Software Requirements Specification

for

Memorizing Notes

Version 1.0

Prepared by

Safa Uslu

Group Name: Group 8

Safa Uslu	1700003646	1700003646@stu.iku.edu.tr

Instructor:	Akhan Akbulut
Course:	Software Engineering
Lab Section:	*D*
Teaching Assistant:	*Büşra Kocaçınar*
Date:	11.05.2022

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Safa Uslu	Menu scene added, levels menu added. Game F.	11.05.2022
0.9	Safa Uslu	8 more levels added. E/N/H levels designed	07.05.2022
0.5	Safa Uslu	Level passing scene's coded.	04.05.2022
0.4	Safa Uslu	Some levels designed.	26.04.2022
0.3	Safa Uslu	Random notes will be shown in scene.	19.04.2022
0.2	Safa Uslu		12.04.2022

0.1	Safa Uslu	Score now shown in scene. Piano tiles and sounds created.	21.03.2022

• Introduction

Memory games has been developing for many users since game history. The memorizing notes project is a game for improving and keeping fresh of memory for children.

• Project Purpose and Scope, and Objectives

Memorizing Notes game developed on Unity. The Project can be executed in different formats. For windows, exe. For Android, apk etc.

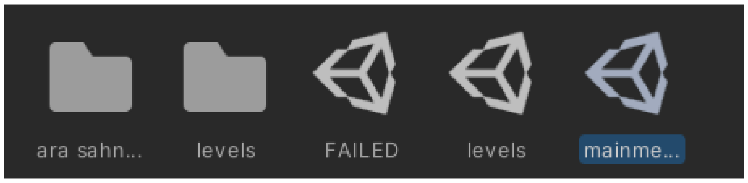
Software requirements may change depends on the operating system the game works on. However game has 7 note file, a couple of image and Unity's basic UI elements so it may work on old systems too. Only requirements are a screen, an audio output, and a mouse or any clickable input devices.

• Roles and responsibilities

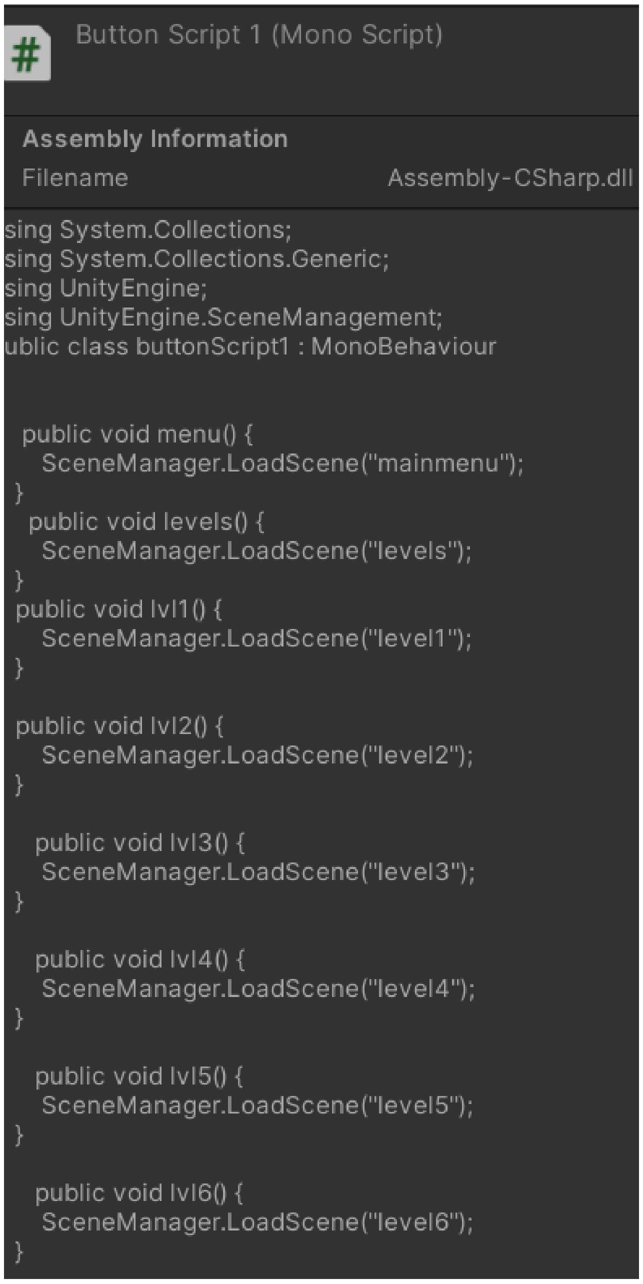
Safa Uslu
-Developement
-Design
-Code
-Idea

