Platform Simulation Project

Architecture / Design Document

Table of Contents

Change History 2
Introduction 3
Design Goals and Objectives 4
System Behavior 5
Logical Design 7
High-Level Design 7
Mid-Level Design 8
  Detailed Class Design 8
User Interface and User Interaction 9
  Using On-Screen GUI 9
  Using Keyboard 9
Lessons Learned 15
Summary 17

Change History

Version: 0.3
Modifier(s): Iyad Shaheen, Muhammad Hilmi Badrol Hisham, Paula Navea
Date:12/12/18
Description of Change(s):

1. Change was made to the UI. Simulate scene was added, which include buttons that allow the user to navigate from a scene to another, read the data from the data file and simulate the data across the platform.
2. Changes were made to all parts of the write up to accommodate changes that were made to the program and new functionalities that were added to it.

Team B (Iyad Shaheen, Muhammad Hilmi Badrol Hisham, Paula Navea)
Platform Simulation Project (Final Phase)
COMP465
Professor: Vahe Karamian

Platform Simulation Project

(Final Phase)

Introduction

The main objective and goal of this program is to simulate “Kinetic Art” for a Mechanical Hydraulic System, using multiple cubes that can freely move about the y-axis.

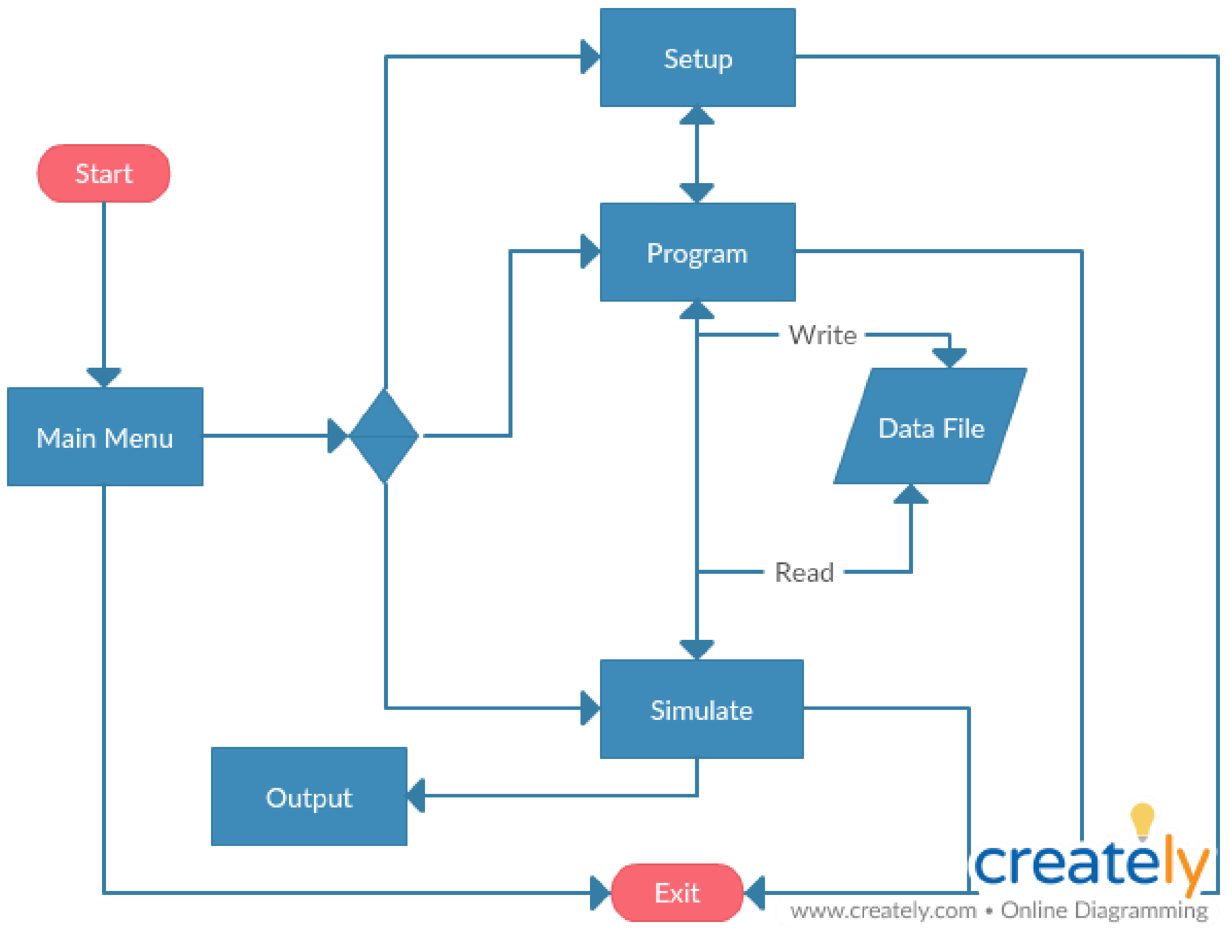
At this point, we have reached the last phase of this program. The program now allows user to interact via the on-screen graphical user interface (GUI) that we have created, giving user the ability to set up a platform, with specific dimensions, spacing and maximum node height that would be used in the Programming scene. Then, in Programming scene, user will be able to set up each node to their preferred height by selecting the node and specifying the height. Lastly, in the 3rd phase of this project that we have been working on recently is the Simulation scene. In this scene, the program will read the saved formatted data that was previously programmed by user, store it in the appropriate data fields in the program’s back-end, and simulate the data, enabling the programmed nodes to move across the platform until the user change the scene or exits the program. More details about the design of the program and objectives will follow in the sections down below.

