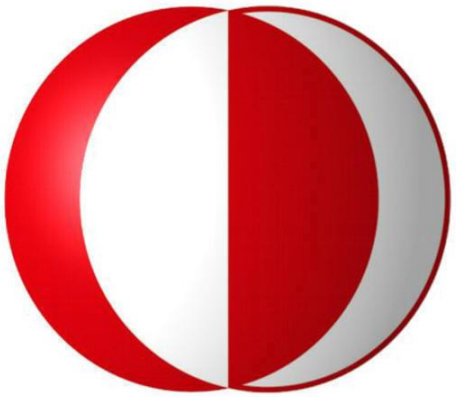
BIG CENG THEORY

Detailed Design Report

Middle East Technical University



09.01.2012

INDEX

1. INTRODUCTION

This detailed design document is prepared to give detailed design descriptions for single-player educational physics game project Fizbot which is sponsored by Words To Inspire for Fatih project [1] and carried out by Big Ceng Theory project group.

1.1. Problem Definition

Most of the high school students have difficulties in learning physics. The major reason is that the physics in high schools is taught as it is only a course subject which they have to pass and not related with the real world. Therefore, students cannot associate the knowledge they learned in class with real life and the knowledge cannot go further than a memorization. However, we know that the better way to learn physics is to visualize the concepts and relate them to daily life. Considering that students at that age are more proned to play games than studying physics which is also a severe problem, we can propose a solution that combines physics with computer games. From childhood to adulthood, games have an important role in learning, since it helps increasing imagination, creative thinking, curiosity and exploring the environment with fun. Therefore, a game that uses ninth grade physics as rules may have a great contribution for the solution of this problem. The reason why we chose ninth grade is it covers the very basic and significant parts of the physics and has too many examples in real world. Despite there are many educational physics game in the market, the need for such game still exists. This is because the existing games are either question answer type which most students find boring or very funny but not sufficient for the level of high school students.

1.2. Purpose

The main purpose of this document is to explain all the design descriptions of Fizbot. The explanations are primarily targeted for programmers who will develop this project to satisfy all the requirements.

1.3. Scope

This software design document includes design patterns giving brief explanation about the goal of the project, design constraints, assumptions, dependencies, detailed data description with data dictionary, system architecture with its components, user interface, detailed design, libraries and tools and time planning. The main objective is to provide design description so that by following this document, the construction of the system will easily satisfy the requirements in the SRS.

1.4. Overview

This document has nine sections to state the detailed design of the project. In the following section, we will give a brief system overview. In section 3, we will state the design constrains, assumptions and dependencies. In following section, detailed data description and data dictionary will be provided. In section 5, system architecture will be covered with its components. Section 6 will give information on interface design. Detailed design is explained in section 7. Libraries and tools that will be used in the project are explained in section 8. Finally, a revised Gantt Chart is provided in section 9 in order to state the time line determined for the progress of the game.

1.5. Definitions, Acronyms and Abbreviations

SRS: Software Requirements Specification
DDD: Detailed Design Document

1.6. References

[1] FatihProjesi.(n.d.)Retrievedfromhttp:http://fatihprojesi.meb.gov.tr/site/

2. SYSTEM OVERVIEW

In our game, the user will start to play the game first by selecting an avatar. This avatar will be the main character of the game. There will be two different choices for avatars. The user can choose either a girl or a boy avatar. After choosing an avatar the general story is as follows. The character will firstly encounter with a witch that curses the world by imprisoning two scientists who are very significant for world to develop namely Archimedes and Sir Isaac Newton. With the effect of the curse, those two scientists will not be able to work on their inventions and today's world will be in a chaos.

The character is expected to save those two scientists by going back in time and following the clues provided by "Fizbot" which is a robot, wanting to help the character. There are two levels hence two scientists to rescue. In every level there are two quests we want the player to solve. These quests we chose are related to the physics topics taught in 9^{th} grade and also to the scientist's research area. During solving two quests using the clues, the character will gain a key to open the door in which scientist is kept and move on to save the other in the same manner. Saving each scientist will develop world further and after saving both of them it will be the same as the one before the witch's curse.

The curse of the witch, the development of the world and the final scene after finishing the game will be provided as videos at the beginning, in between the levels and at the end of the game respectively.

The topics of the quests are: buoyancy of liquids, simple machines, conservation of energy and momentum and equilibrium respectively.

In the quest of buoyancy of liquids, the character will encounter with a pool which is filled with water. Inside the water there is a long, thin wood whose half is sank into the water and the character is not able to reach it. At the bottom of the wood there is a key attached and character must take this key to save the scientist at the end of the level. Nearby the character there are four valves containing different liquids. The aim of the character is to change the density of the water by adding different liquids to the pool by using valves in order to reach the key. If the density of water is decreased, the wood will sink more and character will not be able to reach it. In this case the liquid added to the pool will be thrown from the hole at the top of the pool and character will be able to add new liquids to the pool. Therefore, character needs to add a liquid that will increase the density of water in order to reach the key. When character reaches the key, he/she can move to the second task which is the simple machine quest.

The next task is a simple machine problem which the player has to solve a puzzle composed of rotors. In order to open a door, the player needs to find a code related to the rotors' number of rotation. After he/she solves three puzzles, he/she will be able to open the door using the key gained in the previous step. After saving the scientist player will move to next level.

The first task of the second level is related to the conservation of energy. In this quest the player will encounter with a deep rift. Therefore, he/she is not able to pass across to the other side. At the other side, there is a ruined university which is destroyed by the witch at the beginning of the game where Newton had studied. In order to enter the university, a bridge must be dropped from the entering of the university to cover the rift. There is also a ball that character will throw. Bridge will drop slowly when the ball is thrown by the character and reaches the corresponding buttons which are on the other side of the rift. For the first shoot there will be no friction on the rift. However, when a ball thrown by the character touches the button, bridge drops a little bit and some friction will be added to the environment. After character presses all four buttons, the bridge will be down completely with a key at the tip of it. This is the end of this task.

The next task is related to the momentum and equilibrium. In this task player needs to walk on a seesaw without disturbing the balance of it. In order to achieve this, he/she has to change the positions of the objects on the seesaw.

Passing all the four steps means completing the game and saving the world from the curse.

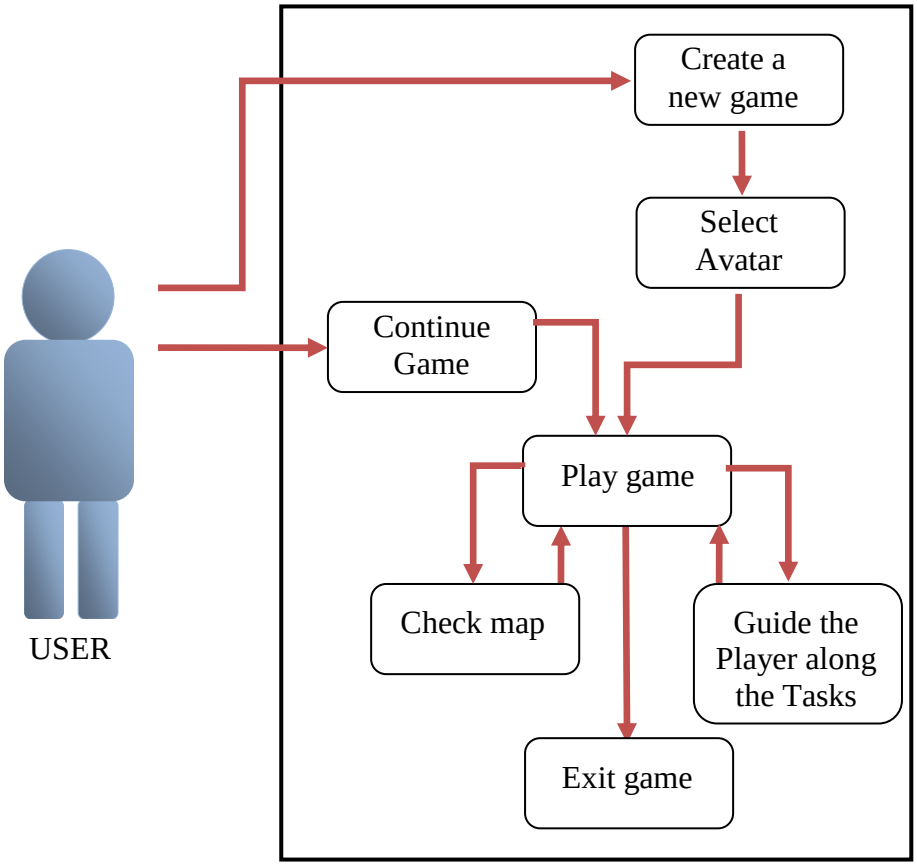


Figure-1 Product Functions

3. DESIGN CONSIDERATIONS

3.1. Design Assumptions, Dependencies and Constraints

The game will be developed on tablets and the operating system of these tablets is assumed to be at least Android OS.2.0. Since the game is a part of the “Fatih Project” [1], the tablets are determined by The Ministry of National Education of Turkey. The game will be developed using “Unity3D”, because it is a good way to design a game on Android platform. The tablets that are provided by Ministry of Education are able to meet the minimum requirements of Unity3D. The processor of the tablets is not determined yet, however it will be single or two core processor. The core of the processor will not affect the performance of the game significantly. It is assumed that the game will respond to the user in 0.1 second however this time can be changed according to the processor used in the tablets. The tablets have the necessary two-dimensional input hardware such as touch surface. The tablets supports also multi touching.