• Technical Assumptions and Constraints

Coding language is C# and Game Engine is Unity. Each button, level, and logic of the game are wrote in different scripts.



An example of script for button in the games



Button Script 1 (Mono Script)

Assembly Information

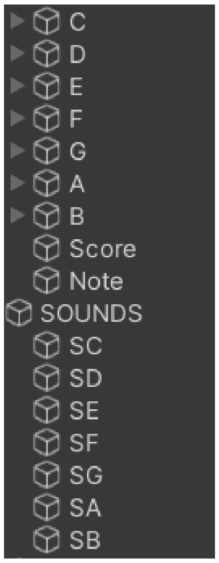
Filename Assembly-CSharp.dll

```
sing System.Collections;
sing System.Collections.Generic;
sing UnityEngine;
sing UnityEngine.SceneManagement;
ublic class buttonScript1 : MonoBehaviour


  public void menu() {
    SceneManager.LoadScene("mainmenu");
 }
   public void levels() {
    SceneManager.LoadScene("levels");
 }
 public void lvl1() {
    SceneManager.LoadScene("level1");
 }

 public void lvl2() {
    SceneManager.LoadScene("level2");
 }

   public void lvl3() {
    SceneManager.LoadScene("level3");
 }

   public void lvl4() {
    SceneManager.LoadScene("level4");
 }

   public void lvl5() {
    SceneManager.LoadScene("level5");
 }

   public void lvl6() {
    SceneManager.LoadScene("level6");
 }
```

- **Naming Conventions**

Each button, image and the texts are UI element of unity.
C,D,E,F,G,A,B are the names of the piano's buttons

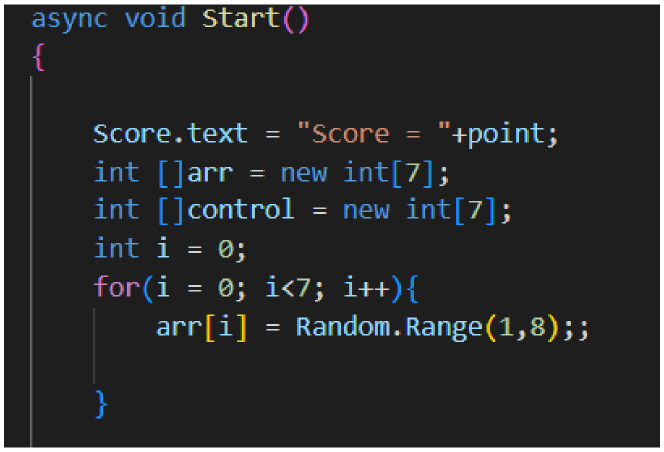
SC,SD,SE,SF,SG,SA,SB are the names of the sound elements
button, button1 are the names of buttons in level passing
score, note are the text elements holding the information of the level's notes and scores.



