CMIS 2720 Data Structures and Algorithms for Games

Ying Zhu

Assignment #3

Due date: March 8, 2023

In this assignment, you will learn the following:

- How to manipulate a queue
- How to manipulate a tree

General information

1. You must write your programs in C#.
2. This assignment contains two separate programs: A3a and A3b.
3. Submit a zip file to iCollege under the folder Assessments → Assignments → Assignment3. The zip file should contain two separate C# files:
    a. firstname_lastname_A3a.cs
    b. firstname_lastname_A3b.cs

A3a requirements

Background: You need to shoot shells using a cannon in a game. A certain number of shells (**numShells**) are created at the beginning of the game, and you will recycle these shells. No new shell will be spawned, and no existing shell will be destroyed. This is usually more efficient than creating and then destroying shells over and over.

The shells are stored in an ammunition queue. Each time you need to fire a shell, a shell is dequeued from the queue. Because it takes time to reload, this shell will not be available for certain seconds (**delay**). After a shell is dequeued for **delay** seconds, the shell is enqueued into the ammunition queue again. This applies to all the shells.

If the ammunition queue is empty when you try to fire a shell, you will need to wait until at least one shell is inserted into the queue. To balance the game, you want to control how quickly a player can fire a cannon. If a player can just hold the fire key to fire endless shells rapidly, the cannon is too powerful. On the other hand, if the cannon is too slow, it's not fun to play. You need to find the right balance so that the cannon is not too powerful nor too slow. To do this, you need to find proper numbers for **numShells** and **delay** through experiments.

1. Write a program to simulate an ammunition queue discussed above. If a user presses the Enter key, display a message saying that a shell is fired. If the user presses the ESC key, the program exits. Adjust the numbers for the variables **numShells** and **delay** so that this virtual cannon is not too fast nor too slow. It's your call to find the right balance.

2. You do not need to create a game. Do not create any graphics. Your program just needs to display a message saying a shell is fired. The important thing is to find the right firing speed for the cannon.

3. You must implement the program with queues. You don't have to implement the queue from scratch. You can use the Queue<T> class in C#/.NET.
    a. I will check the source code. Just printing out the messages without implementing the above idea will get 0 credit.

4. Write comments in your code. This part will be graded. If there is no comment, I will deduct 5 points.

5. Here is a sample output for this program. Your output does not need to exactly the same but should be similar to this.

A3b requirements

In this assignment, you will simulate a simple scene graph. A scene graph is often used in game engines to organize game objects in a scene.

1. Implement a tree. You should be able to insert nodes, delete nodes, search nodes, and print the tree.

2. Your tree must have a root node, with the name "scene"

3. Your program should be able to accept three types of commands from keyboard.

    a. "insert A B". For example, "insert capsule player" means inserting a node "capsule" as a child of the node "player". If there is no node "player", do not insert the node and display a warning. If a node named "capsule" already exists in the scene graph, do not insert the node and display a warning.

    b. "delete A". For example, "delete wall" means delete a node named "wall" from the scene graph, including all its children. If there is no node named "wall" in the scene graph, display a warning.

    c. "print": print the tree.

4. Your program should run a loop waiting for a user to enter these commands. Your program will need to read the user input, separate the words, and perform the necessary work based on the command.

5. You must use a tree in your program. I will check the source code. If a tree is not used, you will 0 credit.

6. Write comments in your code. This part will be graded. If there is no comment, I will deduct 5 points.

7. Here is a sample output for this program. Your output does not need to exactly the same but should be similar to this.