.

- Character Selection Screen

6.2.3 Environment Selection Screen



Screen3: Environment Selection Screen

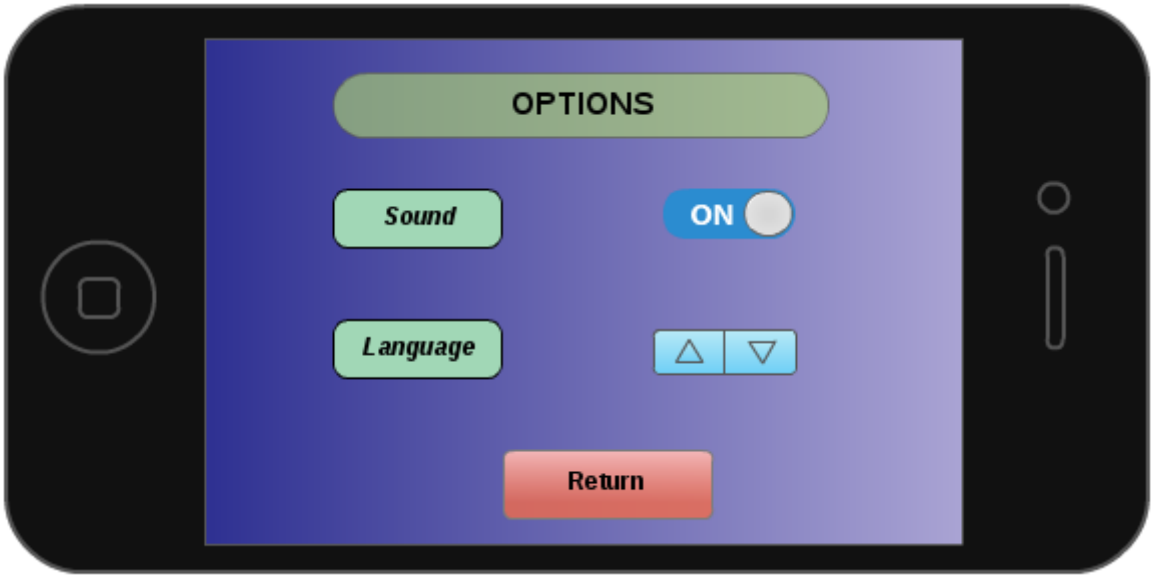
The Environment Selection Screen will enable the player to select the desired environment of the play.

Icon selection:

o Environment Icon: When selected, the player will immediately be transferred to the environment specified by the icon to begin game play.

o Return Button: When selected, the player will be transferred to the Character Selection Screen.

6.2.4 Game Options Screen



Screen 4: Game Options Screen

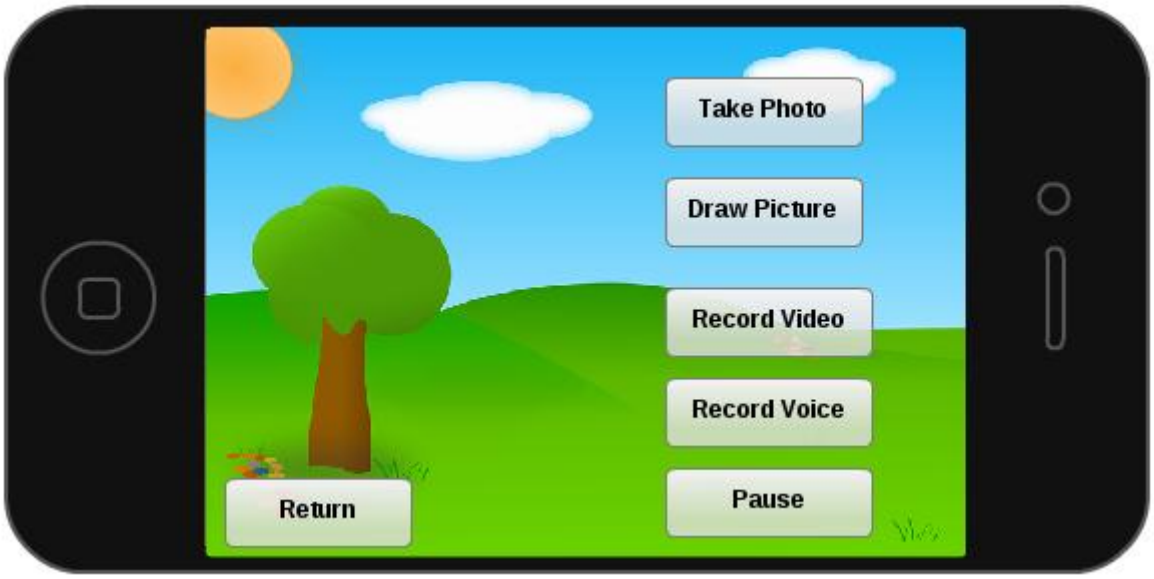
The Game Options Screen allows the player to adjust settings to their specifications.

o Sound: Selecting “On” will enable sound, selecting “Off” will disable sound.

o Language : There will be 2 language choice button for the player based on Turkish or English

o Return Button: When selected, the player will be directed back to the Title Screen.