• Requirements

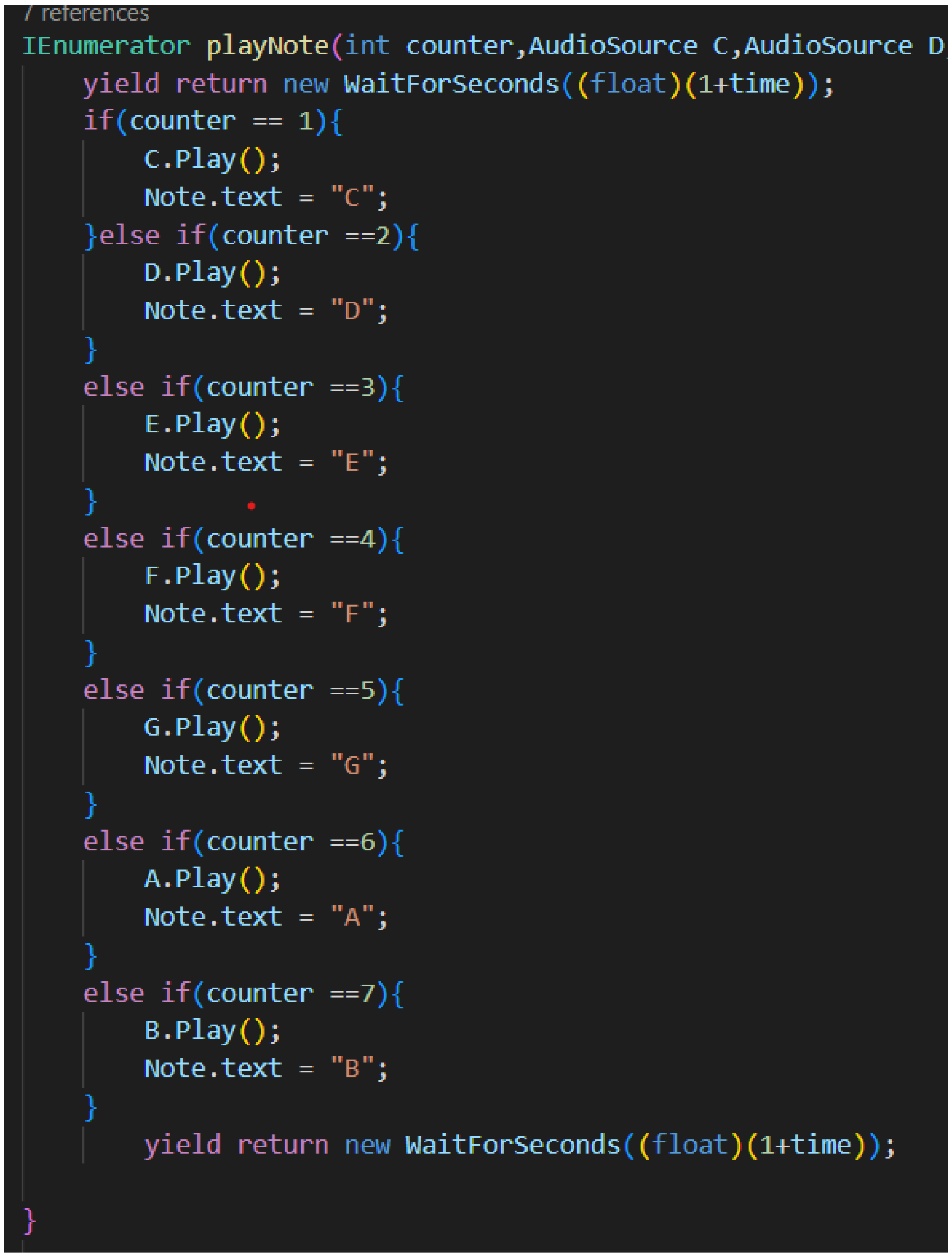
• Functional Requirements

When a level starts, loop arrange for the note order.



