CS 4730: Computer Game Design
Lab 5: Collision Detection, Physics, and Sound

Overview

This week, you will be implementing collision detection, some basic physics (you can take this as far as you'd like), as well as sound. For the most part, these pieces of functionality are straightforward, but they will be important to your games.

PART 1: Adding Collision Detection

Start by adding collision detection to your DisplayObject class. This will likely involve doing the following:

- Add a getHitbox() method to the DisplayObject class. You may need to be careful about what coordinate system this hitbox is in. You might prefer to have separate methods that return the same hitbox in different coordinate systems.
- Add a collidesWith(DisplayObject other) method.
- ***NOTE: Make sure your hitboxes work even when the sprite is rotated, especially around distant pivot points. How will you handle this? I HIGHLY recommend drawing your hitboxes so you can visually see what is going on while you test your programs.***
- Have the DisplayObject throw a colliding event when collisions occur.
- Optional: You may wish to allow one display object to have multiple hitboxes, OR this could just be handled by adding children Sprites of the parent. Up to you.
- Optional: You may very well wish for a collision with a child object to be considered a hit on this object. Thus, you would alter collidesWith() in the DisplayObjectContainer class to account for collisions with children.
- **The most difficult part about this is keeping the coordinate systems consistent. Keep this in mind as you code and as you debug issues.*

PART 2: Adding Physics

Add the ability to make Sprites contain physics. You may want to extend AnimatedSprite into a new class called PhysicsSprite or something like that, OR give DisplayObjects a "hasPhysics" field. Consider adding some of the following:

- Reacts to gravity. Global gravity amount can be set.
- Has mass, acceleration, velocity, forces acting on it, etc.
- Some kind of 'bounciness' when colliding with items, deals with conservation of momentum.
- What else can you add that is cool?

PART 3: Adding Sounds

Probably the easiest of the three. All games have sound, and it is nice to have a SoundManager class whose responsibility is to manage and play sounds. Implement this class with (potentially) the following methods:

- LoadSoundEffect(String id, String filename)
- PlaySoundEffect(String id) //sound effects are short and are removed once complete
- LoadMusic(String id, String filename)
- PlayMusic(String id) //music loops and plays forever, consider adding a parameter for looping
- Etc…

- **You MIGHT consider making this class a Singleton class. It is up to you.*
- **Some methods might be unnecessary depending on the language you implement in.*

Test Demo
Now that you have a much more complete engine, we should be able to make a realistic game! This week, make a simple platformer game.

Have a character (Mario or other) start at the bottom of the screen (You might want to make the screen bigger for this). Place a coin (or something to grab) near the top of the screen. Then, place some platforms around the screen. The game is simply that Mario has to jump up the platforms and grab the coin.

You'll need to use your collision detection, sound effects (some music and sound effects should play), animations (character should be animated), physics (character obeys some basic physics when moving), and events in order to make this game work.

Turn In
If you finish during lab, show your working code to the TA with the demo to prove it. Show your code and describe how it works. Submit on Collab before the deadline.