

Platform Simulation Report

Table of Contents

1 Introduction	2
2 Design Goals	2
3 System Behavior	3
4 Logical Design	4
4.1 High-Level Design	5
5 User Interaction	7
6 User Interface	8
7 Screenshots	9
8 Lessons Learned	10
9 Summary	11

Team Member 3
Team Member 2
Arnold Escobedo

COMP 465 Graphic Systems
Professor Karamian

17 December 2018

1 Introduction

This document describes the architecture and design for the Simulation Platform application that is being developed at California State University, Northridge (CSUN). The Simulation Platform is

meant to create a platform using nodes (cubes) with given spacing, height, and placement of each cube. The user will be able to perform camera movement functions by pressing specific keys on their keyboards. Primarily, everything the user interacts with will be housed and controlled with the user interface. The purpose of this document is to describe the current architecture and design of the Simulation Platform for the final stage. The project is complex, and we need to break down the project's entirety into different sections for the full project to be understood. We will be breaking down this phase into:

- **Design Goals and Objectives:** define what the purpose of this simulation and what is it supposed to create at this phase along with brief descriptions that lead into the final product
- **System Behavior:** define how the system is behaving and what is talking with one another for each key command
- **Logical Design:** explain each step of the system and what is happening to the entirety of the application
- **High-Level Design (Architecture):** showing the actual representation of the application without showing any code, since we do apply a greater scope of classes in this phase of the application
- **User Interaction:** how the platform reacts according to user pressing certain buttons on the user interface.
 - User interaction will span across three different scenes, each with their own specific function
- **User Interface:** designed to be the platform that allows the user to interact with the platform or manipulate the platform to do certain functions.
 - User Interface will be covering three different scenes
- **Screenshots:** display images demonstrating how the user interface looks along with how the platform looks while active.
- **Lessons learned:** detail our overall collaborative learning experience from the beginning of phase 1 to the end of the simulation project.

We break it up into these components to further explain how the system is in each of these categories so when overviewing the entire system, the design and architecture can be understood.

2 Design Goals

Design goals can vary determining on who is programming the project. We need the design to output correctly. We need to design correctly because if we have a bad design than we will wasting more money than we need to do to complete the project. We need to effectively design our project, so it can be presented thoroughly and with detail. Now when it came down to designing this ourselves, we had to have goals of some sort to lay a foundation down and improve the design as we continued the system.

For this phase of the Simulation Platform project, we set these goals:

- The design should run and complete with the appropriate outcome
- Reusability of the current design is desirable. Essentially work smarter, not harder.
- The design should be able to be understood with small comments explaining each function, method, or line of code that was used if others tend to work on this project.

- The design should handle complexity with as few lines as needed, but long lines isn't rejected either because there are multiple ways to solve one problem.
- The design of each code file should be able to effectively communicate with other files
 - User Interface information is going to be handled solely in the UIManager file
 - Platform movement and manipulation is going to be handled solely in the PlatformManager file
 - The information regarding each node of the platform will be handled in the PlatformDataNode file
 - Camera control will be continuously update the camera position
 - Platform Configuration will house all the information regarding the platform which includes the size, spacing, and max height when programming nodes. Programming will be mentioned as we continue through the document
 - Platform Generic Singleton will make sure only one instance of the platform is created

