Interactive Engineering, A Gaming Approach to Increase the Motivation for Engineering Studying

Christian O. Pérez Pérez
Master in Engineering in Computer Engineering
Advisor: Nelliud Torres, DBA
Electrical and Computer Engineering and Computer Science Department
Universidad Politécnica de Puerto Rico

Abstract *— This project focuses on providing a solution for the increase in demand for engineering professions. It will present a methodology that could be used for implementing a potential solution in the form of a game. This article focuses on the programming design aspect of building such a solution. It will also discuss a prototype built in Unity Engine and C# using the design mentioned in this paper. This paper briefly discusses other aspects crucial for this project's completion, such as animation and presentation. Since only a prototype was built, it concludes the discussion on future works for this project to improve the game graphics and expand topics and the game capabilities aspects of the presented solution.*

Key Terms *— Engineering, Gaming, Interactive, Unity.*

INTRODUCTION

In recent years, there has been a high demand decline in engineering as a field of study [1]. This demand has been increasing and is expected to continue rising. Moreover, there needs to be more interest in studying Science, Technology, Engineering, and Math (STEM) related fields [2][3]. This project targets a potential solution to increase the engineering field's interest. The solution involves a prototype game-like experience that will aim to increase interest. This article will present a class diagram that will define the game's overall structure, followed by an explanation of the class diagram.

BACKGROUND

There needs to be an increase of 15% in engineering professionals to meet the demand [1]. There are several ways to tackle this problem. For example, there is an underrepresentation of Women, Black, and Hispanic workers in the STEM fields [1].

Problem

In addition to the increased need for engineering professionals, there has been a decreased interest in Engineering Studies since the 1980s [2]. According to Forbes [4], there was an increased interest in video games in the overall population during the pandemic. Forbes states that 93% of people under 18 years old play video games regularly. This work proposes a solution to this need of engineering professionals by creating a game that will attract high school students to the engineering fields.

METHODOLOGY

The methodology will focus on a general structure for building a cross-platform design (Android and IOS) 2D game oriented to the engineering fields. The solution proposed is implemented for Unity Engine. Unity is a 2D/3D cross-platform game engine (Android, iOS, Windows, macOS, Linux, PS4, Xbox One, HTML5, and more) developed by Unity Technologies. Games can be programmed using C# programing language and structured the game in different scenes [5]. The game will be structured in Electrical, Chemical, and Mechanical Engineering sections, and each section will have levels that add additional topics and difficulty.

Class Diagram

A class diagram for the implemented prototype game can be observed in Figure 1. The class diagram is aimed to be implemented using Unity Engines features.

