RE – IT – SEP4 C – F18
The memory palace game

Project Report

Date: 24th August 2018

Group 18: Andrei-Mihai Surueanu student number: 240370
Josip Drinovac student number: 254089

Supervisors: Jakob Knop Rasmussen
Kasper Knop Rasmussen

Table of Contents

List of figures and tables

Abstract

This project is a 4th semester Information & Communication Technology (ICT) Engineering project at VIA University College in Horsens, aimed to develop a serious game for memorization training. The company Ensight Games asked the students of ICT Engineering under the Cross-media specialization, which have a focus on serious games, to develop a serious game for memorization training.
The game itself is developed as being part of the 4th semester "Semester Project" for the Cross-media specialization. The 4th semester "Semester Project" is part of the ICT Engineering Bachelor's degree curriculum and takes place over a period of three weeks. The game is developed using the Unity engine and is intended for the mobile platform (phones/tablets).
For the project there have been used various methods and techniques. In terms of how the project was conducted and how the workflow management aspect was handled, SCRUM was chosen as the main framework for the development and the implementation of this project.

1. Introduction

This project is intended to create a game that addresses the problem of memory, but to a limited extent.

The memory is a complex phenomenon and it helps us in many different areas and in many different ways. The problems that arise with memory have various causes, they range from serios medical illnesses to just a lack of training. The reason for the lack of training when it comes to the process of memorizing something is that our current modern tools aid the people in their daily tasks so much, that the people do not use their mind to memorize things, as much as they used to, because now the information is as accessible as it has never been before. (Project description) However, this project will not address the serious medical conditions and illnesses such as Alzheimer or other memory disfunctions that could arise from car accidents or other causes, rather this project will deal with the lack of memory training, and more specifically the project will try to improve the process of memorizing information.

From a technical stand point with a simplified approach, the memory and memorization process could be seen as being composed of two separate processes, “the In process” or the process of inserting new information into the memory, and “the Out process” or the process of retrieving the information that was previously stored in memory. From a medical standpoint there can be identified an additional element for the memory which is the process of retaining the information into the memory or holding it in place.

The performance of retaining information into the memory is affected by various causes as stated above, such as medical conditions like Alzheimer, genetics, car accidents etc.

The project will address the process of inserting information into the memory and will try to improve the performance of this process. Additionally, the project will contain an element that will help to verify if the information was correctly inserted into memory and if the information can be retrieved from memory, even though the main focus of the project are not the processes of retrieval and retention of the information into the memory, but rather the process of inserting the information into memory. The element will serve as a tool to measure if the performance of the memorization process was improved and to see to which extent the performance of the memorization process was improved.

2. Analysis

2.1. Requirements

Below are the full set of requirements. The requirements have been divided into two subsections corresponding to their priority.

- The application must improve the user's ability to memorize something
- The application must be developed using Unity
- The application must be developed for phones/tablets
- The group should consist of 2-4 people
- The project should include the memory palace technique as the core mechanic around the game
- The game should have several levels that the player can choose from with different topics for the process of memorization
- The game should be made in a 3D environment
- The topics of memorization could include real topics such as chemistry or math and practical purposes such as remembering the grocery list or the ingredients in a cooking recipe

Below there are the requirements with a higher priority. These requirements have been offered by the stakeholders (Ensight games) and the two supervisors for the project Kasper Knop Rasmussen and Jakob Knop Rasmussen.

2.1.1. Requirements with high priority

- The application must improve the user's ability to memorize something
- The application must be developed using Unity
- The application must be developed for phones/tablets
- The group should consist of 2-4 people

Below there are the requirements with a lower priority. This is an additional set of requirements that has been added after the process of analysing the domain scope and designing and conceptualizing the idea for the project.

2.1.2. Requirements with low priority

- The project should include the memory palace technique as the core mechanic around the game
- The game should have several levels that the player can choose from with different topics for the process of memorization
- The game should be made in a 3D environment
- The topics of memorization could include real topics such as chemistry or math and practical purposes such as remembering the grocery list or the ingredients in a cooking recipe

2.2. Use Case Diagram

The use case diagram is a way of communicating in a simple way what the game or project is supposed to do, so that other people can understand. Below is the Use Case Diagram for the Memory palace game. The actor which in this case is the player can perform the following three operations.

The first functionality is the option of selecting a level. From the main menu the game offers the user to go into a scene where there are several options for selecting a level.

The second functionality is: once the player has selected the desired level the game will start, and the player will go through the gameplay following the instructions displayed on the screen.

The third functionality is: once the time for the level that the player is currently playing has passed, the player will receive a quiz game, where there will be displayed several questions on the screen. Once the player answered all the questions, the score for the quiz game will be displayed on the screen and the player will have the option to go to the main menu.

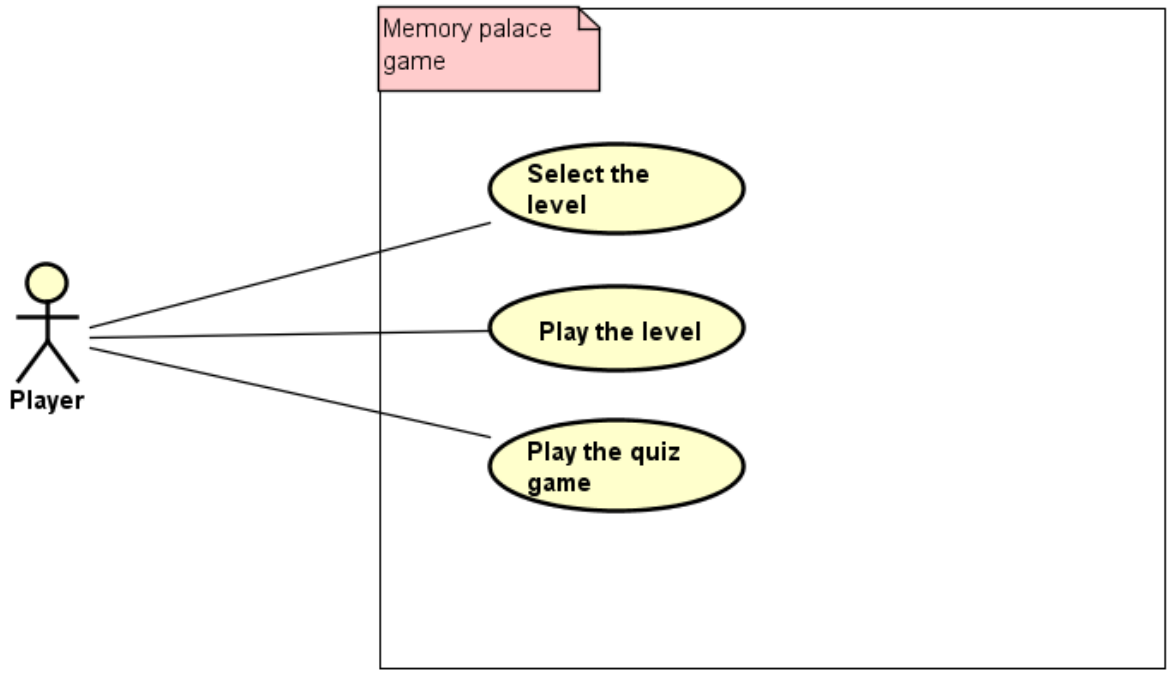


Figure 1 – Use case diagram

2.3. Use Case Descriptions

Below is the Use Case Description for the “Select the level” Use Case. The Use Case Description explains what is the process that the actor goes through the performing the Use Case. In this case the player goes to the screen of selecting a level by pressing a button. The Player can look through the levels by dragging with a finger on the screen on the area where the levels are displayed. The player can select the level by tapping once on the picture that corresponds to the level and the level will start loading immediately.
In the base sequence point number 3, there should be written “from right to left” and not “from right to left”.