It is assumed to make two levels for completing the game and for each level students have to solve two different tasks in each level. However, according to the going process the number of the level or tasks can be increased. Moreover, the number of avatars is assumed

to be two for this moment. However; this number can also be increased according to the process.

The game will be used only with one user. No other users can connect to the game at the same time. The number of avatars that the user will be able to select before starting to the game will be limited. The language that will be used in the game will be in Turkish. No other language will be provided to the user. Since the game will be first used for "Fatih Project", it will only be used in tablets.

3.2. Design Goals and Guidelines

- Maintainability: It is the ease that a product can be maintained in order to correct defects easily, meet new requirements, cope with a changed environment and make future maintenance easier. Since after finishing the project we may be will not be able to develop this project further: therefore, it needs to be maintainable at the beginning to decrease other developers' effort.
- Reusability: helps the developer to add new functionalities to the code with slight or no modification. Using reusable modules and classes reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs and localizes code modifications when a change in implementation is required. Since our project will be spread through other universities and also or game will be developed further by us or another teams in the future, it is really important for us to implement our project in a reusable manner.
- KISS principle: it will be a guideline the help us to maintain the design simple as possible during the design process. Moreover, avoids unnecessary complexity of the product. In order to understand each other's work and guide the developers who will take this project over after us, we will try to implement KISS principle in our development.
- Usability: Since the product will be used by 9^{th} degree student, it is one of the most important design considerations for our project. Therefore, the tasks used in the game should be simple, understandable and informative for students.

4. DATA DESIGN

4.1. Data Description

Since our game is heavily story based, the main data object in the project is Game and Avatar objects. Avatar, Sound, Transition, Item, Task, Map, Scene objects are all associated with the Game object. Also Avatar object is associated with Task and Item objects so that the story will be shaped according to the actions of the character on items in specific tasks. Since there is not database in the project we will use a txt file to keep the information of the saved game. Also we will have graphics files from a professional graphics designer to compose our scenes. Following parts will introduce these data objects and their attributes in detail.

4.1.1. Data Objects

Avatar: This data object is representing the avatar choice of the leading character of the game that is the center element and controlled by the user.

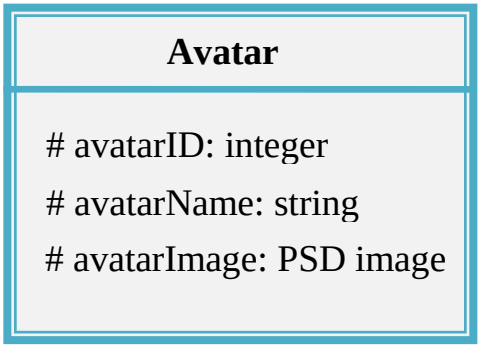


Figure-2 Avatar object

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

avatarName: holds the name of the leading character 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.

avatarImage: holds the avatar graphics designed by the graphics designer which represents the avatar of the player.

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

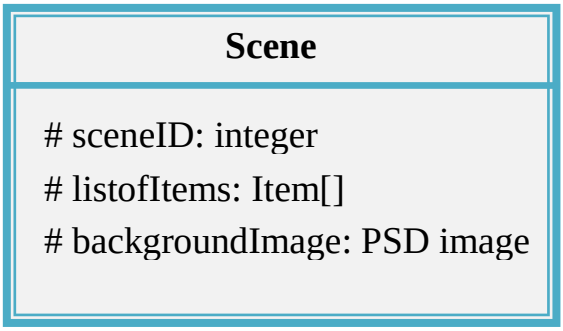


Figure-3 Scene object

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 problems and also the items which are going to be displayed in the current scene of the game. From witch and Fizbot to arbitrary objects in the game are all defined as items and separated by the type parameter.

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.

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

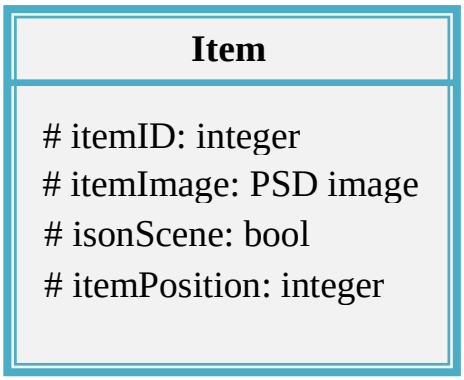


Figure-4 Item object

itemImage: 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.

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

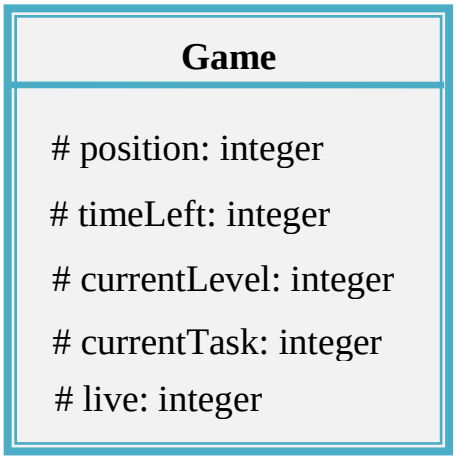


Figure-5 Game object

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

timeLeft: holds the time remained in order to be able to give feedback to the player at the end of the game as points.

currentLevel: holds the level information which the character is at in the game currently.

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

live: holds the number of lives the player has during the game. This number will determine the number of trials the player has. He/she will lose one live when he/she fails to accomplish one task.

Map: This data object is representing the map screen of the game in order to show user which year he/she is currently at and which tasks are completed, which are not.

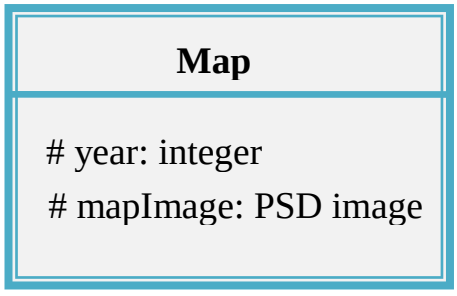


Figure-6 Map object

year: holds the year information of the game according to the levels. The reason to store this data is the game includes time travels and player may need to know which year he/she is

currently at.

mapImage: holds the graphics for representing the map which is designed by the graphics designer.

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

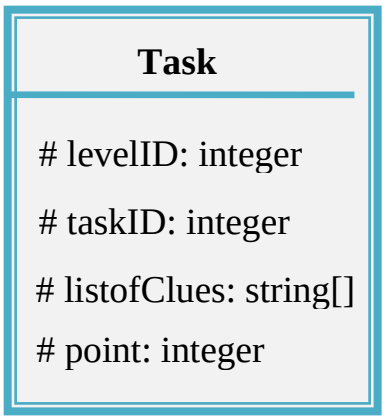


Figure-7 Task object

levelID: holds the id information of the level to load proper quest.

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 level and task id.

listofClues: holds the clue items that the character picked until he/she reaches the task. These items will be displayed on the screen and character will be able to read these clues whenever he/she wants.

point: holds the score of the player for each task.

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.

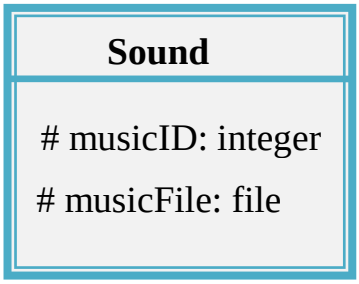


Figure-8 Sound object

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 videos that integrate the tasks to each other. This object will be very helpful for the player to understand the story in a continuous manner.

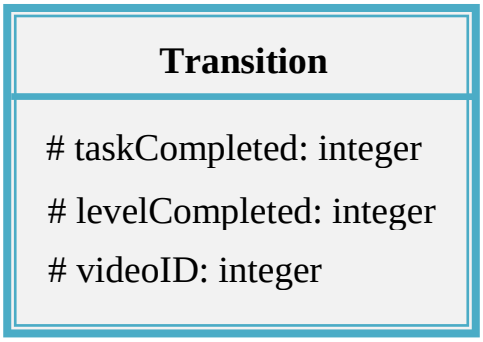


Figure-9 Transition object

taskCompleted: holds the task information which is lastly passed to show the related video to the player.

levelCompleted: holds the level information which is lastly passed to show the related video to the player.

videoID: holds the video id which will be displayed when needed.

SavedGame.txt: This file holds the position, timeLeft, live, currentTask and currentLevel attributes of Game object and avatarID and avatarName attributes of Avatar object in order to resume a saved game from the position where the player quit the game. The attributes will be separated by new line character in the txt file. Here is a sample file:

position: x
timeLeft: 15
live: 3
currentTask: 1
currentLevel: 2
avatarID: 1
avatarName: Deniz

Figure-10 SavedGame.txt file

The format of the graphics files which will compose our gaming environment and will be prepared by the graphics designer is determined as Photoshop files. However, this format can be changed according to our needs.

4.1.2. Relationships

In this section we will introduce the relationships between all the data objects described above.

Avatar-Game: Game object has only one Avatar 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.

Map-Game: Game object has only one Map object that is dynamically changed.

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

Avatar-Task: Avatar object will be interactive with a Task objects to pass in order to proceed.

Avatar-Item: Avatar 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.

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.