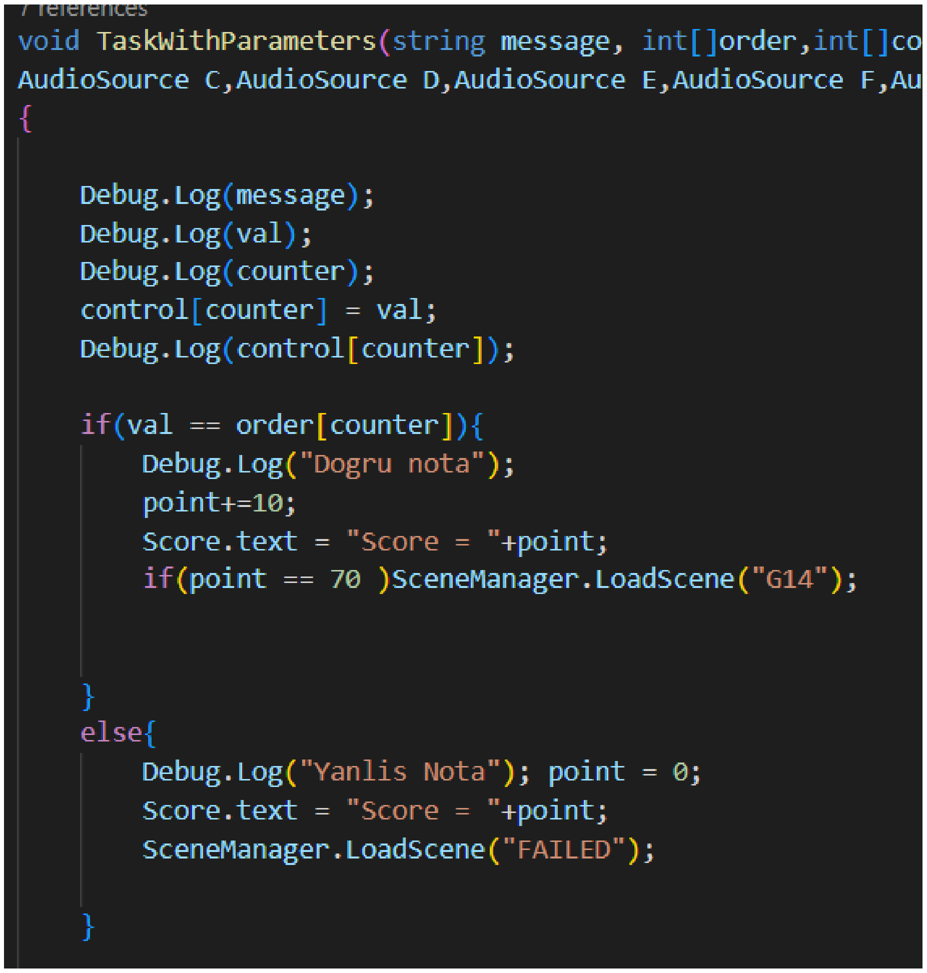
```
async void Start()
{

    Score.text = "Score = "+point;
    int []arr = new int[7];
    int []control = new int[7];
    int i = 0;
    for(i = 0; i<7; i++){
        arr[i] = Random.Range(1,8);;

    }
```

playNote function
When a level starts. Play note function get the information of note's order and time as parameters. Then play the sounds beginning of the level.



```
7 references
IEnumerator playNote(int counter,AudioSource C,AudioSource D
    yield return new WaitForSeconds((float)(1+time));
    if(counter == 1){
        C.Play();
        Note.text = "C";
    }else if(counter ==2){
        D.Play();
        Note.text = "D";
    }
    else if(counter ==3){
        E.Play();
        Note.text = "E";
    }
    else if(counter ==4){
        F.Play();
        Note.text = "F";
    }
    else if(counter ==5){
        G.Play();
        Note.text = "G";
    }
    else if(counter ==6){
        A.Play();
        Note.text = "A";
    }
    else if(counter ==7){
        B.Play();
        Note.text = "B";
    }
        yield return new WaitForSeconds((float)(1+time));

}
```

When player click a button(piano tile, button called TaskWithParamaters function so game logic may continue.



```
7 references
void TaskWithParameters(string message, int[]order,int[]co
AudioSource C,AudioSource D,AudioSource E,AudioSource F,Au
{

    Debug.Log(message);
    Debug.Log(val);
    Debug.Log(counter);
    control[counter] = val;
    Debug.Log(control[counter]);

    if(val == order[counter]){
        Debug.Log("Dogru nota");
        point+=10;
        Score.text = "Score = "+point;
        if(point == 70 )SceneManager.LoadScene("G14");


    }
    else{
        Debug.Log("Yanlis Nota"); point = 0;
        Score.text = "Score = "+point;
        SceneManager.LoadScene("FAILED");

    }
```

Safety and Security Requirements

If user use headphone must be carefull of the voice level.

