Assignment 4: Flight Simulator

Released: February 20
Due: March 13 @ 4:00 PM

Please start early as this is long assignment with a lot of details. We simply want to make sure that you have started the assignment, and we want to provide some feedback to you in time. Furthermore, you will be able to see some good designs from other students on Piazza.

Overview

Welcome to Assignment 4! In this assignment, you are going to build a flight simulator in Unity. You are going to construct a new environment, write script to support your plane, make a UI, and optimize the game to fulfill requirements from the Oculus Best Practice Guide! (What is Oculus Best Practice Guide? Your Assignment 3.)
Tasks to do in this Assignment - *(Read the Q&A at the end. READ IT NOW!)*

**Part 1: Environment**
Build an environment for your flight simulator
Rubric: (45 pts)
A. Skybox: 5pts
B. Terrain and details: 10pts
C. Fog effects: 5pts
D. Water: 10pts
E. Reference photos: 15pts

**Part 2: Flight Simulator**
Build your flight simulator and Scripts
Rubric: (55 pts)
A. Plane: 5pts
B. Acceleration / Deceleration: 10pts
C. Rotate (comfort): 10pts
D. Ray Casting: 10pts
E. Spheres for shooting game: 10pts
F. Shooting Game: 10pts
G. Stationary model - (Extra credit) 10pts

**Part 3: Make it a game!**
UI and tutorial (100 pts):
Have start menu: 5pts
Have tutorials/Instructions: 5pts
Displaying Score: 5pts
Sound: 10pts
Graphics: 25pts
Controls: 25pts
UI/Instructions: 25pts
Missiles - (Extra credit) 10pts

**Part 4: Optimization**
FPS seldom drops below 60: (-50pts~0pts)
Part 1: Environment

First of all, we need a unique environment for your flight simulator. The environment you build should have at least a sky, water, and a terrain. Remember, the purpose of this part is to let you play with Unity graphics and make your environment look good. Therefore, you are always welcome to add other things into your scene besides these requirements! Please go over all of the requirements before you start and take a look at the reference screenshots at the end of the document to get an idea of what we expect.

All the techniques you need for this part of the assignment can be found online. There are tons of useful tutorials on Youtube too.

1. When working on your terrain, make sure that you use at least 3 real life photos (or concept art) as your reference images, and consider the following questions: Where should you place the trees? What is the height of the trees compared to the mountain? How does your fog look, e.g. color, density?
2. Use a Skybox for the sky. It is even better if you combine it with other techniques, such as importing the cloud meshes into the game or write a script to simulate the moving of real clouds.

3. Use Unity’s Terrain Editor and build at least one terrain in the game (Similar to the terrain shown in the picture above). The terrain should at least:
   a. Have enough height details (it is suggested that you load it from a height map and further craft it, or else you will spend much more time adding the height information to this terrain using Height and other Tools)
   b. At least 3 different types of textures (search online or use the one in the package: /assets/import/environment, and you will find it in your assets)
   c. Normal map for the each texture.
   d. Plants (/assets/import/environment, and you will find it in your assets), with shadow

   You should use the Terrain Editor in Unity to build your terrain.

   **Do not directly import a terrain mesh or use Terrain Composer or World Composer to help you with it.**

4. Unity’s fog can be controlled in “Lighting”. Play with it, such as the color and density. However, you must be very careful that effects can be extremely expensive and ruin your game. You are likely to come back to fix your fog effect in Optimization.

5. Add some water to your scene (/assets/import/environment, and you will find it in your assets). It can be an ocean, it can be some stream. Use your imagination and make the scene look good!
Part 2: Flight

A flight simulator must take flight. You are going to build a plane in this part and write scripts to realize some functions: move, rotate, and shoot. In this Assignment, you can simply use a cube as your plane. After part 3, when you are able to build some simple models, you can create your own planes and replace this box with your modeled plane.

The next step is to write some scripts to make the plane fly.

1. You should use an Xbox controller instead of keyboard. Of course, you can implement both of them during your development, but we are only going to test the Xbox controller version.

2. Your plane should be able to do the following:
   a. Acceleration / Deceleration
   b. Rotate

You should implement these functions in your own way with the Xbox controller and Oculus Rift. Think carefully: What is the best way to rotate the plane to minimize discomfort? Should you use buttons or sticks? How fast should the rotation be? Remember to consult the Oculus Best Practice for suggestions.

3. Ray Casting is a very interesting technique. We are going to use it as the weapon of your plane, so that when you press the right trigger, your flight can “shoot” a ray. You should decide where the ray should shoot from.

4. Your simulator looks good, but we can make it better! You should scatter some spheres in the sky, so when your flight shoot at a ball, it should disappear. Also, these spheres should reappear after 5s after you shoot them. You decide the texture and the color of the spheres, but these spheres must be transparent, e.g. transparency from 0.4 to 0.7 is acceptable.

“Extra” credit opportunity (10 points):

Include any additional stationary models in your scene - pick from the asset store - for a 10 point extra credit. Try to make the model relevant to your scene.
Part 3: Make it a Game!

UI & Tutorial

Interface is an important part of a program. Before implementing, think carefully: should the UI/Tutorial follow head movement? Should the UI/Tutorial stay on the ground? How to present your UI/Tutorial so that people can see it in the Rift and don’t feel interrupted? Remember to consult the Oculus Best Practice Guide and share ideas with your teammates. If you are still confused, think about the games you have played before, and check This.

1. You should have a start menu, so that players can start by selecting that option or press a specific button.
2. Your Tutorial should explain to your players how to play the game. It can be either in game or in the sub menu, or both.
3. Your UI should show the score to players. After hitting a ball in your scene, your score should increase. There are multiple ways of displaying the score.