3 System Behavior

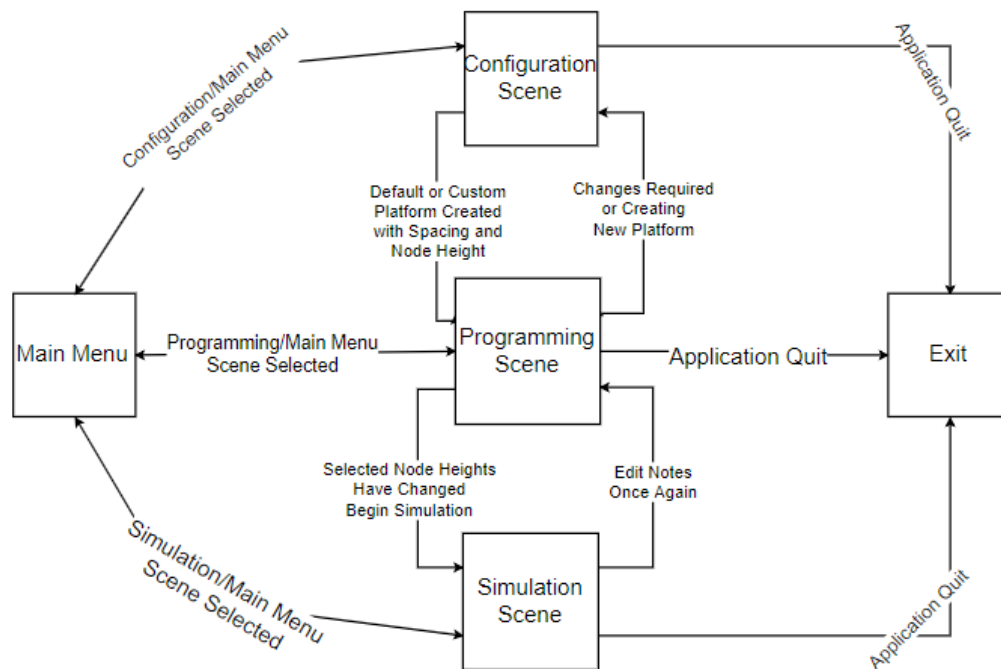


Figure 1 System Behavior

The User Interface has been modified to have one main panel which will be present in all scenes, with a separate panel depending on the scene the user currently will be at. We can demonstrate a simple example for the user when start the application. We start at the Main Menu which contains four buttons that are clickable events that each lead into a new scene with their respective functions. The user may be able to exit the Main Menu, or load into the following scenes:

- Configuration Scene
- Programming Scene
- Simulation Scene
- Exit

From the Configuration Scene, the user can create the default platform or create a custom platform. The programming scene allows the user to edit the height of their selected nodes on the platform. The simulation scene will demonstrate the programming of the edited nodes and move the programmed nodes forward. Here the user is only given the options to go back to the main menu and start from the beginning, back to the programming scene and edit which nodes to be programmed, or exit the application overall.

4 Logical Design

In this section, the logic of the application is explained in better detail. Since there were multiple key inputs, colors, and positions to handle, the application had to be able to keep track of that data information. Since we were using a two-dimensional array was being used for the cubes, it made logical sense to have a two-dimensional array for both nextPosition and nextColor. It made logical sense designing it this way since we were already keeping track of the cubes, we can use the same row and column points to locate the cube, color, and the next cube to move on to.

With the addition of the user interface, the logical design behind the user interface was attempting to make it as simple as possible, user friendly, but handling each input field entry, slider value change, button clicks, and current information being displayed. Now the application is capable of being modular and eliminating dependency as much as possible. The application is now capable of handling platform manager data and user interface data separately. Logically, this is the better design when completing the second phase of the simulation platform manager since we need to make sure each everything that is changing needs to be handled properly.

4.1 High-Level Design

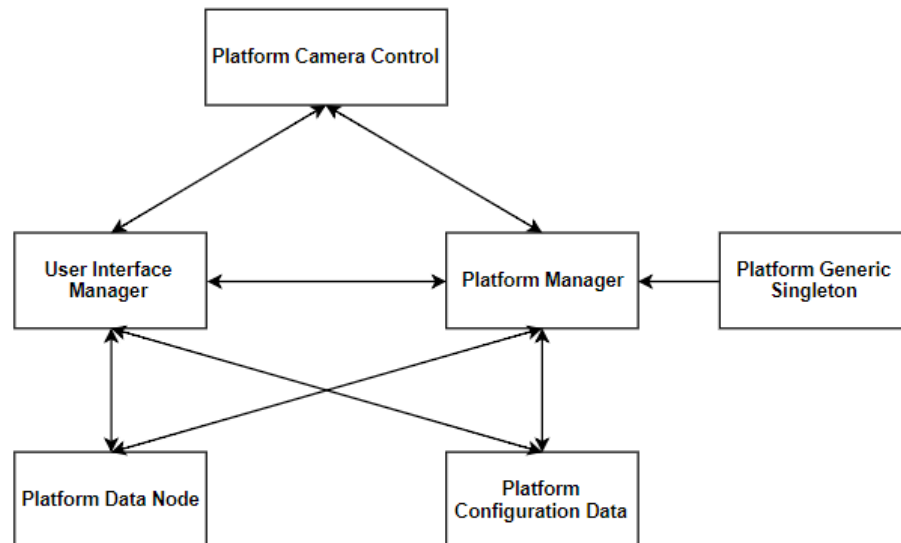


Figure 1 High Level Design

The figure above represents all the classes that need to communicate with other classes, so the application can function properly. User Interface Manager, Platform Manager, Platform Data