4.2. Data Dictionary

avatar: is a visual object in the game that represents the player. Players can separate themselves from other characters in the game by this visual. The most important part is that the avatar is under the control of the player.

task: is the each problem need to be solved by the user in levels. There are two tasks in each level.

Item: means all the objects in the game environment available for player to use in solving the problems, for example; bridge, key, tree, Fizbot, witch etc.

transition: this is the object which will be used to integrate the levels so that the player will understand the whole idea of the game.

sound: this object is for the audios at some specific situations like falling.

map: can be thought as a little map to show where is our character in the game, which year and which task especially.

game: the whole project which is depending on solving physics problems.

scene: one screen shot at a time, in total composing the whole game, when the game is running. For example: items and character in the current screen.

5. SYSTEM ARCHITECTURE

5.1. Architectural Design

Our system will have a modular structure; therefore it is divided into subsystems which will accomplish the goal of the whole system. There will three components or subsystems, which are namely:

PreGame Module

DuringGame Module

PostGame Module

Component diagram of these subsystems is given below:

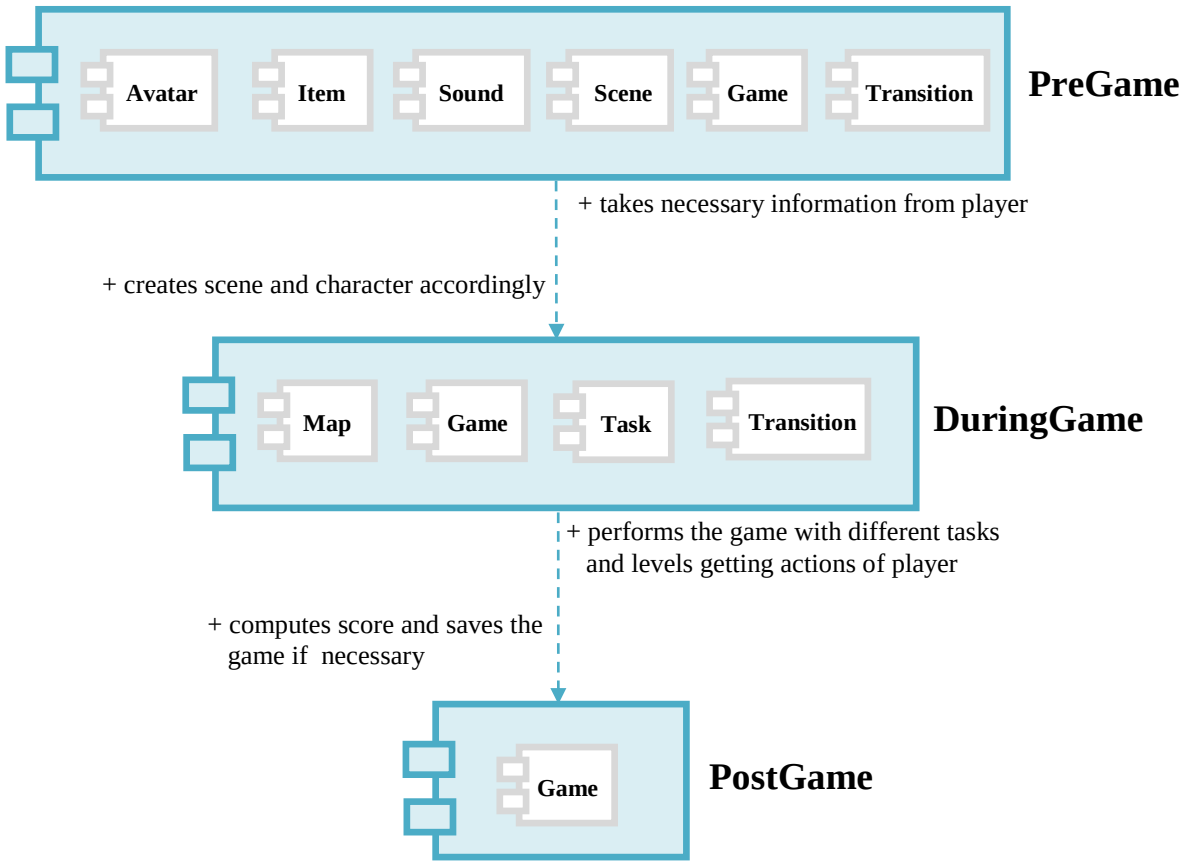


Figure-11 Component diagram of the system

First component of the system is PreGame. In that stage, necessary information from the player will be taken, based on which the game will be prepared for the player. After taking those information, scenes, characters, sounds, and if necessary items will be adjusted and placed accordingly. Then the informative video will be shown.

The second one is DuringGame component, which is composed of four sub-modules, namely Map, Game, Transition and Task. In the Transition as the name reveals itself, player will be informed about the previous and upcoming level of the game and system will make sure that player follows the game appropriately. In the Map module, there will be configurations about the map to give place and time information to user during game. The Game and Task subsystems are going to be where the actual game scene, problem and characters will take place.

The last one is PostGame subsystem. This subsystem gives us the opportunity to save the game in txt file format. For this reason, operations on these files will be simpler. If desired by the player, game will be saved for the future play. If not, current condition will be shown.

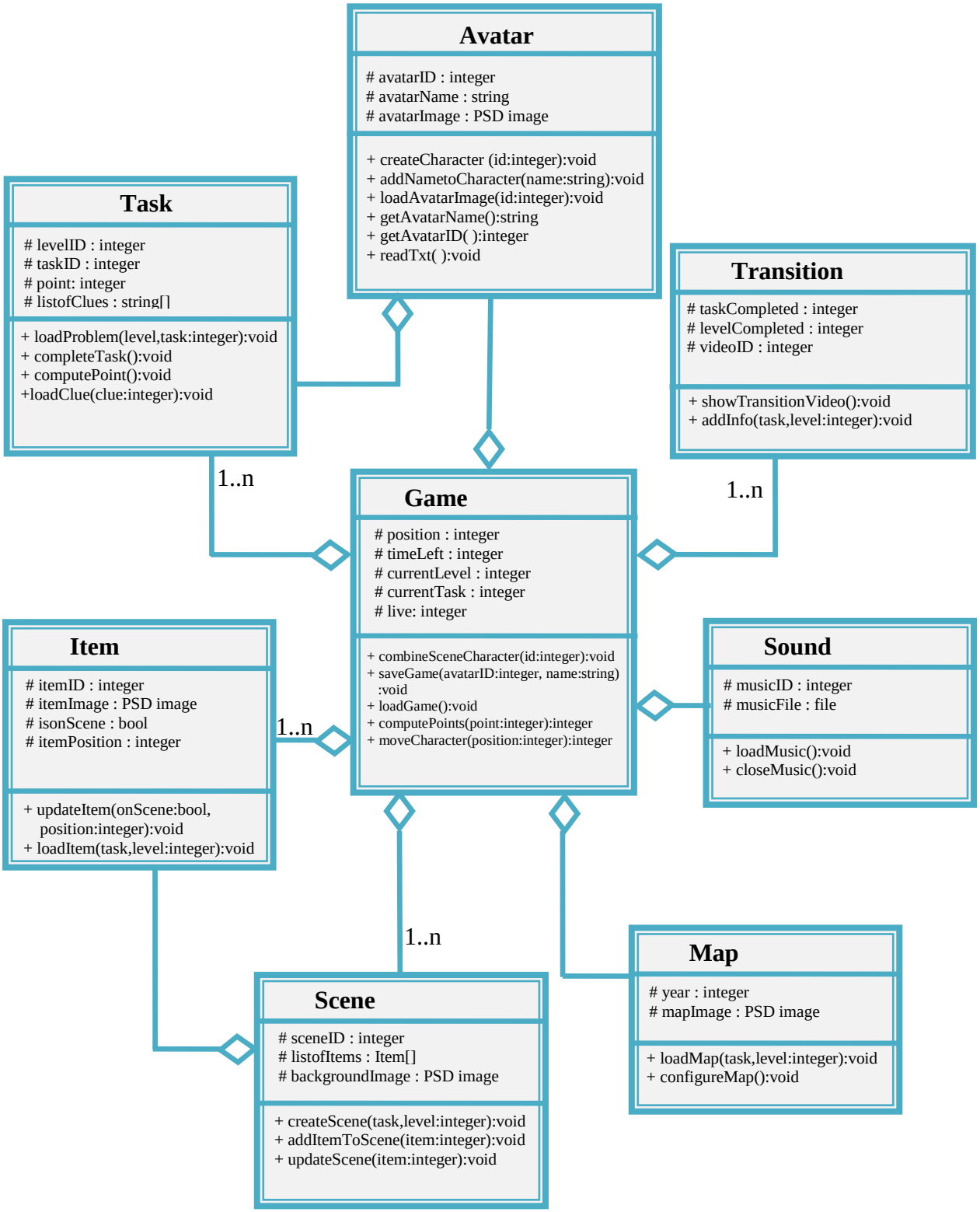


Figure-12 General class diagram

5.2. Description of Components

5.2.1. PreGame Component

5.2.1.1. Processing narrative for PreGame component

This is the subsystem 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, game or settings. 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 of some systems actions such as loadItem, loadMusic, createScene.

