Folkecenter Quiz Game

Internship Project Report

Supervisor:

Daniele Pagani

Trainee:

Ionel-Cristinel Putinica

Number of characters - [14371]

Table of Contents

Abstract

Many visitors arriving at Folkecenter have little to knowledge about renewable energy and the center itself. It is of great importance that they have some base knowledge about renewable energy and Folkecenter's whereabouts before receiving a guided tour, thus they will have a better understanding of the tour when they take part into it.

The game has been fully designed in order to give the end-user a small introduction into the world of renewable energy and Folkecenter, in an interactive and eye-catching way. The game has been designed and implemented by a single person, with the main focus of keeping it simple and straight-forward.

1 Introduction

Games are a ubiquitous part of life in our culture, and experts suggest they will become even more deeply embedded in the coming years. Games help people develop a disposition toward collaboration, problem-solving, communication, experimentation, and exploration of identities, all attributes that promote success in a rapidly-changing, information-based culture (2011 Horizon Report).

Games seem to be particularly successful in helping people develop problem-solving and decision-making skills and encouraging innovation. Without a doubt, gaming prompts people to do a tremendous amount of research and inspires participants to spend an extraordinary amount of time on task.

The developed product aims to educate the end-user, being the result of the task to create a product that gives a person information in a fun and interactive manner.

2 Requirements

2.1 Functional Requirements

ID	User Story Description	Priority
1	As a user, I want to be able to install and run the game on my computer.	High
2	As a user, I want to be able to test my knowledge on renewable energy and Nordic Folkecenter and gain new knowledge about those subjects.	High
3	As a user, I should be able to have access to a straight-forward user interface.	Medium
4	As a user, I want to game to feature background audio/music.	Medium
5	As a user, I want the game to give a sense of reward	Low

Table 1: User Stories

ID	Requirement	Priority
1	The user should be able to install and run the application on his personal computer.	High
2	The user should be able to receive information in a straight-forward UI.	High
3	The user should be able to interact with the UI.	High
4	The user should be prompted with background audio when running the game.	Medium
5	The user should be able to run the game without bothering with launch settings.	Low

Table 2: Functional Requirements

2.2 Non-Functional Requirements

ID	Requirement
1	The game should be programmed in Unity using C#.
2	The game should be easy to modify and to expand in the future.
3	The game should be designed for a single user experience.

Table 3: Non-Functional Requirements

3 Analysis

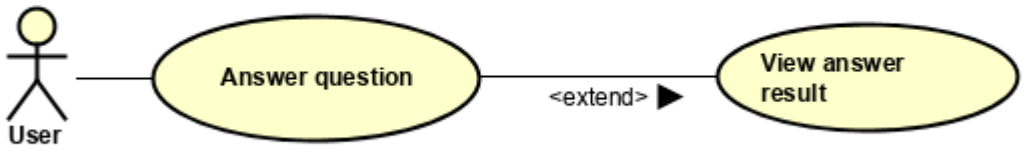

Figure 1: Use Case Diagram

3.1 Use Case Diagram

The above Use Case Diagram represents the operations that the user can perform in the system, and it is also a showcase of the simplicity of the game, since the only actions the user can perform in the game are answering the questions and then viewing the result of the answer.

3.2 Use Case Descriptions

The following Use Case descriptions explain how the functionality of the game was envisioned and described thoroughly.

Use Case	Answer Question
Actor	User
Pre-Condition	The user needs to be on a question window
Post-Condition	The animation engine will run, displaying if the answer was wrong or correct, then the game will clear out the scene and display a new question
Base sequence	1. User clicks either the True or False button. 2. The answer is analyzed. 3. The animation engine is called and displays the correct answer. 4. The system refreshes the scene with a new question.
Exception sequence	User stops the game from running.

Table 4: Use Case Description – Answer Question

Use Case	View Answer Result
Actor	User
Pre-Condition	The user needs to answer a question first
Post-Condition	The animation engine will run, displaying if the answer was wrong or correct.
Base sequence	1. User clicks either the True or False button. 2. The answer is analyzed. 3. The animation engine is called and displays the correct answer.
Exception sequence	User stops the game from running.

