Homework 1 Report

CPSC 466

Summer 2022 - 6/1/2022

Instructor: Chang-Hyun Jo

Eric Chu , kiyuen88ec@csu.fullerton.edu

Nitish Lele, nitishlele@csu.fullerton.edu

Ruilin Wang, rwang18@csu.fullerton.edu

Javier Canedo, jcanedo23@csu.fullerton.edu

Deangelo Aguilar, dlo55@csu.fullerton.edu

Louis Kim, louiskim19@csu.fullerton.edu

Table of Contents

Revision History

Date	Name	Revision Done
06/06/2022	Eric Chu	● All buildings modeled in Blender
06/06/2022	Louis Kim	● Diagrams Completed
06/08/2022	Eric Chu	● All buildings imported to Unity
06/08/2022	Nitish Lele	● Traceability Matrix complete
06/09/2022	Louis Kim	● Updated Diagrams
6/10/2022	Javier Canedo	● UP Diagram completed
6/11/2022	Ruilin Wang, Deangelo	● Functional Requirements done ● Non functional requirements done

	Aguilar	
06/12/2022	Eric Chu	• Game Finished
6/13/2022	Deangelo Aguilar	• Application Description
06/13/2022	Louis Kim	• Completed Project Plan • Attempting to Organize Table of Contents • Completed Description of Application
06/13/2022	Eric Chu	• Construction iterations 1 and 2 completed • Technology Preparation section completed • AIFs section completed
6/13/2022	Nitish Lele	• CMMI graduate section complete
06/14/2022	Eric Chu	• Architectural Design Document Finished • User Guide Section Finished • Deployment Section Finished
6/14/2022	Louis Kim	• Uploaded Updated Detailed Class Diagram
6/14/2022	Deangelo Aguilar, Ruilin Wang	• Use Cases
6/14/2022	Javier Canedo	• Assisted in the completion of the Use Cases
6/14/2022	Nitish Lele	• Assisted in the completion of the Use Cases

Project Plan

As a group we decided we would not be using Roblox to create the game, but instead will be using Unity and Blender.

Eric Chu has a better understanding on how to use Unity and Blender to design an environment that is similar to California State University Fullerton campus. Using Blender, Eric will design specific buildings that are located on campus as a way to mark where you are on campus in real time. Completed the model of the building on June 6th. From there he will implement those buildings into Unity which will be our main platform in creating the game itself. June 8th, imported the buildings into Unity. Finally on June 12th, completed the game.

Nitish Lele is a Graduate Student who will be working on the Capability Maturity Model Integration (CMMI) as required by the document. Tasked with doing the documentation, while doing so will be working on Scrum Artifacts to talk about the game's backlog, plan of release, impediment list, and how to deploy the game to other schools to hold their own virtual commencement. And finally will be designing the traceability matrix.

Ruilin Wang will also be working on the documentation. Tasked with completing the Non-Functional Requirement (NFR), and use cases.

Javier Canedo was tasked to work on the documentation. With that his section would be the Development Case.

Deangelo Aguilar tasked with documentation. Will be working on the description of our application. The functional requirements that the application will allow the player to perform in the game.

Louis Kim will keep track of the required work products. As well as working on the documentation of the application. Tasked with creating the Use Case Diagram, High Level Class Diagram, Traceability Diagram, and Sketching a Detailed Design. All diagrams will be related to the application that was created by Eric Chu through Unity and Blender which does not connect to any server or database, but just through the application itself. All of which were completed as of June 6th and updated on June 9th.

Inception

Description of Application

This game was created for just a single player use and not multiplayer as it is not utilizing a server or database like it would with Roblox. The game is currently using Unity and with Blender to create models of the buildings. The buildings in the game are similar to those you would find on campus at California State University, Fullerton. You are able to control a character and move around visiting some buildings as well as

being able to attend a commencement ceremony. While attending the ceremony you are able to join in on the fun and throw your cap alongside your fellow graduates.

Development Case

Discipline	Techniques	Artifact	Incep	Elab	Const	Trans
		Iteration	I1	E1..En	C1..Cn	T1..Tn
Requirements	Working game for the graduation ceremony. Game will function well and meet the functional requirements set forth. Nonfunctional requirements will try and be met as well.	Vision	S			
		Supplementary Specification	S			
Design	Flow chart based diagram describing different features for the player. Fully functional simulation of CSUF graduation ceremony.	Design Model		s		
		SW Architect Document		s		
Project Management	Weekly Iterations to keep up to date on everyone's tasks. Discord will be used as meeting grounds to discuss project.	Risk List	s			
Implementation	Testing the game to make sure it runs well. Look at map of CSUF to get accurate representation.	Unity, C#		s		

Functional Requirements

The application shall allow a user to be a player.

The application shall allow a user to to control the direction and action of their player.

The application shall allow a user to pause the game.

The application shall allow collision detection between a player and the ground.

The application shall have buildings from California State University, Fullerton.

The application shall allow a user to adjust the volume of the application.

The application shall allow a user to save their current location in the campus.

The application shall display a minimap of the campus.

The application shall allow a user to choose their spawn location in the campus.

The application shall allow a user to teleport to different buildings in the campus

Non-Functional Requirements

1. **The game should run in Windows 10 and above.**
2. **The game should run at a minimum of 15 frames per second.**
3. The size of the game should be less than 2 gigabytes.
4. The resolution of the game should be displayed at 1080p.
5. The game should load in less than five seconds on launch.

**Requirements in bold are the ones implemented in the game.*

Architecturally Influential Factor (AIF)

AIF	**Reasoning**
Non-functional #2	The goal is to design software architecture that takes into consideration the performance of the game.
Functional #5	Collision matrix should be set up in order to define the physics behavior of the game. This will influence what classes and objects

	need to be created.
Knowledge in Unity	The member assigned with creating the game is most familiar with Unity and Blender. Therefore, software architecture will be created in Unity with C# and will be based on the Unity component system

Elaboration: Iteration One

Requirement Analysis

Use Cases

Use Case #1	User becomes a player in the game
Requirement #	Corresponds to functional requirement 1
Description	The game allows a user to play the game as a player
Goal	User is able to launch the game and play as a player in the game
Pre-condition	The user has clicked on the “play” button in the main menu.
Post-condition	The game scene is loaded and the user is now a player in the game.

Use Case #2	Users are able to control the direction and action of their player.

Requirement #	Corresponds to functional requirement 2
Description	The game allows the user to control the movement within the game
Goal	User is able to interact and explore the environment
Pre-condition	They have the hardware for the game to receive inputs
Post-condition	The user is now able to explore the environment we constructed

Use Case #3	User is able to pause the game
Requirement #	Corresponds to functional requirement 3
Description	The game allows the user to stop the game at the user's discretion
Goal	User can pause when they need to take a brief break
Pre-condition	User will click on “escape/E” to pause/resume game
Post-condition	The user is now able to pause/play the game

Use Case #4	Application should allow collision with the ground and the surrounding environment
Requirement #	Corresponds to functional requirement 4

Description	The player's character should have the necessary in game physics to interact with the application
Goal	The player should be able to walk on the ground and be stopped by structures
Pre-condition	The game physics should prevent the player from walking through buildings
Post-condition	The game now has the proper physics to prevent the player from walking through buildings.

Use Case #5	Application will have building from California State University, Fullerton
Requirement #	Corresponds to functional requirement 5
Description	The game environment should consist of the major buildings found on campus of California State University, Fullerton
Goal	The player is able to virtually explore the campus within the application
Pre-condition	The game will allow the user to explore campus and see the major buildings and reference them to the actual buildings of California State University, Fullerton
Post-condition	The game environment consists of all the major buildings found on Campus and the player will be able to explore them freely.

Use Case #6	The application shall allow a user to adjust the volume of the application.
Requirement #	Corresponds to functional requirement 6
Description	The user should be able to control their volume of the in game sound

Goal	The player should be able to control the volume setting and increase or decrease the volume as he wishes.
Pre-condition	The game will allow the user to use designated button on the device to lower or increase the volume.
Post-condition	In game voice from the game example background noise , footstep voice etc. will be lowered.

Use Case #7	User can save their location on the campus
Requirement #	Corresponds to functional requirement 7
Description	The game will allow the user to save their current location on campus and will allow the user to load at the exact location of where they saved.
Goal	The player should be able to save their location on campus at any time
Pre-condition	The game will allow the player to save their current location on campus
Post-condition	The game allows the player to save/load their location on campus

