OYUN DİYARI

SOFTWARE REQUIREMENTS SPECIFICATION

CEMAL AKER

OĞUL CAN ERYÜKSEL

OĞUZHAN TAŞTAN

ANIL GENÇ

Table of Contents

1 Introduction

1.1 Problem Definition

Games have always been used to improve children's skills, since they have been very attractive for every child especially for the preschoolers. However, in today's world, computers are becoming kindergarten of children because the attractiveness of video games increased sharply with the advances in video games industry in few years. Although this kindergarten is capable of providing facilities that can support early childhood development, the companies in the area does not prefer to develop such solutions. This leads to a generation with high possibility of being obese, and experiencing physical and intellectual disabilities, which is due to the lack of movement in front of the computers, and educational content in the video games.

Parents are the other group of people who are also the actors of the problem. All parents want their children to develop normally in terms of physical and intellectual abilities, and learn everything that children of that age should learn. However, due to reasons mentioned above, the children live in the world of video games instead of playing with their friends, and educating themselves.

1.2 Purpose

This document details the software requirements specification for Oyun Diyarı. It will later be used as a base for the extension of the existing software itself. This document follows the IEEE standard for software requirements specification documents.

This software requirements document is intended to supply these:

1. Developers reviewing capabilities of the tool might understand better where their efforts should be targeted to improve or add more features to it.
2. Testers might use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. This way testing becomes more methodically organized.
3. It might give an idea to users who would like to read more about what this tool is.
4. It sets the guidelines for future development.

1.3 Scope

The development process will be carried out in cooperation with Kerem ÇALIŞKAN, project mentor and Co-Founder of InfoDif, preschooler educator Aysun UĞURTAY ÜSTÜNEL and children development psychology consultant Asst. Prof. Başak ŞAHİN ACAR. Brainstorming has been done with Kerem ÇALIŞKAN. Oyun Diyarı was born from his feedback and his experience. Later this idea expanded and improved by meetings with Başak ŞAHİN ACAR and Aysun UĞURTAY ÜSTÜNEL. We consulted these people since they are experts in their area, which is children education.

Oyun Diyarı is a game that includes many levels. Player can go through in the game by completing levels one by one. Also, player can play levels that s/he passed again. In these levels, we have numerous games, which are fruit picking, connecting the dots, dancing, card matching, puzzle, pet caring, balloon popping, and counting animals. We will explain these games briefly:

- Fruit Picking: The player pick fruits from tree by jumping and with using his/her hand.

- Connecting the Dots: The player connects two dots with using his/her hand.
- Card Matching: The player tries to match two same cards by flipping them one by one.
- Puzzle: The player tries to finish puzzle by matching pieces one by one.
- Pet Caring: The player can clean, feed and give water to his/her pet.
- Balloon Popping: The player will pop randomly moving balloons on the screen with his/her hand.
- Counting Animals: The player will count animals by jumping them over the fence with his/her hand.

We will explain these games in next sections with all details.

Oyun Diyarı helps children to learn numbers, basic shapes, and basic colors, and memory development. Furthermore, we help children to emit the excess energy by entertaining him or her. For example, we offer a game that let children to dance. Moreover, Oyun Diyarı will record the activities of children, which will enable parents, teachers, experts, etc. to follow development of children. Recording feature will also bring more demand to the design since many of the games in this field don't provide it.

Oyun Diyarı will be a game that allows children to play with their body gestures. To do that the player must have a camera that gives 3D information like Microsoft Kinect camera. Up to now, three consoles integrate the necessary technology; Nintendo Wii, Sony PS3 and XBOX 360 Kinect. Also, Dell Company released a tablet that has 3D motion sensor quite recently. We want that our game shall run with all of these devices if it is possible. Since Nintendo Wii needs a device that player must hold it look like it is out of our scope. However, we thought that others are still applicable.

Oyun Diyarı will help children, age between 3 and 6, their cognitive, physical and emotional education. Our purpose is improving the children education in ages between 3 and 6. Also, while improving their educational aspect we are not trying to remove them from computer screen which as a really big problem for parents. Moreover, in every stage of this project our team will consult all of mentors frequently to keep educational aspect of project properly. Since we have a lot of games we are not mentioning from them in here. This document will be explaining our games in detailed in the proper section.

Oyun Diyarı will be based on OpenGL, Point Cloud Library (PCL). These libraries are compatible to different platforms that will make the game cross-platform. OpenGL will be used for graphical part of the game and PCL will be used to process 3D data that come from 3D motion camera. Also our game will be available in Turkish language for now. We will add English language support in future.

1.4 Definitions, acronyms, and abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS of by reference to other documents.

Term	Definition or Abbreviation
CENG	Computer Engineering
DBMS	Database Management System
ER Diagram	Entity-Relationship Diagram
IEEE	The Institute of Electrical and Electronics Engineers

Linux	An open source operating system
METU	Middle East Technical University
SRS	Software Requirement Specification
TCP/IP	A communication protocol for the internet and similar networks
UML	Unified Modeling Language
Kinect	3D sensor camera that that produced by Microsoft company.
PS3	Play Station – game console created by Sony company.
Nintendo Wii	Game console that created by Nintendo company.
Xbox	Game console created by Microsoft company.
PCL	Point Cloud Library
OpengGL	Open Graphic Library

1.5 References

- Use Case-based Requirements
  http://agile.csc.ncsu.edu/SEMaterials/UseCaseRequirements.pdf
- IEEE Std 830-1998
- For mockup tool (Balsamiq Mockup)
  http://balsamiq.com/products/mockups/
- For class diagrams and use cases
  http://www.sparxsystems.com.au/

1.6 Overview

This document conforms to the recommended practice for software requirements specifications, IEEE std 830-1998. For simplicity, the same names in the standard are used for section names. The functional requirements are organized by the features of the system.

Section 2 contains general information that is not too specific and is provided as a background for the following sections.

Section 3 contains more detail and presents the requirements with many different diagrams that illustrate the functional requirements of the Oyun Diyarı. Some of the types of diagrams that are used here include, use case diagram, and state diagram.

Section 4 contains information about the game classes and database requirements. Some type of diagrams that are used here include, class diagram and ER diagram.

Section 5 contains state transition diagram and its explanations.

Section 6 includes planning of the project and future works. Finally, section 7 concludes the SRS document.

2 Overall Description

This section provides the reader with background information about the specific software requirements. The specifications will be done in the Section 3 of the SRS; however, this section describes the general factors affecting the software and its requirements as stated in the IEEE Std 830-1998.

2.1 Product Perspective

Since the game is a stand-alone product, it will not be a part of any other system or deploy into another system. However, it makes use of a 3D sensor camera on Windows OS to get 3D data. Moreover, the system uses DBMS in order to store and manage the data. It will eventually run on Windows OS, allowing the users to access the system via a graphical user interface which is also used for responding to the user's commands. Fig. 1 shows the context diagram for the Oyun Diyarı. This diagram explains the system parts in an abstract way.

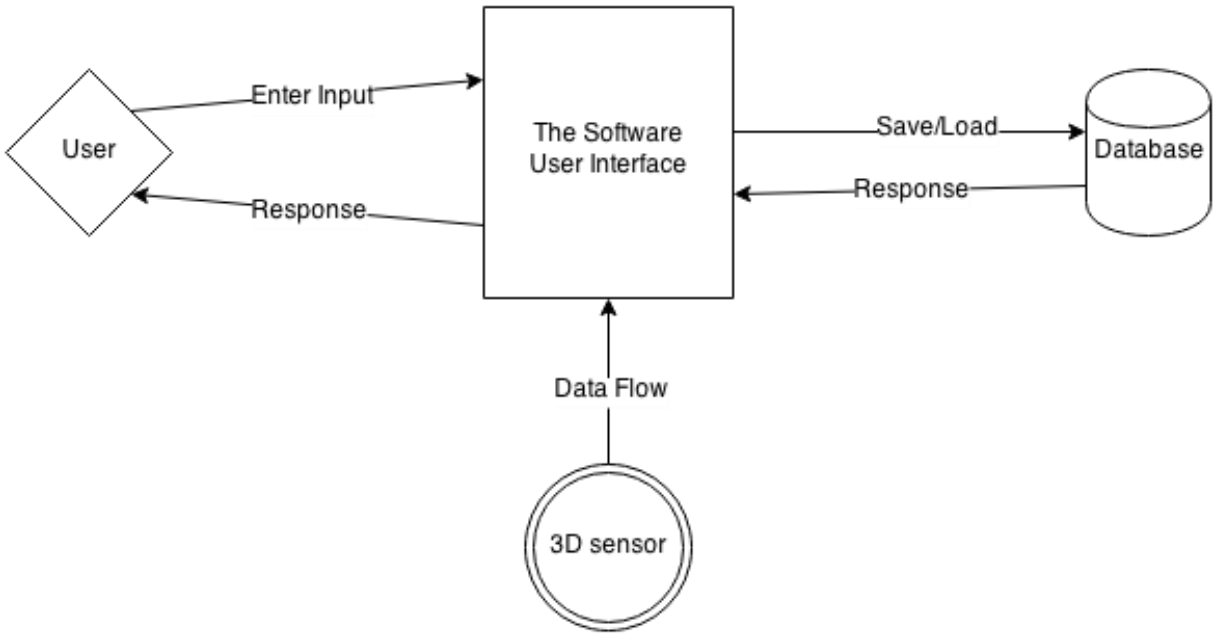

Fig. 1 – Context Diagram of the System

2.1.1 System Interfaces

One of the system interfaces in the system is the interface between the system and 3D sensor. The libraries supplied by 3D sensor camera are main component of this interface. Thanks to the interface, developers can grab the 3D point cloud data, store them in their own data structure and process them. After processing, the developers can show the results via another system interface, namely Graphic interface, which consists of a Graphics Library.

2.1.2 User Interfaces

User interfaces and their functionalities will be examined in this section. Although all screens haven't been created yet, main samples will be provided. The screenshots of this section are created by using Balsamiq Mockup. software up to now. It is likely that any other tool will not be needed for his purpose since it is important to keep screen design as simple as possible. All of the functionalities in this section user will control a cursor with his/her hand.

2.1.2.1 Welcome Screen

This screen will be showed when the game started. It is our opening page. How it will see is shown in Fig. 2. In the welcome screen player can go to the main menu, can go to the profile manager screen and can exit from the game.