Node, Platform Configuration Data all need to communicate because the user interface updates the information about the platform whenever changes are made. Platform Camera Control communicates with both the Platform Manager and the User Interface Manager since it needs to grab data from both classes to effectively center the camera whenever the platform is built along with centering it when loading into different scenes. Platform Generic Singleton is active everything when the application begins because this class dictates that is to only be one instance of the platform and not have multiple platforms display at the same time. To start off, these two functions are in all the classes except Platform Generic Singleton and Platform Configuration Data:

- **OnEnable()** raises the event when a specific button is pressed and the class it needs to call when fulfill the event.
- **OnDisable()** frees the event after fulfilling the event has been completed allowing the event to be called again if need be.

The User Interface is the class that consists of its own separate methods with each method handling the user interface components. These are the methods with a brief explanation on what they do and handle:

- **OnPlatformDataNodeUpdateUI(PlatformDataNode pdn)** updates the information in the programming when the user selects a node and changes the height of that node
- **OnPlatformManagerUpdateUI(PlatformConfigurationData pcd)** updates the main panel in each scene to display the correct, up-to-date information about the platform
- **ButtonClicked(Button b)** handles which button is being pressed. It knows which button is being clicked based on the name of the button. It will then load a new scene or raise an event to initiate a function in another class to fulfill the event.
- **SliderValueChanged(Slider s)** handles which slider's value is being changed. It knows which slider is being changed by utilizing the name of the slider
- The **OnApplicationQuit()** quits the program once the user wants to end the simulation

The Platform Manger is the class that handles all the information regarding the platform and communicates with the other three classes to have correct up-to-date information.

- **Update()** continuously handles the position of the mouse and if the current the user is in is the Programming Scene
 - If a node is behind the User Interface, the mouse will be able to click on that node until it is outside the User Interface
 - It also continuously sets the current node's next position from the game object platform array equal to the same node's next position from the platformData array.
 - If will also set whether that node should be simulated or not.
- **UIManager_BuildPlatformOnClicked(PlatformConfigurationData pcd)** takes care of building the platform by setting the dimensions of the platform after the Build Button has been pressed
- **SceneLoaded_sceneLoaded(Scene arg0, LoadSceneMode arg1)** handles the scene management but only sets whether the user has the ability to select nodes on the platform
- **BuildPlatform()** is where the actually node is being created along with the coordinates and color of the cube are saved for the Node Information panel
- **DestroyPlatform()** essentially destroys the platform if the platform is not empty prior to the start of the application

- **IsPointOverUIObject()** checks if the mouse is over the User Interface and disables whether the platform can still be clicked on.
- **UIManager_OnWriteProgramData()** saves the platform information and writes it to a text file once the Program Button has been pressed
- **ReadFile()** essentially does as stated. It reads from the same text file that has been written to so the user can load in previous data that was entered.
- **UpdatePositions()** continuously updates the platformData array with the platform array so we can simulate as if the nodes that were selected move forward.
 - Since we have to look ahead, the method also handles when the array reaches the end so it can set it back to the beginning and continue the simulation.

The Platform Data Node class houses the information regarding the coordinates and height value of the node.

- **Update()** is just controlling the lerp when the node is changing height.
- **SelectNode()** updates the Node Panel to update Node Panel by updating the coordinates of the node that was selected.
- **OnNodeProgramChanges(Slider s)** is capturing the value of the height node and making sure to save that new height for that selected node.
- **ToString()** have proper format when writing to the text file

The Platform Generic Singleton is the class that creates one instance of the platform upon start of the application. Once the platform has been built no other instances of the platform can exist. If the platform takes on new dimensions, the new platform is now that one instance.