Table 5: Use Case Description – View Answer Result

4 Design

The purpose of this part is to define the architecture, technologies, and UI choosing of the game.

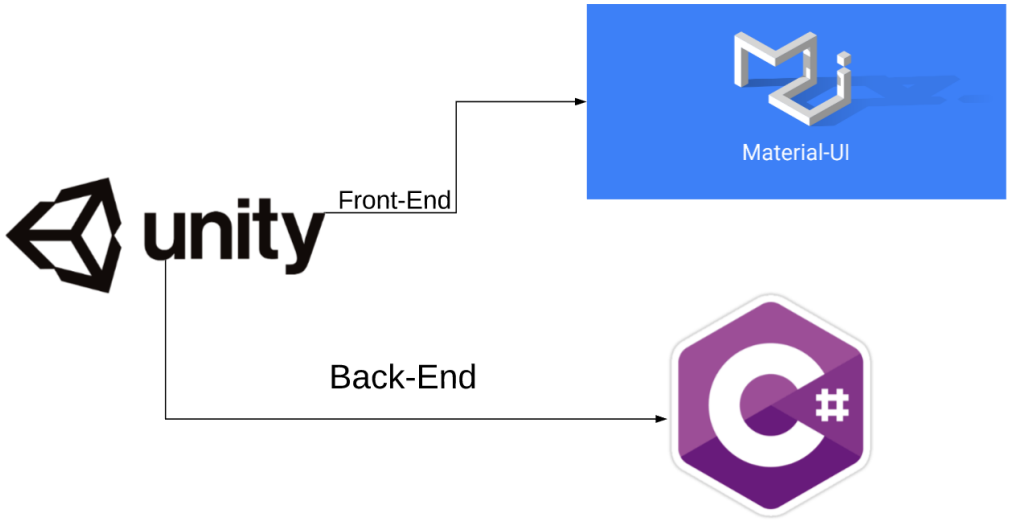

Figure 2: Design Diagram

The above diagram displays the Frameworks and different technologies and programming languages that were used to create the game, its User Interface and the logic (back-end) of the game.

4.1 Game engine – Unity

Choosing a game engine for the Folkecenter Quiz Game wasn't a hard task. The game had to be a simple one, and also cross-platform, so it can run on different computers and operating system, thus, Unity came as an obvious choice. Unity is excellent for cross-platform development, being a platform that made it easy from the very beginning to run and compile a single script on multiple platforms. It is an effective tool for rendering 2D scenes – which is what the Folkecenter Game is after all, with a rich asset store and detailed documentation.

4.2 User Interface – MaterialUI

The Unity Game Engine offers multiple choices for a user interface, but for this particular project, a standalone UI kit was chosen. MaterialUI is a UI kit for Unity that follows Google's official Material Design guidelines. The version that was used on the game was the most recent free version of MaterialUI, which can be accesed with the following github link: https://github.com/InvexGames/MaterialUI/releases.

4.3 Scripts

The language that's used in Unity is C#, which is a object-oriented scripting language. A script must be attached to a GameObject in the scene in order to be called by Unity. Scripting tells our GameObjects how to behave; it's the scripts and component attached to the GameObjects, and how they interact with each other that creates the gameplay for the user.