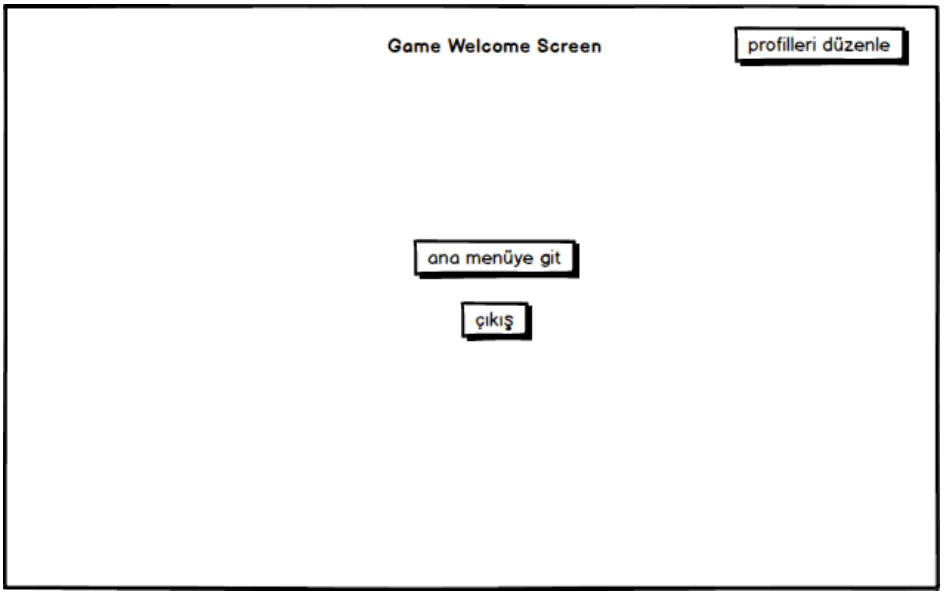

Fig. 2 – Game Welcome Screen

2.1.2.2 Main Menu Screen

In this screen user can go to game map, go back to the welcome screen, can show his/her profile page and can exit from the game. This screen can be seen in Fig. 3.

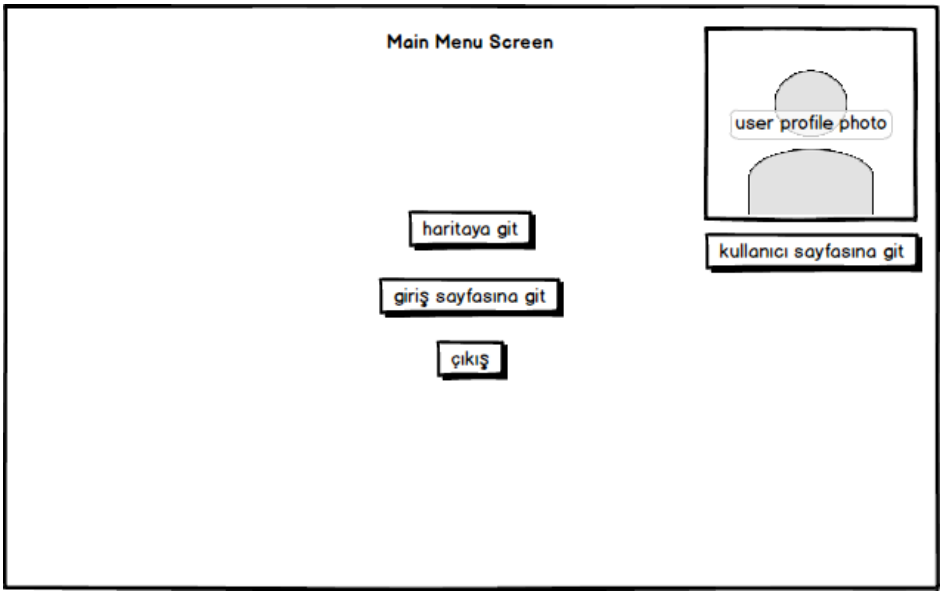

Fig. 3 – Main Menu Screen

2.1.2.3 Profile Manager Screen

In this screen user will see current user list in his/her game. The user can add a new user, delete an existing user, edit an existing user, or read the progress report of the user in this screen.

This screen can be seen from Fig. 4. When the user click to the add user, the “add new user pop-up” will be seen on the screen. The pop-up screen of this functionality is shown at Fig. 5. In the add user pop-up screen user shall fill the user’s name, last name, age, and gender. Also, s/he must take the photo of the user from the existing camera. After giving all the information, user clicks the save button and adding operation has been finished.

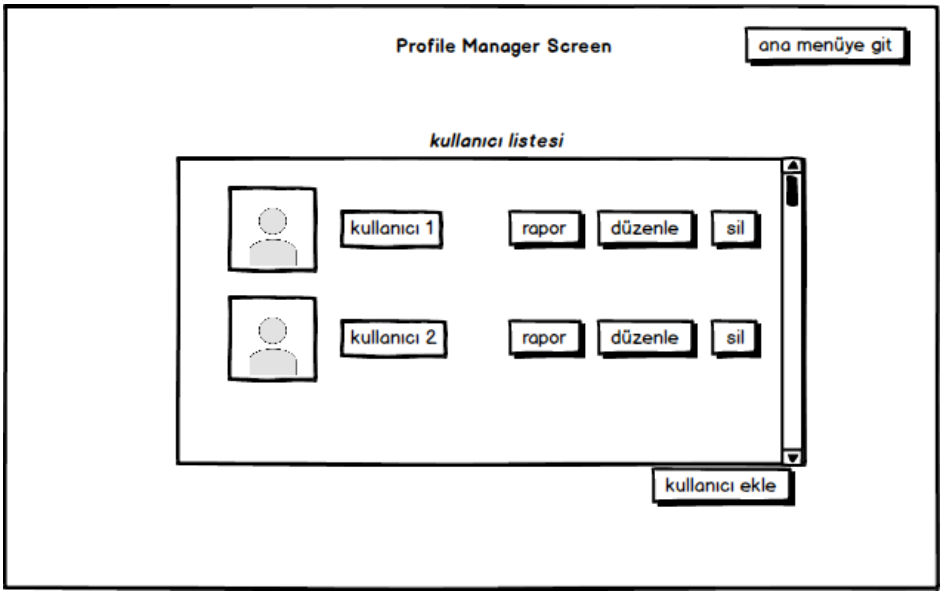

Fig. 4 – Profile Manager Screen

Moreover, user can edit an existing user on the profile manager screen. Since the screen will almost look like “add new user pop-up screen” we will refer to this pop-up screen. Editing operation is almost same as adding operation. The only difference this time pop-screen will be shown with pre-filled with chosen user’s info. Also, user can delete an existing user from user list. When the user is deleted, it shall disappear from the list.

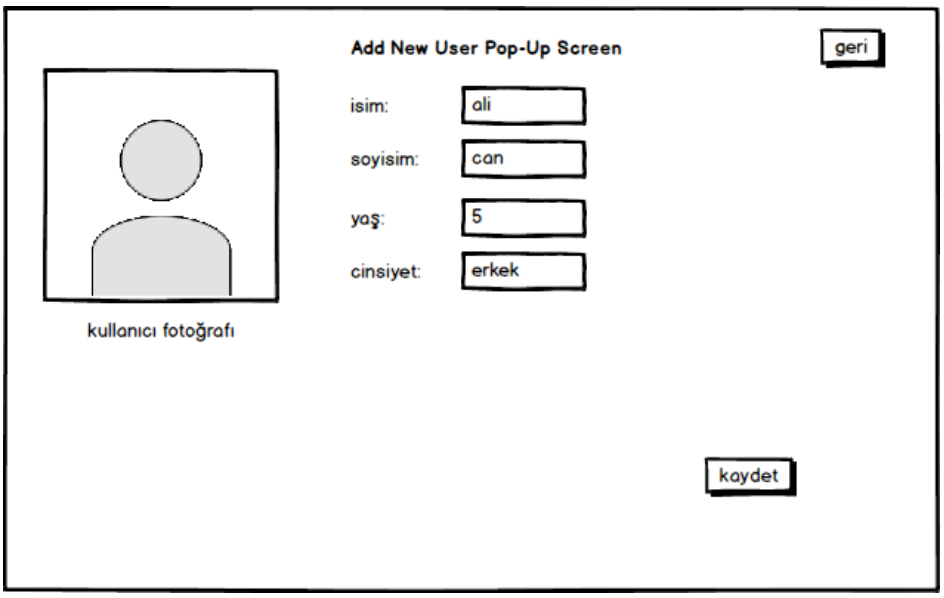

Fig. 5 – Add New User Pop-Up Screen

Read report functionality of the game will be created with the domain experts. What it includes is not certain for now

2.1.2.4 Map Screen

In this screen user will see our game levels as a map. Mock up organization of this screen is shown at Fig. 6. In this screen, the user can go back to the main menu, can exit from the game, or can select a level that s/he wants to play. The organization of these levels will be arranged by consulting the child experts. The hardness of levels and which level includes which educational purpose will be also done by consulting the experts.

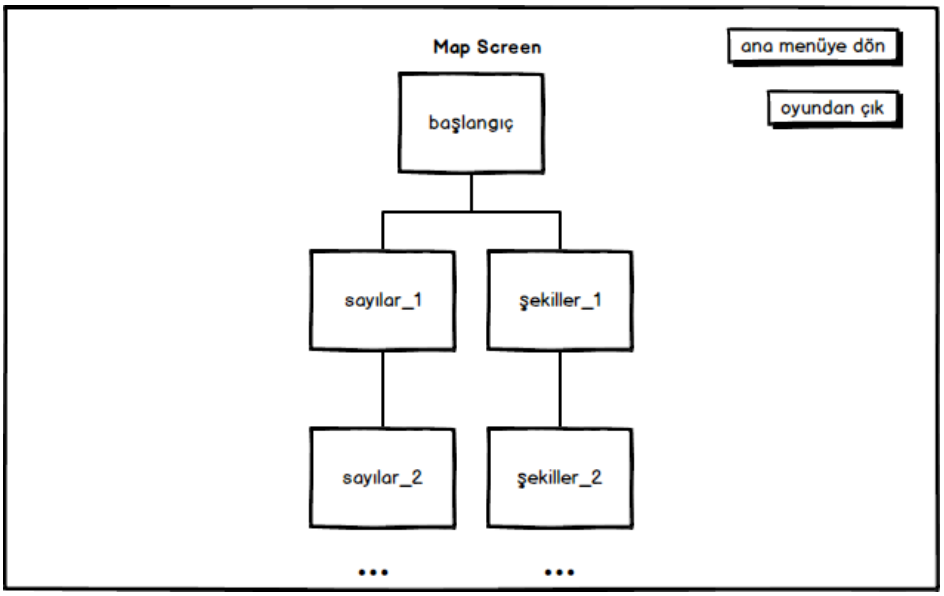

Fig. 6 – Map Screen

2.1.3 Hardware Interfaces

The game is designed for any platform which supports 3d sensor camera and is functional on any operating system. Therefore, there are two hardware interfaces for the game: 3D sensor camera and the platform on which the camera works.

2.1.4 Software Interfaces

The use of the following software products is required for the game:

- Windows OS (7,8)
- 3D Sensor/Camera APIs

2.1.5 Communication Interfaces

The product and the DBMS communicate with each other using TCP/IP protocol.

2.1.6 Hardware Constraints

The main constraint is the game's integration with all devices having 3D sensor camera. The frame rate shall be at least 25 frames per second so that the player can play the game fluently. A speaker is needed to give instructions to the player about the game.

2.1.7 Operations

Since the operations are described in Section 2.1.2, they will not be mentioned again.

2.1.8 Site Adaptation Requirements

There are no site adaptation requirements for the game.

2.2 Product Functions

Functions of the product are best described by use case diagrams. The actors of the system are player and a parent who can manage the profiles of the players of the system and read the developmental report of the players. The actor parent can be anyone who takes care of the children that are players of the system. A parent can be an actual parent, a preschooler educator or developmental psychologist. Fig. 7 shows the use case diagram for the major functions for the actor player.