Design Goals and Objectives

The design priorities for the platform simulation application are the following:

- The design shall minimize the complexity of the program and minimize the development effort.
- The design shall allow user to manipulate the available data via on-screen graphical user interface (GUI).
- The design shall allow user to create and build a platform in the Setup scene during runtime, using user's input of its dimension, spacing between individual nodes, maximum height the nodes can travel, and the initial simulated nodes' color.
- The design shall allow user to program each node individually by clicking on it and using the slider from on-screen GUI to change its height.
- The design shall allow user to control the camera and move it around for a better view using the keyboard arrow buttons.
- The design shall have multiple classes, and delegates and events are used to handle changes and allow each of the nodes to be responsible for its own state.
- The program shall be able to write the details of the platform to a text file for a later use.
- The program shall be able to read the data from an existing save file and use that to simulate the saved programmed platform.
- The program shall have an exit button, allowing the user to terminate the program at any time from any scene.
- The program shall allow the user to navigate between scenes easily.

System Behavior



The behavior for this program is as follows:

1. The program will transition between scenes as the user click the button that correspond to the scene. The starting scene of the program is the Main Menu with the four buttons on it - Setup, Program, Stimulate, and Exit - for user to choose.
2. The program will behave by jumping to Setup scene when the user clicks Setup button which is visible on every scene except for the Setup scene (because it's the current active scene, and thus not need any button to jump into it). The same button behavior will occur for other scenes as well.
3. The program will exit when the Exit button is clicked. This Exit button is visible in every scene.
4. In Setup scene, after user have chosen the specification for their platform and clicked Build Platform button, the program will build the platform with the chosen measurement from the given data. In the top panel, the details of the platform will then appear, reflecting the platform that has just been build.

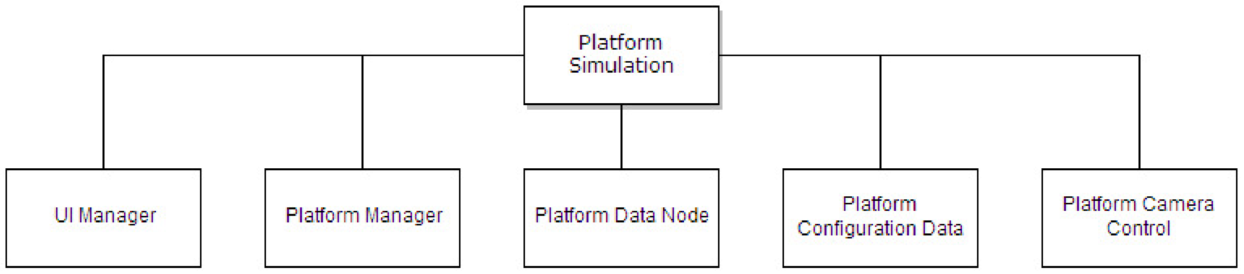
5. Inside the Programming scene, the program will react by changing the color of the tile selected by the user to blue and it will change the y-position based on the user’s choice of height for each node. The user will also be able to see and manipulate the selected node’s information from the side panel, such as the position of the node relative to the platform and programming the height of that particular node.
6. The program will change user’s view of the platform by changing the camera angle. By using the arrow keys on the keyboard, the user will command the changing of camera position, thus the program shall react by changing the view of the scene.
7. Upon clicking the Simulate button either in the Main Menu, or in any other scene, if the save data file is exists, the program will read the data from the pre-written save file. Using that data, the platform simulation will start, and the nodes will react by moving across the platform.
8. Upon clicking the Exit button from any scene, the program will terminate and exit.

Logical Design

The logical view describes the main functional components of the program. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.
In this section the modules of the program are first expressed in terms of high-level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.