4.4 System Sequence Diagram

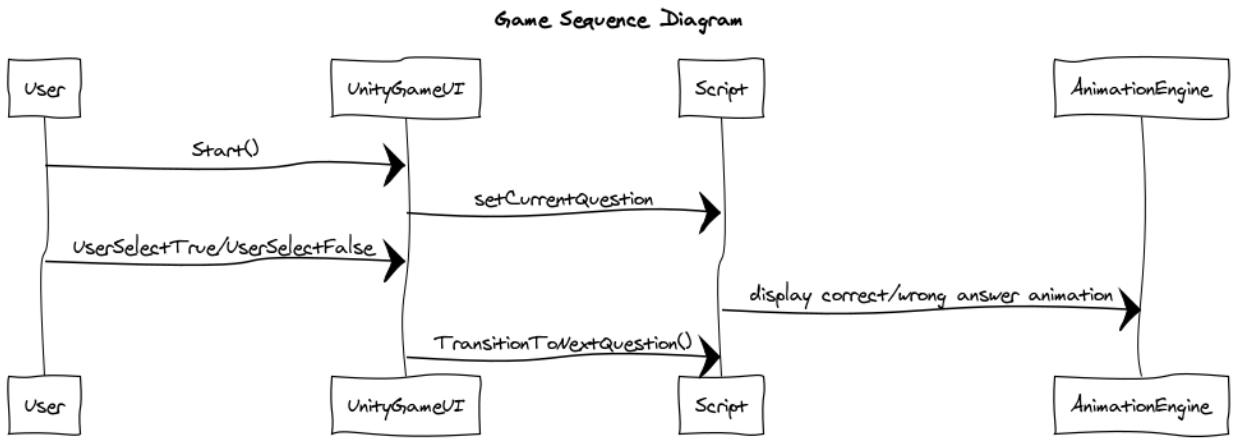

Figure 3: System Sequence Diagram

The sequence diagram above showcases all the different interactions that happen in the game, between all the actors and components, in this game, the actor being the user of the game, and components such as the User Interface of the Unity game, the scripts and the animation engine. The first step is the user starting up the game. Then, the set current question method is called, which displays a random question on the User Interface, from the questions array. After that, the user can select either answering the question with true or false, and, depending on his answer,

the animation engine will display the according animation, letting the user know weather his answer was correct or wrong. After the animation is displayed, a method is called, which has the purpose of transitioning and updating the UI with a new question from the questions array.

5 Implementation

5.1 User Interface

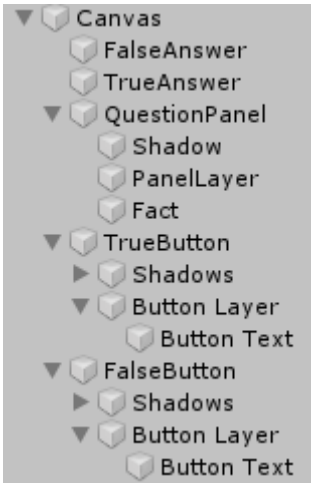

Figure 4: User Interface Contents

The user interface is a simple 2D Unity Scene Canvas, with 3 main components: a panel, where the text for the question is stored, and two buttons: the true and the false button.

All the elements of the User Interface have been created with the MaterialUI tool kit, and, in addition to this, the elements and the relations between them have all been made responsive, thus, in the case of a screen resizing, they will resize as well, so this way overlapping of elements and similar events are avoided.

All elements have been customized for a more pleasant and eye-catching experience, such as the font type and size, the background colors of the elements, and for this part, no coding was required, since the customization of the element proprieties can be done directly in the Unity interface.

