Seminar Sheet 2

Dr Aaron Bostrom

September 16, 2019

In Lecture 2 we considered how functions and classes work at a basic level. We look at the ideas behind passing values by reference and by value into functions. How functions can change and manipulate the state of objects and variables outside of their scope. We considered how we can build upon structs to create classes that contain member variables similar to structs, but also contain methods that can act upon them. Finally we learned about public and private access modifiers. How we can use access modifiers for variables and functions to protect member variables from outside tampering.

BSc Games Development is about learning core computer science techniques, and how they can apply directly into developing games mechanics, or systems. One such system are interactive objects. Within our games we may want to create particular objects (in UE4 they are called Actors) that we can interact with. These objects will need a common set of variables and functions so that we can manipulate them or collect them.

Specify a simple game concept, decide on the game play and decide how the game play will affect the types of classes and data you may need to represent the collectibles, and perhaps the types of functions you can use to interact with them.

One such item that you may want in your world is a button. This button depending on it's state may do two different actions.

Problem 1. Design and create the UML diagram for the Button-like object in your game. What UE4 class will you inherit from, what type of data and functionality will you need to store and manage.

Problem 2. Implement your UML diagram. You can create this in either/both C++ and Blueprint.

Creating a simple test level or arena is ideal for this. Try to ensure your code will be re-usable. Think about how non-programmers may interact with this.

Problem 3. Now that you have thought about and created a toggle button in your game. Think of an additional interactive object that you may want to create. This could be a door or an elevator. Try and build an object that may need to have a state, or be usable etc.

First design and create the UML diagram. Then implement it in either/both C++ and blueprint.

Problem 4 (Extension). Design and extend your button (or similar) class such that it can link to your state based object. Show how your UML diagram will need to change to accomodate this additional functionality.

For example: Extend your button to be able to open your door. Make this sufficiently generic that your button and door can be linked together. Think about how non-programmers may interact with this.