High-Level Design

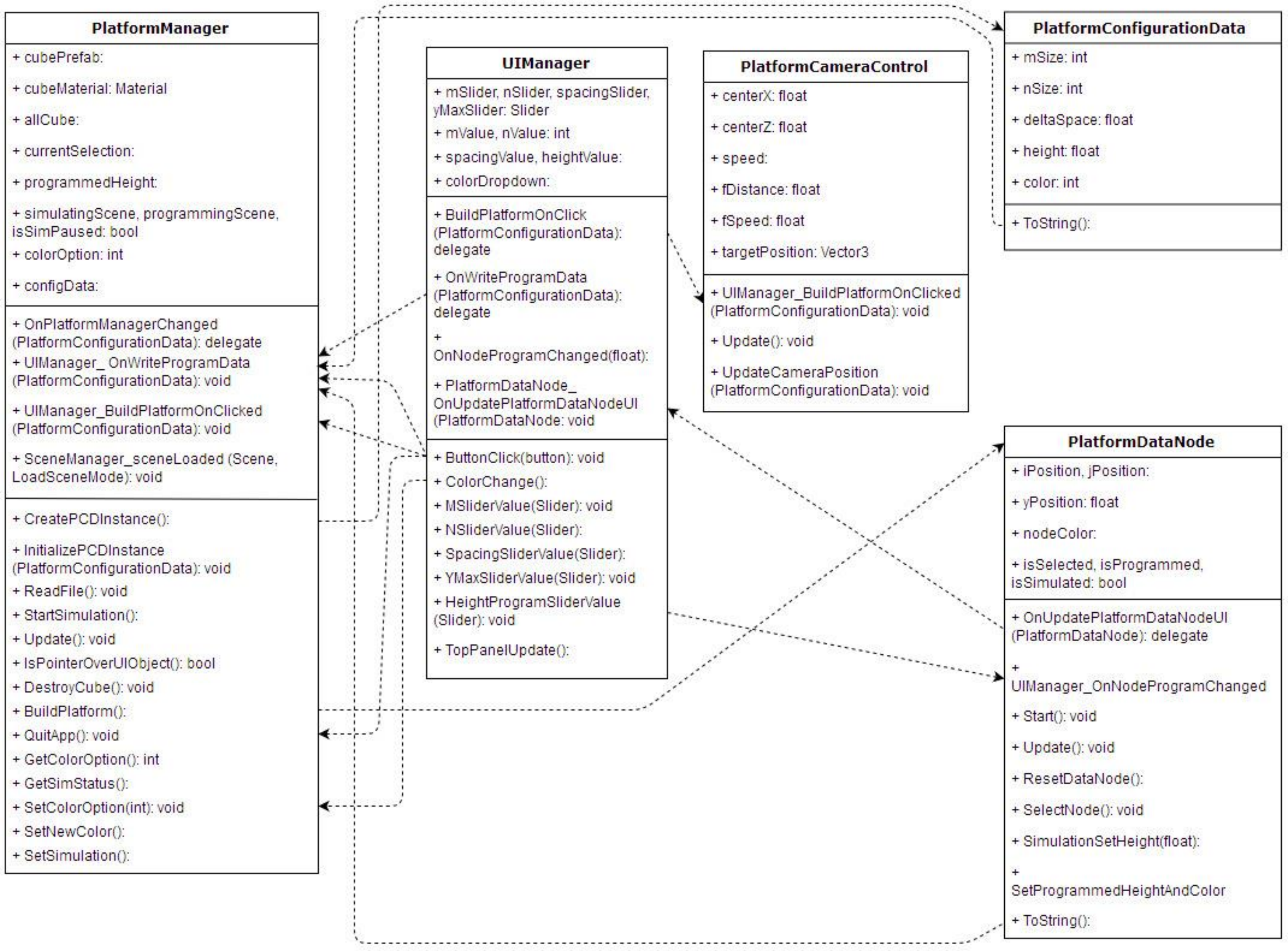
The high-level architecture of the program consists of the following components:



- UI Manager
  - Handling on-screen GUI display and user's interactions.
- Platform Manager
  - Handling platform movement and key input in Simulation scene. This class inherits Singleton pattern.
- Platform Data Node
  - Handling data for individual nodes, including their movement up and down about the y-axis.
- Platform Configuration Data
  - Holds the specification of the platform - total dimension, spacing between each of the nodes, maximum height that each node can reach, and initial color of the simulated nodes.
- Platform Camera Control
  - Handling the movement of main camera, including key input from user.