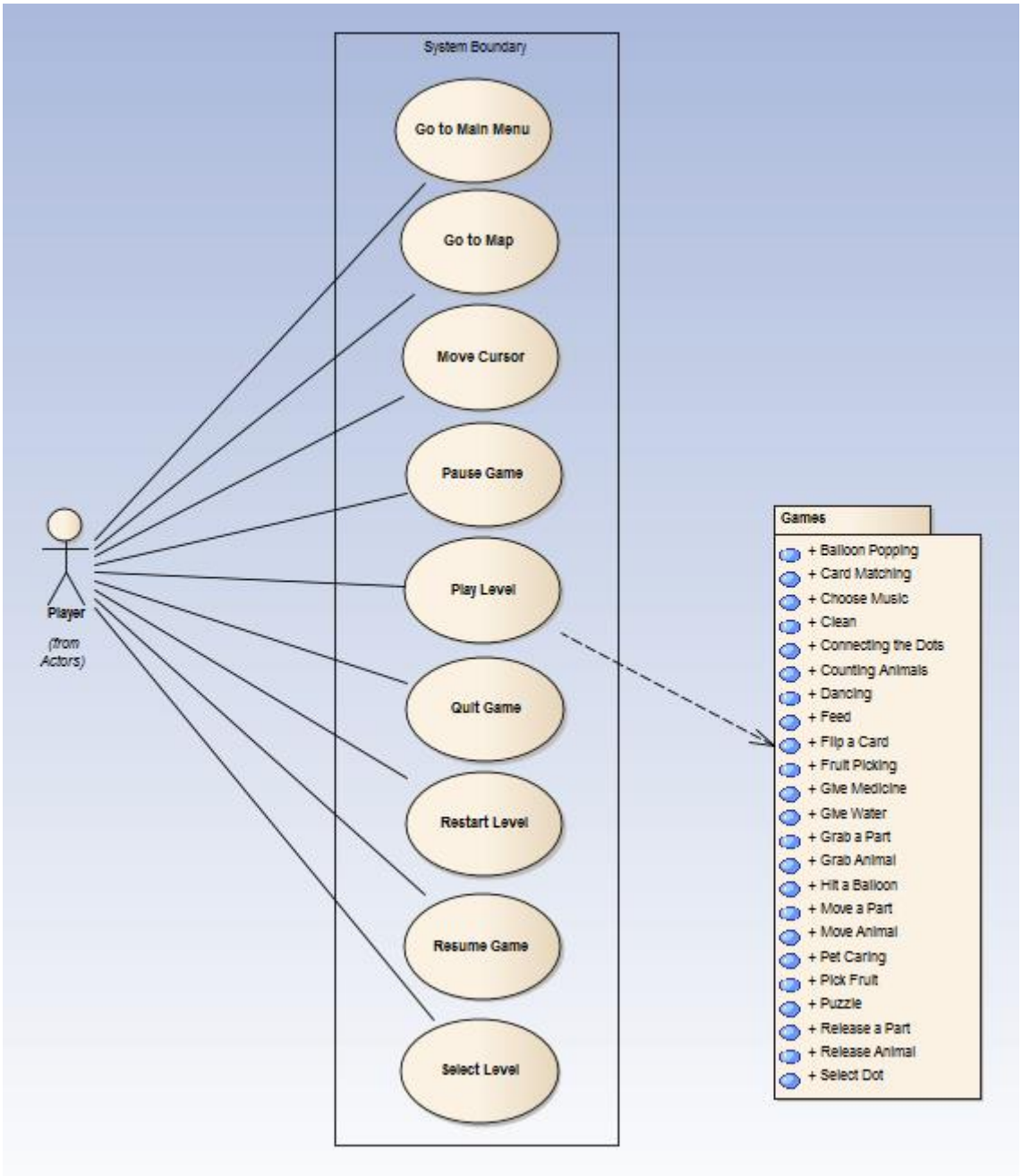

Fig. 7 – Use Case Diagram for Actor Player

Fig. 8 describes the functions that are included in “Play Level” use case for the actor player.

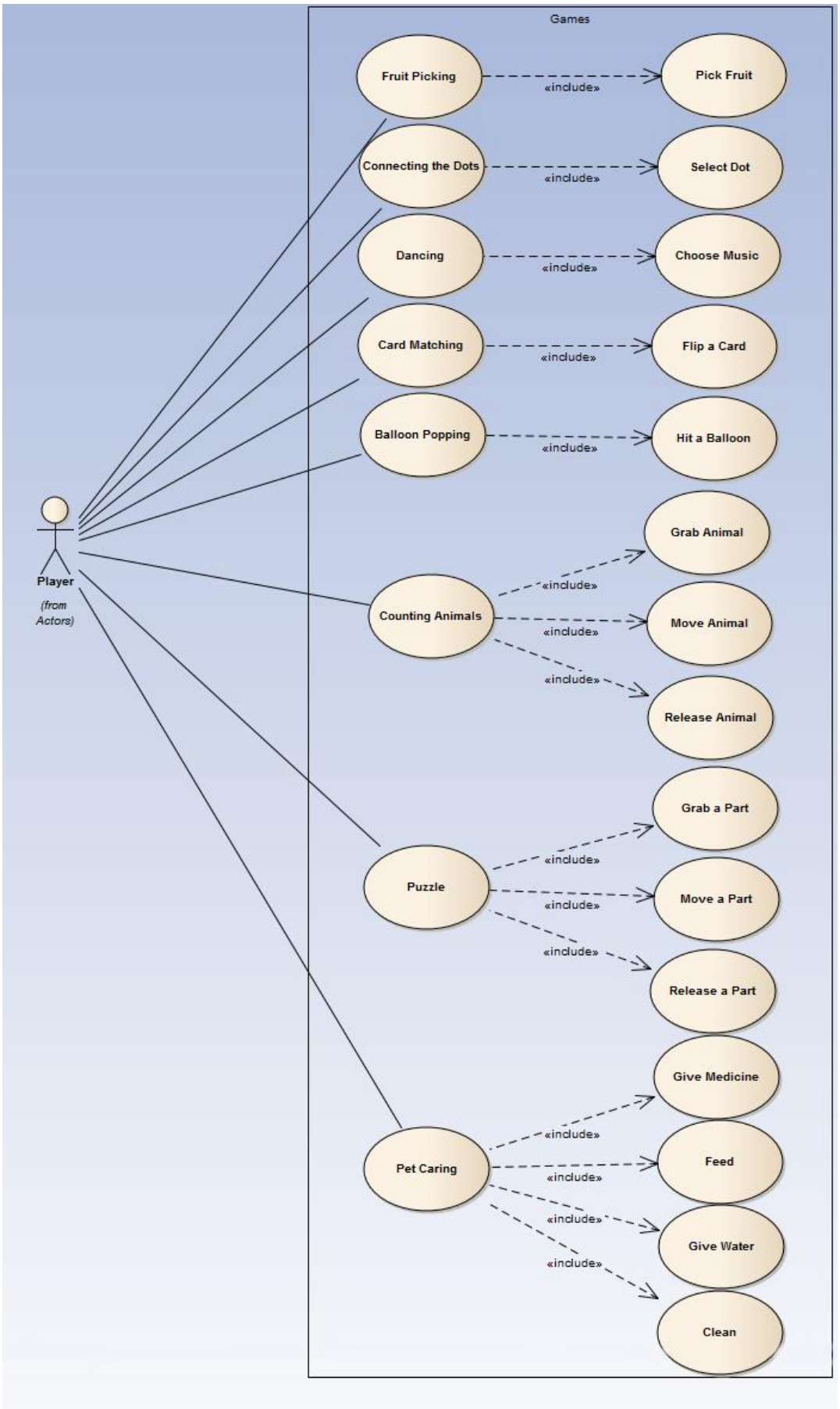

Fig. 8 – Use Cases Included in Play Level Use Case for Actor Player

Finally, Fig. 9 shows the functions that a parent can realize.

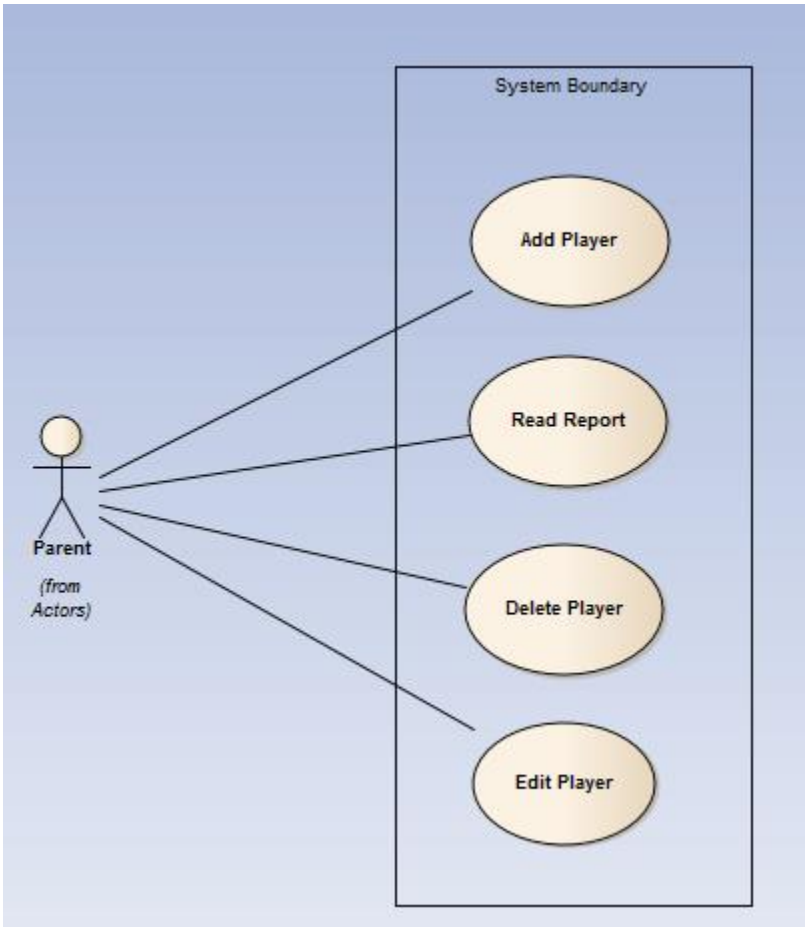

Fig. 9 – Use Case Diagram for Actor Parent

2.2.1 Add Player

Any parent can add player by using Profile Manager Screen.

2.2.2 Edit Player

Any parent can edit an existing player by using Profile Manager Screen.

2.2.3 Delete Player

Any parent can delete an existing player by using Profile Manager Screen.

2.2.4 Read Report

Any parent can read the report about the growth of their children.

2.2.5 Go to Main Menu

Any player can return back to Main Menu from Profile Manager Screen or Map screen.

2.2.6 Go to Map

Any player who is playing a level can go back to the Map.

2.2.7 Move Cursor

Any player can move the cursor at any time by moving his/her hand.

2.2.8 Pause Game

Any player who is playing a level can pause the game by using pause button.

2.2.9 Quit Game

Any player can quit the game from “Profile Manager Screen” or “Map Screen”.

2.2.10 Restart Game

Any player who is playing a level can restart the level by using restart button.

