**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://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. *(This includes other pages linked.)* 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:**

In Project Settings - Input

Player 1:

Bind Fire1 to Q  
Bind Fire2 to E  
Bind Fire3 to Z  
Add Fire4 to C

Setup Player 2:

Add P2Horizontal to J/L  
Add P2Vertical to I/K

Add P2Fire1 to U  
Add P2Fire2 to O  
Add P2Fire3 to M  
Add P2Fire4 to >

Import the BasicFramework Package into a new unity project.

* Create 2 Projectile Prefabs
  + A Projectile script has been given to you in *PWFramework – Classes*.
* 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
      * Make sure you use the Factory Method to spawn the projectile with an Owner. (This is a Static Method in Actor)
* Create 2 Car Prefabs.
  + You have a red and blue car to use.
    - Use the Car Model Prefab as a child
  + CarPawn Script is at the root game object.
  + Have RigidBody at the root 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:**

* Replace the Car Pawn with Space Ship Pawns
  + Movement stays on the 2D Plane.
  + Space Ships add thrust in a direction and keep moving in that direction
  + When Vertical value is negative reduce velocity of the ship.
    - Have a public member to define % to decelerate at.
    - Have a public member that defines value when the velocity speed is below, you’ll set the velocity to zero.
* Replace the Car Pawn with Airplane Pawns
  + Break free of the 2d Plane!
  + Have Arcade like controls (Not Flight Sim Controls!)
    - Fire 1 – Thrust Up
    - Fire 1 – Thrust Down
    - Vertical – Pitch the plane up and down
    - Horizontal – Rotate the plane left and right