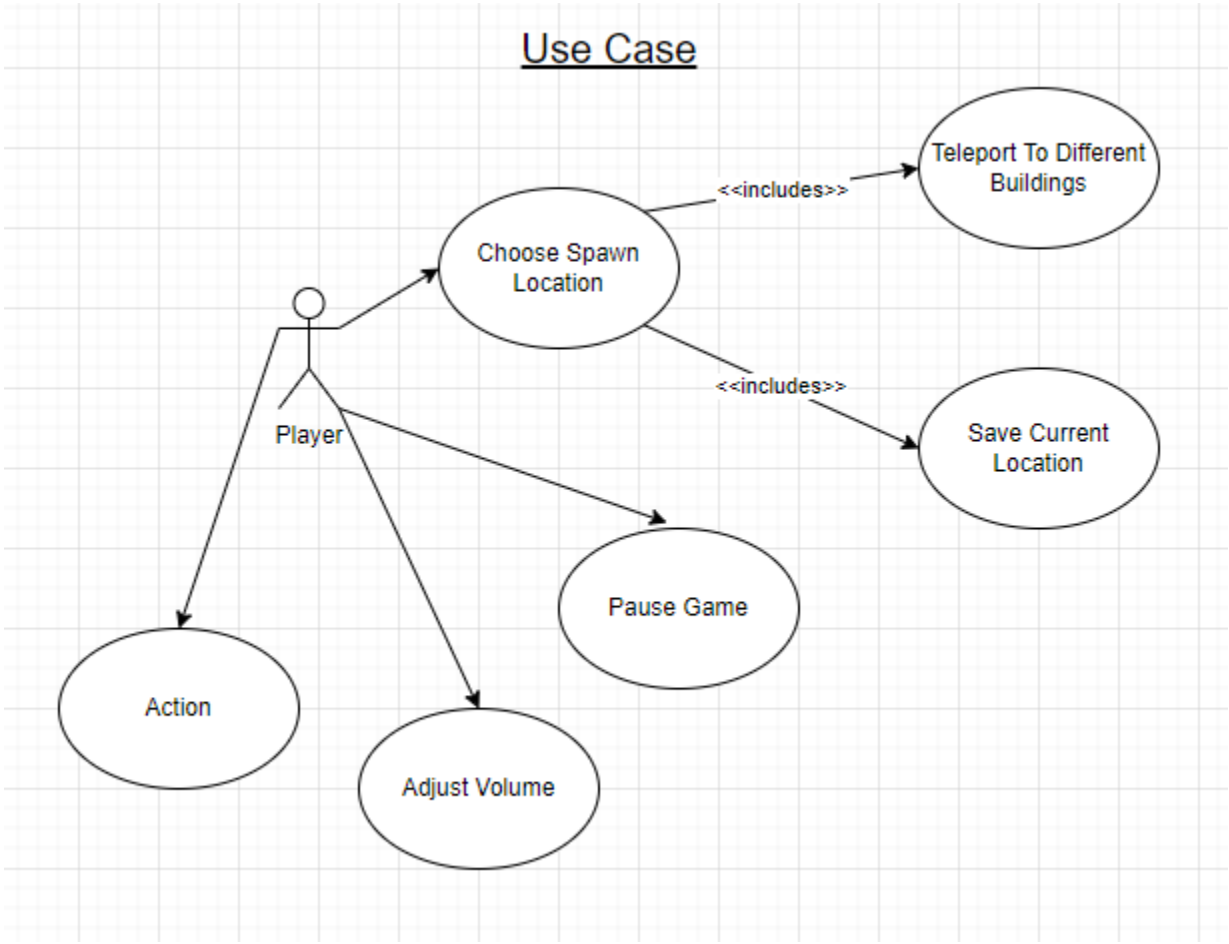
Use Case #8	Application shall display a minimap of the campus
Requirement #	Corresponds to functional requirement 8
Description	The user will be able to view a minimap of the Campus
Goal	The user will be able to have access to a minimap of the Campus
Pre-condition	The game will allow the user to access a minimap of the Campus to aid the user in pinpointing their exact location in reference to the Campus

Post-condition	The game now allows the user to view a minimap of the Campus

Use Case #9	Application should allow the user to set a spawn location
Requirement #	Corresponds to functional requirement 9
Description	The player can choose where they want to spawn in to quickly visit their desired location
Goal	The player can originate himself anywhere in the map.
Pre-condition	At the starting of the game, game software ask the player where he/she wants to spawn.
Post-condition	The game allows the player to originate at any location and pixels and create environment around the ingame player accordingly.

Use Case #10	Application should allow to teleport to the different buildings of the campus
Requirement #	Corresponds to functional requirement 10
Description	The player is able to choose any location on minimap of the campus and Teleport on any floor on any building or in the ground or garden.
Goal	If a player is stuck at some place he is able to teleport any ground or building in the minimap.
Pre-condition	Player moved the teleport arrow to the desired location.
Post-condition	Game loads the new pixel and environment arong the player and on the minimap player's location is changed to desired location.

Use Case Diagram



Architecture Design

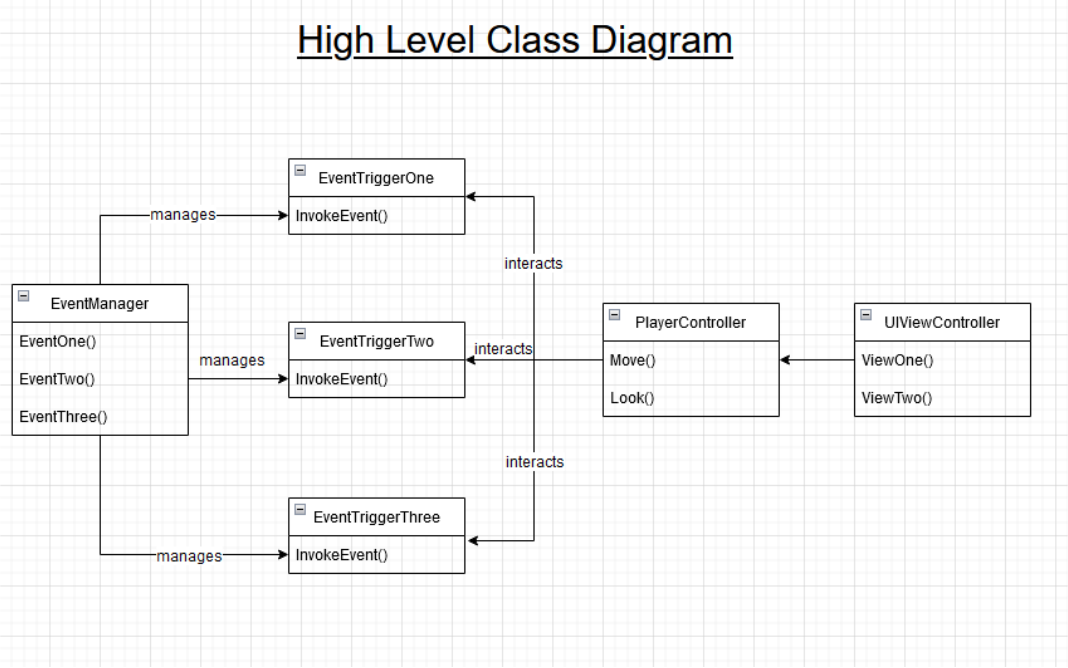
Scope of the Problem

Fullerton Commencement is a game where students can experience the graduation ceremony, which takes place at California State University Fullerton. There are no constraints or special features that the professor wants. The only requirement is that the setting of the game must have buildings from the campus of California State University

Fullerton and they must be positioned similarly like the buildings on campus. Any other additional features can be creatively built.

System Context

Based on the scope of the problem, Fullerton Commencement fits the game genre called open world. Open world games allow the player to explore freely and interact with objects in a virtual world. In addition, Fullerton Commencement is an offline single-player game. This means that we do not have to incorporate any server side components. Therefore, the architecture of the game should be event-driven**.** The following high-class diagram represents an event-driven architecture for Fullerton Commencement



Since Unity uses a component system for its game objects, a multi-class approach is preferred instead of using the traditional inheritance in object-oriented programming. Each class will represent a component that can be attached to a game object.

EventManager

- A singleton instance that manages the events in a single game scene. It contains events that other classes can subscribe to. Subscribed classes will have their methods invoked once the events are triggered.

EventTriggerOne, EventTriggerTwo, EventTriggerThree, etc..

- These classes are responsible for triggering events. They will call the InvokeEvent() method from EventManager to trigger their corresponding event. The trigger of these EventTrigger classes are dependent on the actions and inputs of the user.

PlayerController

- This class represents actions and commands that the user inputs

UIViewController

- This class contains the methods that the UI interfaces will call whenever a user interacts with them. These are most likely placed on the root canvas object.

Although an inheritance approach can still be used in Unity, it requires a lot of time and thought in order to construct an multi-inheritance system that functions with Unity's component system. Therefore, it is more time efficient to create multiple non-inherited classes and attach them to game objects.

Elaboration: Iteration Two

Technology Preparation

Unity and Blender are the two platforms that we will use to create the game. Unity is a game engine developed by Unity Studio. It can be used to create 2-Dimensional and 3-Dimensional games. Blender is a 3-Dimensional computer graphics software tool used for 3D modeling, visual effects, composition, animations, etc. For this game, Unity will be the engine that the game runs on and Blender will be used to create the models used for this game. We will use Unity version 2021.3.3f1 as it is the latest long term support version of Unity.

After researching and looking up what the entire campus of CSUF looks like, we found out that there are around 35 main buildings on campus. With this many buildings to model, it is important to have a pipeline for creating these models. Since one of our AIFs is performance in terms of frames per seconds, it is essential to create models that do not have a lot of vertices. In addition, we need the amount of textures for every model. This will reduce the number of draw calls made by Unity and reduce the overall file size of the game. Therefore, it was concluded that every model should contain less than 3,000 vertices, all models created should use textures from a pool of ten textures, and only the exterior of the buildings should be modeled.