5.2.1.2. PreGame Component Interface Description

For this subsystem, user will encounter a user interface with a menu which he/she can configure some settings and take some actions mentioned in the previous sections to pass data to Game module. 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 choose an avatar that will be used throughout the game. For this purpose output of such action will be the avatar 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. Therefore, this component is important at constructing required data for DuringGame module according to the data taken by the users. 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

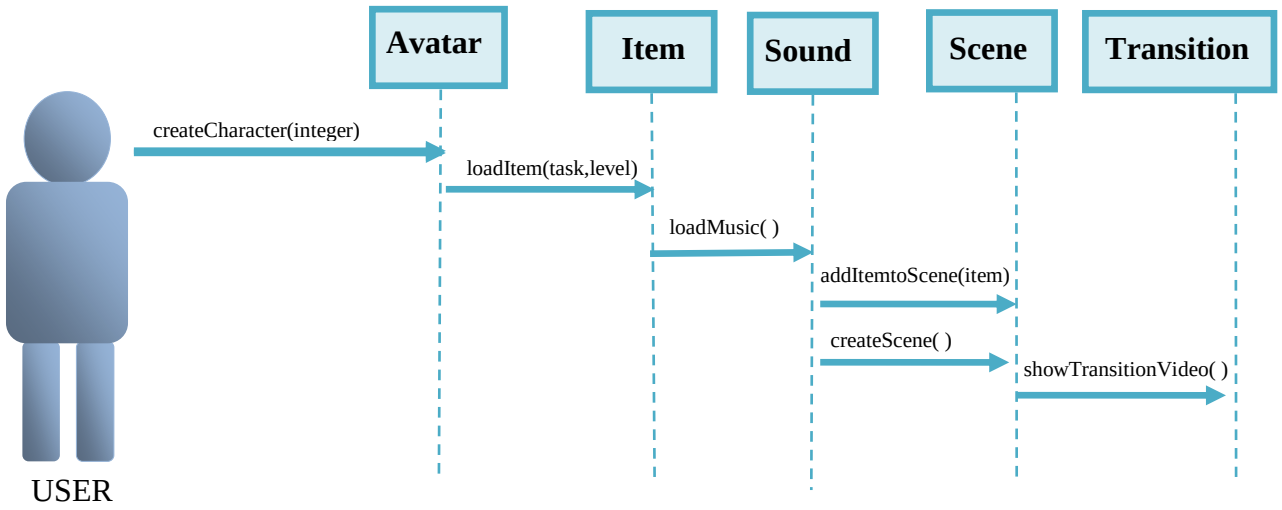


Figure-13 Sequence diagram of “Create a New Game”

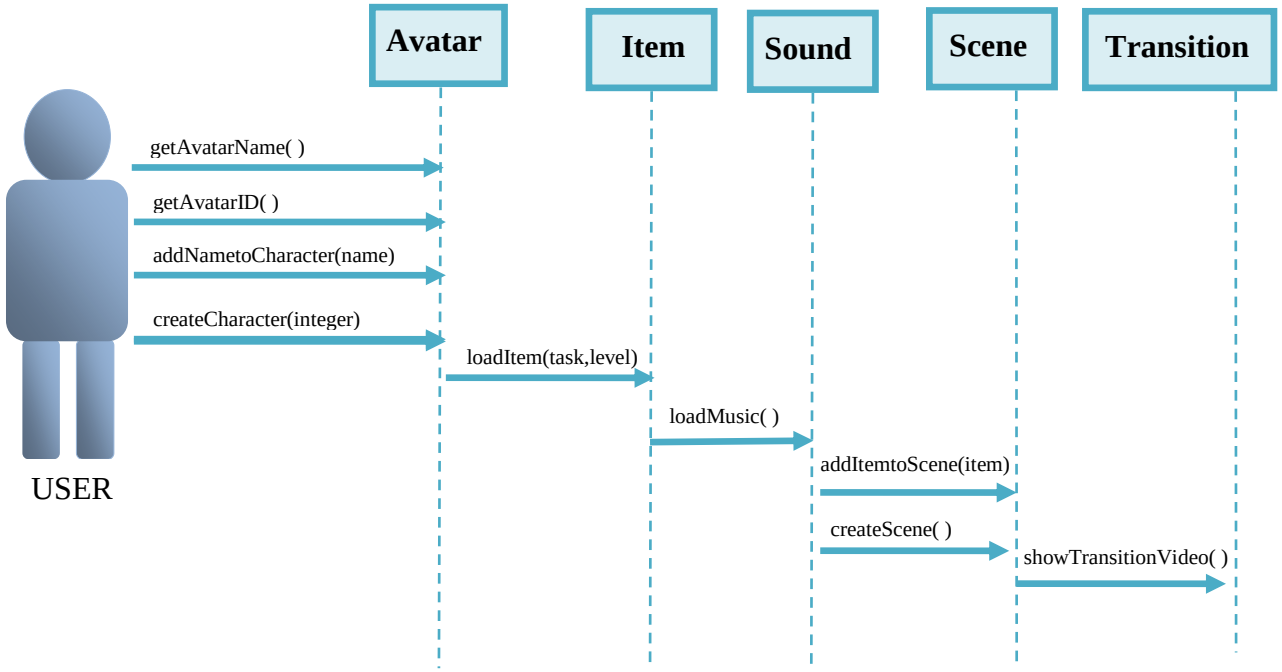


Figure-14 Sequence diagram of “Select Avatar”

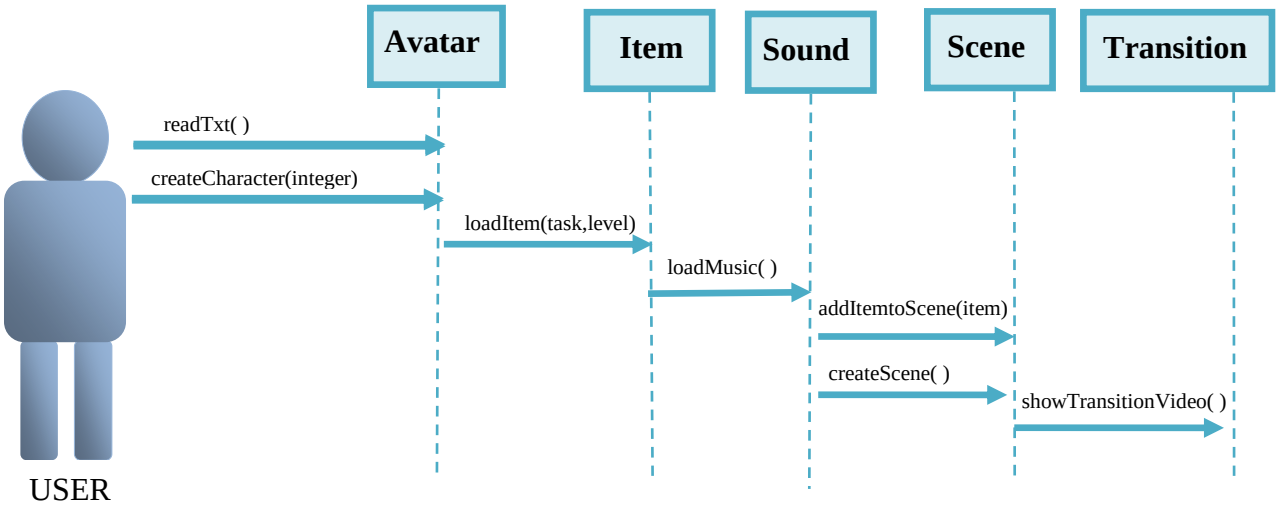


Figure-15 Sequence diagram of “Continue Game”

5.2.2. DuringGame Component

This is the topmost 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 DuringGame component

This subsystem is where the actual game playing takes place; however we find it useful to divide this subsystem into four subcategories, which are namely Map, Task, Transition and Game. In the Transition, transition between levels and parts within the levels will be done smoothly, related actions will be taken, and system will be prepared from the new level or part. On the other hand, in the Game 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 in the available Task. Also the user will be able to see a map whenever he/she wants to understand where he/she is in the game.

5.2.2.2. DuringGame Component Interface Description

This module also has a user interface. 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 behind the game.

In the current Task module within the Game module, user will have two controller buttons, as he/she touches those character on the screen will act accordingly, and user will

be able to complete the game. Also map interface will be supplied to the player in order to make him be able to keep track how much he/she accomplished in the game so far.

5.2.2.3. DuringGame Component Processing Detail

The algorithms in this stage will be implemented depending on the problems which player will have to tackle with. In our game there exist two levels with two tasks. So, there will be a control mechanism to decide whether the player did the right thing to solve the problem in the current task which is part of the current level. So for each level, there will a different algorithm implemented. This component will call itself to implement the new task and new level if necessary.