It is easy to implement a UI/Tutorial (some simple words), but in order to receive high points for the Instruction part in Overall Design, your UI/Tutorial needs to be clear and self-explanatory. If your user doesn’t know what to do when looking at your UI/Tutorial, then it is probably not a good design. Ask other people to try your design if possible. If you think words are insufficient to express your ideas, you can choose to display images to your users instead. (Is audio a good choice?)

You should assume that the user doesn’t know anything about Xbox controller. You should assume that it is user’s first time to use Oculus/Gear VR to play the game. You can assume that the user knows something about shooting game.

Please devote some effort into the UI and menus, as you can easily transfer them to your final project.

Sound

Sound is essential to a simulation. Add a sound file to your simulator. It can be
anything, such as ambient sound or background music. However, it should not decrease player’s overall experience. (For instance, you probably would not like to hear tiger moaning during your flight simulator)

**Overall Design**

“Overall Design” is a vague definition, but we are going to evaluate it along the following dimensions:

1. The detail of your scene. A generic terrain with very few trees and low quality textures will lose you points (unless it is your art style). Check Piazza if you are still confused. Please also note that sometimes screenshots cannot reflect the quality of your scene.
2. Your interface/instruction system works as expected: a first time player without any knowledge of your simulator should be able to know how to play the game.
3. The overall experience of the flight: How is the rotation? What is the limit of the speed? Is collision implemented? Is my plane acting like a real plane (not a car, a box, etc.) (hint: it is difficult for a plane to perform one kind of rotation.)?

**“Extra” credit opportunity (10 points):**

Remember the ray cast in part 2? Now, you are going to replace the ray cast with real missiles. When pressing the fire button, your missile will be fired from your plane to the target, and when they hit, the ball will disappear.

These factors will be considered when giving you the extra credits, but generally, the more realistic the better:

1. The speed of the missile: It should not be linear.
2. The shape of the missile. You can use a box, but be careful about the size of the box.
3. It is not necessary to implement the explosion effect, but you will rock if you do it!
4. Add trails to your missile?

**Optimization**

We have a Titan Black graphic card, but it is still not good enough. In fact, nearly all graphic cards have to work hard to satisfy Oculus Rift.
The first objective is to increase the FPS (frames per second) so that it rarely drops below 60. According to Oculus Best Practice, we need stable FPS to maximize our experience. If your simulator has overall FPS above 60, then you are probably fine for this part. Note that if your FPS is only around 30 and has produced obvious lag, you will lose at most 50 pts, depending on whether your game is playable or not.

You will need to consider the following strategies:
1. Delete some of your assets or lighting in the simulator.
2. Lower the effect (shadow, fog) of your scene
3. Check pixel error of terrain (see terrain settings)
4. Are there some assets in your scene that have too many polygons?
5. Here

Submission instructions: (remember the data folder!)

Step 1: Create a .unitypackage file

1) Save your Unity scene in the Assets folder with the title “CS498HW2”
2) Using the editor, find the created scene in the Project menu
3) Right click on the scene and select Export Package…
4) Export the file using default settings (“Include dependencies” should be checked by default)

Step 2: Create a standalone game build

1) Go to Edit → Project Settings→ Player. Make sure the “Virtual Reality Supported” box under Other Settings is checked.
2) Go to File → Build Settings
3) Click “Add Current”. This will add the current scene to the build. You must have saved the scene to the Assets folder for this to work (you should do that anyways).
4) Hit “Build”. Save the project to C:\Users\student’s netid\project name, rather than your networked folder.
5) This should create an executable (.exe) for running the build, as well as a folder containing your scene data. Make sure this executable runs correctly on the Rift before submitting.
**Step 3: Zip the files and submit them through Compass**

1) Create a zip file containing 3 items:
   a) The .unitypackage created in Step 1
   b) The .exe and data folder created in Step 2
   c) A README.txt file containing any special instructions or notes you think are relevant for evaluating your assignment.

2) Name the file by separating NetIDs with underscores._cs498sl_HW4.zip EXAMPLE: If john1 and carmack2 worked together, the file should be called john1_carmack2_cs498sl_HW4.zip

   **DO NOT SUBMIT YOUR ENTIRE PROJECT FOLDER**

**Sample terrain screenshots**

*Fine*

Scale seems a little off in some places but good otherwise
Good

Acceptable
Q&A

Q: Can you debug my code?
A: NO

Q: There are not enough textures and maps inside the package provided by Unity.
A: Google and download the maps and textures

Q: I don’t know what a map is
A: Google

Q: Why there are negative points on optimization?
A: When people start to play your VR game and your game’s FPS makes them sick (Yes, ‘some’ game companies usually mess up on this point), their interests on the game will highly decrease, especially when there are tons of other alternatives out there.

Q: Will there be an extension?
A: NO

Q: Can I develop them in Unreal/CryEngine instead of Unity?
A: Good Idea! You are highly encouraged! You can learn some Unreal/CryEngine stuff and develop in the new environment. Note that Unreal’s coding standard is different from the C++ programs we write. And in fact, a large majority of people (including me in most cases) who uses Unreal only uses its Blue Print visual script, which simplifies program’s logic into nodes, so you do not need to “write” any code. Read this article before considering learning new stuff.
As for CryEngine, I personally do not recommend it to people who has deadline of around two weeks – it is not that friendly to new users. However, if you are more experienced in CryEngine, you are more than welcome to use CryEngine. Post on Piazza if you have any question concerning Unreal/CryEngine (except for coding).