With those rules in mind, the pipeline for creating the models is:

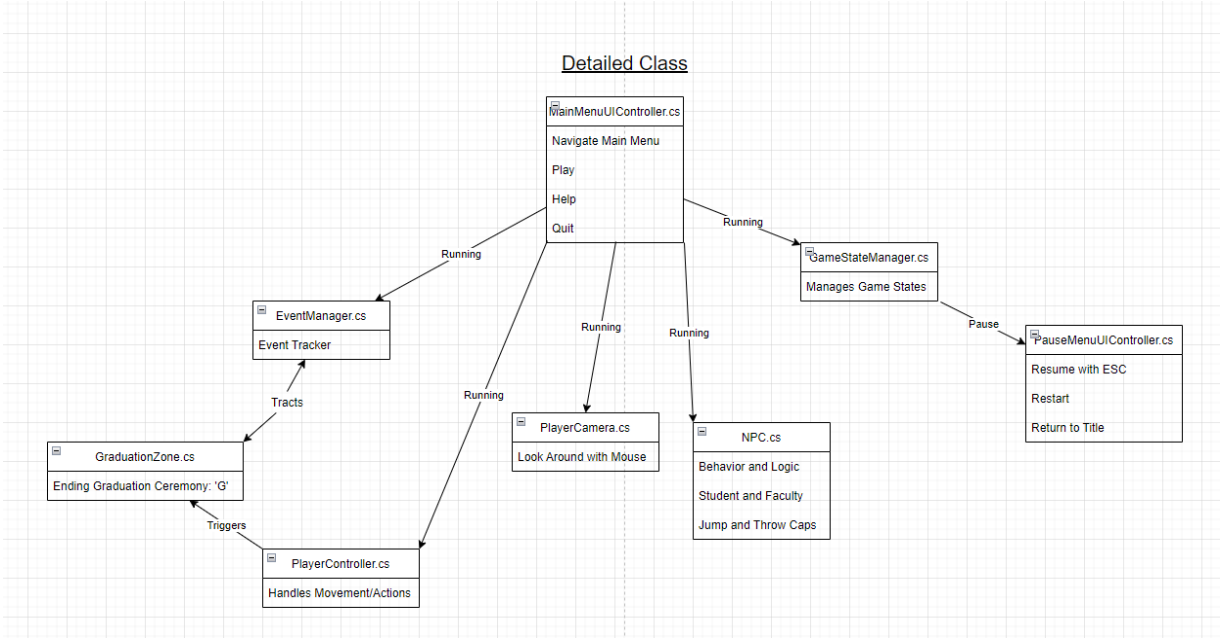
1. Model a building on campus
2. Texture the building
3. Import to Unity

4. Scale the size of the building if needed.
5. Mark the building as completed on the building list

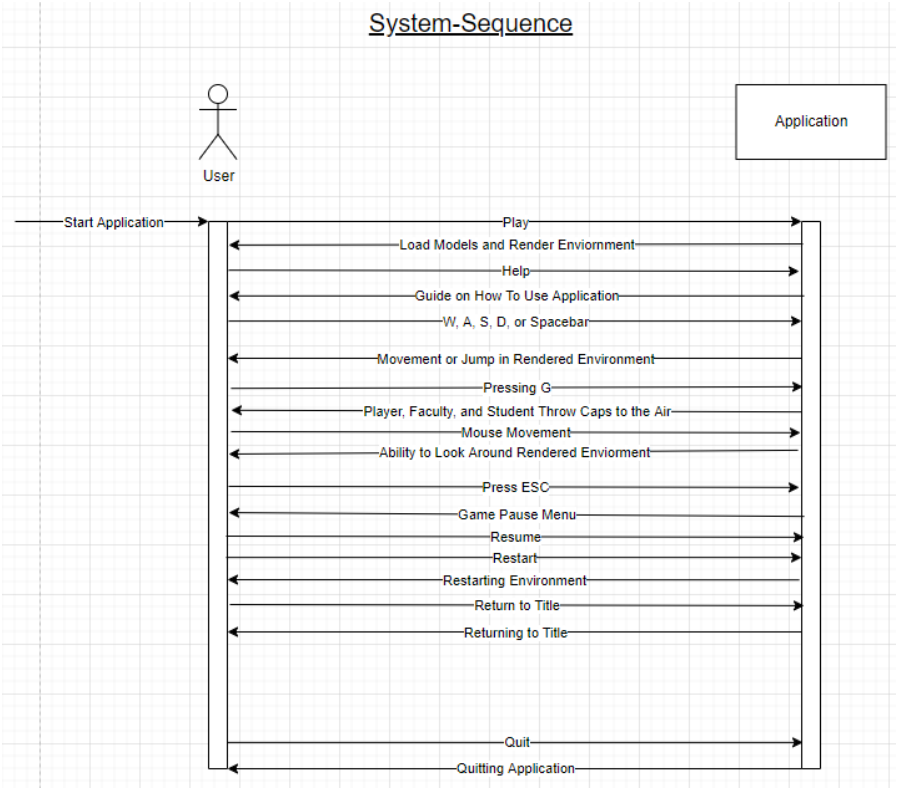
Since one of our AIFs is our knowledge in Unity, we did not need to put in much time into learning Unity. Only a few refresher videos on lighting in Unity were needed in order to create our desired game.

Detailed Design

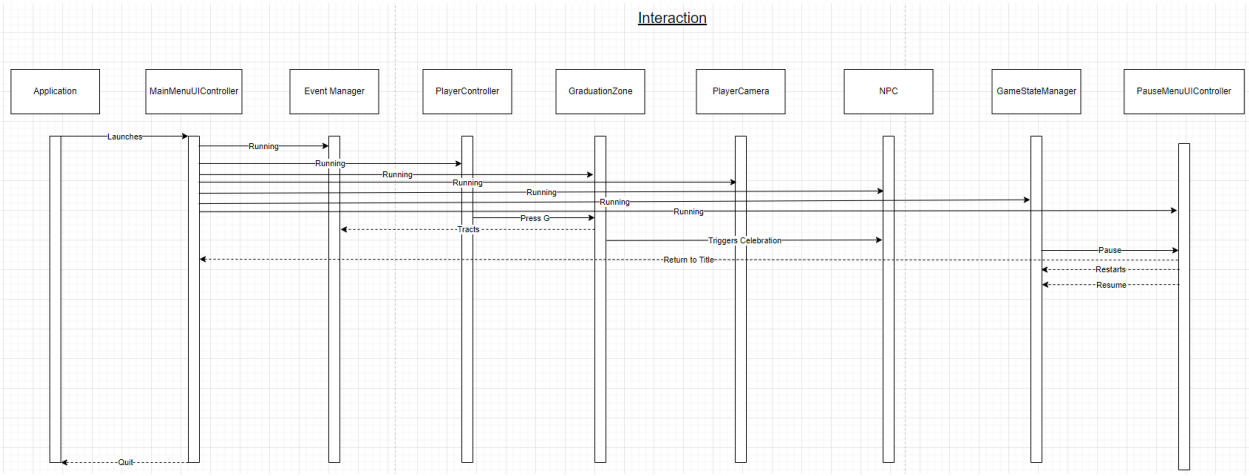
Detailed Class Diagram



System-Sequence Diagram



Interaction Diagram



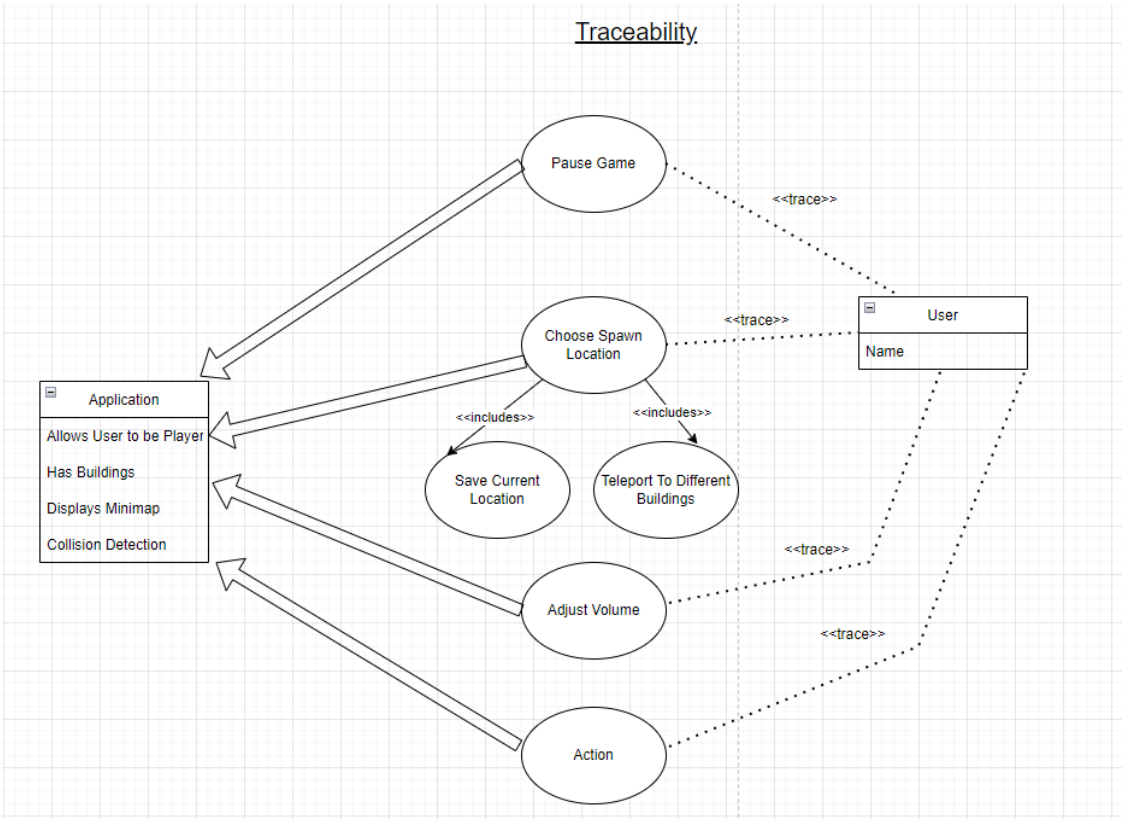
Traceability Matrix

Traceability Matrix

Project name: Fullerton-Commencement-Team 5

Functional Requirements			Use Cases									
Req #ID	Requirement Name		Usecase #	Usecase Name		Assign To		Priority				Current Status
FR-1	**The application shall allow a user to be a player.**		UC-1	User is able to launch the game and play as a player in the game		Eric Chu		High				complete
FR-9	Application should allow the user to set a spawn location		UC-9	**The player can visit specific locations and not waste time traveling on foot .The player can choose where they want to spawn in to quickly visit their desired location**		Eric Chu		Medium				Testing
FR-3	The application shall allow a user to pause the game.		UC-3	**User can pause when they need to take a brief break**		Eric Chu		Low				In Development
FR-10	**Application should allow to teleport to the different buildings of the campus**		UC-10	If a player is stuck at some place he is able to teleport any ground or building in the minimap.		Eric Chu		High				complete
FR-5	**Application will have building from California State University, Fullerton**		UC-5	**The game environment should consist of the major buildings found on campus of California State University, Fullerton**		Eric Chu		Low				In Development