Select the level / UseCase Description

ITEM	VALUE
UseCase	Select the level
Summary	User selects the level that he will play.
Actor	Player
Precondition	The game is loaded and it's in the main menu.
Postcondition	The player enters the level.
Base Sequence	1. The player presses the button for selecting the level 2. The screen for selecting the level appears. 3. The player scrolls through the levels using a finger to drag the screen on a horizontal line from left to right and from roght to left on the area where the l evels are displayed 4. The player selects the level by tapping once on the level that he wishes to p lay.
Branch Sequence	
Exception Sequence	Main menu: 1-2 as base sequence. 3. The player presses the back button and the screen with the main menu app ears.
Sub UseCase	
Note	

Figure 2 – Use case description (Select the level)

Below is the use case description for the “Play the level” Use Case. The use case description explains the functionality of the use case where the player can move around in the scene by using a joystick displayed on the screen. The player can jump in the scene by pressing a button corresponding to the jump functionality that is displayed on the screen and can change the camera angle by using a swipe gesture on the screen, from right to left (the camera changes to left) or from left to right (the camera changes to right).

Play the level / UseCase Description	
ITEM	VALUE
UseCase	Play the level
Summary	The player plays the level that he has perviously selected.
Actor	Player
Precondition	The player has selected the level
Postcondition	The screen with the Quiz Game appears.
Base Sequence	1. The player can move in the game using a joystick displayed on the screen. 2. The player can jump by pressing the jump button. 3. The player can change the camera angle by swiping from left to right to ch ange the camera angle to the right, and from right to left to change the camer a engle to the left.
Branch Sequence	
Exception Sequence	Main Menu: 1-3 as base sequence 4. The player presses the pause button. 5. The player presses the Main Menu button. 6. The Main Menu screen is displayed. Restart: 1-3 as base sequence 4. The player presses the pause button. 5. The player presses the restart button. 6. The level restarts. Resume: 1-3 as base sequence 4. The player presses the pause button. 5. The player presses the resume button. 6. The level resumes.
Sub UseCase	
Note	

Figure 3 – Use case description (Play the level)

This use case description presents what happens when the actor performs the “Play the Quiz game” Use case.

After the countdown in the level has reached 0, the player will be offered a quiz game where there will be displayed a set of questions based on the environment that the player played in the last level. At the end of the questionnaire the player will receive the score for the quiz game displayed on the screen.

Play the quiz game / UseCase Description	
ITEM	VALUE
UseCase	Play the quiz game
Summary	The player enters the Qiuz game section and plays the Quiz game by answeri ng the questions.
Actor	Player
Precondition	The 60 seconds in the level section have passed and the Quiz game screen is displayed
Postcondition	The player returns to the Main Menu and the Main Menu screen is displayed.
Base Sequence	1. The player presses the button to start the game. 2. The player answers the questions by pressing the button coresponding to t he answer that he wihses to pick. 3. The player receives his score displayed on the screen 4. The player presses the button coresponding to the Main Menu
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Figure 4 – Use case description (Play the quiz game)

2.4. Activity Diagrams

The Select the level activity diagram shows the process that the player goes through when selecting the level. In the main menu panel, the player pressed the Level Selection button which takes the player to the level selection panel. Here the player can choose to press the back button which takes him to the main menu panel or to press the level thumbnail button which takes the player to the beginning of the level.

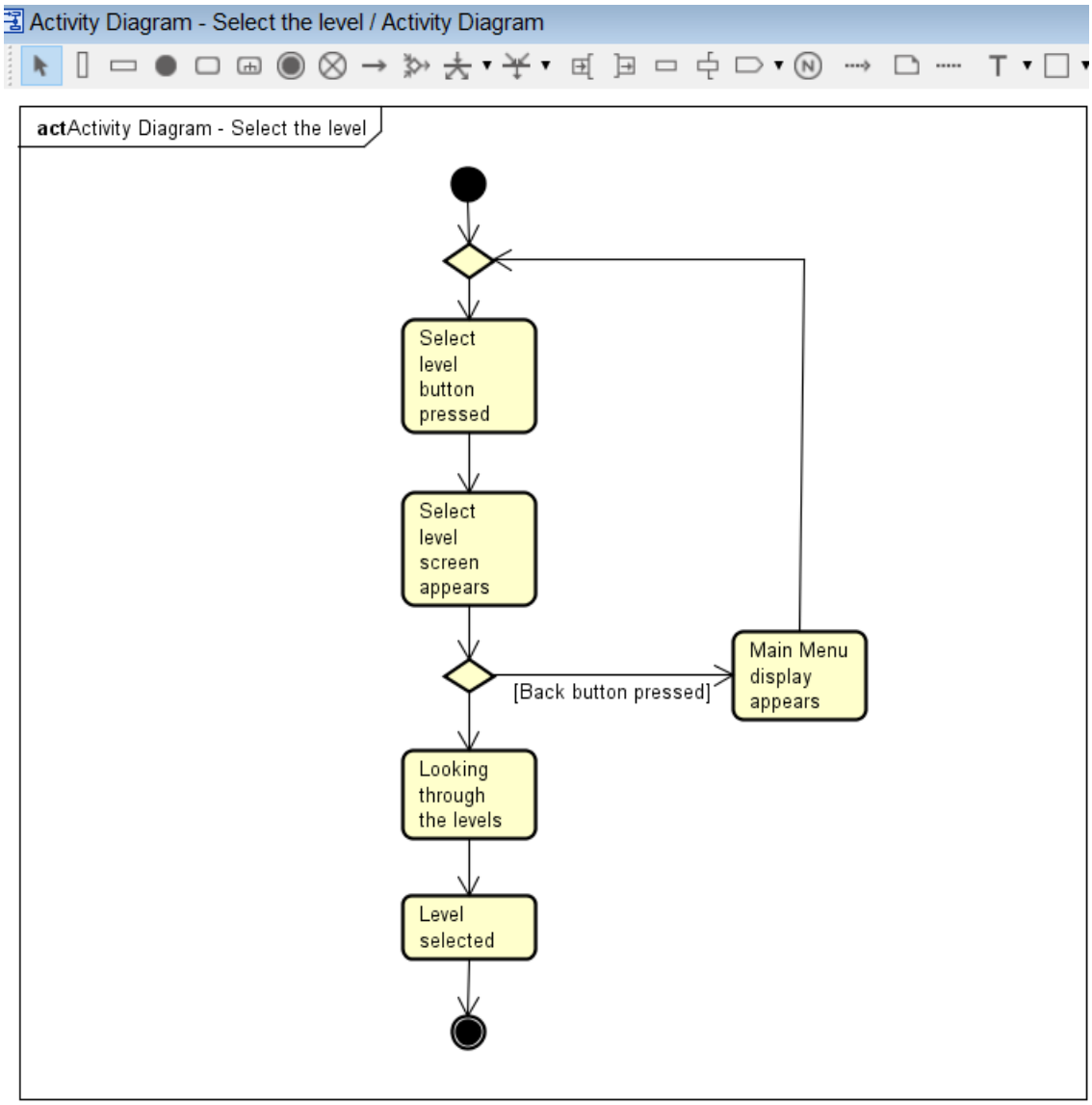


Figure 5 – Activity diagram (Select the level)