Mid-Level Design

Detailed Class Design


PlatformManager
+ cubePrefab:
+ cubeMaterial: Material
+ allCube:
+ currentSelection:
+ programmedHeight:
+ simulatingScene, programmingScene, isSimPaused: bool
+ colorOption: int
+ configData:
+ OnPlatformManagerChanged (PlatformConfigurationData): delegate
+ UIManager_ OnWriteProgramData (PlatformConfigurationData): void
+ UIManager_BuildPlatformOnClicked (PlatformConfigurationData): void
+ SceneManager_sceneLoaded (Scene, LoadSceneMode): void
+ CreatePCDInstance():
+ InitializePCDInstance (PlatformConfigurationData): void
+ ReadFile(): void
+ StartSimulation():
+ Update(): void
+ IsPointerOverUIObject(): bool
+ DestroyCube(): void
+ BuildPlatform():
+ QuitApp(): void
+ GetColorOption(): int
+ GetSimStatus():
+ SetColorOption(int): void
+ SetNewColor():
+ SetSimulation():
UIManager
+ mSlider, nSlider, spacingSlider, yMaxSlider: Slider
+ mValue, nValue: int
+ spacingValue, heightValue:
+ colorDropdown:
+ BuildPlatformOnClick (PlatformConfigurationData): delegate
+ OnWriteProgramData (PlatformConfigurationData): delegate
+ OnNodeProgramChanged(float):
+ PlatformDataNode_ OnUpdatePlatformDataNodeUI (PlatformDataNode: void
+ ButtonClick(button): void
+ ColorChange():
+ MSliderValue(Slider): void
+ NSliderValue(Slider):
+ SpacingSliderValue(Slider):
+ YMaxSliderValue(Slider): void
+ HeightProgramSliderValue (Slider): void
+ TopPanelUpdate():
PlatformCameraControl
+ centerX: float
+ centerZ: float
+ speed:
+ fDistance: float
+ fSpeed: float
+ targetPosition: Vector3
+ UIManager_BuildPlatformOnClicked (PlatformConfigurationData): void
+ Update(): void
+ UpdateCameraPosition (PlatformConfigurationData): void
PlatformConfigurationData
+ mSize: int
+ nSize: int
+ deltaSpace: float
+ height: float
+ color: int
+ ToString():
PlatformDataNode
+ iPosition, jPosition:
+ yPosition: float
+ nodeColor:
+ isSelected, isProgrammed, isSimulated: bool
+ OnUpdatePlatformDataNodeUI (PlatformDataNode): delegate
+ UIManager_OnNodeProgramChanged
+ Start(): void
+ Update(): void
+ ResetDataNode():
+ SelectNode(): void
+ SimulationSetHeight(float):
+ SetProgrammedHeightAndColor
+ ToString():

The class diagram above shows the content of every classes, as well as the association of some methods with other method located in different class. The arrows from one method to another as drawn here shows how the data flows from one to another by the use of delegates and events method.

User Interface and User Interaction

The program allows user to control the behavior of the program in two different ways. The first is through the on-screen GUI, and second is via keyboard command entries.

Using On-Screen GUI

User is able to interact with the program by using the on-screen GUI as follows:

- The Main Menu consists of a panel with four navigating buttons for user to jump into another scene that they choose. The buttons are:
    - Setup
    - Program
    - Simulate
    - Exit
- In Setup scene, user is able to define and configure a new platform by using the provided controls located at the left-hand side of the screen to set the platform's dimension, spacing of the individual nodes, maximum height that each individual node can be programmed to, and simulation color of the programmed nodes. When user clicks on the "Build" button, the platform is then built accordingly.
- In Programming scene, user is able to program individual nodes, where they can define the height of each node. Once the user clicks on the "Program Platform" button, these data will be saved and written to a text file.
- In Simulation scene, the program will then read the saved file and starts the simulation of the programmed node.
- In every scene, user is able to see the details of the platform at the top of the screen, as well as buttons that they can click to jump into different scene and Exit button to terminate the program.