Traceability Diagram



Construction: Iteration One

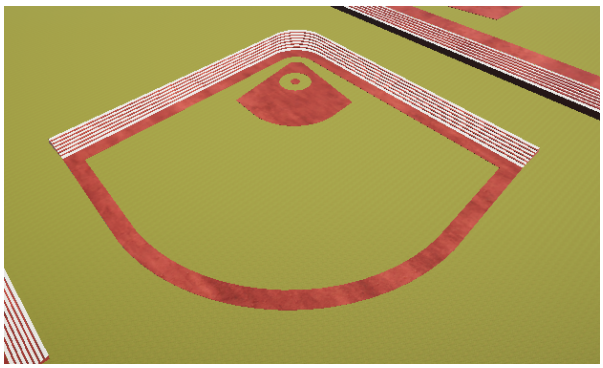
Development Case

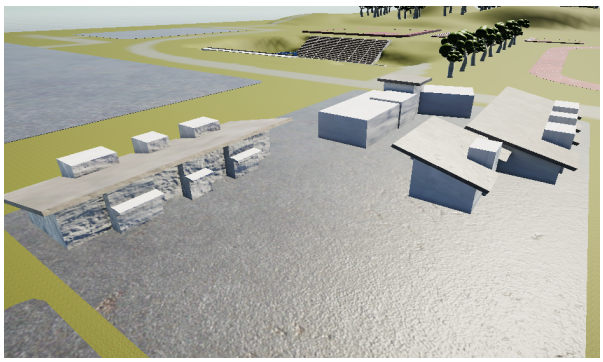
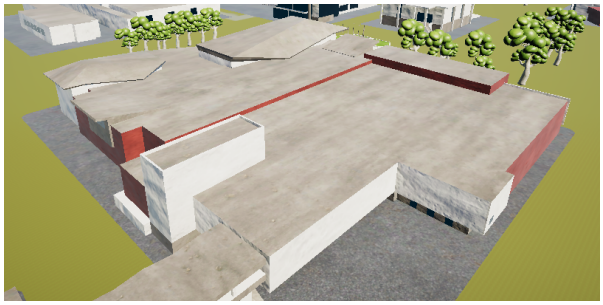
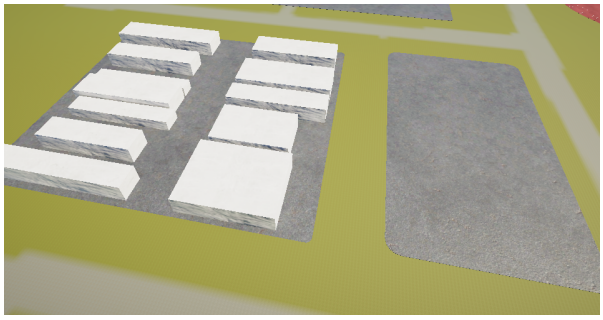
Disclipline	Artifact Iteration	Iteration 1	Elab E1..En	Const CL..Cn	Trans
Requirements	Vision	S		r	
Design	Model	S		r	
Testing	Test Model	S	r		
	Collision Matrix	S	r		
Environment	Development Case	r	r		

Modeling

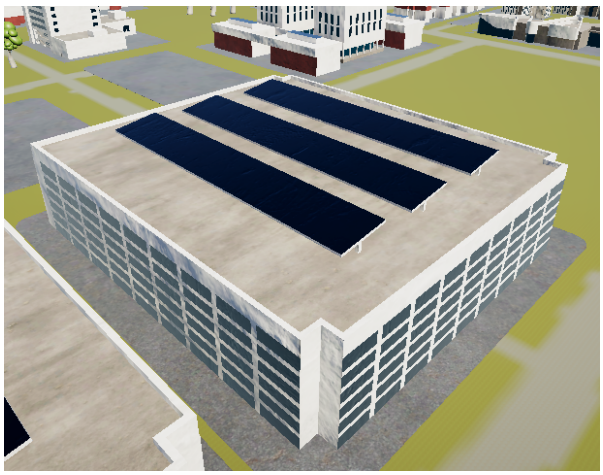
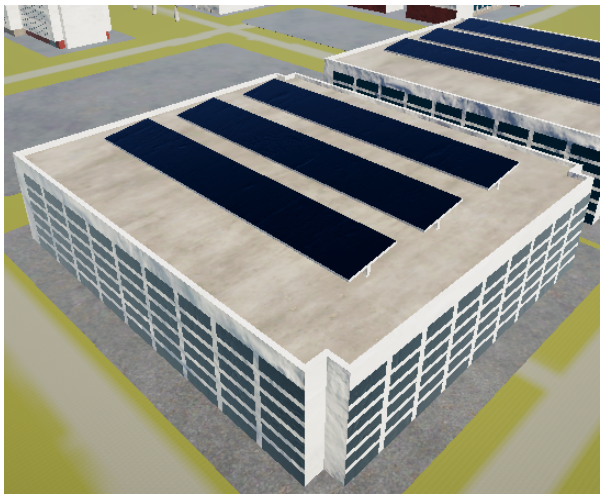
The modeling of the buildings/locations began on May 28th and ended on June 6th. The CSUF map was referenced to determine what buildings/locations to model. According to that map, there are 35 main buildings/locations. After looking at the map, we decided to leave out two buildings since they were separated from the majority of the buildings on campus. The omitted buildings are College Park and Titan Hall. The following images below are the buildings/locations that are included in the game. Google Earth was used to check the exterior of each building/location.

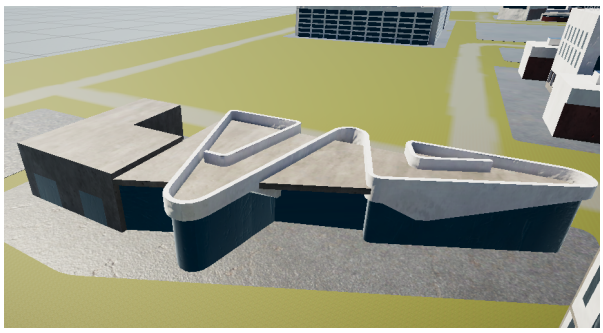
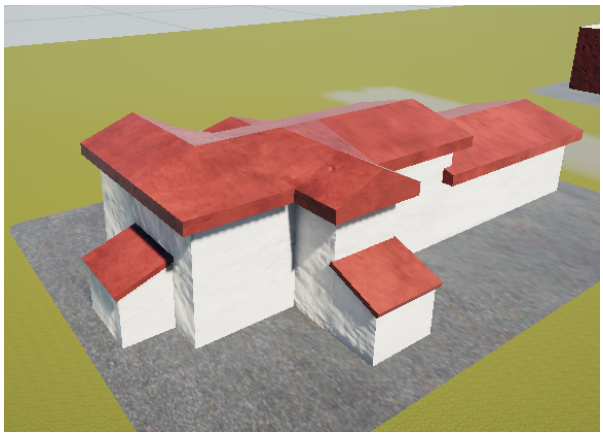
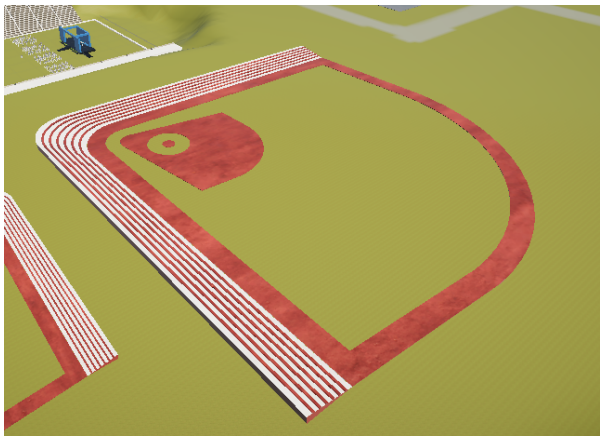
Name of Building/Location	**Image**