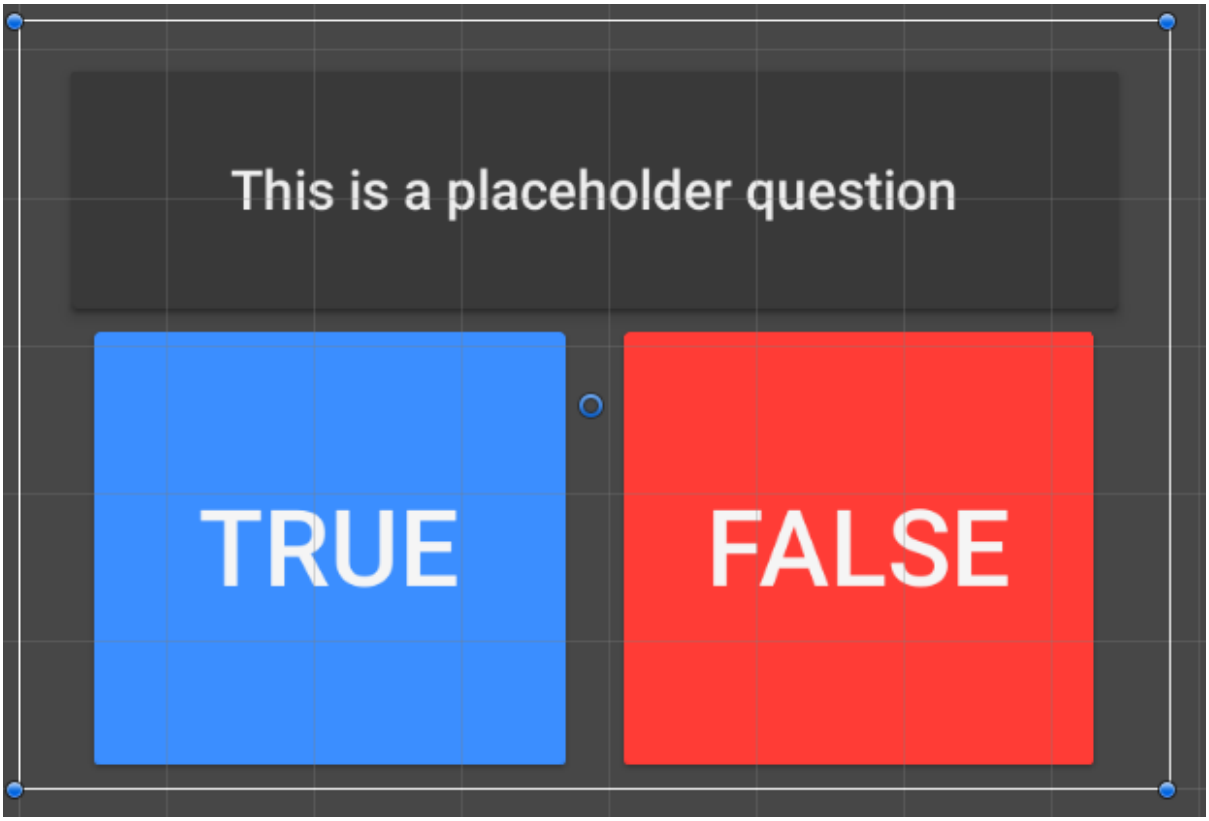

Figure 6: User Interface Template

5.2 Scripts

The logic of the game has been written in C#. There are only two classes that have been written for this game, the Question class and a GameManager class.

Filename Assembly-CSharp.dll

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
[System.Serializable]
public class Question
{
    //the fact string is going to store the actual
question that is displayed to the user
    public string fact;
    //the boolean isTrue stores if the "fact" is
true or false
    public bool isTrue;

}
```

Figure 7:Question Class

The Question class is used to define a question object, specifying that it is going to be a string object, and that it is also going to hold a boolean value.

Most of the logic of the game is found in the GameManager class. Here, multiple methods are defined, that serve different purposes: a method that randomly sets a question (which is retrieved from the Questions array) and replaces the place holder text in the user interface with the according string from the array.

```
void setCurrentQuestion()
{
    int randomQuestionIndex = Random.Range(0, unansweredQuestions.Count);
    currentQuestion = unansweredQuestions[randomQuestionIndex];
    factText.text = currentQuestion.fact;

    if (currentQuestion.isTrue)
    {
        trueAnswerText.text = "CORRECT";
        falseAnswerText.text = "WRONG";
    }
    else
    {
        trueAnswerText.text = "WRONG";
        falseAnswerText.text = "CORRECT";
    }
    }
```

Figure 8:setCurrentQuestion() method

Other methods are present in the game manager, such as the TransitionToNextQuestion() method, which has the purpose of refreshing the current question, taking into consideration the questions that have been already answered. This method is called in the UserSelectTrue() and UserSelectFalse() methods. In these two methods ,the animation engine is also called.

To be able to be accesed, variables in the classes have been serialized, thus they can be accesed from and by the Unity engine.

```
IEnumerator TransitionToNextQuestion()
{
    unansweredQuestions.Remove(currentQuestion);
    yield return new WaitForSeconds(timeBetweenQuestions);
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}
```

Figure 9:TransitionToNextQuestion() method