The Play the level activity diagram shows what the player can do during game play. The player can move in the scene with the joystick on the screen, he can press the jump button in order to jump and change the camera angle by pressing the two buttons on the screen. If the player presses the pause button the player has three options: to press the main menu button which takes the player to the main menu, to press the resume button and the game will resume and to press the restart button and in this case the level will restart.

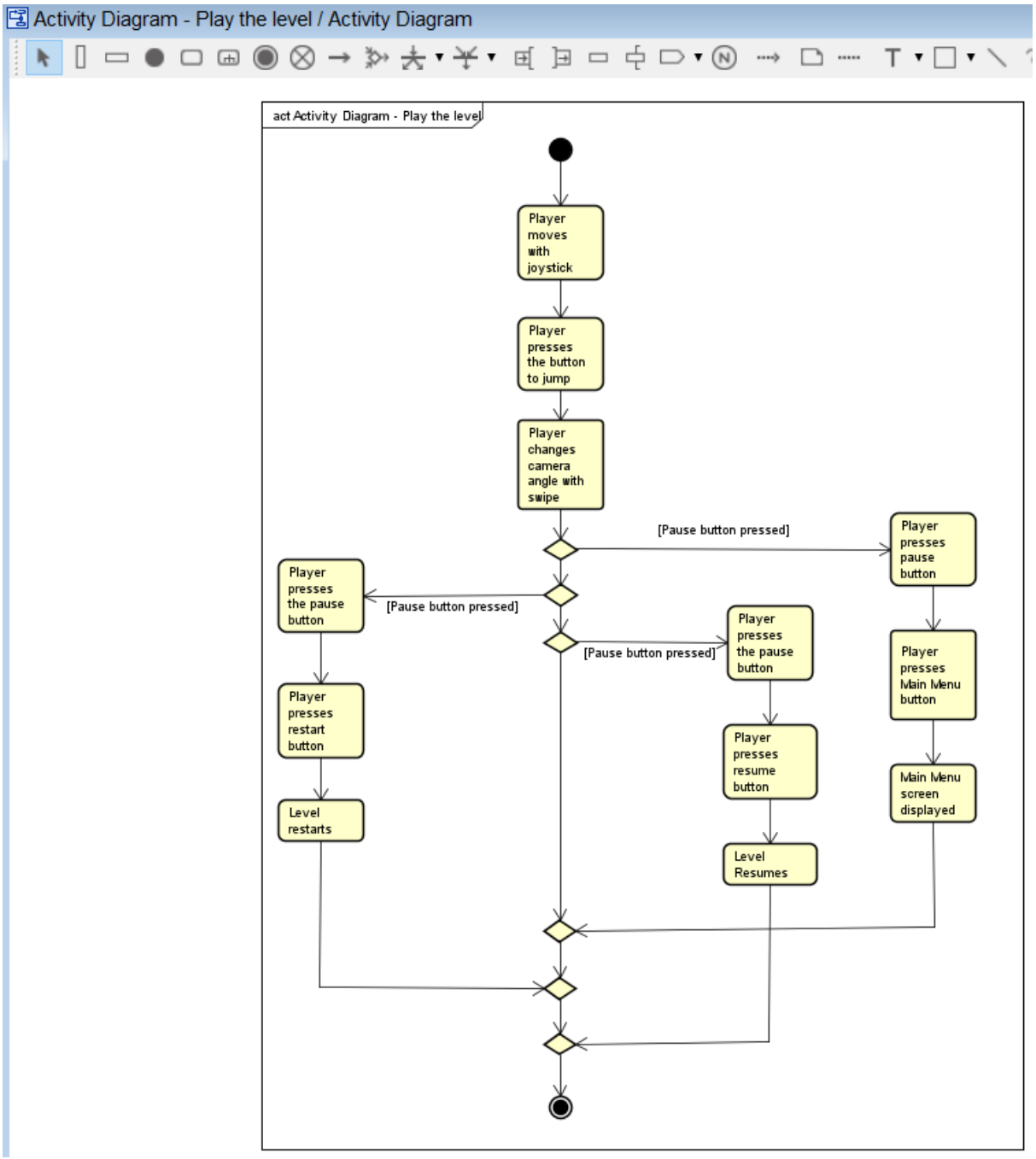


Figure 6 – Activity diagram (Play the level)

The Play the quiz game activity diagram shows the options that the player has during the quiz game. When the level ends the player has the option to press a button to start the quiz game. The player has the option to answer the questions and at the end of the quiz game the final score is displayed. The player has the option from this point to go the main menu by pressing a button.