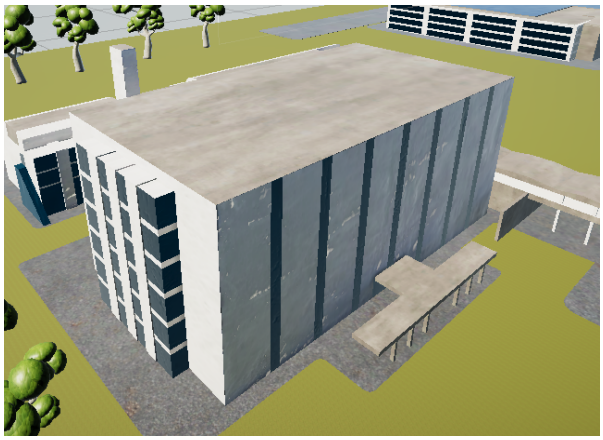
Anderson Field	
Bookstore/Titan Shops	
Becker Amphitheater	

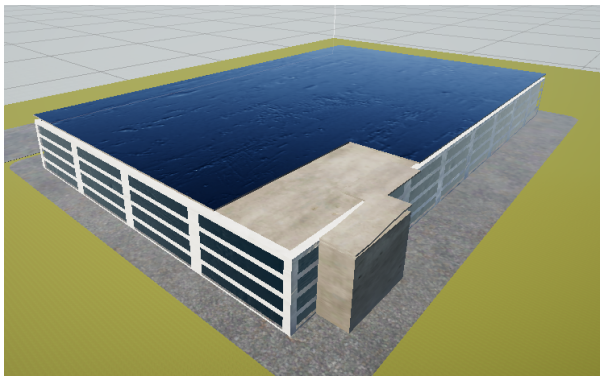
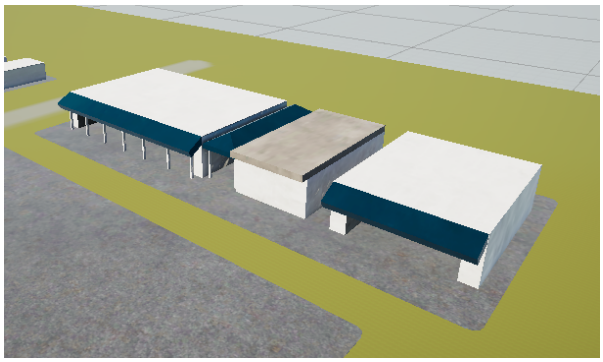
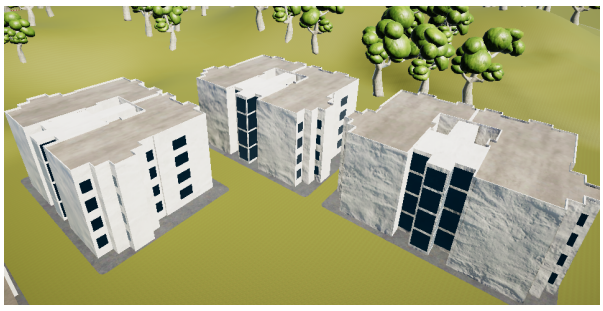
Children's Center	
Clayes Performing Arts Center	
Corporate Yard (Dumbo Downs)	
Computer Science	

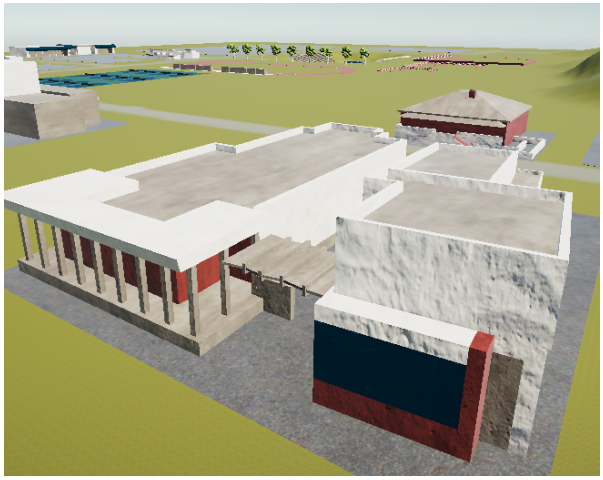
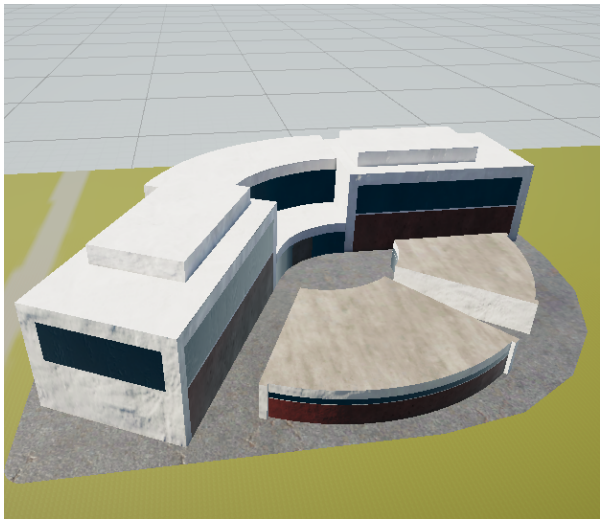
Dan Black Hall	
Engineering	

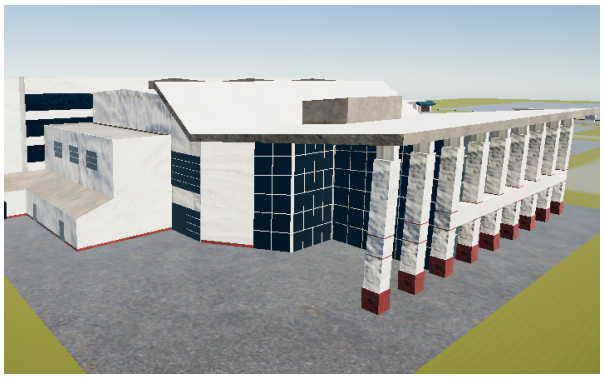
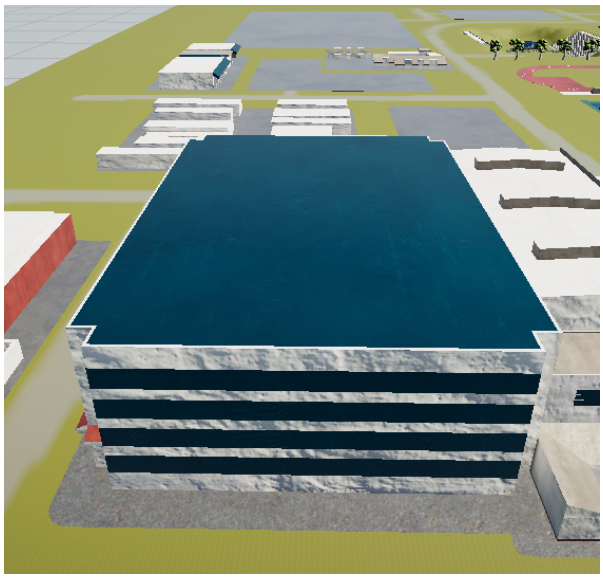
Education Classroom	
Eastside Parking Structure (North)	
Eastside Parking Structure (South)	

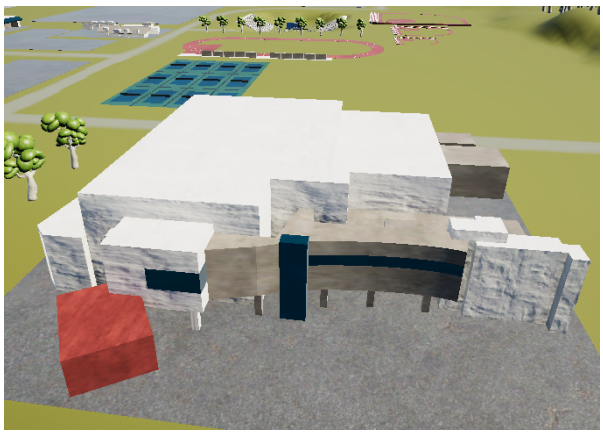
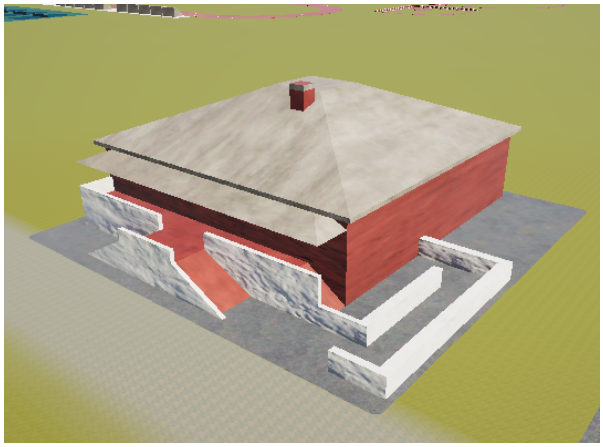
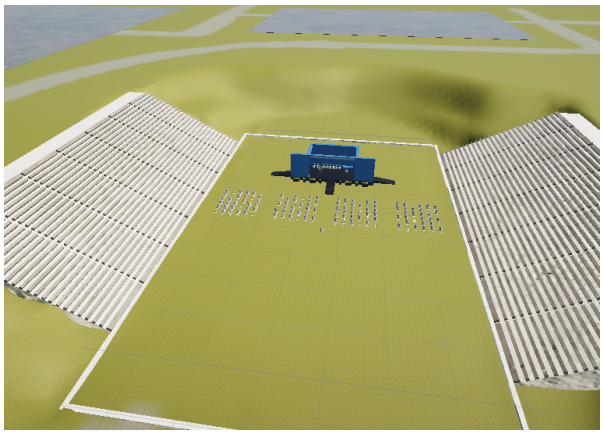
Gastronome	
Golleher Alumni House	
Goodwin Field	

