Seminar Sheet 4

Dr Aaron Bostrom

September 16, 2019

In Lecture 4 we considered how inheritance works, and how we can structure the hierarchy of our objects to gain the appropriate variables and inherit particular functionality.

BSc Games Development is about learning core computer science techniques, and how they can apply directly into developing games mechanics, or systems. UE4 is a large multi-level inheritance hierarchy, consisting of UObjects, Actors, Components, Pawns and Characters etc. All of these objects inherit some functionality from their predecessors.

In a previous worksheet we looked at creating objects to represent interactive items within our world. Often items that we create will have similar and identical properties or data. Designing good inheritance hierarchies can save code creation, and reduce repetitive code.

Specify a simple game concept, decide on the game play and decide how the game play will affect the types of classes, and how they will inherit and form hierarchies. You may want to consider the common, and different functions and variables each of your objects will have.

Problem 1. In the lecture we considered different items in games, swords and guns and potions etc. Design an inheritance hierarchy that could represent different animals, and creatures in a game world. With a starting base Monster, and different child classes.

Produce a UML diagram demonstrating the hierarchy, and how you can use virtual functions with a base class, that will be overridden in your child classes.

Planning inheritance and considering the types of behaviour you want to emulate before you start programming is vitally important in having a flexible code base.

Problem 2. Implement your UML diagram heirarchy in the form of classes.

You can do this in C++ or Blueprint or both. You may wish to show the workflow from blueprint and C++, and how they work together.

Problem 3. Now that you have an inheritance heirarchy and some sub classes, try to refactor your hierarchy. Have you run into any problems or limitations of single inheritance? Have you got duplicated behaviour?

Provide an updated UML, and demonstrate the changes you wish to make to your code and it's associated structure.