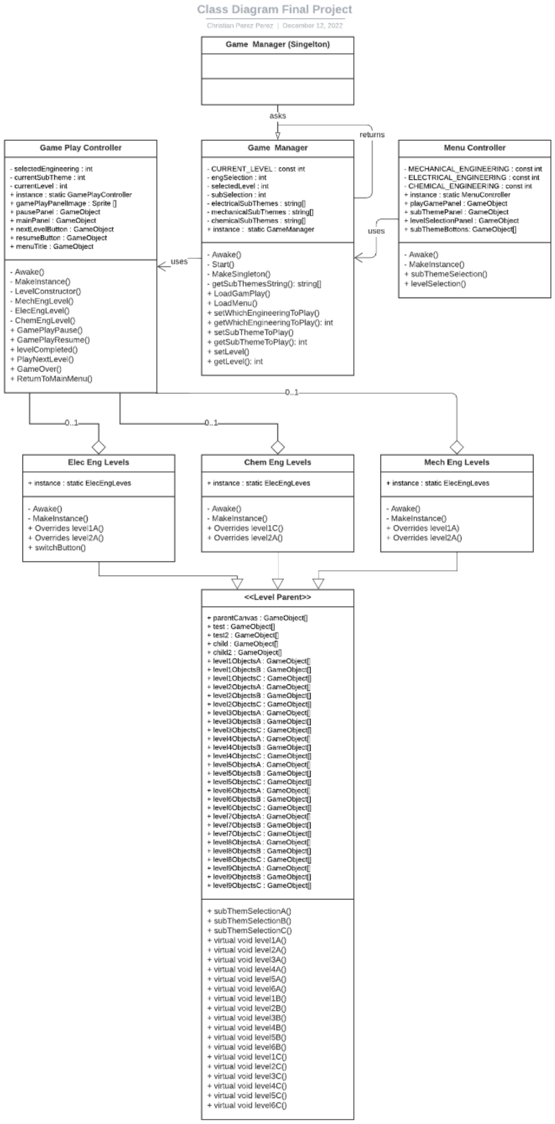

Class Diagram Final Project
Christian Perez Perez | December 12, 2022
Game Manager (Singelton)
asks
returns
Game Play Controller
- selectedEngineering : int
- currentSubTheme : int
- currentLevel : int
+ instance : static GamePlayController
+ gamePlayPanelImage : Sprite []
+ pausePanel : GameObject
+ mainPanel : GameObject
+ nextLevelButton : GameObject
+ resumeButton : GameObject
+ menuTitle : GameObject
- Awake()
- MakeInstance()
- LevelConstructor()
- MechEngLevel()
- ElecEngLevel()
- ChemEngLevel()
+ GamePlayPause()
+ GamePlayResume()
+ levelCompleted()
+ PlayNextLevel()
+ GameOver()
+ ReturnToMainMenu()
uses
Game Manager
- CURRENT_LEVEL : const int
- engSelection : int
- selectedLevel : int
- subSelection : int
- electricalSubThemes : string[]
- mechanicalSubThemes : string[]
- chemicalSubThemes : string[]
+ instance : static GameManager
- Awake()
- Start()
- MakeSingleton()
- getSubThemesString(): string[]
+ LoadGamPlay()
+ LoadMenu()
+ setWhichEngineeringToPlay()
+ getWhichEngineeringToPlay(): int
+ setSubThemeToPlay()
+ getSubThemeToPlay(): int
+ setLevel()
+ getLevel(): int
uses
Menu Controller
- MECHANICAL_ENGINEERING : const int
- ELECTRICAL_ENGINEERING : const int
- CHEMICAL_ENGINEERING : const int
+ instance : static MenuController
+ playGamePanel : GameObject
+ subThemePanel : GameObject
+ levelSelectionPanel : GameObject
+ subThemeBottons: GameObject[]
- Awake()
- MakeInstance()
+ subThemeSelection()
+ levelSelection()
0..1
0..1
0..1
Elec Eng Levels
+ instance : static ElecEngLeves
- Awake()
- MakeInstance()
+ Overrides level1A()
+ Overrides level2A()
+ switchButton()
Chem Eng Levels
+ instance : static ElecEngLeves
- Awake()
- MakeInstance()
+ Overrides level1C()
+ Overrides level2A()
Mech Eng Levels
+ instance : static ElecEngLeves
- Awake()
- MakeInstance()
+ Overrides level1A)
+ Overrides level2A()
<<Level Parent>>
+ parentCanvas : GameObject[]
+ test : GameObject[]
+ test2 : GameObject[]
+ child : GameObject[]
+ child2 : GameObject[]
+ level1ObjectsA : GameObject[]
+ level1ObjectsB : GameObject[]
+ level1ObjectsC : GameObject[]
+ level2ObjectsA : GameObject[]
+ level2ObjectsB : GameObject[]
+ level2ObjectsC : GameObject[]
+ level3ObjectsA : GameObject[]
+ level3ObjectsB : GameObject[]
+ level3ObjectsC : GameObject[]
+ level4ObjectsA : GameObject[]
+ level4ObjectsB : GameObject[]
+ level4ObjectsC : GameObject[]
+ level5ObjectsA : GameObject[]
+ level5ObjectsB : GameObject[]
+ level5ObjectsC : GameObject[]
+ level6ObjectsA : GameObject[]
+ level6ObjectsB : GameObject[]
+ level6ObjectsC : GameObject[]
+ level7ObjectsA : GameObject[]
+ level7ObjectsB : GameObject[]
+ level7ObjectsC : GameObject[]
+ level8ObjectsA : GameObject[]
+ level8ObjectsB : GameObject[]
+ level8ObjectsC : GameObject[]
+ level9ObjectsA : GameObject[]
+ level9ObjectsB : GameObject[]
+ level9ObjectsC : GameObject[]
+ subThemSelectionA()
+ subThemSelectionB()
+ subThemSelectionC()
+ virtual void level1A()
+ virtual void level2A()
+ virtual void level3A()
+ virtual void level4A()
+ virtual void level5A()
+ virtual void level6A()
+ virtual void level1B()
+ virtual void level2B()
+ virtual void level3B()
+ virtual void level4B()
+ virtual void level5B()
+ virtual void level6B()
+ virtual void level1C()
+ virtual void level2C()
+ virtual void level3C()
+ virtual void level4C()
+ virtual void level5C()
+ virtual void level6C()