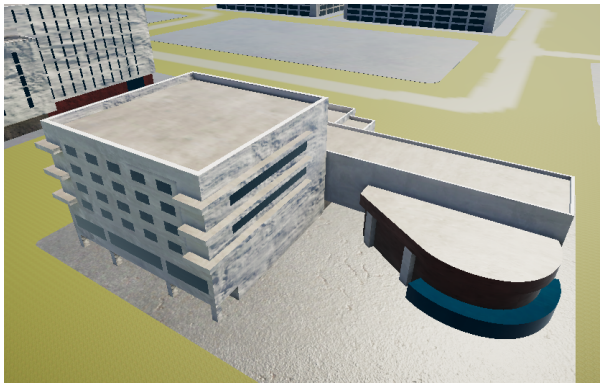
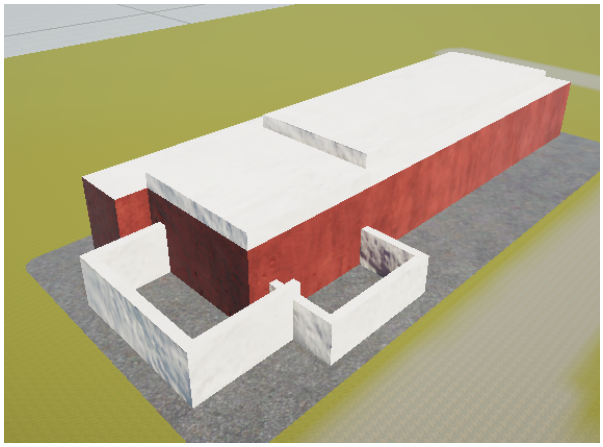
Humanities & Social Sciences	
Langsdorf Hall	
McCarthy Hall	

Nutwood Parking Structure	
Parking & Transportation Office	
Pollak Library	
Residence Hall	

Ruby Gerontology Center	
Student Wellness Center	
Steven G. Mihaylo Hall	

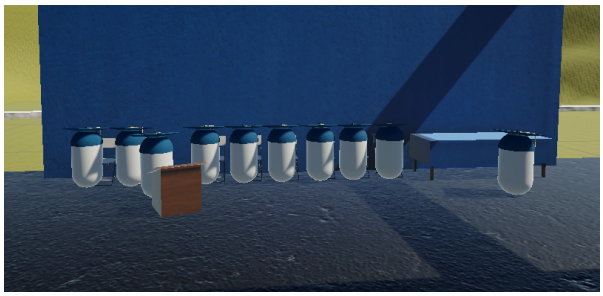
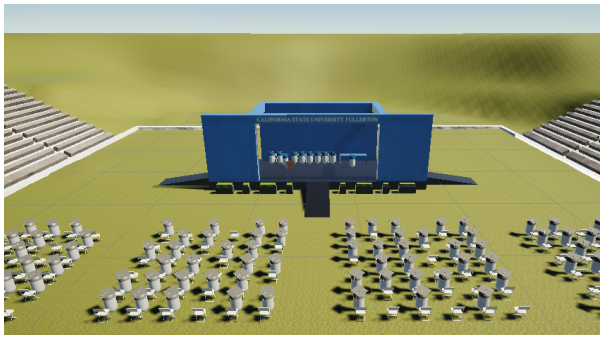
Student Housing	
Student Recreation Center	
State College Parking Structure	

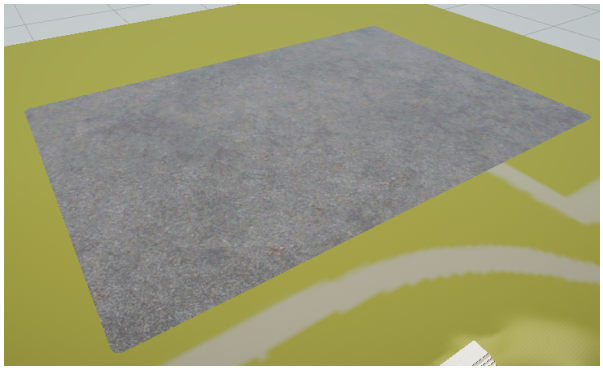
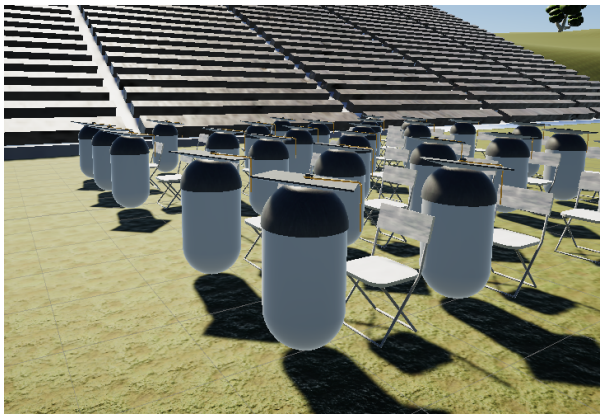
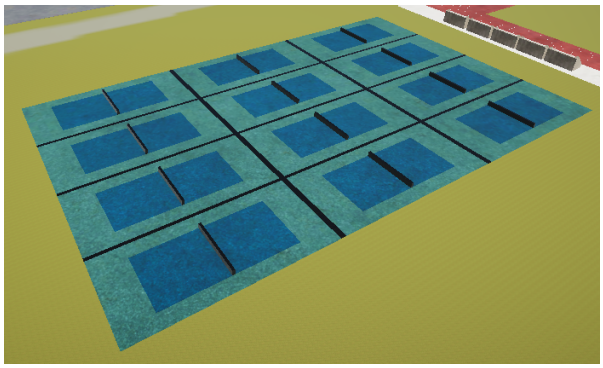
Titan Gym	
Titan House	
Titan Stadium	

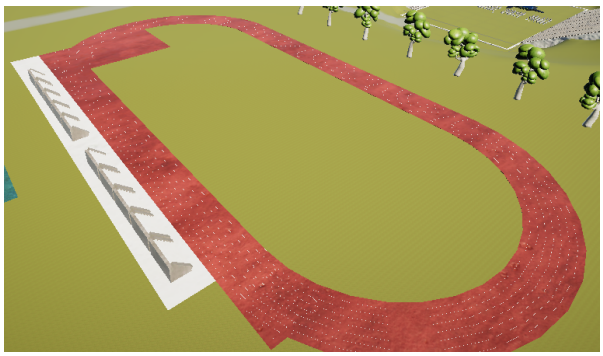
Titan Student Union	
Gordon Hall	
University Police	

Visual Arts	

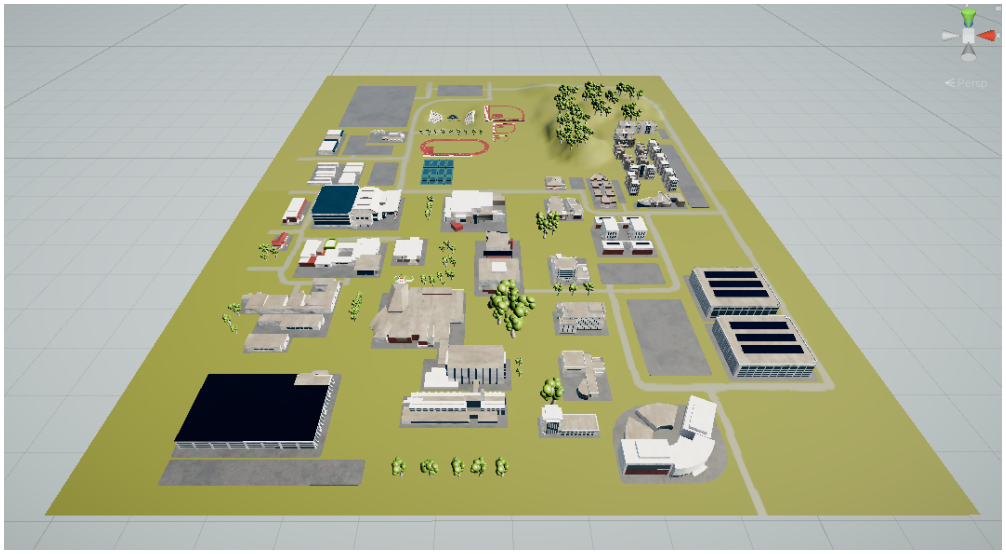
There were also other things modeled, including the graduation location, parking lots, trees, the graduation area. The images of these models are included in the table below.

Name of model	**Image**
Faculty	
Graduation Location	

Parking Lots	
Students	
Tennis Courts	

Track and Field	
Trees	

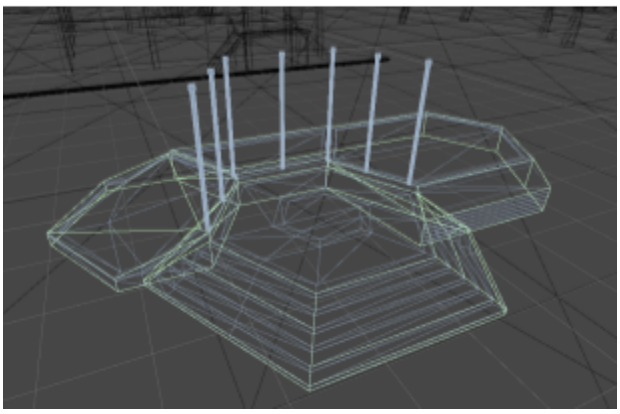
Once all the buildings/locations were created, we began importing the models to Unity. Each building was scaled and readjusted to be proportionally accurate in size to the average human model (2 meters in Unity). The ground was created using the Terrain tool in Unity. The image below is the final layout of the CSUF campus.