2.2.11 Resume Game

Any player who pause the game can resume the game by using resume button.

2.2.12 Select Level

Any player can select a level to play from the Map screen by clicking desired level.

2.2.13 Release a Part

Any player playing a level which is a Puzzle game can release a part.

2.2.14 Move a Part

Any player playing a level which is a Puzzle game can move a part.

2.2.15 Grab a Part

Any player playing a level which is a Puzzle game can grab a part.

2.2.16 Puzzle

Any player can play a level which is a Puzzle game by using corresponding button in Map.

2.2.17 Release an Animal

Any player playing a level which is a Counting Animals game can release an animal.

2.2.18 Move an Animal

Any player playing a level which is a Counting Animals game can move an animal.

2.2.19 Grab an Animal

Any player playing a level which is a Counting Animals game can grab an animal.

2.2.20 Counting Game

Any player can play a level which is a Counting game by using corresponding button in Map.

2.2.21 Fruit Picking

Any player can play a level in the Fruit Picking game by using corresponding button in the map.

2.2.22 Pick Fruit

Any player can pick a fruit in the Fruit Picking game by using grabbing gesture.

2.2.23 Connecting the Dots

Any player can play a level in the Connecting the Dots game by using corresponding button in the map.

2.2.24 Select Dot

Any player can select a dot in the Connecting the Dots game by using grabbing gesture.

2.2.25 Dancing

Any player can play a level in the Dancing game by using corresponding button in the map.

2.2.26 Choose Music

Any player can select music in the Dancing game by using grabbing gesture.

2.2.27 Card Matching

Any player can play a level in the Card Matching game by using corresponding button in the map.

2.2.28 Flip a card

Any player can flip a card in the Card Matching game by using grabbing gesture.

2.2.29 Balloon Popping

Any player can play a level in the Balloon Popping game by using corresponding button in the map.

2.2.30 Hit a Balloon

Any player can hit a Balloon in the Balloon Popping game by using grabbing gesture.

2.2.31 Pet Caring

Any player can care his/her pet by giving water, medicine, food or cleaning.

2.2.32 Give Medicine

Any player can give medicine to his/her pet so that its health level increases.

2.2.33 Feed

Any player can feed his/her pet so that its hunger level decreases.

2.2.34 Give Water

Any player can give water to his/her pet so that its thirst level decreases.

2.2.35 Clean

Any player can clean his/her pet so that its dirt level decreases.

2.3 Constraints

In addition to the hardware constraints which are mentioned in Section 2.1.6, the software has the following constraints:

- The name field of the player must contain at least 1 at most 16 characters.
- The surname field of the player must contain at least 1 and at most 16 characters.
- The age field of the player must be an integer.
- The gender field of the player must be selected.
- The photograph of the player must be taken.

2.4 Assumptions and Dependencies

The only assumption is that the player can understand Turkish.

3 Specific Requirements

This section describes all software requirements, for which background information is provided in the previous sections, in detail.

All requirements are uniquely identified using a signature for each one. The signature is of the form:

< associated section number > : unique identification string

If the requirement has been mentioned in the previous sections, and is being described in detail here, then the associated section number is the section number of the previous one. Otherwise it is the current section number.

3.1 Interface Requirements

In section 2.1 all system, user, hardware, software and communication interface requirements are explained. In order not to repeat information those requirements will not be mentioned again. All the use cases for the actor parent can be associated with “Profile Manager Screen” in section 2.1.2.3. All use cases will be described in detail in the section 3.2.

3.2 Functional Requirements

3.2.1 Requirements for Games

In this section functional requirements that are special to games are documented. For a game, first, the detailed description of the related use cases in tabular form, the requirements applicable to all its modes are listed. Then, functional requirements specific to a single mode are listed under subtitles for that mode.

3.2.1.1 Fruit Picking

XRef	2.2.21
Use Case ID	UC1
Use Case Name	Fruit Picking
Description	This use case describes the event in which the player plays a level which is designed as a Fruit Picking game.
Actors	Player who wants to play a level which is a Fruit Picking game.
Preconditions	Player should load his or her profile.
Trigger	Player chooses the level using corresponding button in the Map Screen.
Basic Flow	1. Player opens the Map Screen. 2. Player pushes the button corresponding to the level. 3. The level starts. 4. A tree and fruits on it appear on the screen. 5. Player plays the level by picking fruits.
Alternate Flow	-
Exception Flow	-

Post Conditions	Player should be able to play the level by picking fruits.

XRef	2.2.22
Use Case ID	UC2
Use Case Name	Pick Fruit
Description	This use case describes the event in which the player picks a fruit from the tree in Fruit Picking game.
Actors	Player who wants to pick a fruit in Fruit Picking game.
Preconditions	Player should be playing a level which is a Fruit Picking game.
Trigger	Player moves cursor.
Basic Flow	1. Player starts a level which is a Fruit Picking game. 2. Player moves the cursor on a fruit by hand motion. 3. The fruit is placed into a basket.
Alternate Flow	-
Exception Flow	-
Post Conditions	The fruit disappears on the tree.

2.2.21: req00001

The tree and fruits on it shall appear on the screen when the game started.

2.2.21: req00002

The child who represents the player shall appear on the screen when the game started.

2.2.22: req00004

The software shall provide a cursor to user for playing game.

2.2.22: req00005

The cursor shall be controlled by hand motion directly from 3D sensor.

2.2.22: req00006

The fruit shall be picked and put into the basket when the cursor comes on it.

2.2.22: req00007

The fruit which is picked shall disappear from the tree.

2.2.22: req00008

The Game system shall give a sound indicating that the fruit was picked when the fruit was picked.

3.2.1.1.1 Picking Specific Number of Fruits

3.2.1.1.1: req00009

The player shall be informed about how many fruits s/he should pick.

3.2.1.1.1: req00010

Player shall qualify to advance to next level if the correct number of fruits is picked.

3.2.1.1.1: req00011

The software shall restart the game if the player fails to pick correct number of fruits.

3.2.1.1.2 Picking fruits For Fun

3.2.1.1.2: req00012

The software shall provide the player with a chance of playing the Fruit Picking game infinitely.

3.2.1.1.2: req0013

The software shall place new fruits if all of them are picked.

3.2.1.2 Connecting the Dots

XRef	2.2.23
Use Case ID	UC3
Use Case Name	Connecting the Dots
Description	This use case describes the event in which the player plays a level which is designed as a Connecting the Dots game.
Actors	Player who wants to play a level which is a Connecting the Dots game.
Preconditions	Player should load his or her profile.
Trigger	Player chooses the level using corresponding button in the Map Screen.
Basic Flow	1. Player opens the Map Screen. 2. Player pushes the button corresponding to the level. 3. The level starts. 4. Dots on the circumference of a shape appear on the screen. 5. Player plays the level by selecting the next dot each time until the shape is completed.
Alternate Flow	-
Exception Flow	-
Post Conditions	Player should be able to play the level by selecting dots.

XRef	2.2.24
Use Case ID	UC4
Use Case Name	Select Dot
Description	This use case describes the event in which the player selects the next dot to complete the shape in Connecting the Dots game.
Actors	Player who wants to select the next dot in Connecting the Dots game.
Preconditions	Player should be playing a level which is a Connecting the Dots game.
Trigger	Player moves cursor.
Basic Flow	1. Player starts a level which is a Connecting the Dots game. 2. Player moves the cursor on a dot. 3. Player makes a grabbing gesture to select the dot. 4. The dot is joined by a straight line to the previous one.
Alternate Flow	-
Exception Flow	4. The dot is not joined to the previous one if it is not the correct dot.

Post Conditions	A line between the previous and the newly chosen dots should appear.

2.2.23: req00014

The predefined number of disconnected dots (or any other symbols) shall appear on the screen when the game is started.

2.2.24: req00015

The software shall provide a cursor to user for selecting dots.

2.2.24: req00016

If the player selects the correct dot, the selected dot shall be enlighten and connected to the previous dot.

2.2.24: req00017

If the player selects the wrong dot, the software shall give the sound indicating wrong dot.

2.2.23: req00018

If all dots are connected, the resulting shape shall be blinking three times and the player shall qualify to advance to the next level.

3.2.1.2.1 Connecting the Dots with Blinking Next Dot

3.2.1.2.1: req00019

The next dot that the player is supposed to select shall be blinking during the game.

3.2.1.2.2 Connecting the Dots Labeled with Numbers in Ascending Order

3.2.1.2.2: req00020

The consecutive numbers starting from one shall appear near the dots, one number per dot.

3.2.1.2.2: req00021

The next dot that the player is supposed to select shall be the dot with the number which is one more than the previously connected dot.

3.2.1.2.3 Connecting the Dots Labeled with Numbers in Descending Order

3.2.1.2.3: req00022

The consecutive numbers starting from the number of dots shall appear near the dots in descending order, one number per dot.

3.2.1.2.3: req00023

The next dot that the player is supposed to select shall be the dot with the number which is less than the previously connected dot.

3.2.1.2.4 Connecting the Dots Labeled with a Pattern of Geometric Shapes

3.2.1.2.4: req00024

The predefined number of disconnected dots (or any other symbols) shall appear on the screen in such a way that they construct a basic geometric shape when the game is started.

3.2.1.3 Dancing

XRef	2.2.25
Use Case ID	UC5
Use Case Name	Dancing
Description	This use case describes the event in which the player plays a level which is designed as a Dancing game.
Actors	Player who wants to play a level which is a Dancing game.
Preconditions	Player should load his or her profile.
Trigger	Player chooses the level using corresponding button in the Map Screen.
Basic Flow	1. Player opens the Map Screen. 2. Player pushes the button corresponding to the level. 3. The level starts. 4. Player chooses the music s/he wants. 5. Requested move appears on the screen. 6. Player makes the move. 7. Go to 5
Alternate Flow	5. Player dances freely.
Exception Flow	-
Post Conditions	-

XRef	2.2.26
Use Case ID	UC6
Use Case Name	Choose Music
Description	This use case describes the event in which the player chooses the music that will be played in Dancing game.
Actors	Player who wants to choose the music in Dancing game.
Preconditions	Player should be playing a level which is a Dancing game.
Trigger	The level starts.
Basic Flow	1. Player starts a level which is a Dancing game. 2. A popup menu appears. 3. Player chooses one of the provided music files by pushing the corresponding button. 4. The popup menu disappears.
Alternate Flow	-
Exception Flow	-
Post Conditions	The music that is chosen should be played.

2.2.25: req00022

The software shall provide a character that mimics moves of the player on the game screen.

2.2.26: req00023

The software shall provide a list of music to the player.

2.2.26: req00024

The software shall play the music chosen, while the player is dancing.

3.2.1.3.1 Dancing with the Requested Moves

3.2.1.3.1: req00025

The software shall show the appropriate dance figures that s/he is going to perform to the player.

3.2.1.3.1: req00026

The software shall evaluate the figure of the player to inform his/her about the correctness of the figure after.

3.2.1.3.1: req00027

The software shall give appropriate sound effect about the correctness of move.

3.2.1.3.1: req00028

At the end of the level if the player made at least 75% of the all moves, s/he succeed in the level and qualifies to advance to the next level.

3.2.1.3.1: req00029

At the end of the level if the player cannot make at least 75% of the all moves, the software shall request to the player to play the level again.