Figure1
Class Diagram

The class diagram consists of eight classes. The idea behind the classes is as follows:

- Game Manager- The purpose of this class is to act as a manager across all scenes in the game. For this reason, a singleton design pattern was used to preserve the instance across all scenes. This class will communicate with the Game Play controller and the Menu Controller.
- Menu controller- The primary purpose is to control the game's main menu. It will have the function for the button click. It will communicate to the Game Manager the player's choice selection.
- Game Play Controller- The primary purpose is to control the game's Game Play. It will have the function for the button click, game pause, game over, and stage change. It will communicate to the Game Manager, among other things, if a level was completed. It will structure the game according to the level structure-related classes.
- Level structure-related classes –Parent Classes and 3 Child classes. Their purpose is as follows:

a. Level Parent – This Abstraction class works as a blueprint for the overall structure of the Electrical, Mechanical, and Chemical Engineering classes.

b. Elec Eng Levels- This contains the structure of the levels per section for the Engineering related gameplay.

c. Chem Eng Levels- This contains the structure of the levels per section for the Chemical related gameplay.

d. Mech Eng Levels- It contains the structure of the levels per section for the Mechanical Engineering-related gameplay.

RESULTS AND DISCUSSION

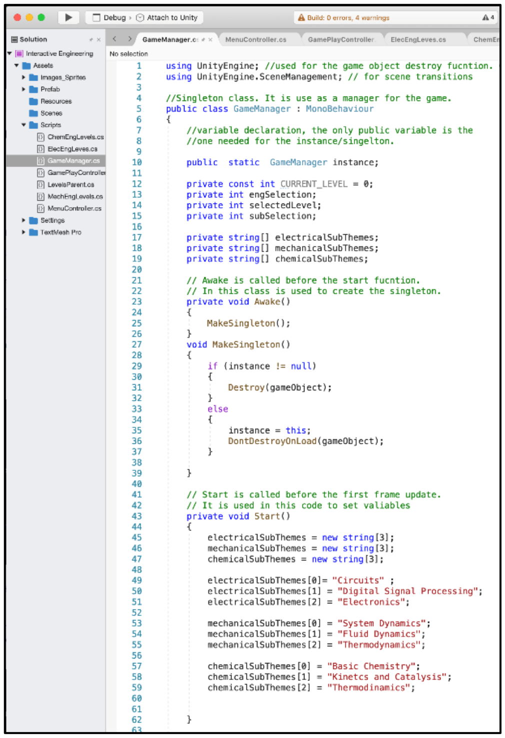
The class diagram was implemented into a prototype using a C# on Unity version 2021.3.8f. In Figures 2 and 3, you can see the code for the Game Manager singleton class implementation.

The Game Manager class is the heart of the code [5]. The following can be said about this class:

- The Unity Engine library is used to destroy the game object. It is crucial to create the singleton.
- The Unity Engine Scene Management library is used to transition between scenes.
- Awake function is called before the start function. In this class, it is used to create the singleton.
- Start function is called before the first frame update. It is used in this code to set variables.
- The gameplay manager calls the getters functions to specify the user selection: Engineering field, Sub-theme, and specific

level.