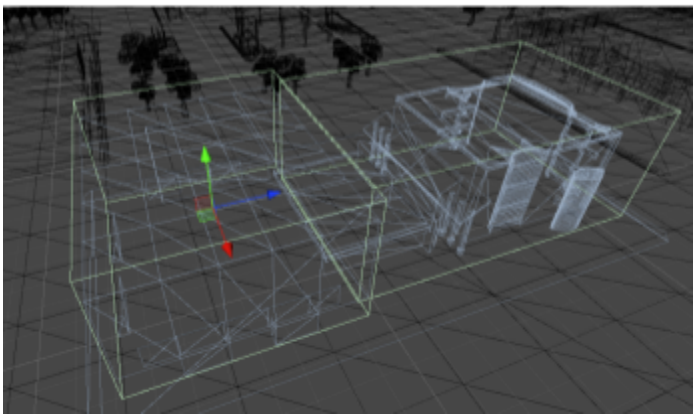
Collision Detection

All objects that the player object can collide with are attached with a type of collider component. Objects that are relatively rectangular in dimensions have a box collider. Objects that are more curved and have a lot of interactable corners have a mesh collider. The images below are examples of the box collider and mesh collider, colored in green.

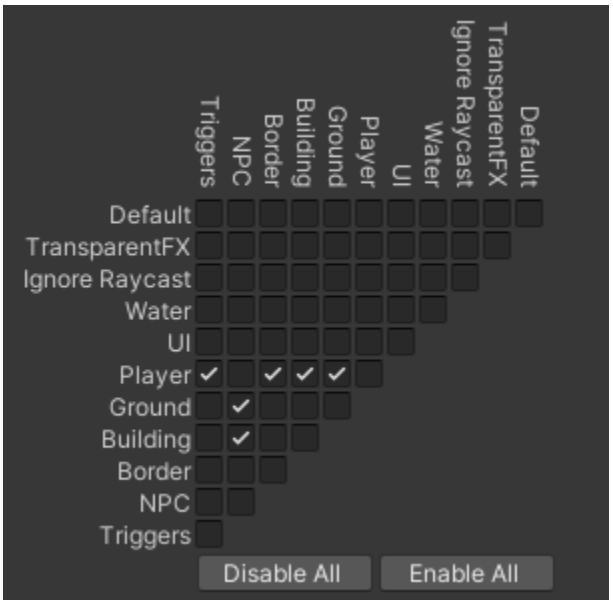
Mesh Collider



Box Collider



A collision matrix was then created in order to set how collisions are detected. The image below represents the collision matrix for the game.



The axes represent the layers inside the game. Every object is put in a layer, which allows us to define how objects interact with each other. A checkmark indicates that those corresponding layers are able to collide with each other.

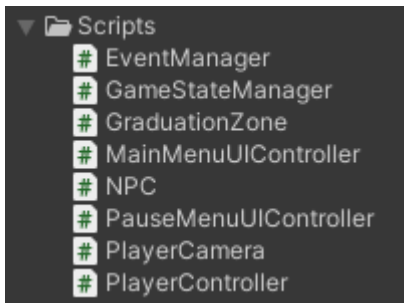
Construction: Iteration Two

Development Case

Discipline	Artifact Iteration	Iteration 2	Elab E1..En	Const CL..Cn	Trans
Requirements	Vision	r	r	r	
	Scripts/Code	S			
Design	Model	r	r	r	
	CMMI	S	r		
Testing	Test Model	r	r		
	Collision Matrix	r	r		
Environment	Development Case	r	r		

Scripts/Code

The image below represents all the scripts written for the game. The table below represents the descriptions of each script and their role in the game



File name/Name of Script	**Description**
EventManager.cs	EventManager contains all the events that occur in the game. There is currently a single event in the game, which is the Ending Graduation Ceremony. Events are

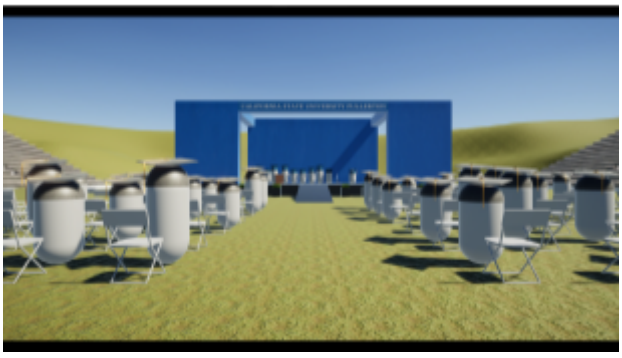
	invoked whenever their trigger occurs.
GameStateManager.cs	GameStateManager manages the various states of the game. There are currently two states in the game scene: Running and Paused.
GraduationZone.cs	GraduationZone is the class that triggers the Ending Graduation Ceremony event whenever the user presses ‘G’.
MainMenuUIController.cs	MainMenuUIController contains the methods that allow the user to navigate the main menu UI. It contains methods that are called by the buttons in the main menu.
NPC.cs	NPC contains the behavior and logic for the Student models and Faculty models. When the Ending Graduation Ceremony event is triggered, all NPC objects will jump and throw their cap into the air.
PauseMenuUIController.cs	PauseMenuUIController contains the methods that allow the user to navigate the pause menu UI. It contains methods that are called by the buttons in the pause menu.
PlayerCamera.cs	PlayerCamera allows the user to use their mouse to

	look around in the game.
PlayerController.cs	PlayerController handles user movement physics.

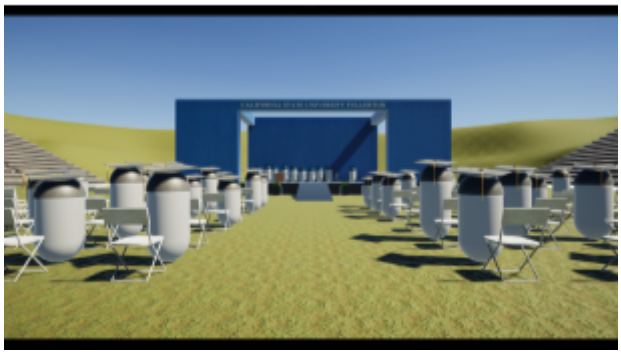
Lighting

Once the models were imported and placed in the correct position, we needed to bake the lighting in the scene. Baking the lighting in a scene pre-calculates the shadows and highlights for static objects in a scene. Although the game does not get a massive performance increase since the game is relatively small, it allows our scene to have ambient occlusion and better lighting.

Before Lighting Bake



After Lighting Bake



Transition

Deployment

Fullerton Commencement is deployed on the Windows platform as an executable and on the web. It is compatible with Windows 10 and above as well as web browsers that support WebGL 2.0.

User Guide

**It is highly recommended that you play the executable version.*

Executable Version

Setting up Executable

1. First, download the “Fullerton-Commencement.zip”
2. Extract the content in the zip file to another folder
3. The game executable is called “Fullerton Commencement.exe”

Playing the game

1. Run the executable called “Fullerton Commencement.exe”
2. When hitting the play button on the main menu screen, it may take a few seconds for the campus to load.

Browser Version

1. Visit https://fullerton-commencement.web.app
2. Wait for the game to load (it may take a few minutes)
3. When hitting the play button on the main menu screen, it may take a few seconds for the campus to load.

Controls

- *Movement***:** the movement keys are mapped to WASD.
    - *W*: move forward
    - *A*: move left
    - *S*: move backward
    - *D*: move right
- *E:* pause/resumes the game
- *G:* interact with mechanic

CMMI

CMMI is an abbreviation of Capability Maturity Model Integration, a process model that defines what an organization should do to set up the systems and other changes so that it can lead the organization to improve the performance. With the five capability levels it determines the baseline on which great products and services can be built on.

- CMMI helps the organization to improve from one maturity level to another level.
- If an organization becomes more mature by upgrading CMMI levels, it becomes competitive in nature via optimization and quality control.
- CMMI helps to reduce re-work and re-planning in the organization.
- Using CMMI organization can improve its product quality due to lower number of defects produced.
- It improves efficiency, speed and quality of software development process thus increasing customer satisfaction.

In CMMI processes are rated according some levels its called as maturity levels and there are five of these levels:

CMMI Maturity levels :

Level 0: Incomplete

At this level all the processes are usually ad hoc . And this level company's work may or may not work.

Level 1: Initial

At this level processes are usually in the very initial stage so processes are very chaotic in nature. This type of organization usually does not provide a stable environment. Accomplishments of the organization depends very much on the ability and competence of the individual employee in the organization. This is its biggest con because success of the organization is not depended on the proven processes but on the individual capabilities.Thus many times organizations are not able to repeat the past successes.

Level 2: Managed

The organization is said to achieve maturity level 2 if all the requirements are managed and all the processes in the organization are planned, performed , measured and also controlled. Companies at level 2 are still very reactive in nature but they are improved from level 1 so all the projects that companies make have product outlines.

Level 3: Defined

Organization who has achieved all the specific and generic goals of the processes assigned as level 3 matured organization.Here processes are well defined, well characterized and understood. In this level the organization set up some well established standards and procedures to follow. These standards and procedures provide guidance for the future projects and programs . At maturity level 3, processes

are written in more detail and more rigorously than at Maturity Level 2. These organizations are more proactive.