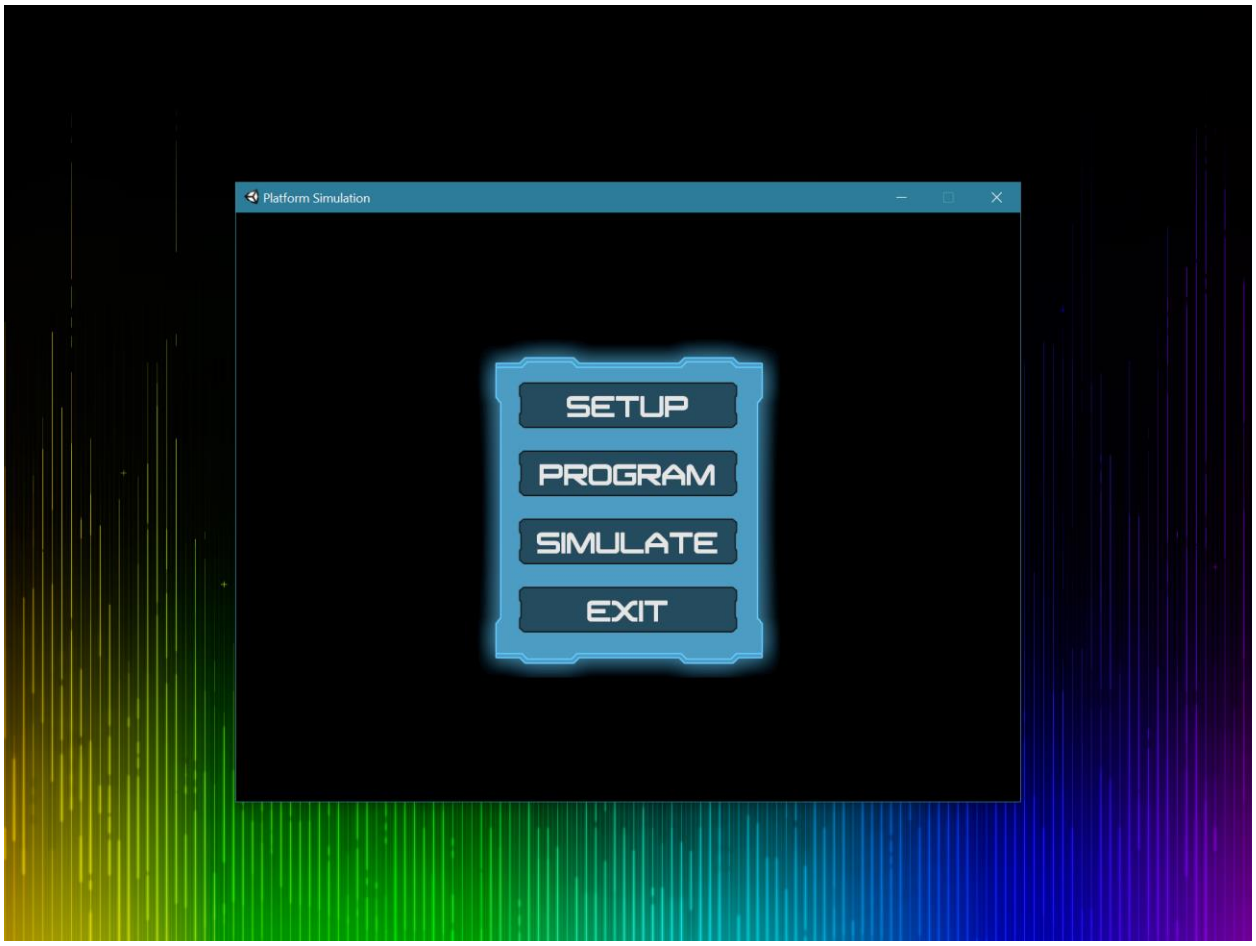
Using Keyboard

User is also able to interact with the program using keyboard input. The available commands are as follows:

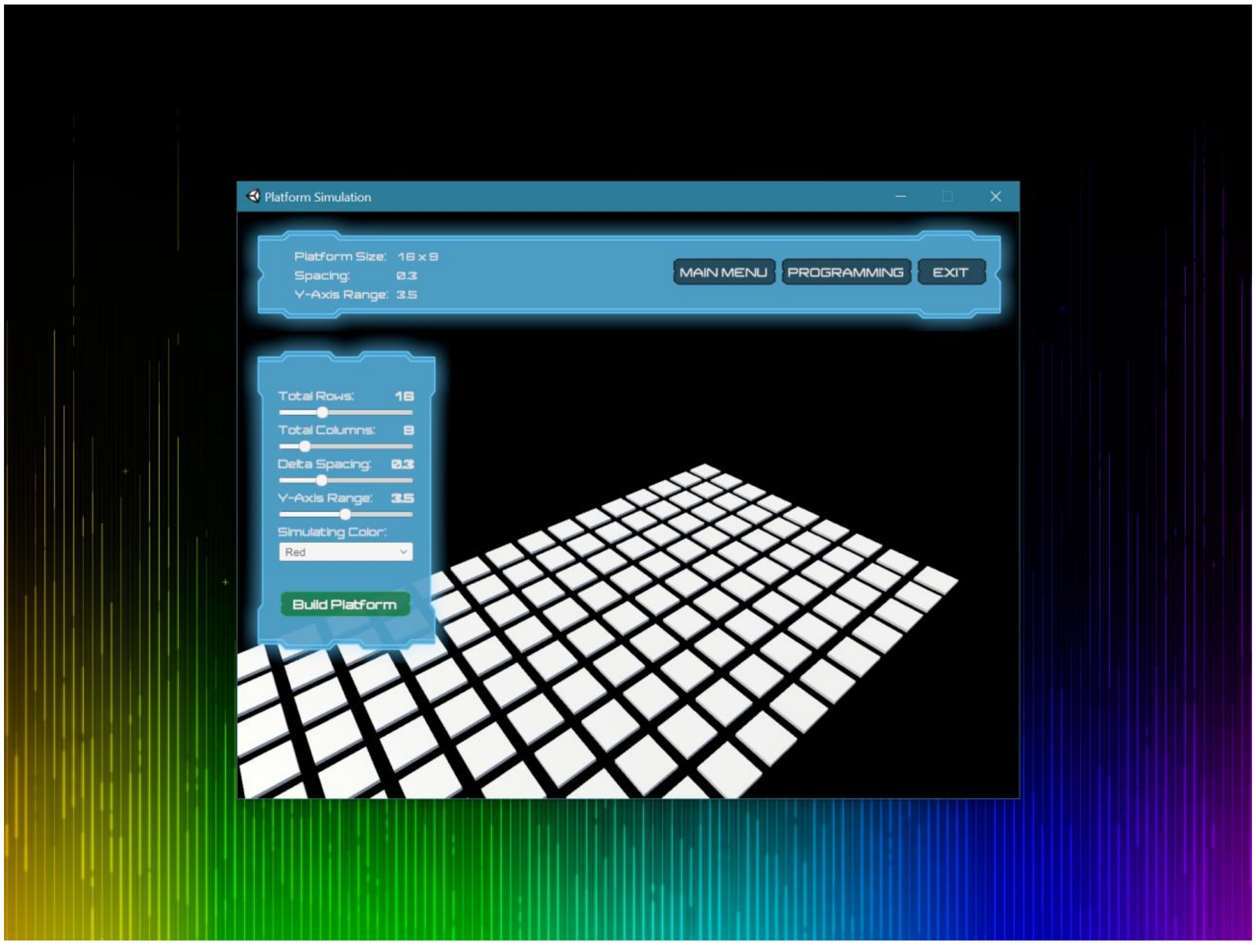
- Keyboard arrows
    - Arrow keys will allow user to change the view by moving the camera around in order to have access to nodes that are not visible or hard to reach.
    - Up and Down arrows can also be coupled with Left Shift key to zoom the camera in and out.
- "T" key (only in Simulation scene)