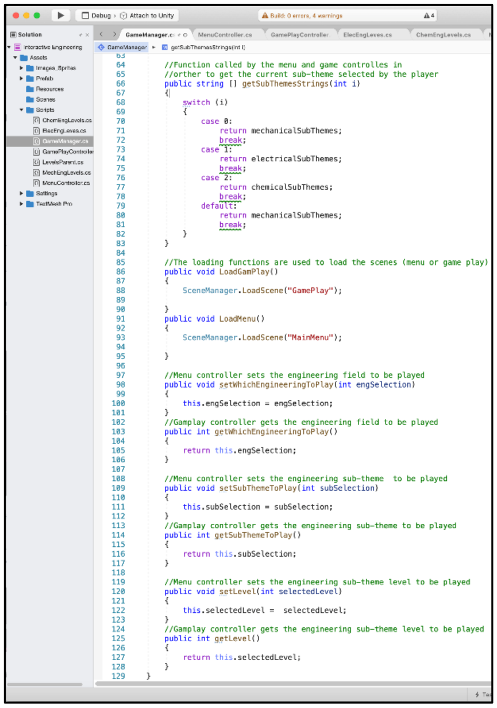
- The loading functions are called to change between scenes.



```
using UnityEngine; //used for the game object destroy fucntion.
using UnityEngine.SceneManagement; // for scene transitions

//Singleton class. It is use as a manager for the game.
public class GameManager : MonoBehaviour
{
    //variable declaration, the only public variable is the
    //one needed for the instance/singelton.

    public  static  GameManager instance;

    private const int CURRENT_LEVEL = 0;
    private int engSelection;
    private int selectedLevel;
    private int subSelection;

    private string[] electricalSubThemes;
    private string[] mechanicalSubThemes;
    private string[] chemicalSubThemes;

    // Awake is called before the start fucntion.
    // In this class is used to create the singleton.
    private void Awake()
    {
        MakeSingleton();
    }
    void MakeSingleton()
    {
        if (instance != null)
        {
            Destroy(gameObject);
        }
        else
        {
            instance = this;
            DontDestroyOnLoad(gameObject);
        }

    }

    // Start is called before the first frame update.
    // It is used in this code to set valiables
    private void Start()
    {
        electricalSubThemes = new string[3];
        mechanicalSubThemes = new string[3];
        chemicalSubThemes = new string[3];

        electricalSubThemes[0]= "Circuits" ;
        electricalSubThemes[1] = "Digital Signal Processing";
        electricalSubThemes[2] = "Electronics";

        mechanicalSubThemes[0] = "System Dynamics";
        mechanicalSubThemes[1] = "Fluid Dynamics";
        mechanicalSubThemes[2] = "Thermodynamics";

        chemicalSubThemes[0] = "Basic Chemistry";
        chemicalSubThemes[1] = "Kinetcs and Catalysis";
        chemicalSubThemes[2] = "Thermodinamics";


    }
```

Figure 2

Part One of the Game Manager Code



```
    //Function called by the menu and game controlles in
    //orther to get the current sub-theme selected by the player
    public string [] getSubThemesStrings(int i)
    {
        switch (i)
        {
            case 0:
                return mechanicalSubThemes;
                break;
            case 1:
                return electricalSubThemes;
                break;
            case 2:
                return chemicalSubThemes;
                break;
            default:
                return mechanicalSubThemes;
                break;
        }
    }

    //The loading functions are used to load the scenes (menu or game play)
    public void LoadGamPlay()
    {
        SceneManager.LoadScene("GamePlay");

    }
    public void LoadMenu()
    {
        SceneManager.LoadScene("MainMenu");

    }

    //Menu controller sets the engineering field to be played
    public void setWhichEngineeringToPlay(int engSelection)
    {
        this.engSelection = engSelection;
    }
    //Gamplay controller gets the engineering field to be played
    public int getWhichEngineeringToPlay()
    {
        return this.engSelection;
    }

    //Menu controller sets the engineering sub-theme  to be played
    public void setSubThemeToPlay(int subSelection)
    {
        this.subSelection = subSelection;
    }
    //Gamplay controller gets the engineering sub-theme to be played
    public int getSubThemeToPlay()
    {
        return this.subSelection;
    }

    //Menu controller sets the engineering sub-theme level to be played
    public void setLevel(int selectedLevel)
    {
        this.selectedLevel =  selectedLevel;
    }
    //Gamplay controller gets the engineering sub-theme level to be played
    public int getLevel()
    {
        return this.selectedLevel;
    }
}
```

Figure 3

Part Two of the Game Manager Code

The other classes will not be discussed under the scope of this paper.

The 2D game consists of 2 scenes, Main Menu and Game Play. The Main Menu looks as follows:

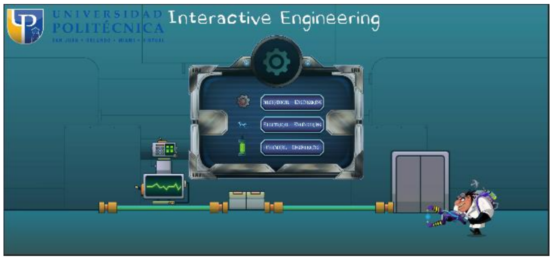


Figure 4

Main Menu Shows 3 Engineering Fields for the Selection

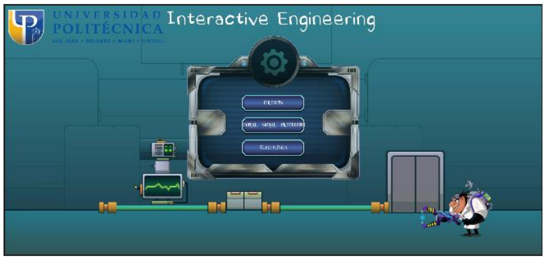


Figure 5

Main Menu Shows Available Sections in the Electrical Engineering Field

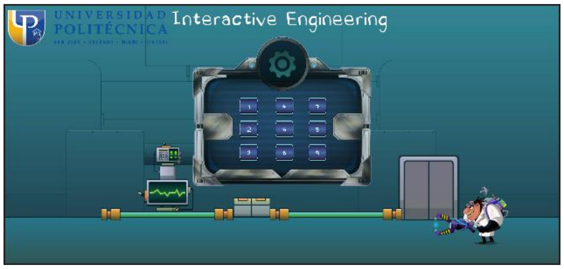
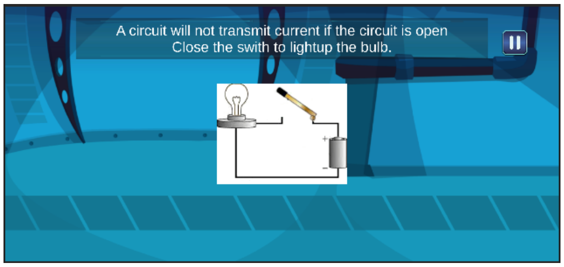


Figure 6

Main Menu Shows the Level Panel in the Section of the Electrical Engineering Field

Here is an example of the Gameplay scene for the electrical engineering field:



A circuit will not transmit current if the circuit is open
Close the swith to lightup the bulb.

Figure 7

Gameplay for the Level on the Circuit Section for the Electrical Engineering Field

CONCLUSION

As mentioned earlier, the class can provide an essential structure for an interactive engineering game. The game was created in a 2D environment to ease the not software related parts of the prototype at a minimum (like animation). Some challenges need to be fully addressed within the project's scope, like marketing and how to make the game more appealing. It is crucial to attract students to the engineering fields making the game more attractive. However, creating an interactive game can aid in the solution to the increasing demand in the engineering fields and the lack of interest in studying this career.

Future Work

The construction of this game was only a prototype, so there are several steps/phases to fulfill its purpose. First, the focus will be on completing all levels for the selected engineering fields. In parallel, an animator should be included in the project to design and create all required sprites to make the game more appealing. The second phase will be the game testing phase. This phase could be implemented by deploying the game to a school where students are incentivized or motivated to play and give feedback. The third phase will be the launch phase. This phase will involve the app's marketing and the different schools' engagement in the game. After the game launch, one could include additional engineering fields. The game can be expanded to include math and science, which are also observed to lack interest.

The structure proposed in this paper can also be used to provide engagement to other non-STEM-related fields. It can be used as an assistance for teachers to make learning more attractive.

Acknowledgments

This project was made possible thanks to the advice of Professor Nelliud Torres and the Electrical engineering undergraduate student Alejandra Caro for providing the idea behind it.

REFERENCES

[1] J. Roman, "National Society of Professional Engineers," *National Society of Professional Engineers*, 2021. [Online]. Available: https://www.nspe.org/resources/pe-magazine/spring-2021/how-meet-the-increasing-demand-engineers. [Accessed 25 November 2022].

[2] W. C. Johnson, "World Expertise," *World Expertise LLC*, [Online]. Available: https://www.worldexpertise.com/Declining_Interest_in_Engineering_Studies_at_a_Time_of_Increased_Business_Needs.htm. [Accessed 28 November 2022].

[3] B. N. Geisenger and D. Raj Raman, "Why They Leave: Understanding Student Attrition from Engineering Majors," *International Journal of Engineering Education*, vol. 29, no. 4, pp. 914-925, 2013.

[4] L. Schardon, "GameDev Academy," *GameDev Academy*, 22 February 2022. [Online]. Available: https://gamedevacademy.org/what-is-unity/. [Accessed 13 December 2022].

[5] T. Kesler, "Foxxthom," *Foxxthom*, 8 April 2021. [Online]. Available: https://foxxthom.medium.com/game-manager-one-manager-to-rule-them-all-1c06afa72b23. [Accessed 19 January 2023].