3.2.1.3.2 Dancing Freely

3.2.1.3.2: req00030

The software shall provide a music list to the player to choose music that s/he will dance.

3.2.1.3.2: req00031

When the song is finished game also shall finish.

3.2.1.3.2: req00032

At the end of the song player shall informed about the score that s/he gained.

3.2.1.3.3 Morning Gymnastics

3.2.1.3.3: req00033

The software shall show the appropriate gymnastic figures that s/he is going to perform to the player.

3.2.1.3.3: req00034

The software shall evaluate the figure of the player to inform s/he about the correctness of the figure.

3.2.1.3.3: req00035

The software shall give appropriate sound effect about the correctness of move.

3.2.1.4 Card Matching

XRef	2.2.27
Use Case ID	UC7
Use Case Name	Card Matching
Description	This use case describes the event in which the player plays a level which is designed as a Card Matching game.
Actors	Player who wants to play a level which is a Card Matching game.
Preconditions	Player should load his or her profile.
Trigger	Player chooses the level using corresponding button in the Map Screen.
Basic Flow	1. Player opens the Map Screen. 2. Player pushes the button corresponding to the level. 3. The level starts. 4. Cards each of which has a pair appear on the screen faced up. 5. Cards are flipped face down. 6. Player flips a card face up. 7. Player flips the pair of that card face up.
Alternate Flow	7. Player flips a card other than the pair face up. 8. Both cards are flipped face down.
Exception Flow	-
Post Conditions	Player should be able to play the level by selecting dots.

XRef	2.2.28
Use Case ID	UC8
Use Case Name	Flip a Card
Description	This use case describes the event in which the player flips a card face up in Card Matching game.
Actors	Player who wants to flip a card in Card Matching game.
Preconditions	Player should be playing a level which is a Card Matching game.
Trigger	The level starts.
Basic Flow	1. Player starts a level which is a Card Matching game. 2. Player moves the cursor on a card. 3. Player makes a grabbing gesture. 4. The card flips face up.
Alternate Flow	-
Exception Flow	-
Post Conditions	The card should appear faced up.

2.2.27: req00036

The software shall create all cards with random positions in a grid.

2.2.27: req00037

The software shall check that each card has available pair.

2.2.27: req00038

The software shall show information of all cards to the player for once at the beginning.

2.2.28: req00039

The software shall open and show the information of flipped card.

2.2.27: req00040

The software shall check both open cards are the same or not if there are two open cards.

2.2.27: req00041

The software shall flip both cards face down if there are two open cards and they do not match.

2.2.27: req00042

The software shall keep matched cards as opened.

2.2.27:req00043

The software shall finish the game if all cards are matched.

2.2.27: req00044

The software shall show completion time at the end of the game.

2.2.27: req00045

The software shall give appropriate sound according that opened cards are matched or not matched.

2.2.28: req00046

The software shall calculate player's score by using complete time of game and number of wrong trials.

3.2.1.4.1 Card Matching Using Pictures

There is no specific requirement for this mode other than the general requirements for Card Matching game.

3.2.1.4.2 Card Matching Using Animal Pictures

3.2.1.4.2: req00047

The software shall create all cards with animals' pictures.

3.2.1.4.2: req00048

The software shall give sound of matched animal if two same animal cards matched.

3.2.1.4.3 Card Matching Using Animal Sounds

3.2.1.4.3: req00050

The software shall create all cards with animals' sounds and without pictures on them.

3.2.1.4.3: req00051

The software shall play the sound associated with a card when a card is flipped.

3.2.1.4.3: req00052

The Software shall provide name and sounds of all matched animals at the end of the game.

3.2.1.5 Balloon Popping

XRef	2.2.29
Use Case ID	UC9
Use Case Name	Balloon Popping
Description	This use case describes the event in which the player plays a level which is designed as a Balloon Popping game.
Actors	Player who wants to play a level which is a Balloon Popping game.
Preconditions	Player should load his or her profile.
Trigger	Player chooses the level using corresponding button in the Map Screen.
Basic Flow	1. Player opens the Map Screen. 2. Player pushes the button corresponding to the level. 3. The level starts. 4. Balloons moving around appear on the screen. 5. Player hits a balloon. 6. The balloon pops and disappears.
Alternate Flow	-
Exception Flow	-
Post Conditions	The balloon that is hit should disappear.

XRef	2.2.30
Use Case ID	UC10
Use Case Name	Hit a Balloon
Description	This use case describes the event in which the player hits a balloon in Balloon Popping game.
Actors	Player who wants to pop a balloon in Balloon Popping game.
Preconditions	Player should be playing a level which is a Balloon Popping game.
Trigger	The level starts.
Basic Flow	1. Player starts a level which is a Balloon Popping game. 2. Player moves the cursor on a balloon. 3. Player makes a gesture to hit. 4. The balloon pops and disappears.
Alternate Flow	-
Exception Flow	2. Player moves the cursor on an empty area. 3. Player makes a gesture to hit. 4. The gesture is ignored.
Post Conditions	

2.2.30: req00053

The Software shall provide a cursor to player for playing game.

2.2.30: req00054

The cursor shall be controlled by hand motion directly from 3D sensor.

2.2.30: req00055

The software shall detect the position where player hits.

2.2.30: req00056

The balloon shall be destroyed when it is hit by player.

2.2.29: req00057

The software shall give a pop sound when the balloon is destroyed.

2.2.29: req00058

The software shall provide total number of popped balloon at the end of the game.

3.2.1.5.1 Balloon Popping and Counting By Color

3.2.1.5.1: req00059

The Software shall detect wrong hits or move of player.

3.2.1.5.1: req00060

The Software shall decrease total points of player when any wrong balloon is hit by player.

3.2.1.5.1: req00061

The Software shall adjust speed and colors of Balloons according the player level.

3.2.1.5.1: req00062

The Software shall finish the game, if enough Balloons are popped.

3.2.1.5.1: req00063

The Software shall do nothing if player hits wrong Balloons.

3.2.1.5.1: req00064

The Software shall do nothing if player hits a space without Balloons.

3.2.1.5.2 Balloon Popping Freely

3.2.1.5.2: req00065

The Software shall change speed of Balloons randomly during the game.

3.2.1.5.2: req00066

The Software shall change colors of Balloons randomly during the game.

XRef	2.2.20
Use Case ID	UC11
Use Case Name	Counting Animals
Description	This use case describes the event in which the player plays a level which is designed as a Counting Animals game.
Actors	Player who wants to play a level which is a Counting Animals game.
Preconditions	Player should load his or her profile.
Trigger	Player chooses the level using corresponding button in the Map Screen.
Basic Flow	1. Player opens the Map Screen. 2. Player pushes the button corresponding to the level. 3. The level starts. 4. Animals behind a fence appear on the screen. 5. Player grabs an animal. 6. Player moves the animal. 7. Player Releases the animal. 8. Go to 5.
Alternate Flow	-
Exception Flow	-
Post Conditions	

XRef	2.2.19
Use Case ID	UC12
Use Case Name	Grab Animal
Description	This use case describes the event in which the player grabs an animal in Counting Animals game.
Actors	Player who wants to grab an animal in Counting Animals game.
Preconditions	Player should be playing a level which is a Counting Animals game.
Trigger	The level starts.
Basic Flow	1. Player starts a level which is a Counting Animals game. 2. Player moves the cursor on an animal. 3. Player makes a grabbing gesture. 4. The animal is attached to the cursor.
Alternate Flow	-
Exception Flow	2. Player moves the cursor on an empty area. 3. Player makes a grabbing gesture. 4. The gesture is ignored.
Post Conditions	The animal should be attached to the cursor to move together.

XRef	2.2.18
Use Case ID	UC13
Use Case Name	Move Animal
Description	This use case describes the event in which the player moves an animal in Counting Animals game.
Actors	Player who wants to move an animal in Counting Animals game.
Preconditions	Player should be playing a level which is a Counting Animals game.
Trigger	Player grabs an animal.
Basic Flow	1. Player starts a level which is a Counting Animals game. 2. Player grabs an animal. 3. Player moves the cursor using hand motion. 4. The animal moves together with the cursor.
Alternate Flow	-
Exception Flow	-
Post Conditions	The animal should be positioned with the same position as the cursor.

XRef	2.2.17
Use Case ID	UC14
Use Case Name	Release Animal
Description	This use case describes the event in which the player releases an animal in Counting Animals game.
Actors	Player who wants to release an animal in Counting Animals game.
Preconditions	Player should be playing a level which is a Counting Animals game.
Trigger	Player moves an animal.
Basic Flow	1. Player starts a level which is a Counting Animals game. 2. Player grabs an animal. 3. Player moves an animal. 4. Player opens his/her hand to release the animal. 5. The animal is detached from the cursor.
Alternate Flow	-
Exception Flow	-
Post Conditions	The animal should stay still where it is released.

2.2.20: req00067

The software shall show animals and the fence on the game scene.

2.2.20: req00068

The software shall provide a cursor that can be controlled by hand of the player to user for playing game.

2.2.20: req00069

The software shall detect the position where the player clicks by her/his hand.

2.2.19: req00070

When the player makes a grabbing gesture, the animal that the cursor is on shall move with the cursor until the player opens his or her hand.

2.2.19: req00071

When the player makes a grabbing gesture, no changes occur if the cursor is not on any of the animals.

2.2.17: req00072

If the animal is on the other side of the fence when released, the software shall give the appropriate animal sound and give points to the player.

2.2.17: req00073

If the animal is released on a wrong place, which is not other side of the fence, the software shall give wrong move sound and put back the animal to the old place.

2.2.20: req00074

When the task is done, the software shall inform to the player about his/her score.

3.2.1.5.3 Having Specific Number of Sheep Jump Over the Fence

3.2.1.5.3: req00075

The software shall tell the player his/her task. This task includes how many sheep is going to jump over the fence.

3.2.1.5.4 Having Animals of Specific Kinds Jump Over The Fence

3.2.1.5.4: req00076

The software shall tell the player his/her task. This task includes how many animals with which order is going to jump over the fence. For example, jump 3 cow then 2 sheep over the fence.

3.2.1.6 Puzzle

XRef	2.2.16
Use Case ID	UC15
Use Case Name	Puzzle
Description	This use case describes the event in which the player plays a level which is designed as a Puzzle game.
Actors	Player who wants to play a level which is a Puzzle game.
Preconditions	Player should load his or her profile.
Trigger	Player chooses the level using corresponding button in the Map Screen.
Basic Flow	1. Player opens the Map Screen. 2. Player pushes the button corresponding to the level. 3. The level starts. 4. Randomly positioned puzzle parts appear on the screen. 5. Player grabs a part. 6. Player moves the part. 7. Player Releases the part.

	8. Go to 5.
Alternate Flow	-
Exception Flow	-
Post Conditions	

XRef	2.2.15
Use Case ID	UC16
Use Case Name	Grab a Part
Description	This use case describes the event in which the player grabs a part in Puzzle game.
Actors	Player who wants to grab a part in Puzzle game.
Preconditions	Player should be playing a level which is a Puzzle game.
Trigger	The level starts.
Basic Flow	1. Player starts a level which is a Puzzle game. 2. Player moves the cursor on a part. 3. Player makes a grabbing gesture. 4. The part is attached to the cursor.
Alternate Flow	-
Exception Flow	2. Player moves the cursor on an empty area. 3. Player makes a grabbing gesture. 4. The gesture is ignored.
Post Conditions	The part should be attached to the cursor to move together.