    - Pause or resume platform simulation.
- “R”, “G”, “B”, and “H” key (only in Simulation scene)
    - Change color of simulated nodes (to red, green, blue, or gray respectively).
- “Q” key
    - Quitting the application.

Screenshots



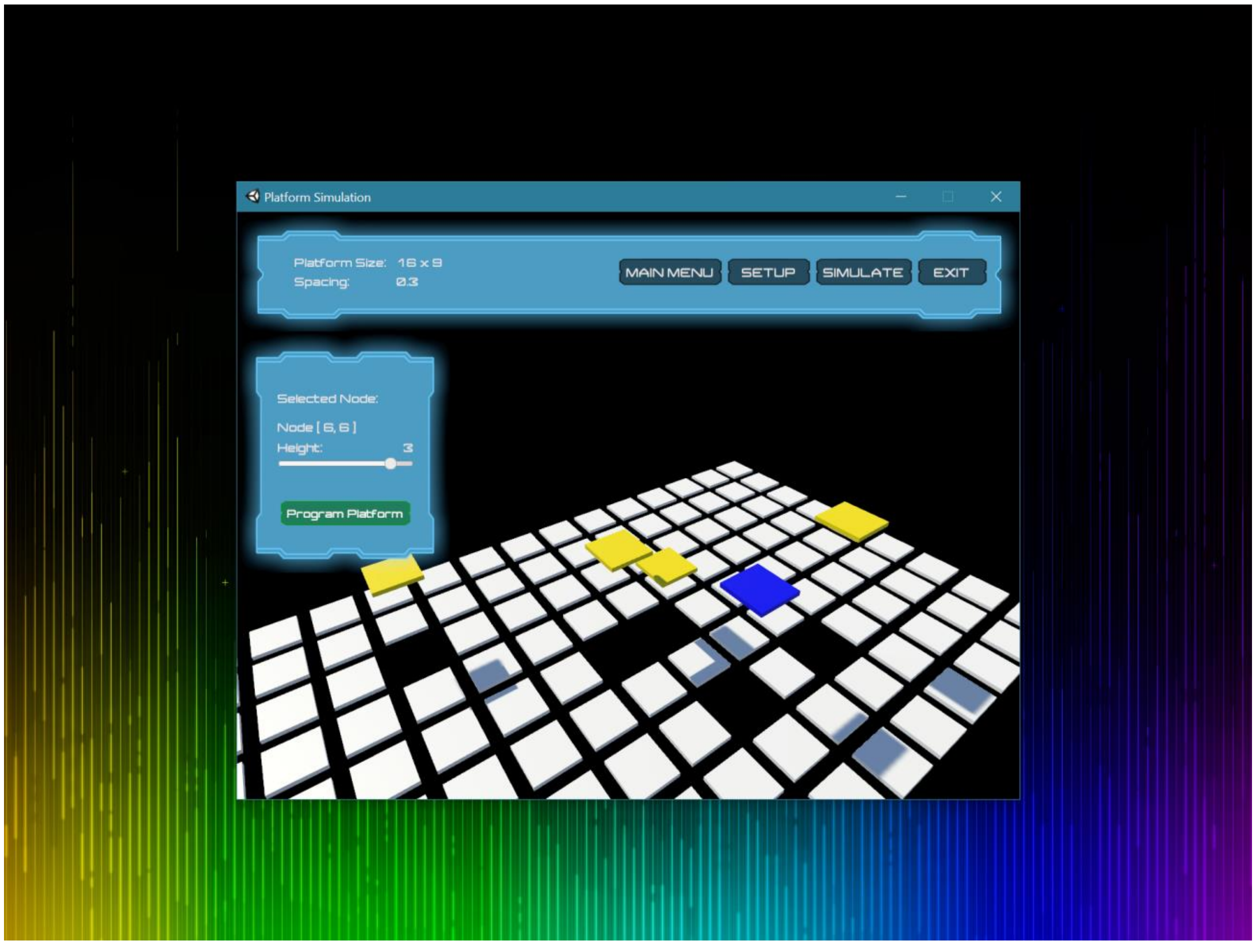
On program launch, user will be greeted by this Main Menu in which they can choose which scene they wanted to jump into. (By default, they will prevented from jumping into Program scene at launch, and Simulation scene if there is no save file found on their system)