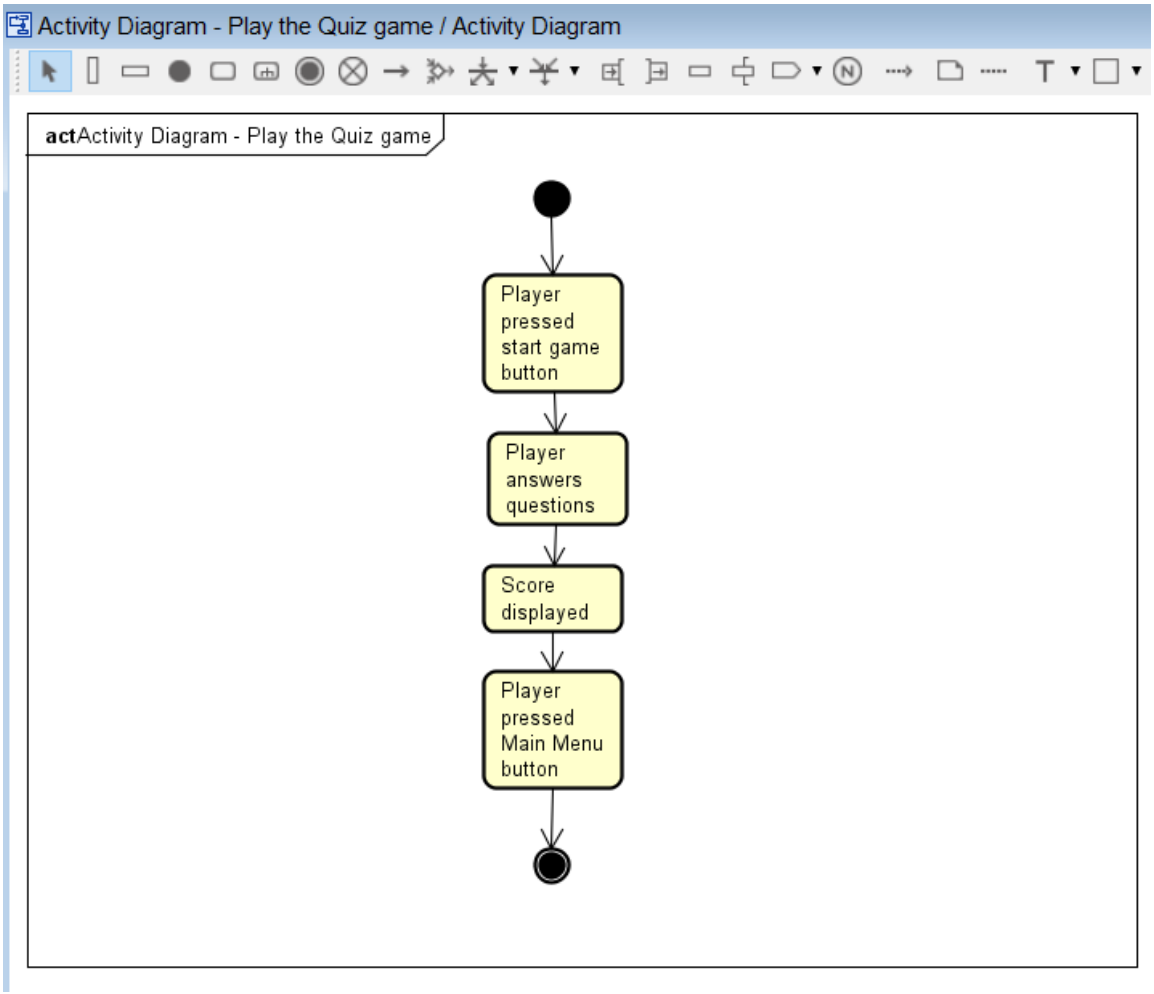


Figure 7 – Activity diagram (Play the quiz game)

3. Design

3.1. Class diagram

The class diagrams will represent the objects in the game and the scripts attached to them, to give an understanding of how the parts that make up the game connect to each other, and how the logic of the game is structured in relation to the code and the different game components.

Below is represented the Class Diagram for the Level_1 scene. Each rectangle together with the note that is inside that rectangle represents a game object listed in the Hierarchy of the Unity project. Part of the class diagrams are only the game objects that have a script attached to it or the game objects that use a script in some way without being directly attached to the game object.

If a game object is connected to the script of another game object it can mean two things:

- If the game object is connected to an attribute of a script attached to another game object, then it means that the other game object has a reference of the first game object. As an example, the Player game object is connected to the lookFor attribute of the CameraMovement script, in return this means that the MainCamera game object has a reference of the Player game object through the CameraMovement script. This is applicable only for the attributes that are noted public or [SerializedField].
- If the game object is connected to a method of a script attached to another game object, then it means that the game object has attached to it the other game object through which the methods of the script can be accessed. As an example, the JumpButton game object has attached to it the Player game object through which it can access the Jump method inside the PlayerMovement script. This option of connecting the game object to the method of a script is applicable only for the game objects that represent a button.

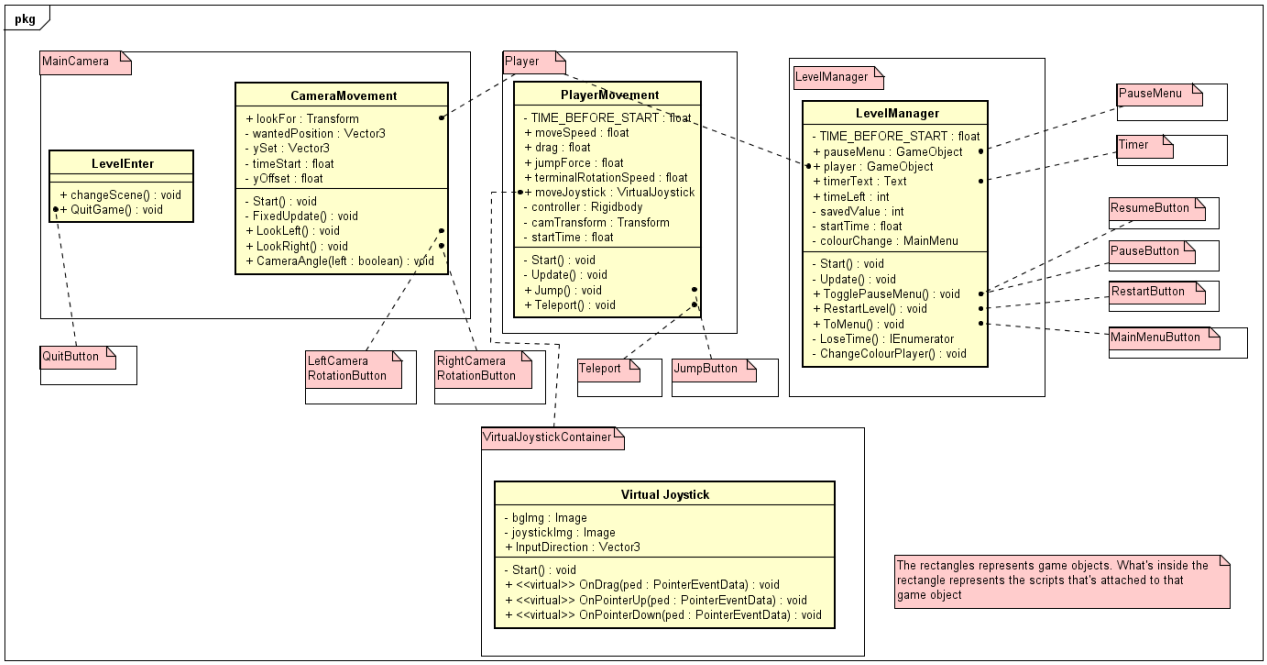


Figure 8 - Class diagram (Level_1 scene)

In the Level_1 scene the MainCamera game object has attached to it two scripts which are LevelEnter and CameraMovement. The CameraMovement script is responsible for making the camera follow the ball character

during gameplay, to rotate the camera when the two buttons on the screen
left and right are pressed, and it is also responsible with the transition in the beginning of the game from a perspective view of the environment to a third person view.

The LevelEnter script is attached to this game object in this scene because it is responsible for the functionality of the QuitButton, which is present in two scenes, in the Level_1 scene and in the MainMenu scene.

The QuitButton, the LeftCameraRotationButton and the RightCameraRotationButton game objects have each a reference to the MainCamera game object through which they use the methods inside the scripts, that are attached to the MainCamera game object. The QuitButton uses the QuitGame method from the LevelEnter script, the LeftCameraRotationButton uses the LookLeft method from the CameraMovement script and the RightCameraRotationButton uses the LookRight method from the same CameraMovement script.

The MainCamera game object has a reference to the Player game object so it can access the transform component of that object and use it to set the camera position to follow the player.

The Player game object has attached to it the PlayerMovement script, which is responsible for the movement of the ball. The Player game object has a reference to the VirtualJoystickContainer game object, which has attached to it a script called VirtualJoystick. This is because the VirtualJoystick script is responsible for the movement of the joystick image on the screen depending on how the player drags the joystick, and the Player game object needs this reference in order to know how the joystick is moving so that it can translate that movement into moving that ball.

