CS 4730: Computer Game Design

Engine Team: Collision System

Overview

For this assignment, you will add one of the most important features for any game: collision detection and resolution!

Basic Collision Detection

To earn a 1 on this assignment, your code must do the following at a minimum:

1. Display Objects should now have a method called getHitbox() that returns the location of the hitbox of the Sprite. A basic version of this method should return the correct hitbox given position changes, pivot point changes, but not scaling or rotation (that is for the advanced features below). Note that not incorporating scaling and rotation is much easier, so this is a good starting point.
    a. Recommendation: I recommend you add a method called drawHitbox() to Display Object that draws the hitbox so that you can visually see where it is located as you debug.
2. Your code should now contain a collision system. You may design this class however you wish. We have provided the header file for our solution, but you may change this as you see fit. It is your job to ensure the following:
    a. The collision detection is accurate and that resolution of collision moves sprites away from collisions in a sensible way.
    b. That level programmers can add collision detection to the game using **as few lines of code as possible**. You want it to be as easy as possible to use your collision system. This is a good litmus test for how strong your design is.
    c. Your collision detection will be working on axis-aligned hitboxes only, making the work easier for this first iteration.

Advanced Collision Detection

If your group wants a 2 on this assignment, you will need to provide the following updated features:

1. Your collision detection should now work correctly given any number of transformations applied to the Sprite (e.g., rotation, scaling). This includes nested transformations due to the display tree hierarchy.
    a. This means your hitbox should probably return a list of points (the corners of the box) or something similar because rotation is potentially involved. A traditional rectangle object (e.g., upper left point, width, and height) will not work anymore.
2. Your collision resolution should be more dynamic in the at least one of the following ways:
    a. Binary search is used to resolve the collision such that the Sprites remain as close as possible.
    b. Resolution should take each axis into consideration independently. This would allow for features like walking up a sloped platform, etc.

Demo

Make a very simple demo to show off collision detection working. The demo should contain the following:

- A parent container that can be moved around, scaled, etc. The conrols for this should be:
    - Move: WASD
    - Rotate: E and R
    - Scale: F and G

- One small sprite that is a child of the container. This sprite's controls are:
    - Move: IJKL
    - Rotate: O and P
    - Scale: N and M
- Another small sprite that is also a parent of the original container. Its controls:
    - MOVE: Arrow Keys
    - Rotate: X and C
    - Scale: V and B

Your demo should not allow the two child sprites to overlap. So, you must detect collisions between the two sprites and resolve those collisions so that they can never overlap. The controls allow you to test your detection and different angles of rotation, etc. to ensure everything works.

Turn In

As always, submit your code on Collab.