XRef	2.2.14
Use Case ID	UC17
Use Case Name	Move a Part
Description	This use case describes the event in which the player moves a part in Puzzle game.
Actors	Player who wants to move a part in Puzzle game.
Preconditions	Player should be playing a level which is a Puzzle game.
Trigger	Player grabs a part.
Basic Flow	1. Player starts a level which is a Puzzle game. 2. Player grabs a part. 3. Player moves the cursor using hand motion. 4. The part moves together with the cursor.
Alternate Flow	-
Exception Flow	-
Post Conditions	The part should be positioned with the same position as the cursor.

XRef	2.2.13
Use Case ID	UC18
Use Case Name	Release a Part
Description	This use case describes the event in which the player releases a part in Puzzle game.
Actors	Player who wants to release a part in Puzzle game.
Preconditions	Player should be playing a level which is a Puzzle game.
Trigger	Player moves a part.
Basic Flow	1. Player starts a level which is a Puzzle game.

	2. Player grabs a part. 3. Player moves a part. 4. Player opens his/her hand to release the part. 5. The part is detached from the cursor.
Alternate Flow	-
Exception Flow	-
Post Conditions	The part should stay still where it is released.

2.2.16: req00077

The software shall provide a cursor to user for playing the game.

2.2.16: req00078

The cursor shall be controlled by hand motion directly from 3D sensor.

2.2.15: req00079

The software shall detect the position where the player clicks by her/his hand.

2.2.14: req00080

When the player makes a grabbing gesture, the puzzle part which the cursor is on shall move with the cursor until the player opens his or her hand.

2.2.15: req00081

When the player makes a grabbing gesture, no changes shall occur if the cursor is not on any of the puzzle parts.

2.2.13: req00082

When the player opens his/her hand, the grabbed part should be detached from the cursor and stay still.

3.2.1.6.1 Classical Puzzle Game

3.2.1.6.1: req00083

If a part is in the correct position when released, the software shall join it with the neighboring parts.

3.2.1.6.1: req00084

If a part is not in the correct position when released, the system shall inform the player by sound and not join the part with any other parts.

3.2.1.6.1: req00085

At startup, the software shall display a rectangle where the parts will be positioned, and all the parts with random positions.

3.2.1.6.1: req00086

When the puzzle is completed, the player qualifies to advance to the next level and the software shall return to the level map.

3.2.1.6.2 Puzzle with Shapes

3.2.1.6.2: req00087

The software shall display the position that the part should be placed and one part with random position.

3.2.1.6.2: req00088

If the part is correctly positioned, the player qualifies to advance to the next level and the software shall return to the level map.

3.2.1.6.2: req00089

If the part is not correctly positioned when released, the software shall inform the player by sound.

3.2.1.6.3 Pattern Matching

3.2.1.6.3: req00090

At startup, the software shall display a picture of a pattern with a missing part and some parts with random positions one of which is the missing part of the pattern.

3.2.1.6.3: req00091

If the missing part is in the correct position, the software shall join it with the pattern, return to the level map, and the player qualifies to advance to the next level.

3.2.1.6.3: req00092

If a part other than the missing one is placed where the missing part should be, the software shall not join it with the pattern.

3.2.1.6.4 Classic Puzzle Game (Endless)

3.2.1.6.4: req00093

If a part is in the correct position when released, the software shall join it with the neighboring parts.

3.2.1.6.4: req00094

If a part is not in the correct position when released, the software shall inform the player by sound and not join the part with any other parts.

3.2.1.6.4: req00095

At startup, the software shall display a rectangle where the parts will be positioned, and all the parts with random positions.

3.2.1.6.4: req00096

When a puzzle is completed, the software shall create another one.

3.2.1.7 Pet Caring

XRef	2.2.31
Use Case ID	UC19
Use Case Name	Pet Caring
Description	This use case describes the event in which the player plays a level which is designed as a Per Caring game.
Actors	Player who wants to play a level which is a Pet Caring game.
Preconditions	Player should load his or her profile.
Trigger	Player chooses the level using corresponding button in the Map Screen.
Basic Flow	1. Player opens the Map Screen. 2. Player pushes the button corresponding to the level. 3. The level starts. 4. The pet appear on the screen. 5. Player cares the pet by giving water, medicine, food or cleaning.
Alternate Flow	-
Exception Flow	-
Post Conditions	Player should be able to play the level by caring the pet.

XRef	2.2.32
Use Case ID	UC20
Use Case Name	Give Medicine
Description	This use case describes the event in which the player gives medicine to the pet in Pet Caring game.
Actors	Player who wants to give medicine to the pet in Pet Caring game.
Preconditions	Player should be playing a level which is a Pet Caring game.
Trigger	Player moves cursor.
Basic Flow	1. Player starts a level which is a Pet Caring game. 2. Player moves the cursor on the medicine picture by hand motion. 3. Player clicks the medicine picture by grabbing gesture.

	4. The pet gets medicine.
Alternate Flow	-
Exception Flow	-
Post Conditions	The pet gets medicine and increase its health.

XRef	2.2.33
Use Case ID	UC21
Use Case Name	Feed
Description	This use case describes the event in which the player feed the pet in Pet Caring game.
Actors	Player who wants to feed the pet in Pet Caring game.
Preconditions	Player should be playing a level which is a Pet Caring game.
Trigger	Player moves cursor.
Basic Flow	1. Player starts a level which is a Pet Caring game. 2. Player moves the cursor on the food picture by hand motion. 3. Player clicks the food picture by grabbing gesture. 4. The pet eats food.
Alternate Flow	-
Exception Flow	-
Post Conditions	The pet eats food and decrease its hunger.

XRef	2.2.34
Use Case ID	UC22
Use Case Name	Give Water
Description	This use case describes the event in which the player give water to the pet in Pet Caring game.
Actors	Player who wants to give water to the pet in Pet Caring game.
Preconditions	Player should be playing a level which is a Pet Caring game.

Trigger	Player moves cursor.
Basic Flow	1. Player starts a level which is a Pet Caring game. 2. Player moves the cursor on the water picture by hand motion. 3. Player clicks the water picture by grabbing gesture. 4. The pet drinks water.
Alternate Flow	-
Exception Flow	-
Post Conditions	The pet drinks water and decrease its thirst.

XRef	2.2.35
Use Case ID	UC23
Use Case Name	Clean
Description	This use case describes the event in which the player clean the pet in Pet Caring game.
Actors	Player who wants to clean the pet in Pet Caring game.
Preconditions	Player should be playing a level which is a Pet Caring game.
Trigger	Player moves cursor.
Basic Flow	1. Player starts a level which is a Pet Caring game. 2. Player moves the cursor on the brush picture by hand motion. 3. Player clicks the brush picture by grabbing gesture. 4. The pet is cleaned.
Alternate Flow	-
Exception Flow	-
Post Conditions	The pet is clean and decrease its dirt.

2.2.31: req00097

At first play, the player shall be able to choose the kind of animal that s/he will take care of from a list.

2.2.31: req00098

The software shall provide a menu with entries to feed, give water, clean and give medicine to the pet.

2.2.31: req00099

The software should display health, hunger, thirst, and dirt levels of the pet.

2.2.31: req00100

If the dirt level of the pet is greater than 90%, health level of the pet shall decrease by 4% at each hour down to 0%.

2.2.31: req00101

The hunger level of the pet shall increase by 4% at each hour up to 100%.

2.2.31: req00102

The thirst level of the pet shall increase by 4% at each hour up to 100%.

2.2.31: req00103

The dirt level of the pet shall increase by 4% at each hour up to 100%.

2.2.33: req00104

When the player choses the entry to feed from the menu, the hunger level of the pet shall become 0% and the timestamp for this operation shall be saved.

2.2.34: req00105

When the player choses the entry to give water from the menu, the thirst level of the pet shall become 0% and the timestamp for this operation shall be saved.

2.2.35: req00106

When the player choses the entry to clean from the menu, the dirt level of the pet shall become 0% and the timestamp for this operation shall be saved.

2.2.32: req00107

When the player choses the entry to give medicine from the menu, the health level of the pet shall become 100% and the timestamp for this operation shall be saved.

2.2.31: req00108

At the shutdown of the Pet Caring game, the software shall save the current values of the health, hunger, dirt, and thirst levels.

2.2.31: req00109

At the startup of the Pet Caring, the software shall load the timestamps for the last feed, give water, give medicine and clean operations and saved values for the hunger, thirst, dirt, and health levels. Then, it shall compute the new values according to time passed.

3.2.2 General Functional Requirements

3.2.2: req00110

The software shall check availability status 3D sensor and speakers.

3.2.2: req00111

The software shall provide a tutorial to teach user how to play.

3.2.2: req00112

The tutorial shall provide an option to skip tutorial.

3.2.2: req00113

The tutorial shall provide an option to restart tutorial.

3.2.2: req00114

The software shall ignore any motion except for player motion.

3.2.2: req00115

The software shall provide a task to complete for player (Except free mode).

3.2.2: req00116

The task shall be explained to player by a voice.

3.2.2: req00117

The task shall be listened again by a restart option.

3.2.2: req00118

The software shall provide to user options which are pause, resume, restart, and exit the game.

3.2.2: req00119

The software shall provide to user an option for increasing or decreasing volume of game music.

3.2.2: req00120

The Game system shall provide a player report to the parent.

3.2.2.1 Requirements Related to Actor Player

XRef	2.2.5
Use Case ID	UC24
Use Case Name	Go to Main Menu
Description	This use case describes the event in which the player who wants to return back to the main menu.
Actors	The player who wants to go back to main menu.
Preconditions	The player should be in “Profile Manager Screen” or “Map Screen”.
Trigger	Player clicks the “Go to Main Menu”.
Basic Flow	1. The player moves the cursor by hand movement to the “Go To Main Menu”. 2. Player clicks the “Go to Main Menu” by grabbing gesture. 3. The player is directed to the Main Menu.
Alternate Flow	-
Exception Flow	-
Post Conditions	The player should be in Main Menu.

XRef	2.2.6
Use Case ID	UC25
Use Case Name	Go to Map
Description	This use case describes the event in which the player who wants to return back to “Map Screen”.
Actors	The player who wants to go back to “Map Screen”.
Preconditions	The player should be playing a level.
Trigger	Player clicks the “Go to Map”.
Basic Flow	1. The player moves the cursor by hand movement to the “Go to Map” 2. Player clicks the “Go to Map” by grabbing gesture. 3. A dialog box for confirming the operation pops up.

	4. The player clicks on the “OK” button. 5. The player is directed to the “Map Screen”.
Alternate Flow	
Exception Flow	3. The player clicks on the “No” button. 4. Game is going on.
Post Conditions	The player should be in “Map Screen”.

XRef	2.2.7
Use Case ID	UC26
Use Case Name	Move Cursor
Description	This use case describes the event in which the player who wants to move cursor.
Actors	The player
Preconditions	-
Trigger	The player shows his/her palm to the camera.
Basic Flow	1. The player moves his/her hand in any direction. 2. The cursor moves in the direction which is the same as the player's.
Alternate Flow	-
Exception Flow	-
Post Conditions	The cursor must be at the correct position according to its movement.