6.2.5 In-Game Screen



Screen 5: In-Game Screen

The In-Game Screen contains in-game user interface regarding player control and actual game play.

There will be a lot of different game scenarios depending on the environment choice and the process of the player, but upper photo shows most important basic icons which the player can choose.

6.3. Screen Objects and Actions

In first screen there is going to be :

Play Game
Options
Open Record
Exit

In second screen there is going to be :

- Batman
- Heidi
- Smurfs
- Sailor Moon
- Mario

In the third screen there is going to be :

- School
- Amusement Park
- House
- Market
- Garden

In the game screen there is going to be :

- Take Photo
- Draw Picture
- Record Video
- Record Voice
- Pause
- Return

In the option screen there is going to be:

- Sound - ON/OFF
- Language - ENGLISH/TURKISH
- Return

7. DETAILED DESIGN

7.1. PreGame Component Design

Classification:
This component can be classified as a module.

Definition:
This component is the module where users will be first in interaction before starting to play the game which help users to take some actions about the game.

Responsibilities:
This is the module which is responsible from taking all the configurations that the player will give to the system before starting to play. It is responsible of loading game, starting new game, creating character, loading sound and items, and also drawing scene.

Constraints:
The important thing about this module is that, player will not be able to change any configuration such as selected character, loaded game or settings during the game. If player wants to change some configurations, the game should be started all over again.

Composition:
The PreGame module has five subcomponents which are Player, Character, Environment and Scene.
The Charcter subcomponent provides user five characters that he/she can select before starting the game. User can select one of them before starting to play the game.
When player chooses a character, an charcter name will be written by the user, an id and image id belonging to the selected character will be provided to the PreGame component. This subcomponent will be loaded by the PreGame module at the beginning of the game.
The Environment subcomponent contains five environments that he/she can select before starting the game.
The Scene component includes the images of the background of the game. This contains different images in Photoshop file format which are going to be loaded while the player is advancing in the game. Each scene in the game will be differentiated from each other from a scene id and also contains items list and that the scene will use in the game.

Uses/Interactions:
The PreGame component is in interaction with the DuringGame component. When player chooses an character and and environemnt or wants to continue to play the previously saved game, these choices will be sent to the DuringGame component and DuringGame component will use these information.

Processing:
This component does not have an important algorithmic implementation. The process includes handling the player choices and makes the necessary adjustments accordingly. If player wants to choose a character, five different characters will be provided to the user. When user chooses one of them, the id of the character will be saved in the program to be used in the PreGame component and also the name that is written by the user will be saved. Also, if a previously saved game is going to be loaded, the correct flow of the game is crucial. Correct parsing of the saved game file is important.

Interface/Exports:
This module mainly has one interface to interact with the user, which is namely the menu from which the player will be able to configure game settings and choose an character

7.2. DuringGame Component Design
Classification:
This component can be classified as a module.
Definition:
This component is the module where the actual game playing will take place. This component will provide all the information which will be used during the game.

Responsibilities:
This is the module which is responsible from updating the game, loading the game and tasks.

Constraints:
Player will not be able to change any configuration such as selected character, loaded game or settings during the PreGame module. If player wants to change some configurations, the game should be started all over again.

Composition:
The DuringGame module has four subcomponents which are Game and Task.
The Game subcomponent contains all the relevant information about the game such as the movements and position of the character. This will take information about the actions taken by the user in the tablet. This subcomponent also contains information about the current level and current task.
The Task subcomponent contains four different quests that are used during the game. These quests will be solved by the player in the game in order to advance in the game. For each game, there will be different tasks. Each task has a unique id and has same level id if they belongs to the same game. DuringGame component will use these ids to load the related task in the game.

Uses/Interactions:
The DuringGame component is in interaction with the PreGame component. When player

chooses an character, the name, id and the image belonging to the character will be sent to the PreGame module and will be used during the game. If player wants to continue to play the previously saved game, these choices will be sent again to the DuringGame component and DuringGame component will load the corresponding game and items to the scene and display the relevant videos according to these information.
The DuringGame module is also in interaction with the PostGame component. If during the game the player wants to quit the game or the game is completed, the DuringGame module will prompt PostGame module that the player wants to terminate the game.