5.2.2.4. Dynamic Behavior of DuringGame Component

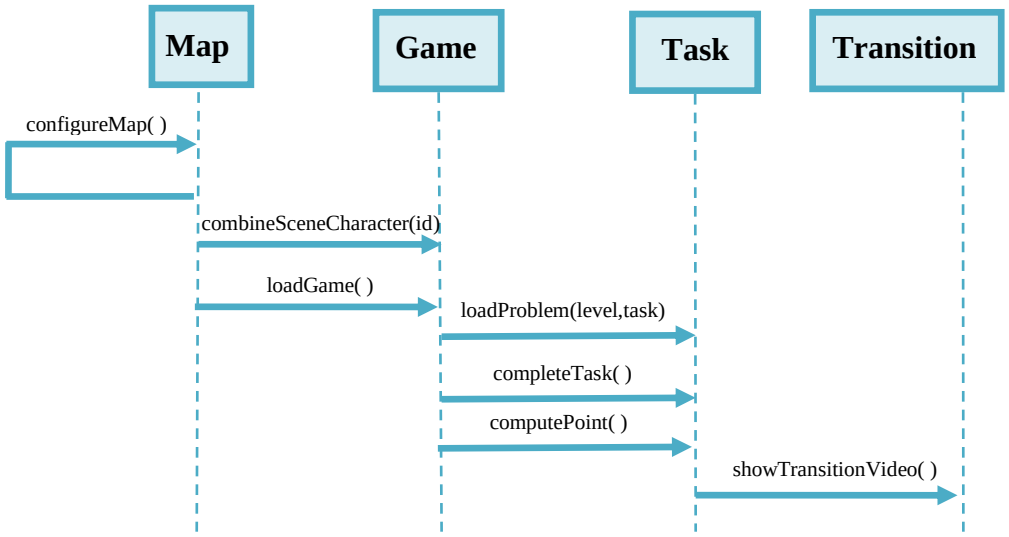


Figure-16 Sequence diagram of “Play Game”

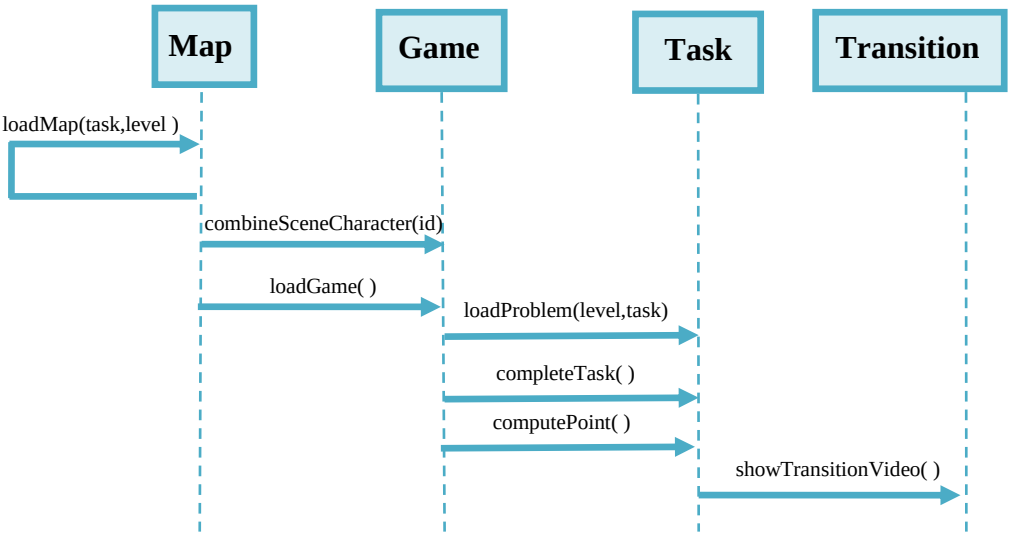


Figure-17 Sequence diagram of “Check Map”

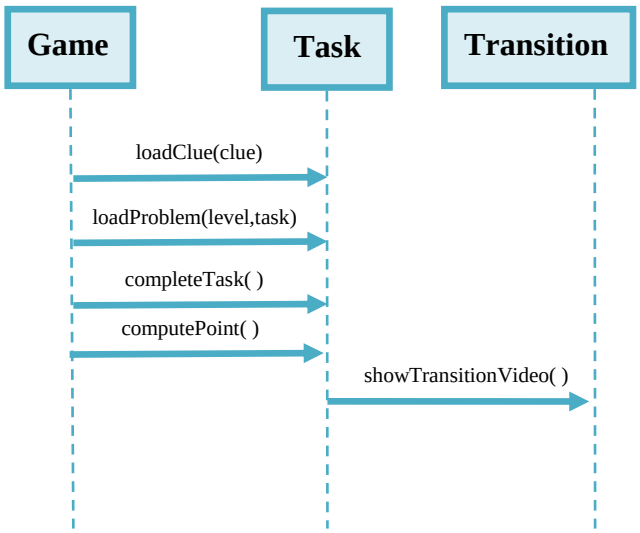


Figure-18 Sequence diagram of “Guide the Player along the Tasks”

5.2.3. PostGame Component

5.2.3.1. Processing narrative for PostGame 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 saveGame, terminateGame, computePoints etc.

5.2.3.2. PostGame 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 wants to return 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. PostGame 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 that information.

The weight of the levels will be different from each other and they will be determined later on according to the difficulties of the levels.

5.2.1.4. Dynamic Behavior of PostGame Component

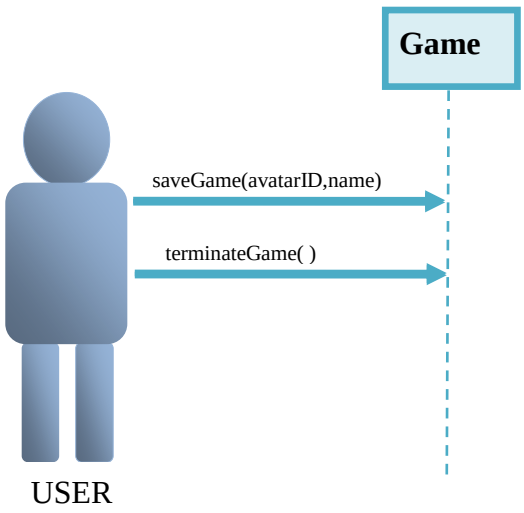


Figure-19 Sequence diagram of “Exit Game”

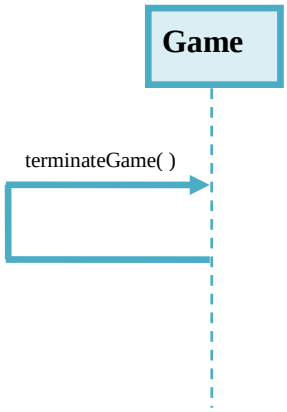


Figure-20 Sequence diagram of “Finish Game”

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 on each other, 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.

Moreover, we chose Unity3D because the most important feature of it is that when a game is developed in Unity3D, it is capable of working with full functionality on multiple platforms and operating systems by only writing a plug in for different platforms and integrating the code. This feature will help us to adapt our code in different platforms easily. Another reason why we chose Unity is it is easy to add new levels to the game later and it is providing a good environment of coding for games designed for Android OS.

5.3. Traceability of Requirements

SRS REQUIREMENTS	DDD COMPONENTS
3.2.1 Create a New Game	5.2.1 PreGame
3.2.2 Play Game	5.2.2 DuringGame
3.2.3 Select Avatar	5.2.1 PreGame
3.2.4 Check Map	5.2.2 DuringGame
3.2.5 Exit Game	5.2.3 PostGame
3.2.6 Continue Game	5.2.1 PreGame
3.2.7 Guide Player along The Tasks	5.2.2 DuringGame

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 two choices whether creating a new game or continuing the game which is saved last time. If user selects continuing game button, he/she will directly start to play the game where he/she was in last time. If new game is chosen, user again has two options. He/she can choose an avatar or start to play game without choosing avatar. When user chooses to play the game without selecting an avatar, a default avatar will be provided by the game. User can also click the help button to be able to learn the actions that the avatar can make or exit the game without playing.

User will play two levels in order to finish the game. In each level two different tasks must be solved by the user in order to continue the game. Before reaching one of the tasks, user will collect some paper of clue that will help the player while solving the task. And also these clues will increase the trial numbers of the player. At the beginning of the game a default trial number will be given to the user. This number will be five. User will be able to try to solve a question until 5 trials if he/she did not collected any clue paper. However, if clue papers are collected, the number of trials will increase according the number of clue papers. When the number of trials is decreased to zero, the game will be over. User can touch one of the clue papers which are situated at the top of the screen to read the information written. In any time of the game, user can touch to map button, which will be unique to each level. The map helps user to see how many tasks are there and in which year is the user is in. Then he/she can return to game and resume playing. The player aim is to save a scientist in each level. Therefore, to save them he/she needs a key. This key is given to the player if the first task of the level is solved. If the second task of a level is solved, player will use the key that is collected in the first task and save the scientist. If user wants to quit the game, he/she can touch the exit button which is again on the screen. At this stage, the system asks user to exit game with or without saving.

6.2. Screen Images

This part gives information about the screen appearance of our game. However, the images used to represent screens in this section are not the actual screen interfaces that will be used in the game. These are only some prototype of screens.

6.2.1. Main Menu Screen

The “Main Menu Screen” has five buttons called “Karakter Seç”, “Oyuna Başla”, “Oyuna Devam Et”, “Yardım” and “Çıkış”. If the player did not save the previously played game “Oyuna Devam Et” button will be displayed but the player will not be able to touch it. If not he/she can choose to continue the game or start a new game.

Figure-21 Main menu screen

6.2.2. Select Avatar Screen

"Select Avatar Screen" helps user to select an avatar. At the moment there will be two different avatars. One of is a girl and the other is a boy. The user should write a name that will be his/her character name for the game. After that user can only touch one of the avatars that he/she wants to play with and then touch the "Oyuna Başla" button. From the avatar page, the player starts to play the game.

