Task 22 Spike: Collisions

CORE

Context

Physics is a major component of 2D and 3D games. A key part of a physics engine is the handling of collisions between moving entities. A game world model typically requires algorithms to test for collision or intersection between entities. There is no single best algorithm and developers need to be familiar with different techniques and their implications to select the right approach.

Knowledge/Skill Gap:

A developer needs to be familiar with simple box and circle-based collision test techniques suitable for use in 2D games and their differences.

Goals

Either extend your Sprites and Graphics spike or create new 2D entities which you can implement a collision management data structure around:

1. Implement at least box-based (axis-aligned rectangles) intersection testing, and circle-circle intersection testing
    a. Display two or more boxes, one of which is a fixed position box and the other(s) moving. Detect, using axis-aligned rectangle testing, one box overlaps another (collision detected) and visually display the “collided” status to the user
    b. Display two or more circles where one or more of the circles are moving. Detect, using circle-circle collision testing, when one circle overlaps another, and change the visual appearance to represent that a collision has occurred.
2. Incorporate extensibility in your collision data structure, so that you would be able to swap out collision detection method for another

Expected Output

Repository

1. Code
2. Spike Report

Canvas

1. Spike Report

Notes

- Keep your entities simple. A box and a circle bouncing around the window is fine
- Find a tutorial in your framework, find examples, reference these
- You do not need to do circle-rectangle collision testing
- You might decide to create two separate applications (one for each test), but a single application with a simple key press of 1 or 2 to select the demo would be best.
- Use a simple object (class) to contain your box and circle (but you don’t need to go crazy with inheritance etc – although there are some useful advantages). Again – keep your demonstration code as simple as possible.