The Teleport and JumpButton game objects have attached to them a reference of the Player game object so that they can access the methods from the PlayerMovemet script. Teleport uses the Teleport method to reset the position of the ball in the 3D world and JumpButton uses the Jump method to apply a force to the ball so that it moves up in a jump motion.

The LevelManager game object has attached to it the LevelManager script, which is responsible for several things: the timer of the game, the pause menu (displaying the pause menu when the pause button is pressed and executing the functionality of the buttons inside the pause menu), setting the colour of the ball at the beginning of the game according to the colour choice saved in the PlayerPrefs, and loading the next scene when the time has elapsed.

The LevelManager game object has references to three game objects, these are: PauseMenu, Player and Timer. The PauseMenu game object is referenced so that the LevelManager can deactivate the pause menu in the beginning of the game and set it active whenever the pause button is pressed. The Player is used to set the colour of the ball in the beginning of the game through the private method ChangeColorPlayer(). The Timer is a text game object used to display the remaining time on the sceen.

There are 4 buttons that have a reference of the LevelManager game object. These are: ResumeButton, PauseButton, RestartButton and MainMenuButton. Both ResumeButton and PauseButton use the TogglePauseMenu method while RestartButton uses the RestartLevel method and MainMenuButton uses the ToMenu method.

The class diagram for the MainMenu scene contains three game objects that have a script attached to it, the rest of the game objects have a reference to another game object that has a script attached to it. The MainCamera game object has attached the LevelEnter script, that has two functions (changeScene() and QuitGame()). LevelOneButton and QuitButton have a reference to the MainCamera game object to use the two methods. LevelOneButton uses the changeScene() function to the Level_1 scene when the button

corresponding to that level is pressed in the level selection panel, and QuitButton uses the QuitGame() function to close the application when the quit button is pressed on a mobile device.

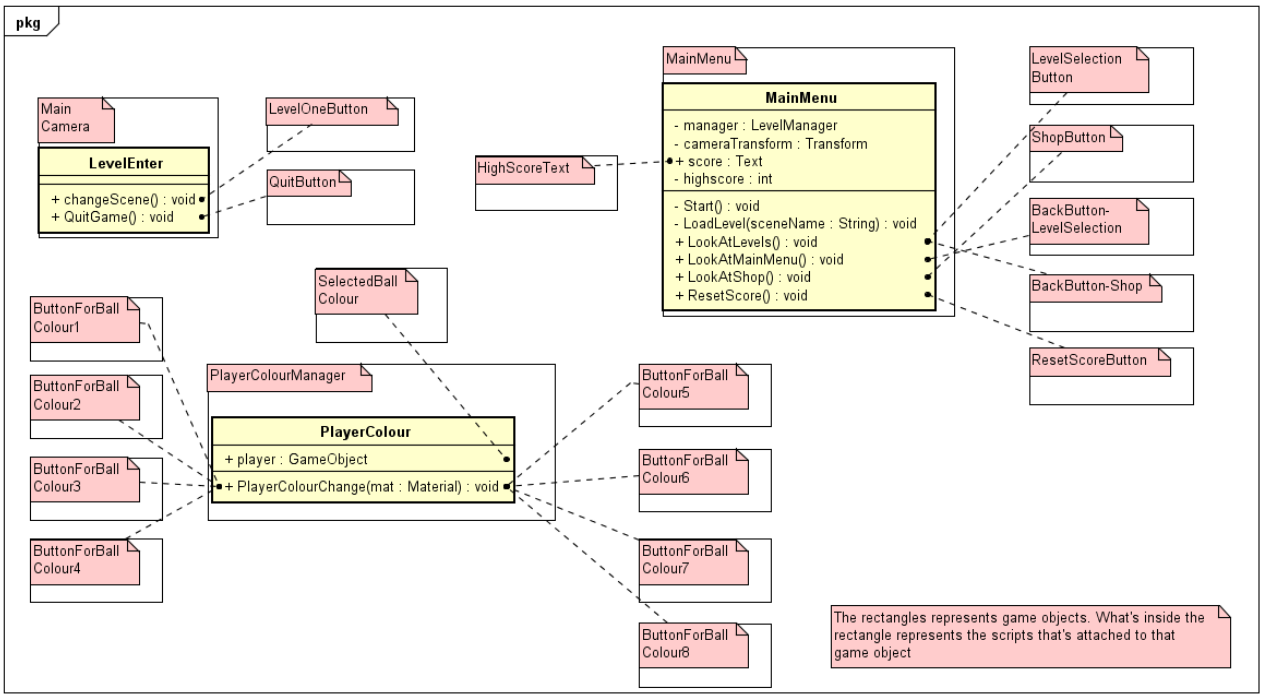


Figure 9 – Class diagram (Main menu scene)

The MainMenu game object has attached the MainMenu script and it has a reference to the HighScoreText game object in order to display the highs crore in the level selection panel. The MainMenu script is responsible for the navigation between panels by changing the camera angle and for displaying and resetting the high score in the level selection panel. As shown in the diagram above LevelSelectionButton and BackButton-Shop use the LookAtLevels method which changes the camera angle to look upwards. BackButton-LevelSelection uses the LookAtMainMenu method to change the camera angle to look at the main menu panel and ShopButton uses the LookAtShop method to change the camera angle to look at the shop panel. ResetScoreButton uses the ResetScore method to reset the score in the PlayerPrefs and on the screen, in the level selection panel.

The PlayerColourManager game object has attached to it the PlayerColour script which is responsible for changing the colour of the ball. This script is used inside the shop panel. The PlayerColourManager has a reference of the SelectedBallColour game object to set colour of the ball inside the shop panel so that the selected colour can be visible to the player. All 8 buttons for the ball colours have a reference to the PlayerColourManager and use the PlayerColourChange method from the PlayerColour script. Depending on which button has been pressed, the method will take the colour corresponding to that button as a parameter of type Material and inside the PlayerColourChange method the colour of the ball on the screen will be changed using that parameter and depending on what material name has been passed a corresponding number will be saved in the PlayerPrefs so that it can be later retrieved in the Level_1 scene to be set to the ball used inside the game.

