CSC 481 Fall 16 Homework 2

Due: 10/18/16 by the start of class

Overview

Your task for this assignment is to add more functionality to your Processing engine. You will submit your code and a writeup of your results, including screenshots where appropriate. This is an ***individual assignment***, you are to work alone. As always, you are expected to abide by the University's Academic Integrity Policy (`http://policies.ncsu.edu/policy/pol-11-35-01`), which includes providing appropriate attribution for all external sources of information consulted while working on this assignment.

There are 125 points available on this assignment. Students enrolled in 481 are required to complete the first 100 points (Parts 1–3), but may also choose to complete Part 4. Students earning scores above 100 will receive a 100 on the assignment. Students enrolled in 591 are required to complete all 125 points, and will receive a grade as the percentage of the 125 points they earn. All students are required to submit the writeup addressing the sections of the assignment they have completed.

Part 1: Game Object Model (40pts)

For the first component of this assignment, you must design and implement a *software game object model* of your choosing. The design is completely up to you. You may opt to use one of the models we discussed in class, or come up with something on your own. The only requirements are that it be extensible enough that you can reasonably implement new types of objects that you aren't necessarily using at this point in time.

Part 2: Multithreaded, Networked, Processing Sketch (30pts)

For this portion of the assignment, your task is to fully integrate a Processing sketch with your multithreaded, networked client and server from Assignment 1. If you are enrolled in 591, this should be integrated with your code from Part 5 of Assignment 1. If you are enrolled in 481 you may choose to integrate with either Part 4 or Part 5 from Assignment 1. At the minimum, you must support the following functionality:

1. You must have a multithreaded server that reads data from and sends data to clients, all while accepting an arbitrary number of incoming connections at arbitrary times. Whether you send GameObjects, Strings, primitive types, or something else isn't critical provided the correct behavior results.
2. You must handle inputs that control the movement of a unique object on each client.
3. You must draw the entire sketch on a minimum of three clients, including the movement of objects controlled by the inputs on the clients.

In other words, you should start a server which will define the "scene," *i.e.*, maintain the abstract game object model including static and moving object locations. Then, you should start any number of clients, where each client will create a player object (square, rectangle, oval, *etc.*) and handle keyboard inputs to move that object around the sketch.[1] Each of those movements should get transmitted to the server, where it will be re-sent to every other client and drawn on their sketches as well. In effect, you're laying the foundation for a multiplayer platformer game which you will continue to develop throughout the course of the semester.

[1] Hint: As we discussed in class, if you choose to use the `java.io.ObjectOutputStream` to send data, make sure you pay attention to the `.reset()` method.

Part 3: A 2D Platformer (30pts)

Your task for this part of the assignment is to use your multithreaded server and client code, as well as your game object model to implement a multiplayer 2D platformer. At the minimum, you need to have the following game objects:

- **Characters** that are the depiction of the client-controlled players. Characters must respond to keyboard inputs to move left and right, as well as jump.
- **Static platforms** that characters can land on.
- **Moving platforms** that function similarly to the static platforms, but move around the screen in a patterned way.
- **Spawn points** that are hidden (*i.e.*, not drawn or rendered to the screen) objects marking the location where characters start from. There may be multiple spawn points in your environment.
- A "**death zone**," which is a different type of hidden object that marks some boundary of your environment. When a character collides with the death zone, they should be teleported a spawn point of your choosing. There may be multiple death zones.

When implementing these game objects, think in terms of the Game Object model you developed. Are you using inheritance to accomplish different behaviors? Components? Properties? What behaviors, properties, or components do you need to get these game objects working right?

In order to make sure your platformer works correctly, you will need to rely upon the collision detection functionality from your first assignment.

Although not required, it may be very informative to implement additional functionality like scoring or side-scrolling.

Part 4 (required for 591, optional for 481): Performance Comparison (25pts)

For this section of the homework you will implement a second networking protocol and do a performance comparison. If for Part 2 of the assignment you opted to send entire GameObjects between server and client, develop an alternative method sending some other Objects or information which provides the same behavior. For example, you may opt to send information encoded in Strings which get parsed on the other side.[2] If you did something similar for Part 2, then make your Game Object Model `Serializable` and send GameObjects instead.

To compare your two methods, you will use the server with two and three clients. You will want to measure the performance in terms of runtime. Given that you're using a fixed number of clients (two and three), you should change the number of static and moving objects in the environment. The moving objects do not have to be controlled by keyboard inputs. You will want to measure how long it takes to complete some number of game loop iterations (say 1,000) under different conditions. How long does it take with 10 moving objects on each client? What about 100? Can you get to 500? How does this change if you're using three instead of two clients?

[2] If you're already sending entire GameObjects, you may find it useful to play with the `.writeReplace()` and `.readResolve()` methods provided by the `Serializable` interface to achieve your second protocol.

Writeup

Now that you have designed and implemented a number of engine components, write a 2-3 page paper summarizing your design. That is a minimum of 2 *FULL* single-spaced pages. It is **strongly** suggested that you do not limit yourself to only answering the questions posed in this assignment. Think creatively about what you have done. Why did you make the design decisions you made? What did they enable for this assignment? What will they enable in the future? The most successful writeups will contain evidence that you have thought deeply about these decisions and implementations and what they can produce and have gone beyond what is written in the assignment.

As an appendix to your paper, please include all relevant screenshots to support your discussion. The appendix does not count toward your 2-3 page requirement. For students completing Part 4 of the assignment, any graphs used to present data should be part of the appendix. Your points for the writeup will represent $\frac{1}{2}$ of the points allocation for each of the above sections (*i.e.*, 20 of the 40 points for the "Game Object Model" section will come from your writeup).

What to Submit

By the start of class on 10/18/16, please upload to moodle a .zip file containing your code, a README file with compilation instructions and execution instructions, and a .pdf of your writeup. If you elected to use the Processing IDE, you may export the Processing project as an executable to include in your .zip file as well, provided you include executables for all three operating systems (Windows, Linux, OS/X).