The game may portable to VR, so user must be careful about the position of his/her in the playing area to prevent hitting objects in the area.

User must be careful about the play time since game is about hearing random notes. Listening random notes for a while may cause nausea.

- **Software Quality Attributes**

The game wrote on Unity and used Unity's basic elements so it is portable for other platforms easly. If a new platform developed Unity will give it support so the game is.

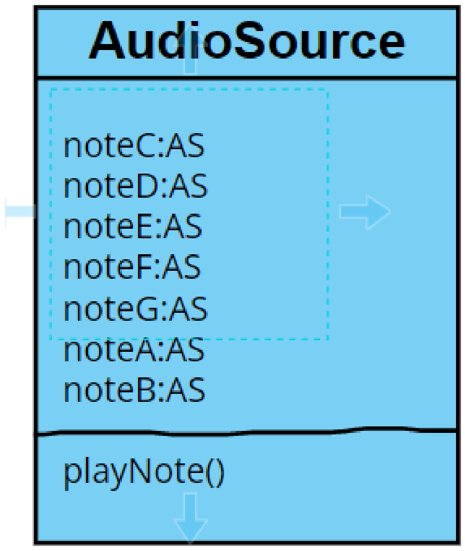
Each level use same logic, and similar script so it is flexible to add new levels, new difficulties and new challenges.

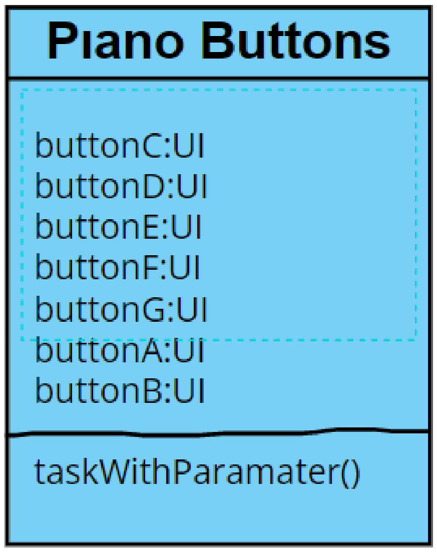
When developing the game, each functional operations and each user operations wrote on logs so it is easy to follow and understand when any bugs or errors happened.

- ## System Architecture and Architectural Design

Event Driven Architecture used for the system. Codes are written by EDA for when the project manager give new request it is easy to implement and code.

