Programming Project 2

Due: Tue, Sept 30.

Objectives:

- Learn how to work with 2D images (displaying, moving, animating, scaling, rotating).
- Develop a sprite library that you can use throughout the rest of the course.

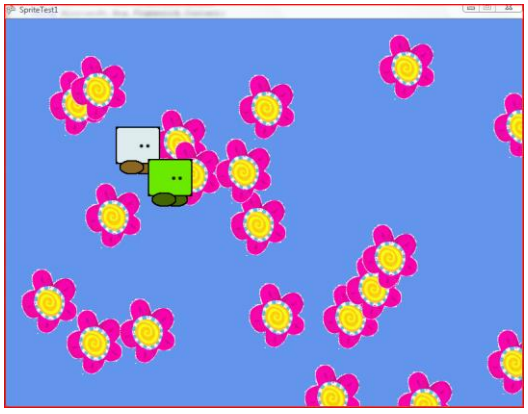
You can download the images needed for this assignment from the Resources page.

Part 1: displaying images (20 points)

Write a basic class called Sprite which can hold a texture and draw (a specified portion of) it to the screen.

SpriteTest1.cs, which you can download from the Resources page, gives an example of how this Sprite class may be used. The picture below shows what you should see when you run SpriteTest1.cs.

When looking at SpriteTest1.cs, you can see that your Sprite class will need the following members: Member variables and getters/setters for the position of the sprite, its z-level, the texture providing the image, the width and height of the area to be displayed, and a rectangle indicating which rectangular area of the texture should be displayed. Finally, you will need a Draw method. As you can see this method should take a SpriteBatch object as argument. You can then use the SpriteBatch's Draw method to actually draw the sprite to the screen (see the link to the documentation on the Resources page). The SpriteBatch's Draw method is overloaded; that is, there are several versions of it. I would recommend to use SpriteBatch.Draw(Texture2D, Vector2, Nullable<Rectangle>, Color, Single, Vector2, Single, SpriteEffects, Single). This is more complex than what we need at the moment, but we will use all of that functionality later on in this assignment.



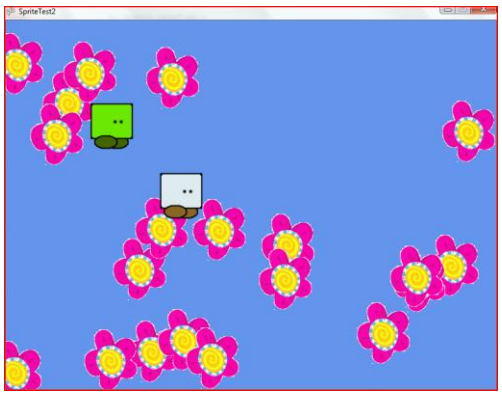
Part 2: moving the sprite (20 points)

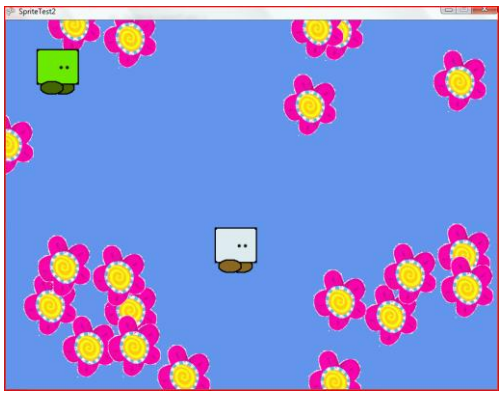
Extend the Sprite class such that it allows for moving sprites. Again, SpriteTest2.cs, which you can download from the Resources page, gives you an example of what your Sprite class should be able to do.

As you can see, you need to add a property for the velocity to the class and add a method called Update which updates the position of the sprite according to the velocity and the time that has passed since the last update.

Then add a method called Bounce that checks whether the sprite has moved beyond a given area specified by an x and a y coordinate and if so moves it back into that area and reverses its direction. Try to make sure that your sprite bounces off its center. This should happen automatically, if you set the origin parameter of the SpriteBatch.Draw method appropriately (namely to the center of the displayed rectangle).

Finally, make sure that your sprite always looks forward. As you can see from SpriteTest2.cs, you need to add a member variable of type SpriteEffects to your Sprite class. If you then use the specified SpriteEffect when drawing the sprite, it should get flipped horizontally whenever it bounces off the left and right walls.

