Bouncy Shapes Project

For this project we will be creating a desktop application that renders shapes as they bounce around the screen inside of a desktop window.

Learning Objectives

- Be able to familiarize yourself with medium-sized projects (100's of LOC) as a starting point for your project.
- Incorporate existing components into a project
- Extend the existing architecture by designing new components that fit into an existing system
- Develop new classes and functions

Simple Rendering Engine

I have provided a *really* simple rendering and windowing engine as a starting point for this project. You can find the starter code here: https://github.com/mokonzi131/bouncy-shapes-initial.

Managing operating system windows and rendering are beyond the scope of this course. As such you are not expected to extend any of the windowing / rendering components of this project. The starter project already handles the following for you:

- Creates and manages a Window's Desktop Application
- Windows message processing and the event loop
- Interfacing with Direct2D and managing device resources
- Running a simple timed update / render cycle

All of this above mentioned stuff is handled in Main.cpp as well as some of the rendering and device management in the CircleRenderer and SquareRenderer classes. There is quite a bit of code in Main.cpp, however you will mostly be interested in the constructor of the SampleWindow class.

The way a rendering engine works is that it cycles through a loop at some pre-defined rate (like 60 cycles per second). On each cycle, it makes two important calls, the first Update is responsible for updaging the objects in a scene, and the second Render is responsible for drawing the current scene to the window.

The starting code for this project has all the framework in place for the rendering loop. In Scene.cpp, the two functions Update and Render will be called 60 times each second.

Project Requirements

For this project there are no restrictions on what you may add or remove from the starter project. You are free to change existing classes and interfaces as you see fit.

The initial project will simply open a window and draw one yellow circle on the screen. You are to add the capability to draw multiple shapes, movement, and simple collision detection. The exact list of requirements is found below in the *Implementation Steps* section.

You must do the following things specifically to complete the project:

- Increase the objects in the scene so that there are at least 10
- Incorporate the square renderer so that circles and squares can both be rendered
- Create additional renderers that use different colors so that the objects in the scene have different colors
- Implement the update function in Scene.cpp so that objects move around the screen
- Add boundary checks so that objects bounce of the edges of the screen

There are some // TODO comments in the code, but one of the goals of this project is that you can read and understand what the given code is doing. Furthermore, there are a number of ways to go about adding the above features, and it is up to you do come up with your own approach.

Implementation Steps

1. Add more GameObject's to the scene so that there are at least 10 objects.
2. There is a single CircleRenderer that uses a yellow color. Add at least two more renderers with different colors and sizes than the provided medium-sized yellow circle.
3. The initial object uses an instance of CircleRenderer, there is another renderer called SquareRenderer that is not currently used. Update your scene so that at least 3 of the objects use the SquareRenderer.
4. In Scene.cpp implement the Update function such that it updates every object in the scene.
5. Provide at least two different update behaviors so that objects in your scene don't all move in the same way. *Hint* this could be accomplished at a very minimum by giving different objects different speeds.
6. As objects move, they will eventually go off the screen. Design a strategy that can be incorporated into the object movement so that they "*bounce*" off the edges of the screen and remain within the window.

Grading Breakdown (100 points)

- **(10)** There are 10 objects in the scene
- **(5)** There are at least 3 different colors of objects in the scene
- **(5)** There are at least 3 different sizes of objects in the scene
- **(10)** At least three of the objects are drawn using the SquareRenderer provided
- **(20)** All the objects move when the program runs (i.e. update has been implemented)
- **(10)** Different objects have different movement behaviors (full points as long as the objects don't all move in exactly the same direction and speed)
- **(20)** Objects remain within the window (they bounce off the window borders) as they move
- **(20)** Good object-oriented design is applied. Specifically, rather than hard-coding all the above behaviors into a single loop, different behaviors are represented using different classes and these classes are composed at object construction time.

Demo

Please come to class to see a demo of what the finished application will look like.