```
public void UserSelectTrue()
{

    animator.SetTrigger("True");
    if (currentQuestion.isTrue)
    {
        Debug.Log("CORRECT!");
    }
    else
    {
        Debug.Log("WRONG!");
    }
    StartCoroutine(TransitionToNextQuestion());
}
```

Figure 10:userSelectTrue() method

5.3 Animations

The animations have been created using the Unity animation tool. “Wrong” and “Correct” text objects have been created and placed behind the “True” and “False” buttons. The answer of the user is analyzed in the userSelectTrue() or userSelectFalse() methods, and, depending on the answer, the pre-recorded animations will be rolled out, and the animation will display the user if his answer to the question was correct or wrong.

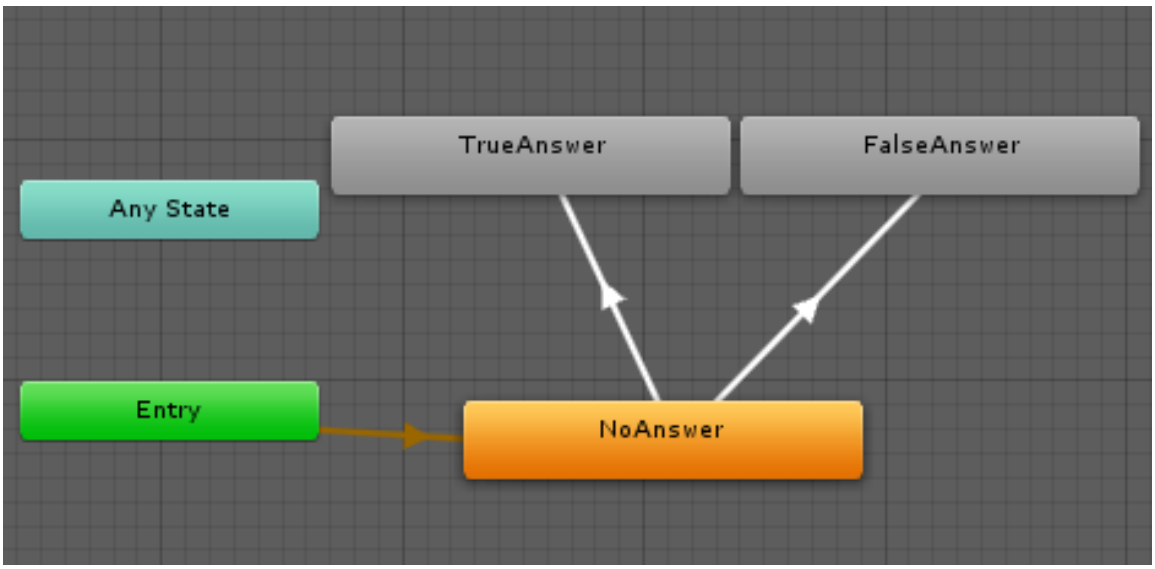

Figure 11:Animation logic diagram

5.4 Background music

For a more pleasant experience, background music has also been added, thus sound has been linked to the camera of the game, so whenever the scene runs a 2D (uniform) sound will play in the background of the game.

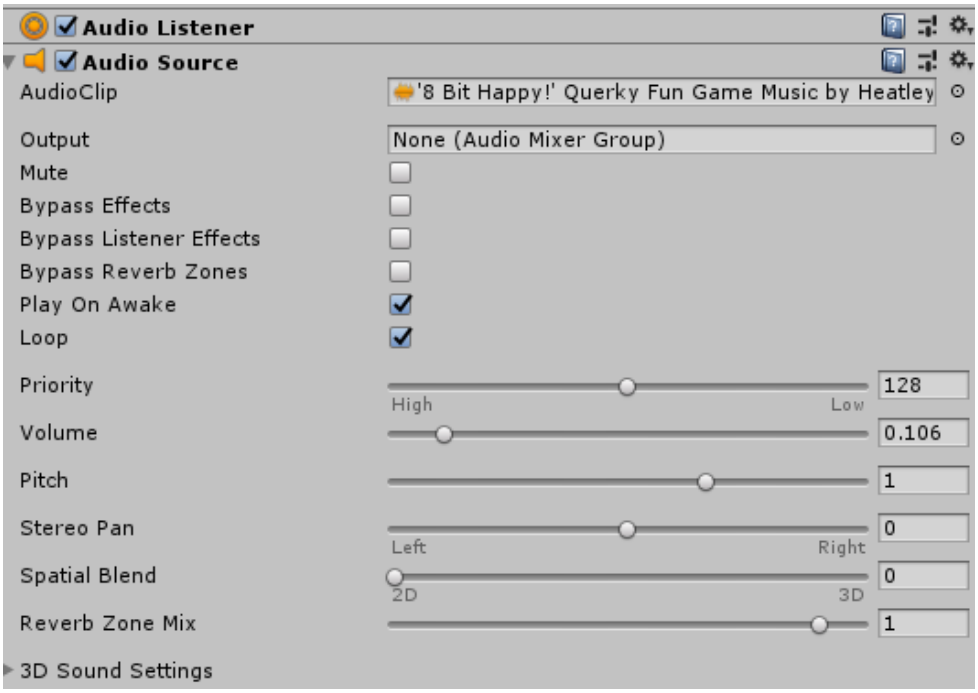

Figure 12:Music component

5.5 Building the game

The game was built and an installation file was created using the vanilla Unity game builder. A custom icon was created for the game, and, because of its scale, some pre-defined settings have been created for the user – windowed mode is forced by default, a 700px width and 550px height are set, and support for Mac Retina displays has also been enabled.

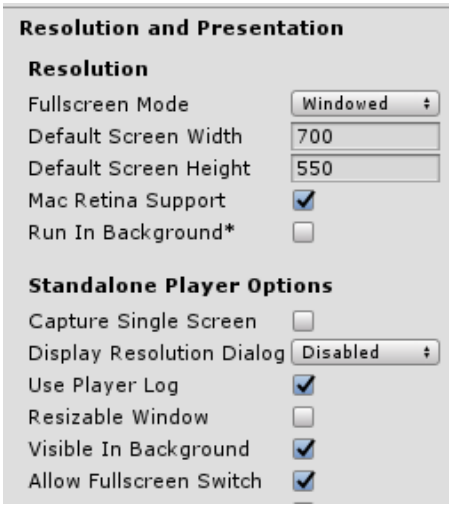

Figure 13:Predefined player settings component

6 Testing

Black Box Testing

The tests check if the game works correctly from the user's perspective. Does each function and feature do what it promises and is the User Interface easy to use and understand?

The general goal for the game test is to check weather the use cases work or not.

Description	Expected Result	Result
The user wants to install and run the game on a Windows running computer	After installation process is complete, the user can run the current build of the game and is sent to the main game scene	Success, works as expected
The user wants to install and run the game on a Linux running computer	After installation process is complete, the user can run the current build of the game and is sent to the main game scene	Success, works as expected