- **T Instance** is determining whether there is an instance already, if not it will instantiate one new instance
- **Awake()** if an instance is null it won't destroy anything since that is the only one, else destroys all game objects.

Platform Camera Control is the class that handles the camera angles. It allows the user to move the camera around the platform to get a better viewing of the platform itself. It continuously centers the camera every time the platform is built. It is centered by utilizing the dimensions of the platform and figuring out where the center is located:

- **UIManager_BuildPlatformClicked(PlatformConfigurationData)** utilizes the dimensions of the platform to calculate the center of the entire platform
- **Update()** is just continuously handling the movement of the camera along with moving around and zooming in and out.

Platform Configuration Data solely houses the information regarding the dimensions and spacing of the platform, and sets the max height for the nodes when being programmed.

- **ToString()** this is meant to write the information of the entire platform to the text file.

5 User Interaction

The user is now capable of interacting with the platform by pressing the keys mentioned above along with clicking the various buttons, entering values into the input fields, and dragging the button along the slider for a new value:

As mentioned in Phase 1 and Phase 2A about these certain interactions that the user can do with the user interface and how the platform functions. We are holding off the keyboard commands during this phase of the project. Along with the temporarily removing the keyboard commands, we added some commands that control the camera view of the platform, so the user can move the platform around. In comparison with the User Interface, we are keeping the functionality of most of the buttons and sliders but either removing completely or moving the button to another scene. These are some of the functions that were kept from the previous phase:

- **Setup Button:** reverts to the configuration scene from the programming scene
- **Exit Button:** quit the entire application in all scenes
- **M Input Field:** enter numeric value for M
- **N Input Field:** enter numeric value for N
- **Delta Spacing:** drag the button along the slider to change the value
- **Y-Displacement:** drag the button along the slider to change the value
- **Build Button:** create a new platform based on the new values

Here are the following components that were added to their respected scene:

- **Configuration Button:** loads into the Platform Configuration Scene from the Main Menu
- **Programming Button:** loads into the Platform Program Scene from the Main Menu
- **Simulation Button:** loads into the Platform Simulation Scene Main Menu
 - All 3 above buttons are part of the Main Menu Scene
- **Program Button:** saves the current information regarding the platform and writes it to a text file
 - Part of the Platform Programming Scene

6 User Interface

Phase 3 of the Platform Simulation project introduced the user interface. This allows the user to manipulate the platform without having to remember each that needs to be pressed to do the certain functionality they want. The user interface comprises of one large canvas, with two panels within the canvas within each scene with their title, components, and their respected functionality. The important additions that the user interface introduces is the ability to create a new platform allowing the user to input new M and N values along with moving the slider to change the spacing between each node on the platform. The following text information, buttons, sliders, and dropdowns can be explained on their respected panel:

Main Menu:

- **Configuration, Programming, Simulate:** Allows the user to select one of the buttons to load into their respected scene.

Just to note that all center panels will usually contain all the following buttons and text information:

- **Platform Size:** displays the dimensions of the latest platform that was built
- **Delta Spacing:** displays the latest spacing between each cube
- **Y-Displacement:** displays the latest max height the nodes can go in the programming scene
- **Main Menu Button:** loads and displays the user the main menu
- **Programming Button:** loads and displays the user the programming scene
- **Setup Button:** load and displays the user the configuration scene
- **Exit Button:** allows the user to exit the application all together