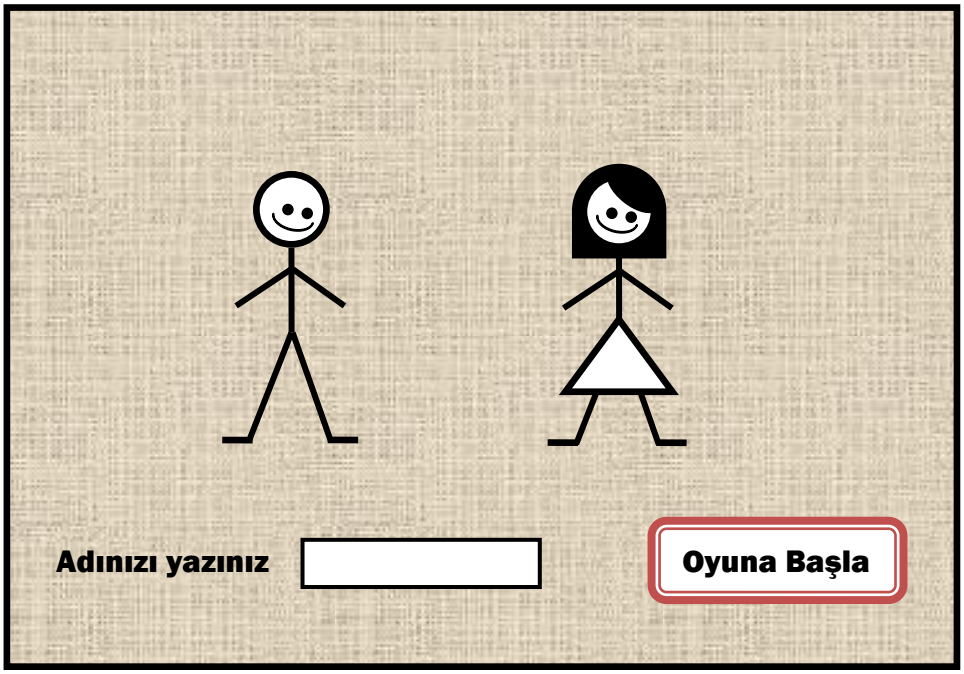


Figure-22 Select avatar screen

6.2.3. Help Screen

The “Help Screen” explains the user how to play the game: how to use buttons to move the avatar, how to collect the paper clues and keys, how to read the inside of a clue paper and how to use the key at the end of the task.

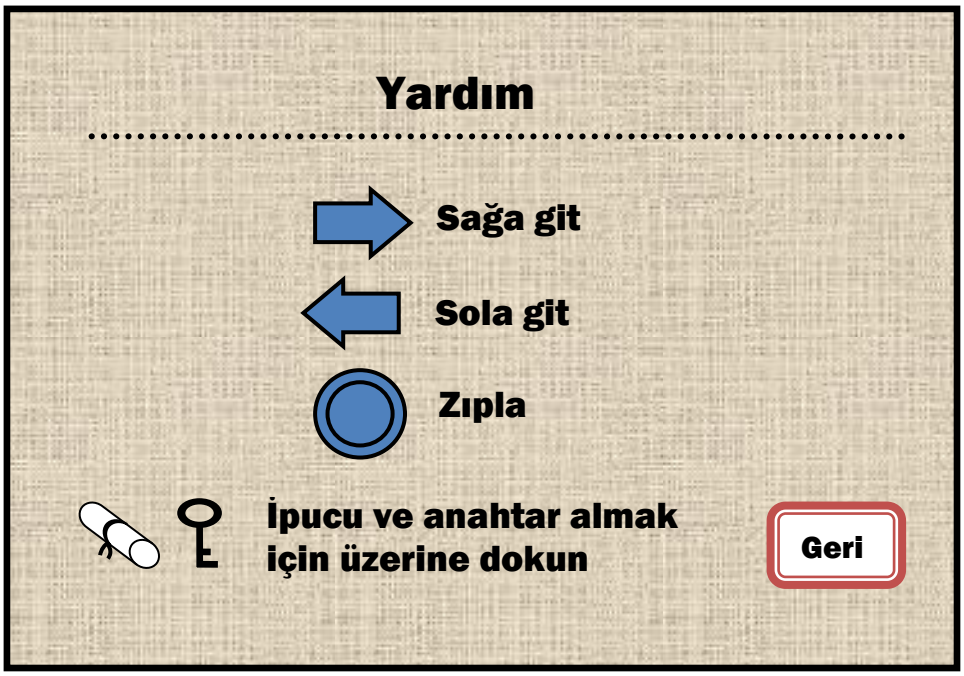


Figure-23 Help screen

6.2.4. Introductory video screen

“Introductory Video Screen” is displayed when the user wants to start the game. Before starting to play the game, this video will appear on the screen and will explain the story of the game to the player.

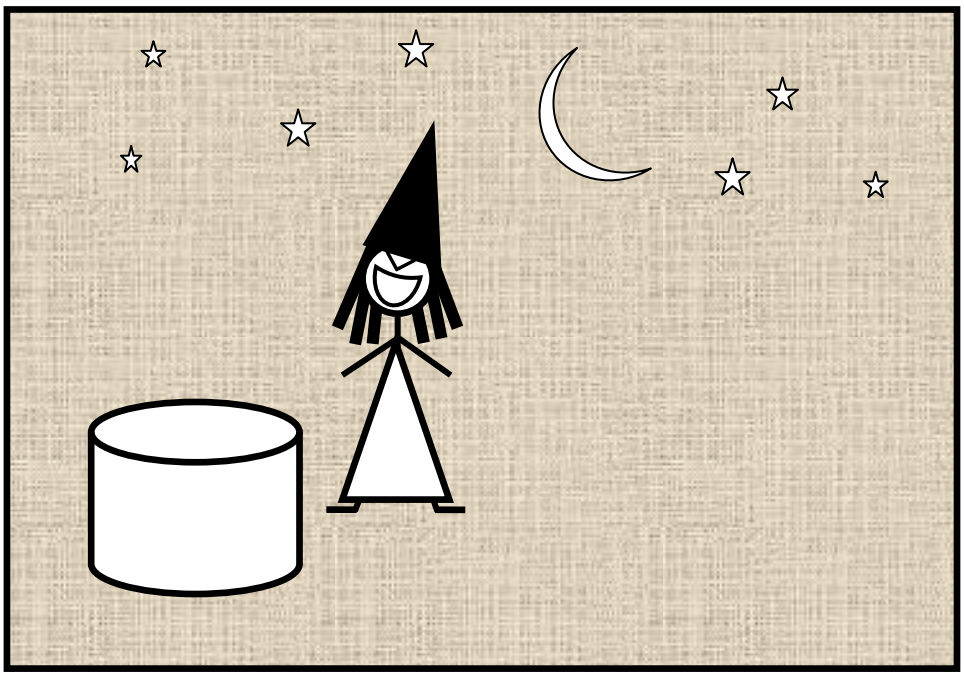


Figure-24 Introductory video screen

6.2.5. Level Screen

There are two levels in the game. The roots and the tasks that are going to be solved are different for each level. However, the places of the items on the screen are the same for both levels. Figure-25 and Figure-26 are the prototypes of two different tasks.