The user wants to install and run the game on a Mac running computer	After installation process is complete, the user can run the current build of the game and is sent to the main game scene	Success, works as expected
The user wants to answer one of the questions and see if his answer was correct or wrong	The user can interact with either the true/false buttons, and, after one is clicked, an animation will pop-up, displaying the user if his answer was correct or wrong	Success, works as expected
The user wants to be able to answer a multitude of questions without them repeating constantly	After answering a question, the game should display a completely new question, chosen randomly from the pre-defined list of questions	Success, works as expected
The user is able to close the game at any given moment	The user can click the "close" button in the top-right corner and the game will shut-down without displaying any error messages	Success, works as expected
The user is able to hear the background music	The user runs the game and the music runs in the background	Partial success, the music is playing, but after answering a question, the scene resets, so the music also resets

Table 6: Test Scenarios – Black Box Testing

7 Results, Discussion, Conclusions and Project Future

The “Folkecenter Quiz Game” project was developed to meet some simple and vague requirements, and developed in a short period of time by a single person. The progress of the development process can be followed on Github, since the project followed the version control protocols (Github link : https://github.com/IonutPutinica/FolkecenterGame) . The way the project was developed, it has left way for future improvements and new features. Some possible changes and improvements could be turning the game into a 4 possible answers game, rather than a true/false game, and also include a lives/score system, to make it feel more challenging.

Ultimately, the game is a small success, and paths the way for more small projects of such kind that can promote Folkecenter, along with the concept of renewable energy.

8 Sources of information

(2019, Oct 14th)
https://citl.indiana.edu/teaching-resources/teaching-strategies/games-learning/

(2019, Oct 15th)
https://unity3d.com/learning-c-sharp-in-unity-for-beginners