CS162 - Programming Assignment 4 - Inheritance

The purpose of this assignment is to give you some experience using inheritance (and seeing the benefits and drawbacks).

Overview

This program is part three of many that will implement an Asteroids-like game. Note: Asteroids is a registered trademark of Atari, Inc.

In this program you'll build upon the previous assignment.

- First you'll split the ship class into two parts.
    - One part will contain code that is common to all on-screen objects
    - The other part will contain code that is specific to the ship class
- Then make use of inheritance to "reuse" the common code for both the ship class and an asteroid class.

When the assignment's complete, you'll have one ship and one asteroid, and the code will be such that implementing other on-screen objects (photons, etc..), and detecting collisions, will be very, very easy.

Details

This entire assignment builds on the previous one, so you'll be working with that project.

Step 1 - Add two new files to the project named:

- SpaceObject.h
- SpaceObject.cpp
  Then cut and paste (that was CUT, not copy) all the Ship code into those two files.

Step 2 - Split your ship class code into the two different classes as shown below. I'd recommend doing this via the following process;

- In the Ship class files
    - Write a new Ship class header as shown below, and copy the#ifndef/#define/#endif lines from the old Ship class files.
    - #include the SpaceObject.h file at the top.
    - Move the 4 function prototypes and functions that belong in the ship class to the ship.cpp file (see below). They'll have problems (things underlined in red) that we'll come back and fix after getting the SpaceObject class code fixed.
- In the SpaceObject source files
    - Fix the #ifndef/#define name to reference SPACE_OBJECT_H
    - Replace all occurances of "Ship" with "SpaceObject" (VS has a replace-all feature).
    - Add the new functions (see below; they're simple)+
- Back in the Ship class

- Fix the broken functions by replacing the references to the SpaceObject private variables with calls to the SpaceObject accessor and mutator functions.

```
class SpaceObject  {
private:
      Vector maxLocations;    //maximum allowable values for location
      Vector location;        //current location (x,y)
      Vector velocity;        //current velocity (in pixels/frame)
      float angleDeg;         //angle ship is facing, in degrees
      float radius;           //gross radius (for collision detection)

public:
      SpaceObject();
      //-----------------------------------------
      //mutators
      void setLocation(float x, float y);
      void setVelocity(float velocityX, float velocityY);
      void setAngle(float angDeg);
      void setRadius(float radius);
      //the following three are new.  They facilitate changing
      //member variable values by the given amounts
      void chgVelocity(float deltaVx, float deltaVy); //NEW!
      void chgAngle(float deltaA);                    //NEW!
      void chgRadius(float deltaR);                   //NEW!

      //-----------------------------------------
      //accessors
      Vector getLocation();
      Vector getVelocity();
      float getAngle();
      float getRadius();

      //-----------------------------------------
      //others
      void updateLocation();
};

class Ship: public SpaceObject {
public:
      Ship();
      void rotateLeft();
      void rotateRight();
      void applyThrust();
      void draw(sf::RenderWindow& win);
};
```

After this code is debugged, your program should work just like it did before we made those changes. But now we can easily implement additional functionality (asteroids! photons! other stuff!). Verify it still works. Fix it if it doesn't.

Step 3 - Create Asteroid source files to implement the Asteroid class as follows:

```
class Asteroid : public SpaceObject {
private:
      int sides;
      float rotationalVelocity;
public:
      Asteroid();
      void draw(sf::RenderWindow& win);
};
```

The Functions should do the following:

Asteroid()	Initialize the radius, location, and velocity by calling the appropriate base class functions.
	● radius should be 50
	● location ideally would be a random spot on the screen not near the middle.
	● velocity would be some low random values. The following should work for x (do something similar for y):
	`float x = 0.01 * (rand()%10+4);`
	`if ((rand()%2) ==1) x*=-1;`
	Initialize sides to 8 (or the number of sides you want your asteroids to have)
	Initialize rotational velocity to .1 or -.1

draw()	Change the angle by the rotational velocity amount. Then draw the asteroid as a regular polygon of the given number of sides. If it's near an edge (within one radius), you should draw it again off the opposite edge so the part that is there shows up.

Step 4 - Add an asteroid to main() and update and draw it just as you did the ship. Both should then show up on the screen. Ship should be flyable. Asteroid should just drift and rotate slowly.

Step 5 - Thoroughly test the code. Make sure that when the Asteroid starts going off one edge of the screen it immediately starts showing up on the other edge. Test all 4 cases. (left, right, top, bottom). Experiment with the asteroid velocities if it seems too slow or fast.

Very Important Stuff

All programs should follow the class's coding conventions.
Submit the following:

- A zip file containing all the source files you created and your executable