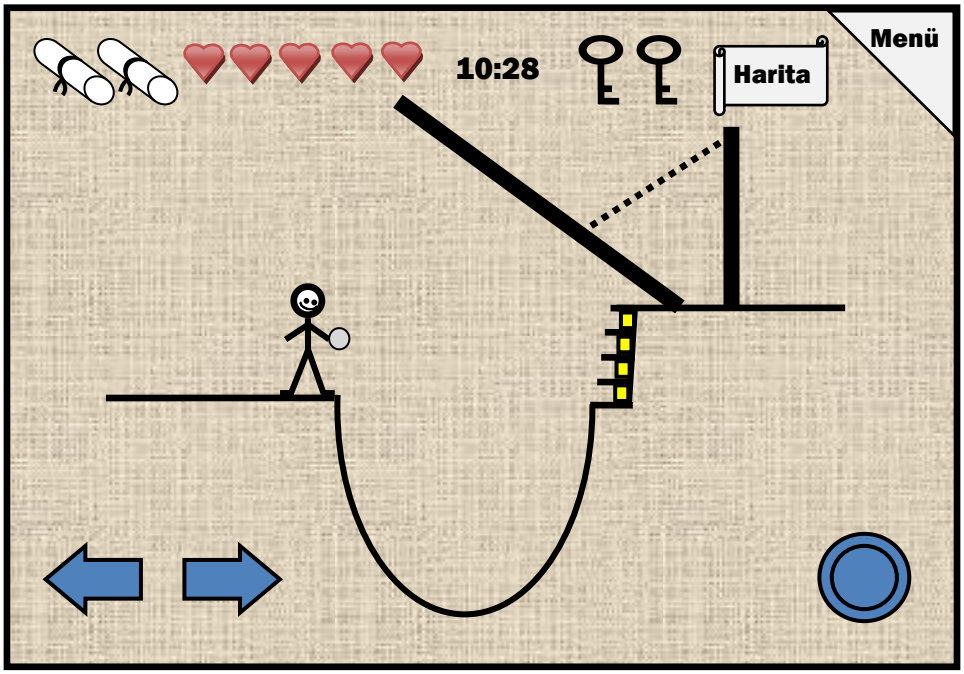


Figure-25 Level screen task - 1

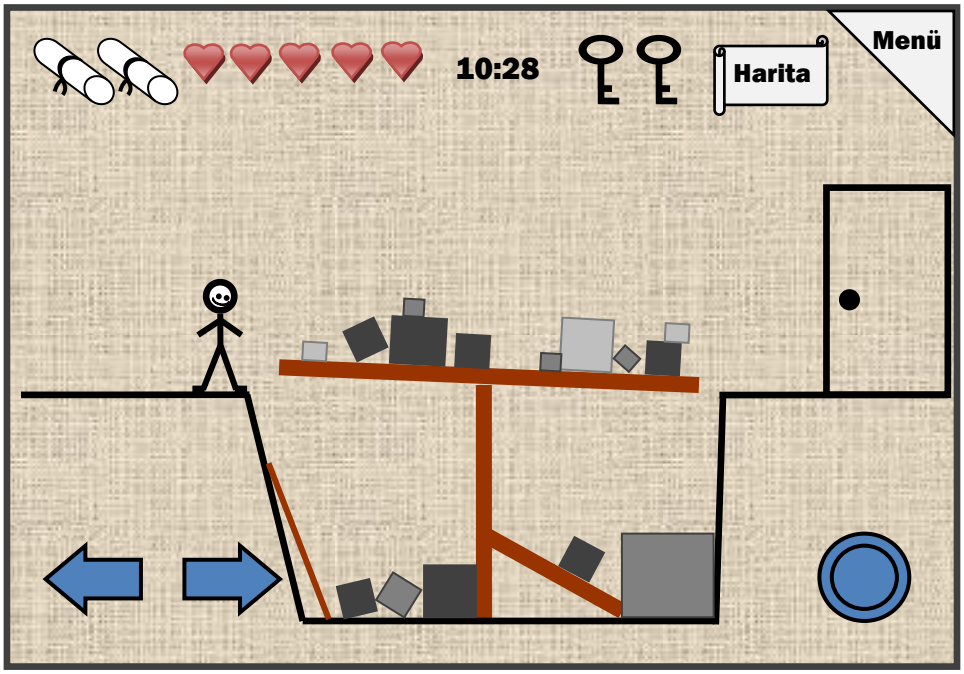


Figure-26 Level screen task - 2

At the top of the screen there will be icons that show the key and clue papers which are collected by the player during the game. If player wants to use one of these icons, he/she should touch on one of the icon. There are also “Menu” and “Harita” buttons. If the player touches the “Menu” a menu screen will be appeared. If the player touch the “Harita”, button a map will be appeared. There is also a counter that shows the time passed during the game and a number of trying rights which are represented with hearts in the screen. If user gives wrong answers to the questions, the numbers of hearts will decreases. If user collects the clue paper in the game, he/she will get rights for each clue collected and the number of hearts will increase accordingly. The right, left arrows and the blue button allow user to move right and left and jump respectively. The figure shows a scene belonging to one of the task which will be in second level. There will be also other scene with different tasks during the game.

6.2.6. Main Menu Screen during the Game

There is a button called “Menu” on the “Level Screen” that helps player to exit game while playing or get some help. If player does not want to quit the game, he/she can touch the “Continue Game” button.

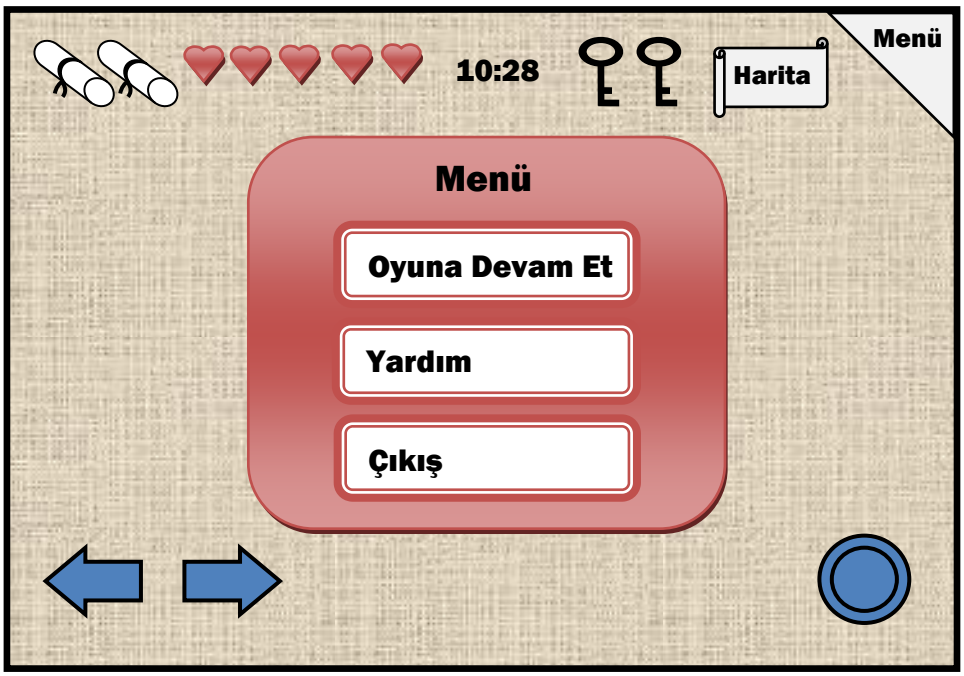


Figure-27 Main menu screen during the game

6.2.7. Map Screen

The player can see the “Map Screen” by touching the button called “Harita” in the “Level Screen”. “Map Screen” contains a map that shows the root that the player should follow during the game and remaining tasks. There is also a time line at the left of the map which indicates the year of the player is in.

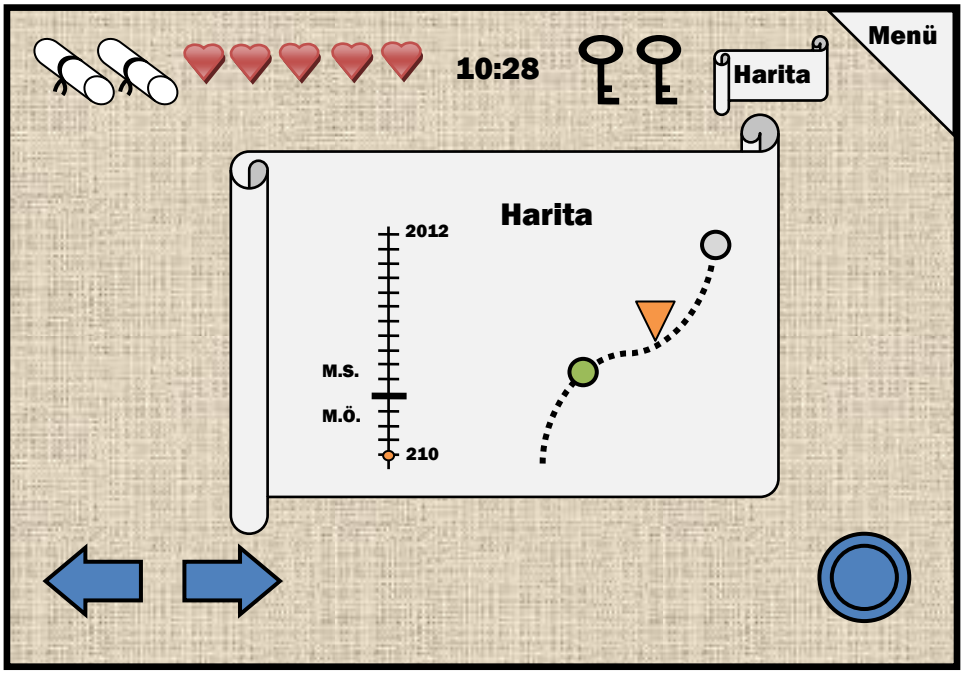


Figure-28 Map screen

6.2.8. Clue Screen

When a player is trying to solve a task in the game, he/she can get some help from the clue paper that are collected on the root. These clues give some information about how to solve a question. If the player wants to read what is written in the paper, he/she can touch one of the clue icons which are at the top of the "Level Screen". After touching, a screen that shows the inside of the paper will appear on the screen and the player will be able to read it.

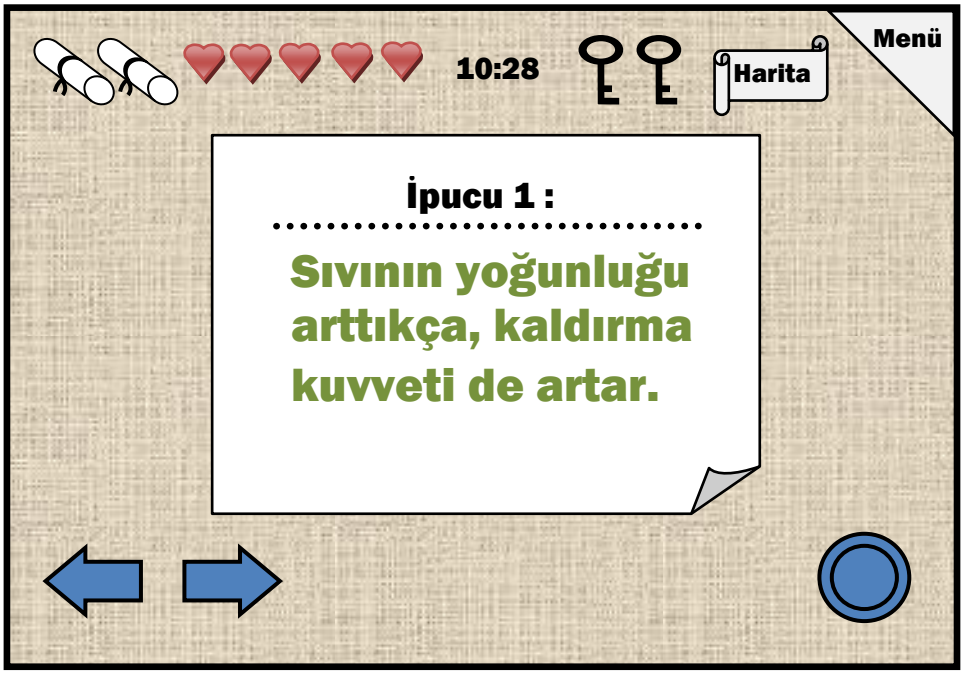


Figure-29 Clue screen

6.2.9. Level Video Screen

"Level Video Screen" is shown between the two levels. When the player completes the first level, there will be a very short video that shows the player that he/she has completed the level successfully. This video will give the player the impression that the world is developed more when a scientist is saved. Moreover, the time that the player is spent in that level will be appeared.