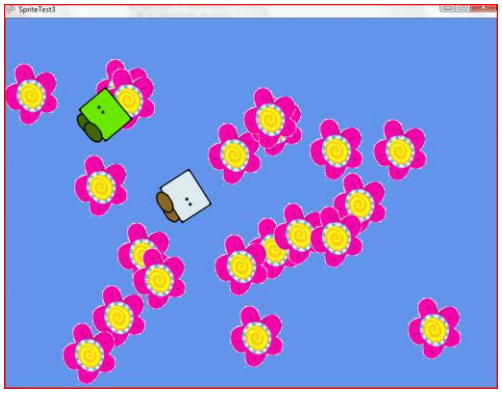


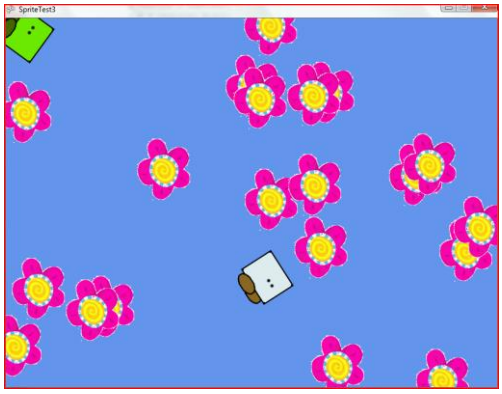


Part 3: rotating the sprite (10 points)

Now add rotation. Your sprite class should allow to position the sprite at an angle (the white sprite of SpriteTest3.cs) and also to continuously rotate a sprite (the green sprite). Your sprite should rotate around its center; again that should happen automatically if you set the origin parameter of the SpriteBatch.Draw method right.

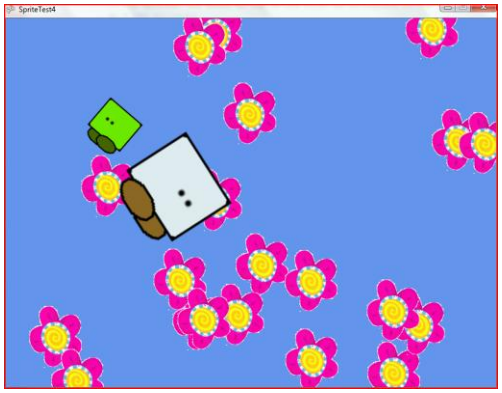
Use SpriteTest3.cs to test your Sprite class.





Part 4: scaling the sprite (10 points)

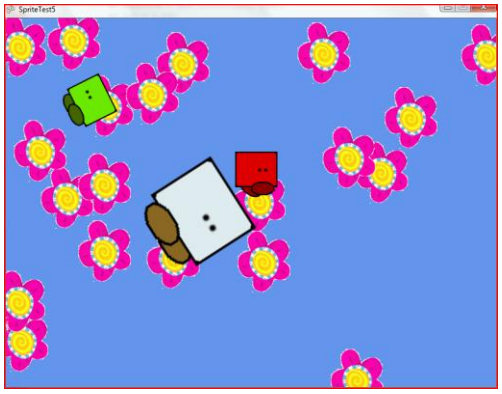
An easy one: add functionality so that your sprite can be scaled. Use SpriteTest4.cs to test.



Part 5: animating the sprite (20 points)

So far, we have only used the first picture in the WalkingSquare image. Now, we are going to animate the walking square by cycling through all of the six pictures in the image.

Add member variables indicating the number of frames and the desired animation speed (in frames per second) to your Sprite class. Then add a method Animate (called by the Sprite's update method) which updates the member variable specifying the rectangle to be displayed based on the animation speed and on the time that has passed since the last time the rectangle was updated.



Part 6: Play! (20 points)

You have two options here:

1) Produce some fancy animation using your sprite library. For example, create a circle of sprites that seems to orbit horizontally around a point by manipulating scale and depth level of the sprites.
2) User your sprite library in a *simple* game. There should be some interaction with the user and a goal the user has to achieve.

Requirements for the Design Document (Due September 16).

You design document should consist of

- a specification of your Sprite class: all member variables with types, getter/setter methods, signature for all methods and brief description of what that method is going to do.
- Tell me what you are planning to do for part 6. I want to know whether you are choosing option 1 or 2 and also a concrete description of what you are going to do.