Seminar Sheet 5

Dr Aaron Bostrom

October 28, 2019

We have considered how inheritance, classes, structs and general data behave in C++.

This seminar sheet is a culmination of all the past weeks of work undertaken. Combining elements form each to be able to create a complex system, that is common in a number of games.

In previous worksheets we looked at items, and actors and components and how they function in unreal engine.

The aim of this work sheet is to create an inventory system to store, manage and handle objects and items in the game world. You will need to consider how your game will handle items and the plyaer owning them. Are items unique, can they only spawn once, and never again. Can you have stacks?

You may want to design a simple inheritance hierarchy for the items in your game. You may want to consider whether to type them with enums instead.

You can choose to implement this seminar sheet in either blueprint, C++ or both.

Problem 1. Design a set of in game items or collectables, outline the types of statistics you want to record, how will you handle the different items, and how do they fit into your game world / idea.

Use UML diagrams to explain the different types of items you want to represent.

Problem 2. Implement and create multiple items that can be picked up and collected by the player. Place them in the world, and allow the player to run over and collect them. (Do not worry about storing them yet).

You may want to have them randomly spawn in your game level, or you may want to place them.

You may want to create an item spawner, consider how a designer might interact with this. (You may want to seek inspiration from DND or RogueLikes etc.)

Problem 3. Design the Inventory system in UML, present a flow diagram for show casing how you for see your API being called.

Create an inventory system. This is required to store the item and it's current quantity. Design and implement an API to deal with adding and removing items, etc. Justify the various functions you may need. Consider how your player class / player controller may want to interface with this system.

Hint: Research a simple Inventory system, and consider how this can inform your API design.

Problem 4. Plugin the functionality from earlier where a player could overlap with items in the world. Allow them to run over and collect items place or spawned, and update the inventory witht the relevant information.

Can you setup a test scene to ensure that you have programmed this system correctly?

Does the player have a limited inventory system? Do they have max stacks of a particular item?

Problem 5 (Extension). Consider how you could design a simple UI to display your Inventory.

What simple functionality can you provide to the elements in this UI. Drop items, merge stacks, split stacks.