Level 4: Quantitatively Managed

A critical difference between Maturity Level 3 and Maturity Level 4 is the predictability of different processes' performance. At Level 4 maturity organizations work on the basis of data. All the processes here are data driven . Here quality of the processes and performance of the processes are understood in statistical terms and are managed throughout the life of the processes.

The performance of processes is controlled using statistical analytical and other quantitative techniques, and is quantitatively predictable.

Level 5: Optimizing

The organization is called level 5 if the organization completed all the specific goals from the previous levels 2,3,4 and generic goals from 2,3,1. At this level the organization's processes are very stable and flexible. This Level focuses on the continually improvement and improvisation. Another center of attention improving process performance through both incremental and innovative technological improvements. At this level quantifiable goals of the organizations are set for example increase software productivity by 10 percent or reduce the cost by 4% ect. The organization’s stability provides a platform for ever changing requirements and adaptability new challenges through innovation.Role of every employee in this

organization is to continual process improvement in every respect such as speed of the testing, reducing defects etc.

Companies that achieved CMMI:

- It takes several months to transition from Level 1 to 2.
- It takes 1 to 2 years for the organization to transition from Level 2 to Level 3.
- It takes even more 2 to 3 years for a level 3 organization to transit to Level 4 or 5.
- It requires a dedicated management commitment to process improvement; this is why to date, only a few organizations have achieved Level 5.
- Over 30,000 businesses use CMMI from over 106 countries, including the U.S,China, Italy, Chile, India, Australia, Egypt, Turkey, and Russia and many more have achieved some level of CMMI.
- The U.S government and its subsidiaries often require the organizations to achieve some kind/level of CMMI for getting the government contracts.

How CMMI is different from the agile processes you learn in this class.

- CMMI is used for optimization while agile processes are used for systematic software development.
- To achieve maturity level by level takes time in CMMI , To achieve higher level of maturity takes even more time on the other end agile processes are quick in nature as organization mature agile processes become even quicker.

- In agile processes the area of attention is on the software development, testing and iteration of above while in CMMI area of attention is on the processes and it's evolution from one level to another.
- Agile methods are circular in nature while CMMI is a linear methodology.
- CMMI is used for improving existing standards and procedures to become more mature while agile is used for developing new products and methods for the future.
- There are many face to face interactions that happen in this process called scrum meeting on the other hand CMMI is a very institution driven process where no daily meetings are needed.

References

Larman, C. (2002). *Applying Uml and patterns: An introduction to object-oriented analysis and design and the Unified Process*. Prentice-Hall.

Larman, C. (2012). *Agile and iterative development: A manager's guide*. Addison-Wesley.

CAL STATE FULLERTON

Team Charter

(Fill in the Team Charter now, follow it always, and submit this with your HW reports and Team Evaluation)

Course Title	CPSC 466 (59) Software Process	*All team members participated in the creation of this charter and agree with its content.* ***Date*** *06/04/2022*
Instructor	Dr. Chang-Hyun Jo	
Course Dates	05/31/2022 – 07/01/2022	

Team Members (Contact Information)

Name	Address (city, state, country)	Phone	Cell	Email
Eric Chu	Chino, CA	N/A	703-991-3480	kiyuen88ec@csu.fullerton.edu
Nitish Lele	Fullerton, CA	N/A	657-799-8272	nitishlele@csu.fullerton.edu
Ruilin Wang	Fullerton, CA	N/A	626-410-7298	rwang18@csu.fullerton.edu
Javier Canedo	Orange, CA	N/A	949-239-8474	jcanedo23@csu.fullerton.edu
Deangelo Aguilar	Ontario, CA	N/A	714-388-4660	dlo55@csu.fullerton.edu
Louis Kim	Irvine, CA	N/A	714-824-1787	louiskim19@csu.fullerton.edu

Team Member Skill Inventory (Areas individual members can contribute)

Eric Chu	• C#, C++ • Experience with game development and Unity 3D • Blender • CPSC 362, CPSC 462 (software engineering)
Nitish Lele	• Technical writing • CPSC 541
Ruilin Wang	Database development
Javier Canedo	C++, CPSC 362
Deangelo Aguilar	Testing
Louis Kim	• CPSC 362 • Qt Creator

CAL STATE FULLERTON

Team Charter

Team Goals (Project goals, team process goals, quality goals, etc.)

- Learn about software processes and software design and architecture.
- Acquire practical software engineering (people, process, tools, and methods) knowledge from teammates.
- Develop a strong, cohesive team and collectively produce the required assignments in a timely fashion.
- Produce and deliver a good final paper to the professor.

Team Roles (Define roles of each member to achieve goals)

Eric Chu	▪ Building the game ▪ architectural design document ▪ user guide, construction
Nitish Lele	▪ Documentation (Graduate Student portion) ▪ Scrum Artifacts ▪ traceability matrix
Ruilin Wang	▪ Documentation() ▪ Non-functional requirement(NFR)
Javier Canedo	▪ Documentation() ▪ Development Case
Deangelo Aguilar	▪ Documentation() ▪ Description of application ▪ functional requirement ▪ play testing
Louis Kim	▪ Documentation() ▪ Keep track of the required work products ▪ Use cases diagram ▪ High Level Class Diagram ▪ Traceability Diagram ▪ Sketch Detailed Design

CAL STATE FULLERTON

Team Charter

Ground Rules (Meeting schedule/locations, attendance expectations, agenda, assignment completion, communication methods, etc.)

- Will be communicating through Discord
- Scheduling meetings every week to go over our progress (Times will vary)
- Expectation for everyone to be able to attend the group meetings
- Agenda for right now is to finish the Team Charter
- Next agenda will be to go over the documentation/start HW1
- Completing our Team Charter assignment today (5/30/22)

Time Commitments/Availability (Pacific Time)

Eric Chu	▪ Monday - Wednesday, 6pm-10pm ▪ Thursday - Saturday, free all day
Nitish Lele	• Monday- Saturday 4pm- 10pm
Ruilin Wang	Monday-Friday, after 6 P.M && Weekend
Javier Canedo	Monday-Friday, after 2pm, Sunday after 6pm
Deangelo Aguilar	Monday -Friday after 6pm and Sunday
Louis Kim	• Monday 2:30PM - 9PM • Wednesday 2:30PM - 9PM

Conflict Management (What are potential conflicts that might arise among or between team members during this course? How will team members deal with these and other conflicts?)

- roles are clearly defined avoid confusion and merge conflicts
- if a team member does not show up for a meeting, message them
- conflicts should be discussed in team meetings
- To avoid the confusion, division of labor will be there before the project starts.

Risk Management (What are potential barriers to the achievement of these goals?)

- Identify threats and risks
- Evaluate each threat and risks (likelihood, dependencies, level of impact, etc.)
- Log and monitor progress to check for potential threats and risks
- Have a response plan for each threats/risks

Team Evaluation Criteria (List evaluation criteria that will be used to evaluate team members objectively.)

- Eric Chu will be evaluated through his progress with creating the game itself
- The rest of the members will be evaluated by their participation/effort of the documentation
- Team performance will be evaluated by scrum artifacts and burndown chart.

CPSC 466 – Summer 2022

Self-Evaluation of Team05

See instructions below.

Do not mess with the cells with red text. They are formulas.

You can customize the form (by adjusting the number of columns and rows, total, and average) according to your team's situation. However, total must be [100 x number of members]

Members / **Evaluators**	Eric Chu	Nitish Lele	Ruilin Wang	Javier Canedo	Deangelo Aguilar	Louis Kim	**Total**	**Comments on Your Evaluation on Team**
Eric Chu	100	100	100	100	100	100	**600**	All members did well.
Nitish Lele	100	100	100	100	100	100	**600**	All members completed their work proper manner.
Ruilin Wang	100	100	100	100	100	100	**600**	All members did their part well.
Javier Canedo	100	100	100	100	100	100	**600**	All members did their work well
Deangelo Aguilar	100	100	100	100	100	100	**600**	All members were active and participated evenly
Louis Kim	100	100	100	100	100	100	**600**	All members did well on their designated parts.
Total	600	600	600	600	600	600	3600	
Max	600	600	600	600	600	600	600	
Average	100.00	100.00	100.00	100.00	100.00	100.00	600.00	
Percent	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	
Signature	EC		RW	JC	DA			
Comments on Your Score Earned from Team								

[Instruction] Self-Evaluation of Your Team:
Submit one page of self-evaluation (by team). Evaluate your team members each other for this course. Give % to each other member that shows how much he/she performed his/her duty for the team.
Total % of a team should be [number of team members x 100%]. For example, assume that your team consists of 3 members, A, B, and C. If A gives 90% to B, then A or C should take A's 10% (for example, A's evaluation: A = 110%, B = 90%, C = 100%, Total = 300%). If A gives 120% to B, then 20% should be taken off from A or C (for example, A's evaluation: A = 90%, B = 120%, C = 90%, Total = 300%). In a similar manner, B and C also give their evaluations. Calculate an average of each member.
Give evaluation fairly. Do not take it as personal matters. For fair evaluation, use the evaluation criteria that you decided in your Team Charter.
Signatures are required regardless of agreement. If there is any missing signature, for convenience, I will assume you agree on it in advance. It is your responsibility to check the evaluation when it is submitted. If you do not agree on the evaluation, you still need to sign and say so in the comment. For this case, additional evidences need to be submitted.
Individual score will be graded based on both your team members' consensus and instructor's judgment. If you and your team members disagree, you need to submit additional documents (such as team meeting log, etc.), so that the instructor can grade it. Before you submit any document, you should show it to all team members (keep documents for this).
If your team members all agree, one page of self-evaluation is enough.