CS 4730: Computer Game Design
Lab 2: Display Objects

Overview

This week, you will be making your DisplayObject class much more usable by adding a great deal of essential functionality. You will also be developing an AnimatedSprite class that allows you to switch the image of a Sprite rapidly to give the perception of animation.

PART 1: Adding DisplayObject Functionality

Step 1 of this lab is to add functionality to your DisplayObject class that allows for important functionality such as positioning, rotation, etc.

TO-DO 1:

Add the following fields to your DisplayObject class:

visible:	should be true iff this display object is meant to be drawn
position:	should describe the x,y position where this object will be drawn
pivotPoint:	The point, relative to the upper left corner of the image that is the origin of this object
scaleX / *scaleY*:	scales the image up or down. 1.0 would be actual size.
rotation:	defines the amount (in degrees or radians, your choice) to rotate this object
alpha:	defines how transparent to draw this object.

TO-DO 2:

Write some getters and setters for these fields.

TO-DO 3:

Now, we will update our display objects to always draw the object as defined by the internal state. Remember that when someone updates an object (e.g., sets the rotation to some new value), the object doesn't actually move immediately. The internal variable changes, and the object is drawn with the new rotation next time a draw() is invoked (on the next frame).

Now, update your draw() method so that the image is drawn according to this internal state of this DisplayObject. *Note that MOST of your changes will actually be placed in applyTransformations() and reverseTransformations().*

Test Demo

You should test out your new functionality by having the following keyboard commands control the following:

- Arrow Keys: Moves your sprite to different positions on the screen.
- Q&W: Rotates your sprite clockwise and counter-clockwise.
- A&S: Scales your sprite up and down respectively.
- Z&X: Makes your sprite more and less transparent.
- V: Toggles visibility of your sprite.
- IJKL: Treated as keyboard arrows. Moves the sprite's pivot point around.

PART 2: AnimatedSprite

Next, you will implement a new class called AnimatedSprite that extends Sprite. Your class should have the following additional functionality:

- Class should contain a list of frames (images, not DisplayObjects) that contain all of the images in the animation(s). You'll need some mechanism for instantiating these (sprite sheet support, or perhaps just indexing the individual images with some naming convention like Mario_run_0.png, Mario_run_1.png, etc.)
- Class should contain a current frame integer as well as a startIndex and endIndex. These last two integers will be used to support multiple animations (see next bullet below).
- Class should support multiple animations (e.g., jump, walk, run)
- Animations can be stopped on a single frame, or playing.
- The speed of the animation can be adjusted through a setter.
- I can set which animation to play through a method, like animate("run");

Test Game: Mario Clicker 2.0

Let's update our Mario clicker game to incorporate our new functionality:

- Mario can now move around and also scale, become transparent, change pivot point, etc. as in the demo for part 1 above.
- Player 2 still tries to click on Mario.
- Let's extend the timer to 60 seconds this time.
- Tweak the parameters on Mario so that the game is as interesting and fun as you can make it.

****NOTE: It is ok if rotating Mario around distant pivot points ruins your collision detection. We will handle this problem when we do collision detection more formally.***

Test Game Report

Write a short pdf (a couple of paragraphs or so is enough) describing how you tweaked the demo game to make it as fun as possible. What strategies existed that made the game too easy? How did you deal with that to make the game more interesting and fair?

Turn In

You will submit 1) Your working code and Mario Clicker 2.0 game (this includes your engine so far) and 2) your short pdf write-up describing how you designed the Mario Clicker 2.0 game to be fun. Submit on Collab. Once again, please include an executable Jar file if you are using Java.