XRef	2.2.8
Use Case ID	UC27
Use Case Name	Pause Game
Description	This use case describes the event in which the player who wants to pause the game
Actors	The player

Preconditions	The player should be playing a level.
Trigger	The player clicks the pause button.
Basic Flow	1. The player moves the cursor by hand movement to the pause button. 2. The player clicks the pause button by grabbing gesture. 3. The game pauses and pause menu is displayed.
Alternate Flow	-
Exception Flow	-
Post Conditions	The game should be paused.

XRef	2.2.9
Use Case ID	UC28
Use Case Name	Quit Game
Description	This use case describes the event in which the player who wants to quit the game.
Actors	The player
Preconditions	The player should be in “Profile Manager Screen” or “Map Screen” or “Main Menu”.
Trigger	The player clicks the quit button.
Basic Flow	1. The player moves the cursor by hand movement to the quit button. 2. The player clicks the quit button. 3. A dialog box for confirming the operation pops up. 4. The player clicks on the “OK” button. 5. The game is shut down.
Alternate Flow	-
Exception Flow	3. The player clicks on the “No” button. 4. The player stays in the current screen.
Post Conditions	-

XRef	2.2.10
Use Case ID	UC29
Use Case Name	Restart Level
Description	This use case describes the event in which the player who wants to restart the level currently played.
Actors	The player
Preconditions	The player should be playing a level.
Trigger	The player clicks the pause button.
Basic Flow	1. The player moves the cursor by hand movement to the pause button. 2. The player clicks the pause button by grabbing gesture. 3. The player clicks the restart level button in the pause menu. 4. A dialog box for confirming the operation pops up. 5. The player clicks on the “OK” button. 6. The level is restarted.
Alternate Flow	-
Exception Flow	3. The player clicks on the “No” button. 4. Game is going on.
Post Conditions	The level should be restated.

XRef	2.2.11
Use Case ID	UC30
Use Case Name	Resume Game
Description	This use case describes the event in which the player who wants to resume the paused game.
Actors	The player
Preconditions	The player should be paused the game.
Trigger	The player clicks resume button.

Basic Flow	1. The player moves the cursor to resume button by hand movement. 2. The player clicks resume button by grabbing gesture. 3. The paused game is started from where it stays.
Alternate Flow	-
Exception Flow	-
Post Conditions	The player continues the level where it stays.

XRef	2.2.12
Use Case ID	UC31
Use Case Name	Select Level
Description	This use case describes the event in which the player who wants to play a level.
Actors	The player
Preconditions	The player should be in Map Screen.
Trigger	The player clicks to a level.
Basic Flow	1. The player moves cursor by hand movement to the level which s/he wants to play. 2. The player clicks to the level by grabbing gesture. 3. The corresponding level starts.
Alternate Flow	-
Exception Flow	-
Post Conditions	The player should be playing the level s/he has chosen.

3.2.2.2 Requirements Related to Actor Parent

XRef	2.2.1
Use Case ID	UC32
Use Case Name	Add Player
Description	This use case describes the event in which the parent who wants to add a

	player.
Actors	The parent
Preconditions	The parent should be in “Profile Manager Screen”.
Trigger	The parent clicks to add player button.
Basic Flow	1. The parent moves cursor by hand movement to add player button. 2. The parent clicks to add player button by grabbing gesture. 3. The add player screen appears on the screen. 4. The parents fill the needed attributes to add a player. 5. The parent moves cursor by hand movement to OK. 6. The parent clicks to OK by grabbing gesture . 7. The pop up indicating the operation is successful appears
Alternate Flow	-
Exception Flow	-
Post Conditions	The player should be inserted into the database.

XRef	2.2.2
Use Case ID	UC33
Use Case Name	Edit Player
Description	This use case describes the event in which the parent who wants to edit a player.
Actors	The parent
Preconditions	The parent should be in “Profile Manager Screen”.
Trigger	The parent clicks to edit player button.
Basic Flow	1. The parent moves cursor by hand movement to edit player button. 2. The parent clicks to edit player button by grabbing gesture. 3. The edit player screen appears on the screen. 4. The parents fill the needed attributes to edit a player.

	5. The parent moves cursor by hand movement to OK. 6. The parent clicks to OK by grabbing gesture. 7. The pop up indicating the operation is successful appears
Alternate Flow	-
Exception Flow	-
Post Conditions	The player in the database should be updated.

XRef	2.2.3
Use Case ID	UC34
Use Case Name	Delete Player
Description	This use case describes the event in which the parent who wants to delete a player.
Actors	The parent
Preconditions	The parent should be in “Profile Manager Screen”.
Trigger	The parent clicks to delete player button.
Basic Flow	1. The parent moves cursor by hand movement to the player who will be deleted. 2. The parent clicks to delete player button by grabbing gesture. 3. The pop up indicating the operation is successful appears. 4. A dialog box for confirming the operation pops up. 5. The player clicks on the “OK” button. 6. The player is deleted.
Alternate Flow	-
Exception Flow	3. The player clicks on the “No” button. 4. The player is not deleted.
Post Conditions	The player should be deleted from the database.

XRef	2.2.4
Use Case ID	UC35
Use Case Name	Read Report
Description	This use case describes the event in which the parent who wants to read the report.
Actors	The parent
Preconditions	The parent should be in “Profile Manager Screen”.
Trigger	The parent clicks to read report button.
Basic Flow	1. The parent moves cursor by hand movement to read report button. 2. The parent clicks to read report button by grabbing gesture. 3. The report appears on the screen.
Alternate Flow	-
Exception Flow	-
Post Conditions	The report should appear on the screen.

3.3 Nonfunctional Requirements

3.3.1 Performance Requirements

3.3.1: req00121

The movements of the player are recognized and responded immediately so that there is no delay which negatively affects the game.

3.3.1: req00122

The frame rate shall be at least 25 frames per second so that the player can play the game fluently.

3.3.2 Availability Requirements

3.3.2: req00123

The Software shall be available at least 99% percent of 7/24 except of power outage.

3.3.2: req00124

The Software shall continue to work properly with a single restart; if there will be unplanned outages.

3.3.3 Maintainability Requirements

3.3.3: req00125

All codes and components of software shall be fully documented.

3.3.3: req00126

Last modified time of codes and components shall be documented.

3.3.3: req00127

All codes shall be prepared to permit future modifications.

3.3.3: req00128

All codes shall be stored in separate files.

3.3.4 Portability Requirements

3.3.4: req00129

The Software shall work on tablets, Xbox, PlayStation and Nintendo Wii with 3D sensor.

3.3.5 Reliability Requirements

3.3.5: req00130

The Software shall never crash or hang, other than as the result of an operating system error.

3.3.5: req00131

The Software shall prepare the player report without any fault.

3.3.6 Security Requirements

3.3.6: req00132

The Software shall encrypt all private information about players.

3.3.6: req00133

The Software shall require any private information from the player except for age, and gender.

3.3.6: req00134

The Software shall protect the 3D camera data from the hackers.

3.3.7 Safety Requirements

3.3.7: req00135

The Software shall not contain any adult contents.

3.3.8: req00136

The Software shall not contain any content that harms child physiology.

4 Data Model and Description

4.1 Data Description

This section will give information about the data objects related to this project, the relationship among them, the attributes of the data objects and the complete data model with data objects' functions included.

The software includes three data objects: Player, Pet, and Level. Fig. 10 shows the Entity Relationship diagram of the software. The details of the data objects are as below:

- Player entity contains all information of players which are id, name, surname, age, gender, photo and report. Each player must have unique id. Other attributes of player may be same with other players. The software gives permission to have more than one level to player. However, having more than one pet does not permitted by the system.
- Pet entity contains all information of the pet which are id, emotion, dirt, health, hunger, thirst. Each pet must have unique id. Other attributes of pet may be same with other pets. Each pet must be had by exactly one player. In other words, pets do not exist without players.
- Level entity contains all information of game levels which are id, kind, and open/lock. Each level must have unique id. Other attributes of level may be same with other levels. Each level can be had by more than one player.

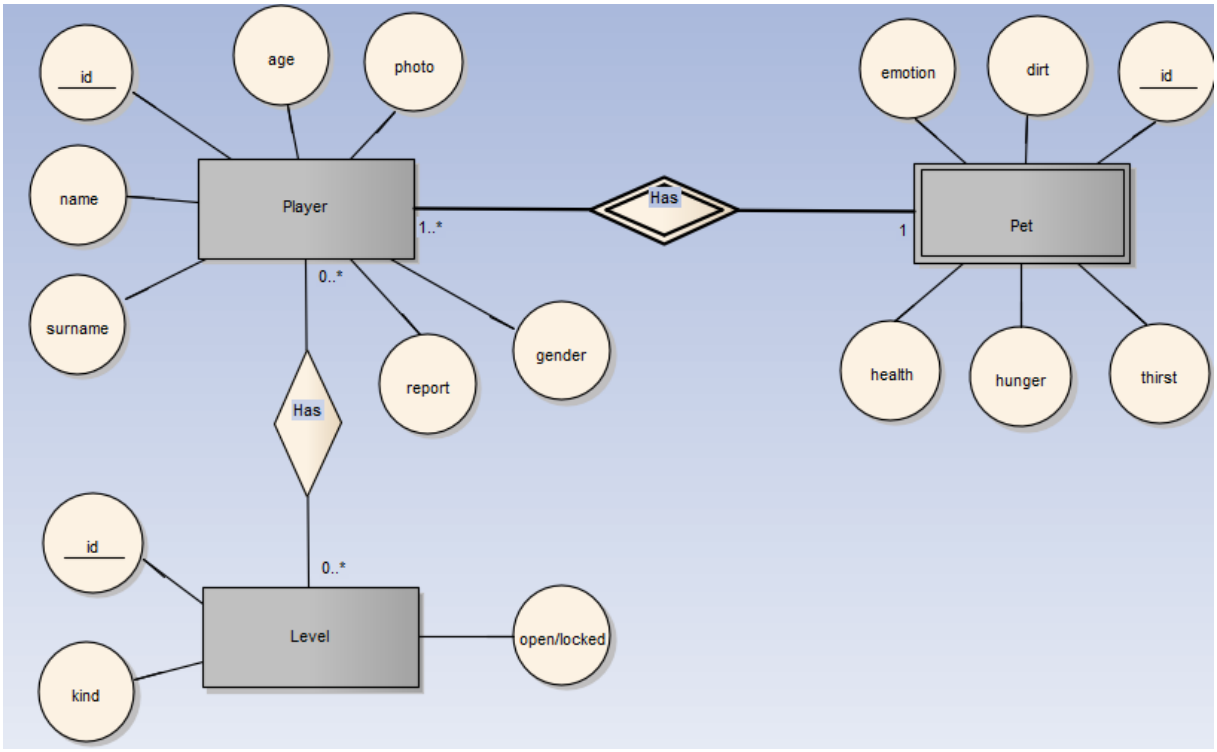

Fig. 10 – ER Diagram

4.2 Data Objects

The software has ten main classes which are Level, Pickable, Hitable, Grabable, Camera, Processor, Database, Map, Player, and Pet. The relationships between classes are shown in the following class diagram, Fig. 11.