Processing:

The algorithm of the DuringGame component uses a loop which starts by updating the game, then loading the game and tasks and displaying the videos in each completion of the level. This component uses the information taken from the PreGame module. PreGame module sends the id, name and image id of the character to the PreGame module and the selected character is loaded in each scene during the game. If player starts to play a new game, game and tasks will be loaded starting from the beginning of the game. If the player wants to continue a previously saved game, the information of the related level will be loaded to the game.

Interface/Exports:

There are mainly 2 kinds of interface, which the player will see during this module: task scene-where the player will actually play the game, road between tasks-where the player will have to pass through in order to play the next task.

7.3. PostGame Component Design

Classification:

This component can be classified as a module.

Definition:

This component is the module where all the post game actions will take place. This component is also responsible from informing the player about the results.

Responsibilities:

This component is responsible from calculating the score, showing the final video, and saving the game if it is asked for.

Constraints:

There exist a few constraints about this module. One of them is if user wants to exit the game right away after it has just started the calculation of the score will not occur. Another one is if there exists a previously-saved game in the system, this module will have to delete that game after asking the player if it is desired so.

Composition:

The PostGame component has only one subcomponent: Game.
Game component will calculate the score in the end, to show the player the performance of himself/herself throughout the game. This will be based on the time
In order to do that, DuringGame component should provide some information about the time consumed.
Also this subcomponent, if the game is not finished with completing all of the tasks, will ask the user if he/she wants to save the game for a future play. If it is desired so, it will check the system if there is a previously-saved game. In the case that there is one, a second question will be asked to the player to learn if he/she wants to overwrite this game. If the player wants to overwrite, actions will be taken accordingly.
To save the game, some crucial information about the state of the current game will be taken and written into the file that will be kept in the system. That information will include things such as character id and name, current task and level, and the performance information about the previous level and tasks in order to calculate the score in the end correctly.
There will be an ending animation to show in the end, to give the player a sense of glory.

Uses/Interactions:

This component will be in interaction with both of other two components mentioned earlier. From the duringGame component, information about the performance of the player will be taken in order to calculate the final score. And also, the information of whether, all of the tasks had been finished or not, will be taken. If they are not finished yet, actions to save the game will be taken accordingly. If they are all completed, the ending animation will be shown to the player.
Also the text file which was written during the save of a game will be used by the PreGame component in order to create the scene, character and other necessary items in order to make player resume that game if he/she wants to.

Resources:

Since this component includes only things like calculation of the score, or the save of the game, only resources that we are going to use in this component is the text file that we are going to use to keep the necessary information to recreate the game all over again to resume, and the animation that we are going to show in the end of the game.

Processing:

There is no algorithm in relation with the actaul game implemented here, only during the calculation of the final score; the weights of the tasks will be different from each others. Those weights will be determined according to how easy or difficult that particular task and level is.

Interface/Exports:

There will be either one of the two interfaces that will be shown in this component.
It will either be the ending video and the score or the page that will ask the player if he/she

wants to save the game for a future play.

8. Libraries, Tools

8.1. Libraries

Since we will use Unity tool in the development phase, we will not need any external libraries. However, we think of using a few libraries of Unity which are written below:

- Character Controller: It includes generic controller scripts which may be used for third-person or first-person games.
- Light Flares: It includes necessary scripts to create light flares.
- Particles: It includes necessary scripts to create particle graphics.
- Physics Materials: It includes scripts to adjust friction and bouncing effects.
- Projectors: It is used to project materials onto objects.
- Scripts: It contains basic scripts such as camera scripts, general scripts and utility scripts.
- Standard Assets: It includes components specific to mobile environments.
- Toon Shading: It includes shader scripts to create realistic shading effects.

8.2. Tools

Unity will be used as development tool, which is a multipurpose tool for video game development, architectural visualizations, and interactive media installations. It is compatible Windows, iOS, Wii, Android etc. It also allows web operations which is highly important to record operations.

9. Conclusion

Software Design Descriptions (SDD) Report is written to provide brief information for all design patterns of Macera Tüneli.

In SDD, firstly, design constraints, assumptions and dependencies are introduced. Secondly, data models are examined with their descriptions. Thirdly, system architecture is focused on with its
components. Finally, user interfaces are examined and the libraries and the tools which will be used in the development phase are introduced.