The class diagram for the QuizGameScene contains one central game object that has attached several other game objects and it is attached to several other game objects. The HighScoreValue game object is referenced so that the high score can be set to the Text component and displayed on the screen. The ScoreTextAndValue

game object is referenced to display the current score of the player, and after each correct answer the updated increased score is displayed on the screen. The 5 question panels and the RoundOverPanel are referenced in order to deactivate the question that has been answered and activate the next question during the quiz game. The QuizGameManager script that is attached to the QuizGameManager

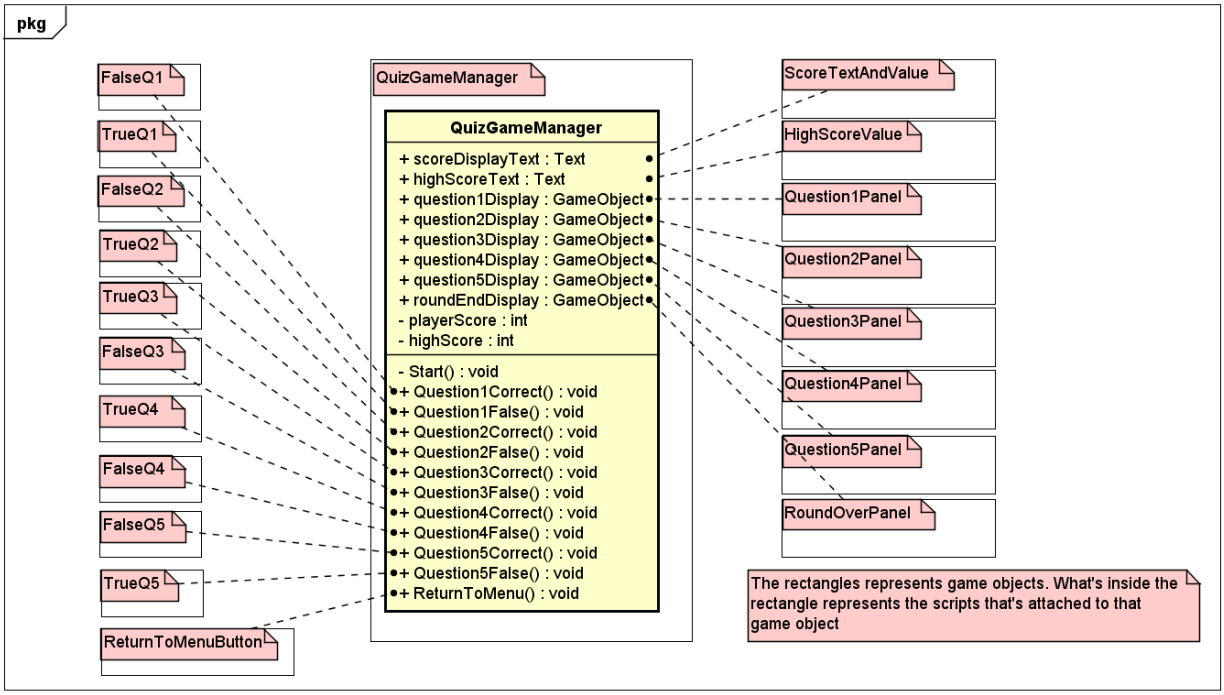


Figure 10 – Class diagram (Quiz game scene)

game object is used to transition from one question to another, to save the high score in the PlayerPrefs, to display the current score on the screen, and to load the MainMenu scene at the end of the quiz game.

Each of the True or False buttons used for the 5 questions are used to add the points to the current score and to display the current score on the screen if the answer to the question is correct, and to move to the next question. Depending on the question the assignment between the buttons and the methods might differ, as an example question one presents a false statement in the quiz game, if the player presses the false button for question one, then this means that the player considers the statement in question one as being false and in this case the player is right so that the false button for question one is associated with Question1Correct() method in the QuizGameManager script and the logic for that method is executed. Question three has a true statement in the quiz game, so if the player presses the true button the Question3Correct() method will be executed because the answer of the player is correct.

The methods that have Correct in the name of the method are adding the points to the current score, displaying the current score on the screen and move to the next question, while the methods that have False in the method name only move to the next question without any other change. The ReturnToMenuButton uses the ReturnToMenu method which saves the current score in the PlayerPrefs if the current score is greater than the high score, and loads the MainMenu scene.

4. Implementation

The Start() method in the LevelManager script is responsible for a few things. In the beginning of Level_1 the pause menu is active, so the first thig in the Start() method is to deactivate the pause menu, then the startTime variable takes the time at the beginning of the game. The colour preference is retrieved form the PlayerPrefs. The LoseTime coroutine is started and ChangeColorPlayer() method is called.

```
0 references
private void Start()
{
    pauseMenu.SetActive(false);
    startTime = Time.time;
    savedValue = PlayerPrefs.GetInt("ballColor");

    StartCoroutine("LoseTime");

    ChangeColorPlayer();
}
```

Figure 11 – Code snippet (Strat method)

```
0 references
IEnumerator LoseTime()
{
    while (true)
    {
        if (timeLeft > 60)
        {
            yield return new WaitForSeconds(5);
            timeLeft = timeLeft - 5;
        }
        else
        {
            yield return new WaitForSeconds(1);
            timeLeft--;
        }


    }
}
```

Figure 12 – Code snippet (LoseTime method)

The LoseTime() method of type IEnumerator is responsible for the timer of the game. If the value of the timeLeft variable is greater than 60 than the program waits for 5 seconds and decreases the timeLeft by 5 and this is to put a smaller pressure on the player if the timer is above 1 minute and if the value of the timeLeft variable is less than 60 then the program waits for 1 second and decreases timeLeft by 1.

This is the first section of the ChangeColorPlayer() method, the rest of the code is the same, the only difference is the name of the material and in total there are 8 if conditions for each material. The ChangeColorPlayer() method is responsible for changing the colour of the ball that the player is controlling.

```
1 reference
private void ChangeColorPlayer()
{
    if (savedValue == 6)
    {
        Material material1 = Resources.Load<Material>("Materials/mat_6");

        player.GetComponent<Renderer>().material = material1;
    }

   else if (savedValue == 1)
    {
        Material material1 = Resources.Load<Material>("Materials/Ground_Texture_Material");

        player.GetComponent<Renderer>().material = material1;
    }

    else if (savedValue == 2)
    {
        Material material1 = Resources.Load<Material>("Materials/House_color");

        player.GetComponent<Renderer>().material = material1;
    }
```

Figure 13 – Code snippet (ChangeColorPlayer method)

```
0 references
private void Update()
{
    timerText.text = ("" + timeLeft);

    if (timeLeft <= 0)
    {
        StopCoroutine("LoseTime");
        timerText.text = ("Time is up!");


        SceneManager.LoadScene("quizMenu");
    }
}
```

Figure 14 – Code snippet (Update method)

The Update() method is responsible with displaying the time left on the screen by assigning the value of the timeLeft variable to the Text component of the timerText game object. If the value of the timeLeft variable is less than or equal to 0 then the LoseTime coroutine is stopped, on the screen it appears the message “Time is up!” and the SceneManager is used to load the scene called “QuizMenu”.

```
public void TogglePauseMenu()
{
    //returns state of pause menu,
    //changing scale 0-1
    pauseMenu.SetActive(!pauseMenu.activeSelf);
    Time.timeScale = (pauseMenu.activeSelf) ? 0 : 1;

}
```

Figure 15 – Code snippet (TogglePauseMenu method)

The TogglePauseMenu() method is used when the pause button or the resume button in the pause menu is pressed. The way that it works is that the method finds out first what is the current state of the pause menu game object in order to deactivate it if it is active and to activate it if it is deactivated using the SetActive() method and the negation sign (!). The TogglePauseMenu() method is also responsible with stopping the time if the pause menu game object is active and to resume the time if the pause menu game object becomes inactive by using the conditional (ternary) operator.