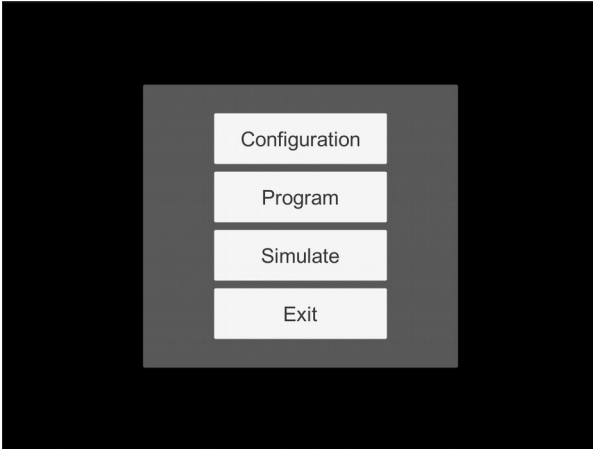
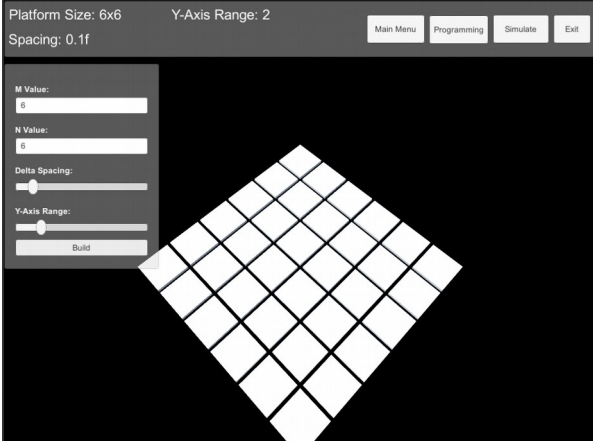
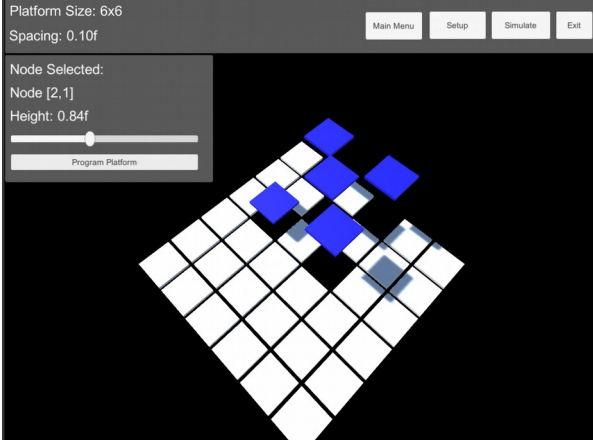
Build Panel in the Configuration Scene:

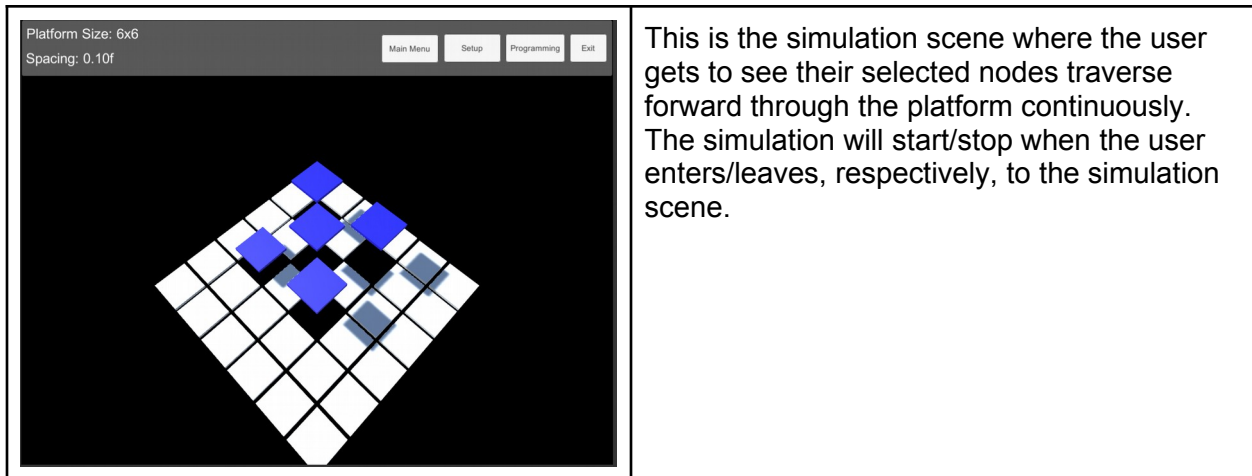
- **M Input Box:** allows the user to enter a new M value, if wanted
- **N Input Box:** allows the user to enter a new N value, if wanted
- **Delta Spacing Slider:** allows the user to drag the button along the slider with Delta spacing continuously updating when the value is changed
- **Y-Displacement Slider:** allows the user to drag the button along the slider to dictate the max height for the nodes to be programmed
- **Color Dropdown Option:** allows the user to select a color to display the on the platform, currently not implemented this phase
- **Build Button:** builds a new platform with the given values if they were changed, if not the default values are inputted

