

CMSC 388G Virtual Reality Game Development
Programming Assignment 1: Introduction to Unreal Engine and C++
Due Date: 11:59 PM, February 22, 2019

Project Submission:

- 1) Delete the “Intermediate”, “Saved” and “Binaries” folders right before submitting.
- 2) Name your folder “INSERTFULLNAME_CMSC388G_Project1.zip” and make a README.md file where you will put your answers to the questions.
- 2) Place all files for the project in the folder and ZIP up the folder. You will submit your project via the submit server. To submit a zip file, login to the submit server webpage and look for the link to make a *web submission*.

Project Description

The purpose of this assignment is to **get acquainted with C++ and use the VR lab space to test your project**. You will be modifying two C++ classes in order to get familiar with working in both Unreal and C++. The starter code for this project can be downloaded from the Github Repository under the folder called “Project1” (<https://github.com/MikhailSorokin/CMSC388G---Virtual-Reality-Game-Development-Class-Examples->). The code in these projects will serve as a template for most of the projects in this course, so it is important that you read and understand what is happening. As always, feel free to ask questions in office hours or on piazza.

In Unreal Engine, there are two ways to develop. One is with blueprints and another way is with C++. In general, C++ is 10 times faster at processing code than blueprints. **The goal of this project is to come up with several techniques to convert existing Blueprint code into C++**. You can see a reference blueprint project by:

1. Clicking the **Yellow Launch Button**
2. Going to the **New Project** tab
3. Going to **Blueprint -> Virtual Reality**
4. Save the project somewhere on your machine and load it.

NOTE: You may be able to finish the project without seeing a blueprint reference, but it may be challenging to do so unless you are already familiar with the UE4 editor.

The specific classes and functions you will have to rewrite in C++ include:

1. MotionControllerActor + Teleportation
 - a. **void ClearArc()** - Clears the blue laser arc every frame.
 - b. **FNavMeshNode TraceTeleportDestination()** - Finds the points on the NavMesh (*RecastNavMesh-Default* object in Map) and teleports to first available point.

- c. **AActor* GetActorNearHand()** - The MotionController interacts with this to get the object nearest the hand to set the animation state of the hand.
2. Pickup Cube
 - a. **Constructor** - Set Simulate Physics. By default, this cube doesn't have physics simulate it. Will need to figure out how to go about doing that.
 - b. **Pickup(USceneComponent* AttachTo)** - Attach the cube to the parent object of the hand controller.
 - c. **Drop()** - Removes the attachment of the cube associated with the specific hand controller that you grabbed the cube object with.
 - d. Placement of several cube objects into the scene with the Blue Material reference that is in the project.

The classes that are made for you are the following. What is italicized is stuff you will have to edit:

- *CustomMotionController*
- CustomPickupActorInterface
- *PickupCube*

They can be found in the following relative path to the root folder: **Source/Project1**

Useful Functions and Classes You can Use

Teleportation:

UGameplayStatics::PredictProjectilePathByObjectType() - Predict the arc of a virtual projectile affected by gravity with collision checks along the arc. Returns a list of positions of the simulated arc and the destination reached by the simulation. Returns true if it hit something.

NavMeshes - Press p to see the navigation grid in the editor. This will be useful for the TraceTeleportDestination class. Look at the following link to see how you can use NavMesh further -

<https://answers.unrealengine.com/questions/541084/changing-navmesh-properties-breaks-vr-teleporation.html> Code reference:

<https://api.unrealengine.com/INT/API/Runtime/NavigationSystem/index.html>

DoesImplementInterface(UObject* Class) - This will be useful for interacting the MotionController actor with the PickupCube.

Pickup Cube:

Simulating Physics -
<https://api.unrealengine.com/INT/API/Runtime/Engine/Components/UStaticMeshComponent/index.html>

C++ Interfaces - https://wiki.unrealengine.com/Interfaces_in_C++

Questions

You have to answer these questions in a separate document to help you get a better understanding of the project. Name this document **README.md**:

- (1) - What classes do APickupCube and ACustomMotionController extend, respectively?
- (2) - In the MotionControllerActor blueprint, in what commented node section is "Clear Arc" and "Trace Teleport Destination" being called? What about GetActorNearHand()?
- (3) - What is the point of the "CustomPickupActorInterface" class? (This should be very brief, 2-3 sentences)

Grading

Your program will be graded based upon correctly performing all tasks that are mentioned above. All code that you write should be well commented. Your code is judged subjectively based on the simplicity and clarity of implementation. An implementation that is easy to understand, but has few minor bugs will be scored higher than a messy implementation with the same number of minor bugs.

Markdown

While you won't really be graded on code comments and markdown style, it is important to make the Markdown readable when answering questions.

<https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>