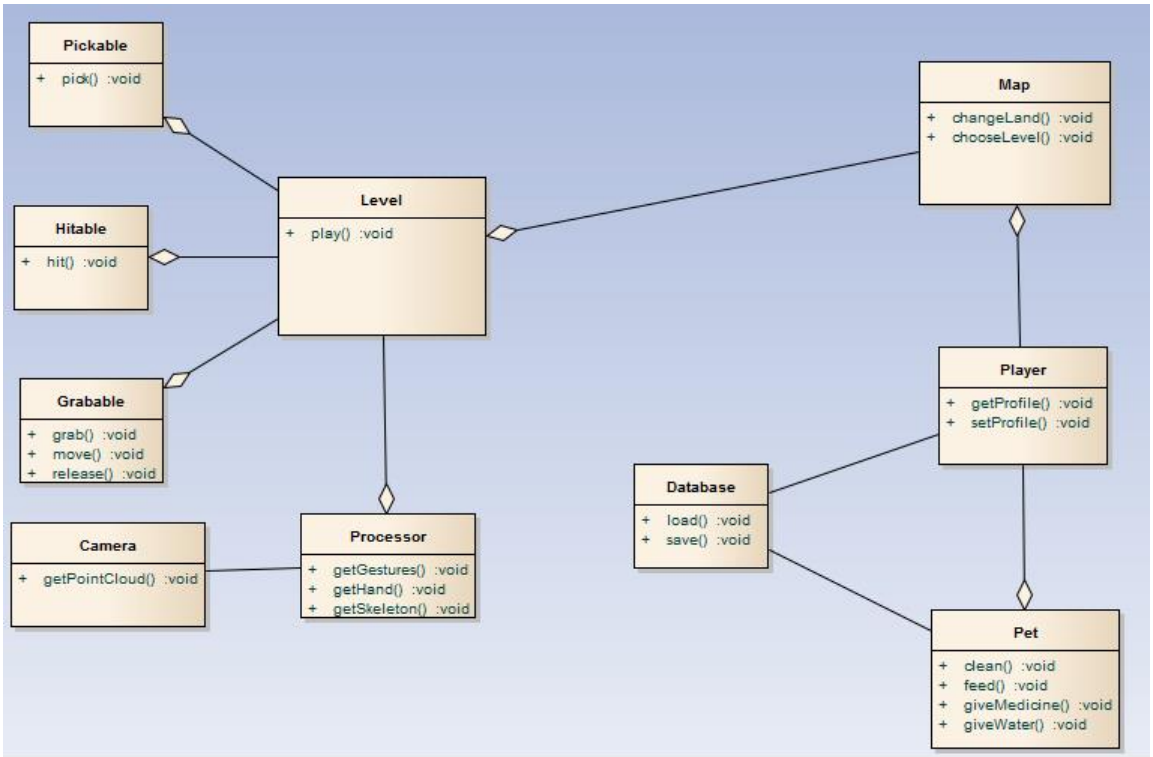

Fig. 11 – Class Diagram

The brief information about classes is described below:

- **Level:** Level class initializes the main settings according to selected game and level. After that, it starts the main loop of the game. This class includes Pickable, Hitable, Grabable, Processor classes.
- **Pickable:** Pickable class represents objects which can be picked in the games and how to control these objects.
- **Hitable:** Hitable class represents objects which can be hit in the games and how to control these objects.
- **Grabable:** Grabable class represents objects which can be grabbed in the games and how to control these objects.
- **Camera:** Camera class gets the 3D point cloud of scene from 3D sensor.
- **Processor:** Processor class process the point cloud which is retrieved by Camera class. By using this class, the software can detect gestures, skeleton and hand of the player.
- **Database:** Database class is responsible for loading and saving system data.
- **Map:** Map class is responsible for controlling the transition between levels and between game groups.
- **Player:** Player class is responsible for controlling player's profile settings.
- **Pet:** Pet class represents the pet which the player has and is responsible for caring the pet.

4.3 Data Dictionary

4.3.1 Player

id: unique id of the player

name: name of the player

surname: surname of the player

age: age of the player

gender: gender of the player

photo: photo of the player

report: report which contains information about growth of the player

4.3.2 Pet

id: unique id of the pet

emotion: happiness level of the pet

dirt: dirt level of the pet

health: health level of the pet

thirst: thirst level of the pet

hunger: hunger level of the pet

4.3.3 Level

id: unique id of the level

kind: kind of the level

open/lock: Boolean value indicating whether the player reach this level or not.

5 Behavioral Model and Description

5.1 Description for software behavior

The Oyun Diyarı starts with “Welcome Screen”. In this screen, users can edit all profiles as super user, or select one of the existing profiles. It is recommended that, users should select their own profiles or create new one. If user wants to add or delete a profile, s/he must enter “Profile Manager”. Otherwise user can go to “Main Menu” by selecting a profile.

After user go into “Main Menu”, s/he is called as player. By using”Main Menu”, player can see his/her profile or go to main game map. If player wants to see his/her profile, s/he must enter “Profile” by using “Profile Control”.

“Map” screen provides list of all games to player with a specified user interface. Player should select a level to play the games. After player select a level, s/he enter automatically “Play Level”

screen. In this screen, the Oyun Diyarı software is started the selected game. “Play Level” references itself that means player can restart the games without returning the “Map”.

“Quit Game” is the final state of the Oyun Diyarı’s state diagram. If player want to exit from the game, s/he should reach this state by using “Quit”. “Quit” is available from “Main Menu”, “Map”, and “Play Level”. Therefore, to quit the game, player should be one of these states.

Note that, Oyun Diyarı’s state diagram also includes “Go to Welcome Screen” and “Go to Main Menu” option. These options provides to player going back previous steps easily. These options are necessary for the state diagram.

5.2 State Transition Diagram

The behavior of the system described in section 5.1. Fig. 12 shows the state transition diagram of the system.

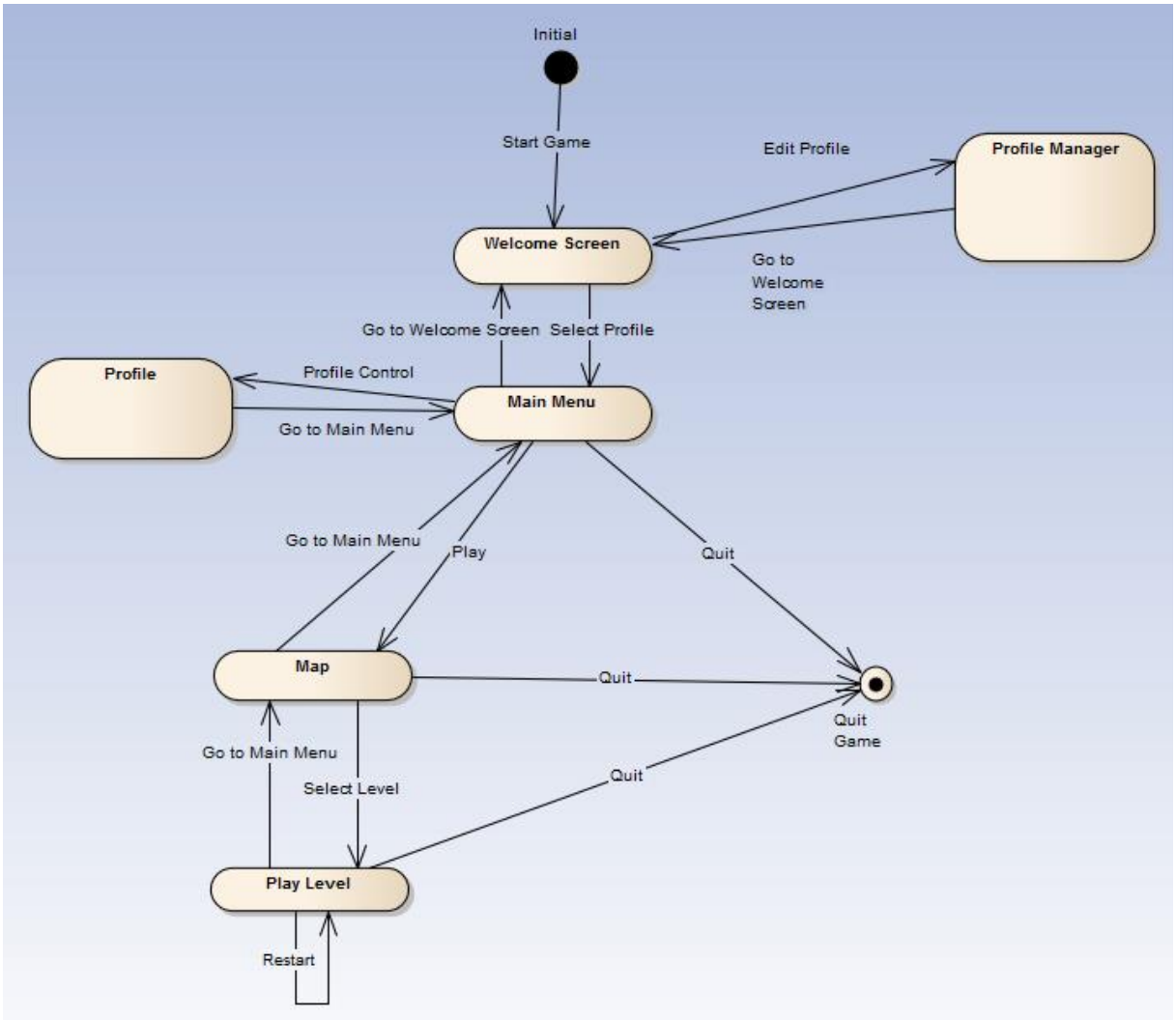

Fig. 12 – State Transition Diagram

6 Planning

6.1 Team Structure

Our team VisionImpossible has a self-contained team structure as it has four members as developers who are Cemal AKER, Oğulcan ERYÜKSEL, Anıl GENÇ and Oğuzhan TAŞTAN; one project manager who is Kerem ÇALIŞKAN, co-founder of InfoDiff, and one advisor Research Assistant Ali Fatih GÜNDÜZ.

6.2 Estimation

We plan to get a clear idea on what will be done and what will be the end product. During that period, we also plan to get comfortable with the technologies we will use in the project. We do not mention about the expectations of the lecturers such as web-page. When we are ready to code, which we presume that we will be ready at the beginning of the second semester, we plan to start coding and finish the project before the end of the second semester.

6.3 Process Model

Our process model is spiral process model which is a non-operational software production process model. The good thing about it from our perspective is that we may need to do lots of updates during our coding period because of the difficulties we could not foresee; then with the spiral process model, we can return back to our specifications and descriptions.

7 Conclusion

This document is a software requirements specification for an educational game project named “Oyun Diyarı”. The game will be played using 3D point cloud sensors. In addition to the features mentioned in this document there are some planned future works:

- Oyun Diyarı will support other foreign languages besides Turkish.
- Oyun Diyarı will analyze player report automatically and detect any disability.
- Oyun Diyarı will be playable from every platform if it supports 3D sensor camera technology.
- Oyun Diyarı will include more new games and levels.
- Oyun Diyarı will be integrated with online game platforms such as Steam and Google Play.

8 Appendix

8.1 Table of Figures