Node Information Panel in the Programming Scene

- Node text information displays the current selected node by the user
- Height Information updates as the user drags the button along the slider to set the height of the selected node
- Program Platform Button: allows the user to save the current information about the platform and each selected node height whether changed or not

7 Screenshots

	<p>This is the main menu and where the user is going to be starting the application. We have the three main scenes as the options along with the exit button to quit the application when the user needs to quit.</p>
	<p>This is the configuration scene and here is where the user can set the platform by providing dimensions, spacing between each node, and sets the maximum Y-Axis Range for each node when in the programming scene.</p>
	<p>This is the programming scene and here is when the user can select which nodes are going to be programmed to a certain height. The limit would be the Y-Axis Range which the user set in the configuration scene. We identify the nodes that have been selected and programmed to be blue and all other nodes to be white.</p>



8 Lessons Learned

Starting from phase 1 of this semester long project we had the opportunity to learn a graphics application that helped create a simulation platform. During phase 1 was the great opportunity to learn the basics of unity and how the Update method worked. We also learned how to take in user keyboard commands to perform simple functions and what combinations were necessary in order to successfully get the correct output. We learned how to create different objects and how to access certain aspects of the game object we had just created. Along with this useful information came the understand of how to have game objects have a smooth transition from start to finish. We utilized the Vector3.Lerp and Color.Lerp functions to successfully have these smooth transitions. Phase 1 was the actual introduction using C# and understanding its respected syntax.

During Phase 2 Part A of the platform simulation project we were introduced to the creation of an actual user interface. The user interface is the front end portion of the application that allows the user to interact with to perform certain tasks. This included input text boxes, buttons, dropdown menus, and actually have a panel dedicated to have the up-to-date information of the current platform in development. We usually only had the platform dimensions and spacing be the information that needed to be continuously updated. Creating the user interface was the fun aspect of the project because we needed to think like the user to create an interface that was going to be easy to understand and maneuver through in order to complete the required task of the user. We also learned that we needed to begin separating class files to avoid housing all functions on one script. We had the UI Manager and Platform Manager class file control their respective functions. During this phase, we also learned about anchoring our user interface components to our main panel so we wouldn't have to spend so much time on individually sizing each and every component.

Lastly, Phase 2 Part B of the platform simulation project we were introduced to creating custom events and how to handle events which then led to which functions were needed to be called

and performed to complete. We learned about the creation one one game object that would be used throughout the entirety of the program to avoid having multiple platforms be created. This became increasingly useful since we learned how to switch between scenes and have our platform singleton be used in different scenes and allowed to be edited. Learning how to switch between scenes was not as difficult as it sounded. We based it on certain buttons to house when to switch to another scene. During this phase, this is when we realized and learned why having the project having modular class files was beneficial. It was beneficial because the project began to grow larger than just a user interface and platform manager, we started house each individual node information, the actual dimensions, spacing, and max height for the nodes, the class file to create that one and only singleton, and the camera controls and information. We also learned how to find different game objects in the world space along with being able to write to a text file and how to read specific information from the text file.

9 Summary

In summary, this semester long project was a great learning experience from beginning to end. As a group, we continuously improved our code and problem solving skills to complete each phase of the project. We started off this semester project individually but the last phase was the group phase and this is where we needed to be able to separate the work amongst ourselves in order to effectively complete not only the coding aspect of the project but our presentation and the technical documentation. We collaborated and combined our minds together to design, implement, and complete the project. This technical documentation is meant to supply the reader with our overall goals for the project, system behavior, logical and high level design, and user interaction and interface. The system behavior describes how the application will behave just demonstrating different options and showing where they lead. The logical design better explains the logic behind completing and handling the information give to this application. The high level design provides an oversight of the how the different C# files communicate with one another. The user interaction is detailing what aspects of the application the user can do with this application. The user interface details the front end portion of the application detailing each component that was incorporated into each scene and what their primary functions were. The screenshots visually depict the application during each scene and how everything forms to create the platform simulation. This project was a great learning experience and we always remembered that if our thinking becomes more complex than needed, we need to take a step back and think of something simpler.