The RestartLevel() method is used to restart the level when the restart button is pressed inside the pause menu. Is doing so by firstly resuming the time, by assigning the value 1 to the timeScale property of the Time class, and secondly by reloading the current scene.

```
0 references
public void RestartLevel()
{
    Time.timeScale = 1;
    SceneManager.LoadScene(SceneManager.GetActiveScene().name);
}
```

Figure 16 – Code snippet (RestartLevel method)

```
public void ToMenu()
{
    Time.timeScale = 1;
    SceneManager.LoadScene("MainMenu");
}
```

Figure 17 – Code snippet (ToMenu method)

The method ToMenu() is used, when the Main Menu button in the pause menu is pressed, to load the Main menu scene. The ToMenu() method is very similar to the previous method and the only difference between them is that the RestartLevel() method was loading the current scene, while the ToMenu() method is loading the “MainMenu” scene.

The Start() method in the QuizGameManager script is used to initialize the playerScore variable to 0, it gets the value stored in the PlayerPrefs for the high score and it assigns the value to the highScore variable and it also displays this value on the screen by assigning it to the Text component of the highScoreText game object.

```
void Start()
{
    playerScore = 0;

    highScore = PlayerPrefs.GetInt("score");
    highScoreText.text = "" + highScore;

}
```

Figure 18 – Code snippet (Start "2" method)

```
public void Question1Correct()
{
        playerScore = playerScore + 10;
        scoreDisplayText.text = "Score: " + playerScore;

        question1Display.SetActive(false);
        question2Display.SetActive(true);
}
```

Figure 19 – Code snippet (Question1Correct method)

The Question1Correct() method is one of the 5 methods that contain the word "Correct" in the name of the method. The Question1Correct() method is responsible for adding the points to the playerScore variable and display this added and updated score to screen. The method then deactivates the question1Display game object for the current question and activates the question2Display game object for the next question. This logic repeats for all the methods that contain the word "Correct" in the name of the method. The methods that contain the word "False" in the name of the method only have the logic for activating and deactivating the corresponding game objects.

The ReturnToMenu() method is used when the "Main Menu" button at the end of the quiz game is pressed. In the ReturnToMenu() method an if statement is used to compare the current score of the player with the high score, if the current score of the player is greater than the high score, then it stores the value of the playerScore variable into the highScore variable, it assigns the value of the new high score to the Text component of the highScoreText game object so it can be displayed on the screen and it saves the new high score to the PlayerPrefs. At the end of the method the "MainMenu" scene is loaded using the LoadScene() method from the SceneManager class.

```
public void ReturnToMenu()
{
    if (playerScore > highScore)
    {
        highScore = playerScore;
        highScoreText.text = highScore.ToString();

        PlayerPrefs.SetInt("score", highScore);
    }

    SceneManager.LoadScene("MainMenu");
}
```

Figure 20 – Code snippet (ReturnToMenu method)

5. Test

The tests that have been performed on the project consist of running the game in the Unity program and building the project and running it on a phone. The tests have been performed only by the programmes that developed the game.

Tests have been performed on an ongoing basis from the moment the implementation phase started to test every new functionality that was added to the project and also tests have been performed to verify that the functionality that was previously added did not change its behaviour after the new functionality has been added.

The camera transition in the main menu has been tested to ensure a proper behaviour. The result of the test is that the camera angle had to be adjusted to rotate each time at the same angle. The transition between the three panels in the main menu, which are the main menu panel, the shop panel and the level selection panel, was not functioning properly. When the camera would transition from the main menu panel to the level selection panel it would firstly show the level selection panel correctly centred but when transitioning back to the main menu panel it would display it either half way up or half way down, changing its behaviour each time, the same would happen also for the shop panel regardless of the panel that was selected first. After adjustment the camera is transitioning each time correctly and displaying the panels centred.

Tests have been performed for the quiz game. The results of the tests are that the quiz game would display the questions overlapped, creating an indistinguishable text, when the answer buttons would be pressed in a certain unknown sequence. The malfunction was solved by properly assigning the methods to the buttons inside Unity.

The score and high score functionalities have been tested to ensure proper behaviour. The score has been tested to ensure that the points would be added to the score and displayed on the screen when the answer to a question would be correct. The result of the test was positive, and the behaviour was proper. The high score has been tested to make sure that at the end of the quiz session, if the current score was higher than the high score then the high score would be updated and saved in the PlayerPrefs. The result of this test was positive, the new high score would be correctly saved and displayed on the screen when returning to the main menu. A test has been performed to verify that if the current score was lower than the high score it would not save the current score and the high score would be kept. The result of this test was positive and the current score that was lower than the high score was not saved.

There have been performed tests also to verify the technical functionalities of the game such as the countdown function to see if the timer misbehaves or not.

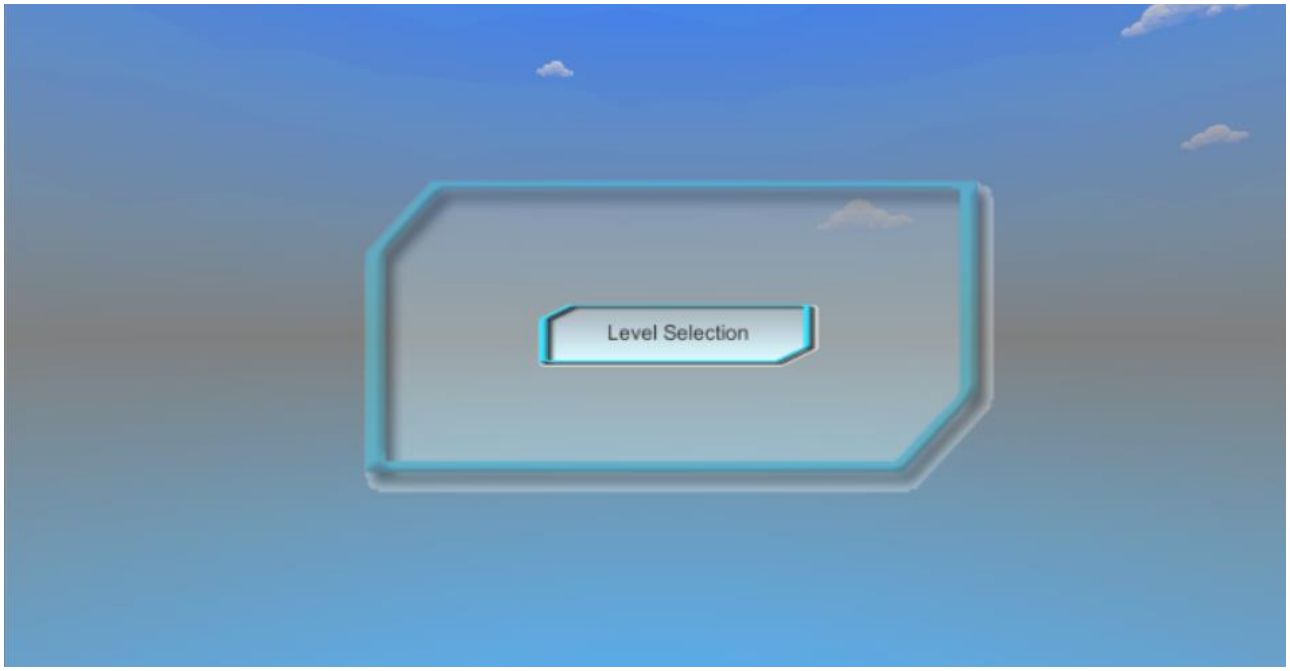
There have been performed tests on the mechanics of the game that include the movement with the on-screen joystick, the jump functionality with the on-screen jump button, and the changing of the camera angle using the two on screen buttons as well as the teleport button which resets the position of the ball.