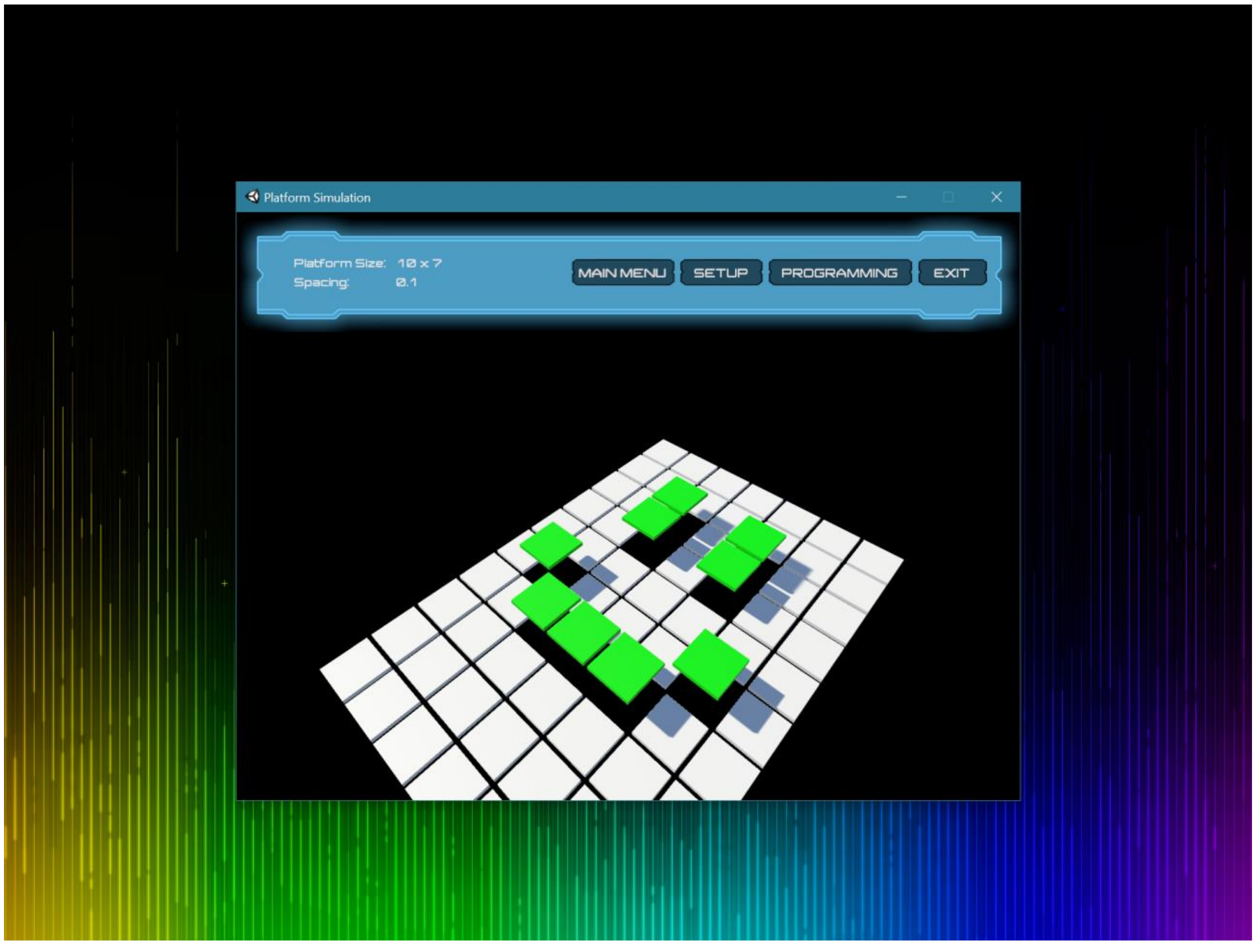
Platform Size: 16 x 9
Spacing: 0.3
Y-Axis Range: 3.5