6.2.10. End Video Screen

"End Video Screen" is displayed when the whole game is completed by the player. This video gives impression to the player that he made something very useful and also to give some courage. Moreover, the time that the player is spent to complete the game will be appeared.

6.3. Screen Objects and Actions

The player first sees the "Main Menu Screen" in which there are several choices. Player can select an avatar, take some help about the game, start the game without

selecting an avatar or continue the previously saved game. When player touches the “Avatar Seç” button, he/she will be directed to the avatar selecting screen. After selecting an avatar the user touches the “Oyuna başla” button and an introductory video will appear on the screen that explains the story of the game. After introductory video, the player starts playing the game from the first level. After completing the first level, a video will appear to show the player that the scientist is saved and the word become more developed and the time passed to complete the first level will be shown. Then the player will start to play the second level. When the second level is completed, an ending video will appear to give the player the impression that the world is saved and his/her total amount of time spent to play the both levels.

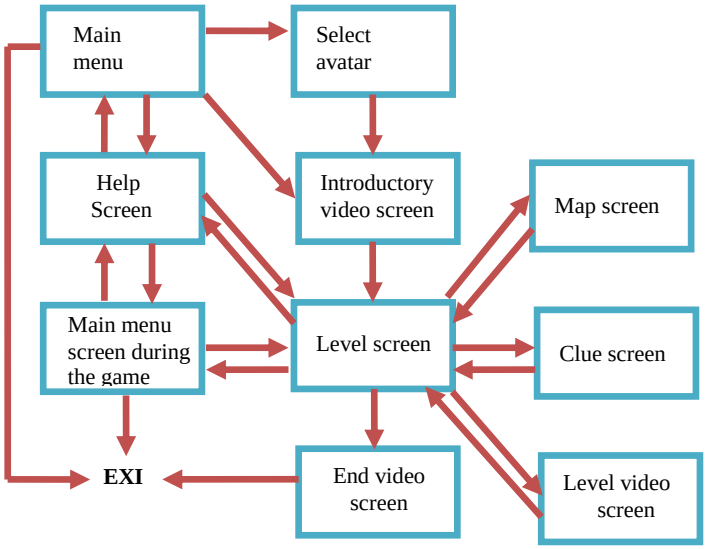


Figure-30 Game screen flowchart

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 Avatar, Item, Sound, Scene and Transition.

The Avatar subcomponent provides user two characters that he/she can select before starting the game. User can select one of them before starting to play the game. If no character is chosen, a default character will be provided by the system. When player chooses a character, an avatar name will be written by the user, an id and image id belonging to the selected avatar will be provided to the PreGame component. This subcomponent will be loaded by the PreGame module at the beginning of the game.

The Item subcomponent contains the images in Photoshop file format which is designed by the graphic designer. These are the images where the player is in interaction with user such as buttons, clues or any other moving items. All items have a unique item id, an image belonging to the item, the position of the item that will be used in the game and also an indicator whether this item is in the scene or not. These items will be loaded into the scene when it is needed.

The Sound component contains the sounds that will be used during the game. These sounds will be loaded to the game by the PreGame component when some actions are taken by the character such as terminating a task or collecting a clue. There will be different sounds for different situations and each sound has a unique music id and a music file belonging th the sound.

The Scene component is different from the item component which 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 clues list that the scene will use in the game.

The Transition component contains video that tells the story of the game to the player. This subcomponent will be loaded by the PreGame component by using the video id of the video to the game before starting play the game.

Uses/Interactions:
The PreGame component is in interaction with the DuringGame component. When player chooses an avatar 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.
Resources:
- Animation used for the description of the game
- This component uses images in Photoshop file format.

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, two 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 avatar.

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 map, loading the game and tasks, and also showing transitions between levels by using some videos that shows the player that the level is completed.

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 Map, Game, Task and Transition.

The Map subcomponent helps the player to show in which year he/she is in and also the places of the tasks that he/she will encounter during the game. This component contains information about the year of the level and also a map image in Photoshop file format. In each level, there will be a different map image. The user will be able to use this component whenever he/she wants to during the game, so this map should be updated by the DuringGame component in the game.

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 level, there will be two different tasks. Each task has a unique id and has same level id if they belongs to the same level. DuringGame component will use these ids to load the related task in the game.

The Transition sub module contains different videos that will be displayed after a level is completed. In each level, a different video will be displayed and the decision of which video will be shown will be determined by the id of the task and the video.

Uses/Interactions:
The DuringGame component is in interaction with the PreGame component. When player chooses an avatar, the name, id and the image belonging to the avatar 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 level 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.

Resources:
The DuringGame module uses the following libraries:

1. Character Controller
2. Light Flares
3. Particles
4. Physics Material
5. Projectors
6. Scripts
7. Standard Assets
8. Toon Shading
9. Water

This component also uses images in photoshop file format.

Processing:
The algorithm of the DuringGame component uses a loop which starts by updating the map, 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 avatar to the PreGame module and the selected avatar is loaded in each scene during the game. If player starts to play a new game, the map, 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. Since during the game the player can see the map whenever he/she wants, the map should be updated each time by the DuringGame component.

Interface/Exports:
There are mainly 4 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, transition videos and the map.

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, and how many lives he/she consumed to complete the tasks so far. 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 avatar 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. This video will show how much the world has changed thanks to player completing all the tasks and saving all the scientists.

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, avatar 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 AND TOOLS

8.1. Libraries

Only thing we are going to use for this project is Unity 3D, hence we are not going to use any external libraries. But still, there will be some libraries of the unity which we are going to use:

1. Character Controller
2. Light Flares
3. Particles
4. Physics Material
5. Projectors
6. Scripts
7. Standard Assets
8. Toon Shading
9. Water

8.2. Tools

Unity: The ultimate tool for video game development, architectural visualizations, and interactive media installations – publish to the web, Windows, OS X, Wii, Android etc.

9. TIME PLANNING

By the end of the first semester, we are planning to finish the menu and the first task of the game, namely buoyancy of the liquids, which we will also present in the demo presentation. In the second semester, we are planning to finish the implementation of the rest of the game. We will mainly implement other three tasks and do system testing.

9.1. Term 1 Gantt Chart

Task	Assigned to	Start	End	Oct	Nov	Dec	Jan
Play with physics	Big Ceng Theory	21.10.2011	09.06.2012				
Analyzing existing games' stories, characters, inspiration elements, experiences, music, interactions with other fields.	All members	21.10.2011	18.11.2011				
Workshop with Words To Inspire, Ministry of National Education of Turkey and Game Developers	All members	19.11.2011	19.11.2011				
Determining physics subjects	Allmembers	20.11.2011	25.11.2011				
Determining the story of game	All members	20.11.2011	25.11.2011				
Determining the music of the game	All members	23.11.2011	03.12.2011				
Updating design and implementation reports 1	All members	03.12.2011	06.12.2011				
Designing graphics of the game	All members	03.12.2011	25.12.2011				
Designing the characters of the game	All members	03.12.2011	25.12.2011				
Implementation of the story of step 1	All members	26.12.2011	15.01.2012				
Testing step 1	All members	16.01.2012	22.01.2012				

Figure-31 Gantt Chart of term 1

9.2. Term 2 Gantt Chart

Task	Assigned to	Start	End	Feb	Mar	Apr	May	June
Implementation of the story of step 2	All members	13.02.2011	05.03.2012					
Testing step 2	All members	06.03.2012	13.03.2012					
Implementation of the story of step 3	All members	06.03.2012	31.03.2012					
Testing step 3	All members	01.04.2012	08.04.2012					
Implementation of the story of step 4	All members	01.04.2012	30.04.2012					
Testing	All members	01.05.2012	07.05.2012					
Implementation to fix problems in the test step	All members	01.05.2012	13.05.2012					
Quality Ensurance	All members	13.05.2012	20.05.2012					

Figure-32 Gantt Chart of term 2

10. CONCLUSION

This DDD is prepared to give detailed information for the all design patterns of Fizbot. First, design constraints, assumptions and dependences introduced. Second data models are given with their descriptions. Then, system architecture is provided with all of its components. In the following sections, user interfaces and actions of objects are stated and detailed design of the game, the libraries and tools which will be used in the project are introduced. Finally, we provide two Gantt charts to demonstrate the detailed time planning we determined for two terms.

This document will be very helpful when implementing the requirements of the project.