6. Results

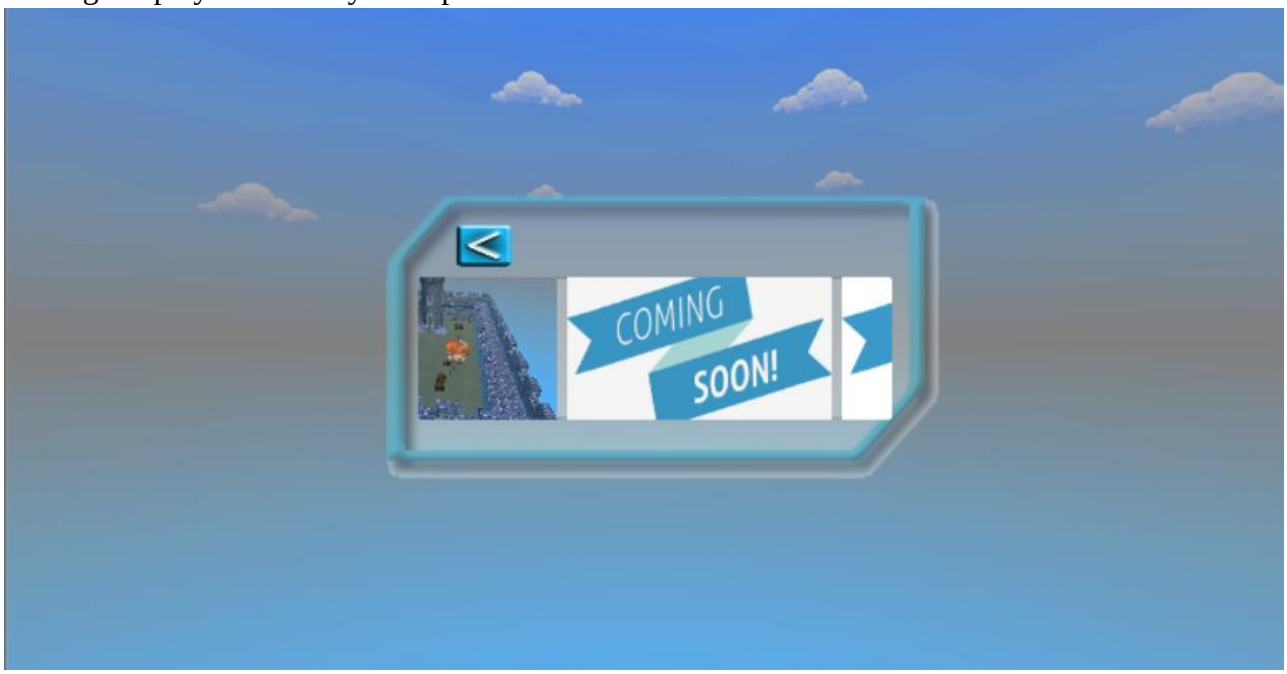
After the 3 weeks project period the result of the project is a game made in Unity for the mobile platform (phones/tablets) specifically Android that has as objective improving the performance of the memorization process using the memory palace technique as the core mechanic around the game.

The game itself has one available level.

The first scene of the game as shown in the picture below, is one in which the player can press a button with the title ”Level Selection” that will transition to the next scene for the level selection.



The next scene presented in the picture below offers the player the option to choose from a number of levels. As seen there are two levels that are currently unimplemented labelled as “Coming soon” leaving the player with only one option to choose from.



When the player selects the level on the screen there will be prompted the message “You have 60 to memorize the place” informing the player that the time for exploring and memorizing the environment is 60 seconds. In the picture below, we can see the visual representation of this process.

When the player presses the button in the top right corner corresponding to the pause functionality the pause menu will be displayed on the screen. The Pause menu displayed on the screen offers the player three options to choose from, each button is labelled with a text that intuitively suggests its functionality. The button labelled with the text “Main Menu” will quit the level and will return the player to the Main Menu screen as shown in the first picture of this section. The button labelled with the text “Restart” will restart the level by restarting the counter and bringing the player to the initial position in the scene. The button with the label “Resume” will close the Pause menu and will resume the gameplay by continuing the countdown from the moment that the player pressed the pause button and the position of the player in the scene from the moment that the player pressed the pause button will not be changed. In the picture bellow is a visual representation of the Pause menu.

7. Conclusion

The result of the project is a game that fulfils the most important requirements. It is made in Unity 3D engine for the mobile platform (phone/tablets) and its main core mechanic is the memory palace technique.

8. List of References

Project Report Guidelines – Studienet course website under IT-SEP4C-S18 course page under Session Material uploaded by Kasper Knop Rasmussen (https://studienet.via.dk/Class/IT-SEP4C-S18/Session%20Material/Project%20Report%20Guidelines.pdf)

Project Report Template – Studienet course website under IT-SEP4C-S18 course page under Session Material uploaded by Kasper Knop Rasmussen (https://studienet.via.dk/Class/IT-SEP4C-S18/_layouts/15/WopiFrame2.aspx?sourcedoc=/Class/IT-SEP4C-S18/Session%20Material/Project%20Report%20Template.docx&action=default)

Class Diagram – Wikipedia (“https://en.wikipedia.org/wiki/Class_diagram “)dwdd

UML Class Diagram Tutorial – Lucidchart, YouTube (”https://www.youtube.com/watch?v=UI6lqHOVHic”)

9. Appendices

9.1. Appendix 1 – User Guide

The user guide will be presented as a demonstration video uploaded on YouTube. The link to the video can be found in a text file uploaded together in the same folder as the project and it can also be accessed by following this link (https://www.youtube.com/watch?v=4nGv_2J0jyU).

9.2. Appendix 2 – Selected Java code

All the selected java code will be uploaded in a PDF file containing all the pictures presented in the report.

The project will be uploaded as a link to the source code in a text file in the same folder together with all the other material.

9.3. Appendix 3 – Analysis documentation

9.3.1. Use Case diagram

All the Use case diagram pictures will be uploaded in a PDF file in the same folder together with the rest of the materials, containing all the pictures presented in the report.

9.3.2. Use Case descriptions

All the Use case descriptions pictures will be uploaded in a PDF file in the same folder together with the rest of the materials, containing all the pictures presented in the report.

9.3.3. Activity diagrams

All the Activity diagrams pictures will be uploaded in a PDF file in the same folder together with the rest of the material, containing all the pictures presented in the report.

9.4. Appendix 4 – Design documentation

9.4.1. Design class diagram

All the Class diagrams pictures will be uploaded in a PDF file in the same folder together with the rest of the material, containing all the pictures presented in the report.

Also all the astah files for the class diagrams will be uploaded in the same folder together with the rest of the materials.