Total Rows: 16
Total Columns: 9
Delta Spacing: 0.3
Y-Axis Range: 3.5
Simulating Color:
Red

Setup scene: User will be able to configure the specification of their platform using the on-screen GUI controls on the left-hand side of the screen. Clicking that green “Build Platform” button will trigger the program to build the platform accordingly.



Programming scene: Here, user will be able to pick individual nodes from the platform that they just built in the Setup scene and set the height movement on them. Current node that the user is working on will be colored blue, and previously programmed node will be colored yellow. All other unprogrammed nodes is colored white and will be defaulted to height of 0. Clicking that green “Program Platform” button will save this programmed platform into disk. If previous save file already exists on user’s machine, clicking this button will overwrite that file to this newly programmed platform.



Simulation scene: If a save file exists in this program directory on user's machine, then user will be allowed to jump into this scene. In this scene, the program will read user's save file and simulate the platform, moving the Kinetic Art across the platform. User also can pause the simulation (stopping the Kinetic Art in place until user hit the pause key again), and change the Kinetic Art color to red, green, blue, or gray using keyboard input.

Lessons Learned

We have learned a lot throughout this whole semester. From a personal perspective, we expanded our knowledge of how to create game object in Unity and writing scripts to control and manipulate the available data, either to move each of the object around, rotate, change color, or control every object simultaneously and put them into motion. We also learned about Game View and Scene View in Unity. For instance, Scene View is the view into the game world, and we use it to edit and reposition the scenery, characters, cameras, lights, and all other game objects. Being able to select, manipulate and modify objects in the Scene View are some of the skills we learned in the beginning of this course which allowed us to properly use and utilize Unity. On the other hand, the Game View is where the final, published game is represented. We learned how to utilize the tools we were provided in each of the object to edit the detailing and set the behavior of each object when user or other game object interacted with it. We also learned how to create multiple scenes and navigating between them within the game. We expanded our knowledge of C# language through this course as we had to use it to write scripts for the back-end of our projects. We also learned how to modularize our code and enhance data management.

From a graphical perspective, we learned how the components interact with each other, the camera, the lights, and game objects (creating primitive object or using Prefab). For instance, the positioning of the camera plays a crucial role on how user will see the program and affects the interactivity and usability of the program if it was not properly handled. The light is also very important as this is a 3D world; if it does not exist, all objects in the scene will be invisible as there is no light directed at them.

Throughout this course, we learned how hard it is to create a graphical program and games and made us appreciate the games we played more knowing how much work was put into it. We also learned about Singleton pattern, its benefits, and how to use it. The Singleton pattern is one of the well-known patterns in software engineering, in which it only allows a single instance of itself to be created throughout the whole time the program runs. We learned more about Raycast and made use of it to select any nodes in the game world when it got clicked.

We also learned how to use delegates and events to call out and execute function from different script. When this got set up properly, it will give us an elegant way to interact between different script without having to worry about any dependency issue. Finally, we learned how to create a save file for the data that we set up during the runtime, and how to read it back into the program so that user can save and continue the program whenever they want without worrying about the works that they have done on this program got lost after it quits.

The mistakes we made throughout this journey came with valuable lessons as well. For example, a small mistake that we spent days trying to fix in the second phase of this project due to small error of not setting the tag of the camera which lead to null pointer errors.

Summary

To summarize, the goal of this semester-long project is to build a program simulating mechanical hydraulic system which user can control and interact by using on-screen GUI and keyboard input. User is able to change the camera view using the arrow keys on keyboard and using on-screen GUI to switch between scenes and give input to this program such as building a platform of size specified by user and programming each node as to draw any shapes on the platform. Finally, user is able to simulate their programmed platform motion, causing the shape that they have programmed to move across the platform.

We have been introduced to Unity in this class and have learned a lot on how to use it as well as fully utilized the tools that have been provided within this software. Other than that, it also introduced us to C# language, since that is the language used by Unity Engine. This project also strengthened our understanding on object-oriented programming, being able to modularize every part of our code and make use of Singleton pattern to design our program better.

Last but not least, we learned about how to work in teams, and splitting works between members, in order to get the finished product deployed within the specified timeframe.