CSC3221 Project 2

Welcome to the practical classes for CSC3221. These classes are built to complement and drive the lectures. The practicals are very important for a programming module. Things which can seem clear during the lectures may just appear not to work at all when in front of a machine. Conversely concepts which seem intractable when described are obvious when seen.

Module Assessment

There are 3 pieces of assessment for this module:

1. Project 1 which is submitted to NESS. (20)
2. Project 2 which is submitted to NESS. (30)
3. An exam in January. (50)

Project 2 (Deadline: Midnight, Friday 8th December, 2017)

Aims

Learn to about abstract classes, inheritance and polymorphism in C++ and collision detection in computer games.

Specification

An important topic in many computer games is determining when 2 object collide with one another. Games looks poor if a car appears to drive into a wall or appear to collide with thin air. Physics Engines devote considerable effort to solving this problem. In this Project we will start to consider this issue

Implement an abstract base class `Shape` to represent moving objects in a game. Derive classes `Circle` and `Square` from `Shape` with appropriate information to store the positions on a flat `2_D` surface. The classes should provide methods to move the position of the shape by a given amount and a method to detect when two shapes overlap. You will need to deal with overlap of 2 circles, 2 squares and circle and square separately (the latter is relatively difficult and will need some research. A basic solution is to embed the square within a circle of the appropriate radius and use it for the overlap algorithm).

Write a test game program that creates several shapes and stores them in an array or list or other data structure of your choosing. The program should then iterate through the data, move each shape by a small random amount (but keeping them within a grid of a given size) and check to see if there are any shapes overlapping. If there are, it should output the details about the overlapping shapes (you do not need to deal with the problem of disentangling overlapping shapes - that is a much trickier problem in game design!). It should then remove the overlapping shapes from the data structure and continue the game until at most one shape remains. NO GRAPHICAL OUTPUT is expected or required and providing some will not score any extra marks (this isn't a graphics course!)

You may choose whichever method you want for keeping the shapes within a grid.

Deliverables

A zip file of the relevant project in the projects directory of Visual Studio containing the following classes:

Shape.h
Shape.cpp
Circle.h
Circle.cpp
Square.h
Square.cpp
Game.cpp
Any other classes you wish to use
Sample output

Mark Scheme

Basic Shape, Circle and Square classes:	6
Overlap algorithms:	9
Game:	15
Total:	30