Task 23 Spike: Collisions Extended

OPTIONAL

Context

Although simple approximations are quite often desirable when it comes quick collision detection, there are situations when a per-pixel level of collision detection is needed for 2D games to create the right gameplay model.

Knowledge/Skill Gap:

The developer needs to know how to perform pixel-level collision detection using their chosen framework.

Goals

Extend your Collisions spike to display multiple moving entities on screen and add pixel-level tests for collisions. Your application must:

1. Perform collision detection with non-basic entities. Use something more than basic squares or circles. Use at least 2 different sprites.
2. Your sprites should all move at a constant velocity within the screen space
3. Sprites must change colour when colliding so that is obvious that collision has been detected.

Expected Output

Repository

1. Code
2. Spike Report

Canvas

1. Spike Report

Notes

- Consider implementing the ability to add more sprites with a key-press
- You should not perform the same collision comparison more than once per update.
- Use a game loop that runs as fast as possible and uses a delta “tick” (game time) to update each of the sprite’s positions proportionally. (Do not have a game loop that waits for new events before updating…)
- Think about creating your own basic sprite class that knows where it is, where it’s going, how to test for a collision with another provided (parameter) sprite, and how to draw itself
- Seriously consider creating a sprite manager class that knows how to look after sprites that have been added to it. So it would have the following methods (or similar)
    - Add(sprite)
    - Update(delta) to update the position of all sprites (loop, call each update())
    - Collisions() to test and then set a “collided” flag on each sprite
    - Render()