**AG 131  
Lab 2 – Cars, Planes, and Polymorphism**

Read the entire lab before starting. The lab write up is a “lose” write-up of required code elements. As such, not everything is covered in the specification. It’s is your responsibility to find these holes and to implement them. Consequently, if you think you can implement something better than written below, try it.

**Readings:**

<https://msdn.microsoft.com/en-us/library/5tbh8a42(v=vs.110).aspx>

<https://msdn.microsoft.com/en-us/library/xfhwa508(v=vs.110).aspx>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/>

<https://docs.unity3d.com/Manual/ExecutionOrder.html>

<https://docs.unrealengine.com/en-US/Gameplay/Framework/index.html>

<https://docs.unrealengine.com/latest/INT/API/Runtime/Engine/GameFramework/AActor/>

<https://docs.unrealengine.com/latest/INT/API/Runtime/Engine/GameFramework/APawn/>

<https://docs.unrealengine.com/latest/INT/API/Runtime/Engine/GameFramework/AController/index.html>

<https://docs.unrealengine.com/latest/INT/API/Runtime/Engine/GameFramework/APlayerController/>

**Part 1: Written Work**

1. Several pages from the Unreal Engine Documentation has been given to you. Review them and other documentation. Provide a diagram that shows the basic relationships of the core gameplay framework classes in Unreal Engine.
2. Compare Unreal’s Actor and Functionality in MonoBehavior. What are the core features that are missing in the MonoBehavior Class?
3. If you were to implement Pawn, Controller, Player Controller classes as scripts in unity, list the most important pieces of these classes you need?
4. What are the essential pieces you will need to implement for “possession” by the Controller classes

*Hint: If you do the readings, you might just find the answer you need…*

**Part 2 – System Diagrams**

* Diagram all of the required basic components for the following game types below.
* Express Each Game mode as an extension or inheritance of the previous one.
* You should have an image document for each game type.
  + Use a UML diagram tool to help build this.
    - A good tool is: https://www.draw.io/
* Simplify your diagram where you can by collecting things into categories.
  + Example: You don’t need to list all weapon types, a “weapon” class to abstract all of them is fine
* Game Modes to Diagram
  + DeathMatch (DM)
  + Team DeathMatch (TDM)
  + Capture The Flag (CTF)
  + Vehicle CTF (VCTF)

**Part 3:**Import the PWFramework Package into a new unity project.

* Write a Projectile Script (if one doesn’t exist)
  + Inherit from Actor
  + Has a Lifetime
    - Default this to 5 seconds.
  + OnTriggerEnter
    - Right now just remove the projectile
    - No other interactions
* Write CarPawn Script
  + Inherit from PWPawn (instead directly from MonoBehavior)
  + Members to Add:
    - Public Transform (or GameObject) for Where Projectile is spawned
    - GameObject for Current Projectile to spawn
  + Override the Input methods in PWPawn
    - Horizontal and Vertical to move and turn (rotate) the car.
    - Fire1 to shoot a projectile
      * Okay to assign a public member in the editor or the projectile
      * Make sure you use the Static Method to in the Actor Class to spawn the projectile with an Owner.
* Create 1 Car Prefab.
  + CarPawn Script is at the root game object.
  + Have RigidBody at the root game object.
  + Use the Car Model Prefab as a child game object.

When you complete this, you should be able to set up a scene with a fixed camera and have one car move around the scene and shoot a projectile.

**Extra Credit:**

* Raid the Asset Store for free assets
  + <https://assetstore.unity.com/search?order_by=relevance&q=category%3A54&q=price%3A0&rows=500>