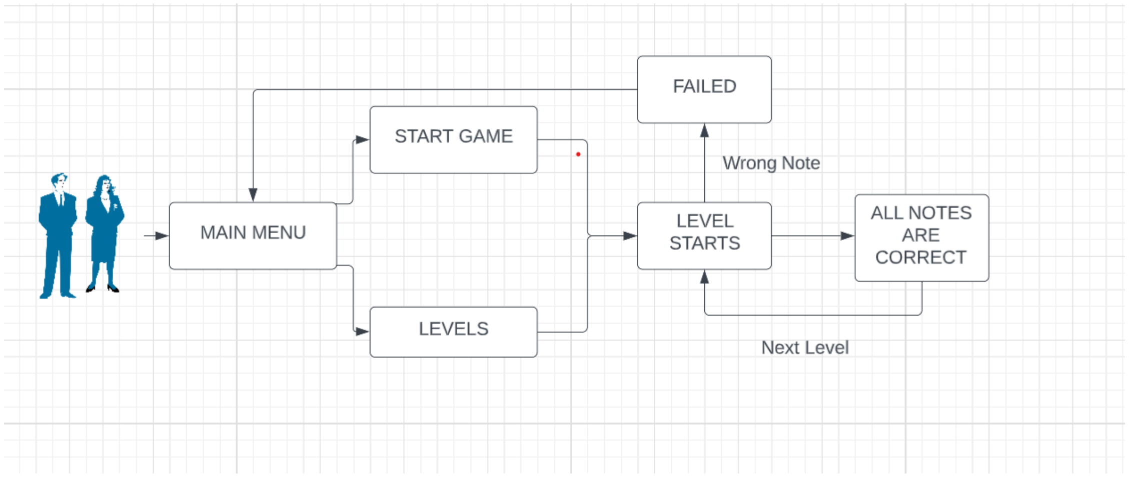
- **Logical View**





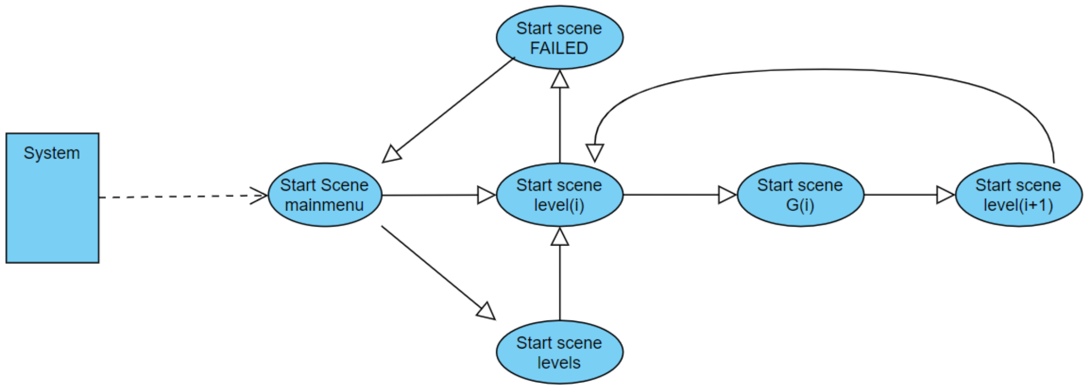
- *AudioSource class for hear to notes. play function makes it.*
- *PıanoButtons class is Visuable objects. Each button call taskWithParamaters function explained above.*

• Use Case View



When player start the game first thing he/she saw is main menu. Main menu has 2 options, starting from level 1 or chosing a level. When user do any of them the level starts and if the user click a wrong note game terminate. If the user do all notes correct game will continue on next level.

• Use Case Scenarios



**Game starts with scene1 which is mainmenu.*
**Mainmenu has 2 options, start the game and select levels.*
**If user start the game, i will be a 1. Start from level1.*
**If user select levels, user will see all levels.*
**If user select any level num at levels scene, game will start and i variable will be that choice.*
**If user pass the level user will see G(i) scene which is succesfull and able to pass next level.*
**If user click next level, i will be i+1 and this loops goes on.*
**If user doesn't succesful at level, FAILED scene will be execute.*

**User can only go back mainmenu scene at this FAILED page.*

• Design and Implementation

***StartCoroutine(playNote(arr[0],SC,SD,SE,SF,SG,SA,SB,0));*

Plays the note with playNote function.

***C.onClick.AddListener(delegate {TaskWithParameters("C'YE BASILDI",arr,control,1,ref counter,SC,SD,SE,SF,SG,SA,SB); });*

C,D,E,F,G,A,B notes has this line. When user click button Unity calls task with parameters function.

```
  **void TaskWithParameters(string message, int[]order,int[]control, int val,ref int _counter,
   AudioSource C,AudioSource D,AudioSource E,AudioSource F,AudioSource G,AudioSource
A,AudioSource B)
   {control[counter] = val;
      if(val == order[counter]){
         point+=10;
         Score.text = "Score = "+point;
         if(point == 70 )SceneManager.LoadScene("G14");
      }
      else{
         Score.text = "Score = "+point;
         SceneManager.LoadScene("FAILED");
      }
   }
}
```

game logic works in this function

• References

https://docs.unity3d.com/2019.1/Documentation/ScriptReference/UI.Button-onClick.html

https://archive.org/details/24-piano-keys