CPSC 3400 PROJECT 3: Java Asteroids FALL 2015

Due Dates: This project is due in two pieces, both through Canvas (**ZIPPED** source code). *Also, for the final submission (only) bring a paper printout of your source code to class.*

Part A: due Monday, November 23rd, 30 project points
- ability to draw the ship, use keys to move ship forward, and when it reaches the edge it wraps around the screen.

Part B: due Friday, December 4th, 100 project points (as shown on rubric)
- complete project as specified, along with any optional extra credit.

*** If your program does not compile, you will receive no credit!! Double check
*** that your program will compile before submitting it.

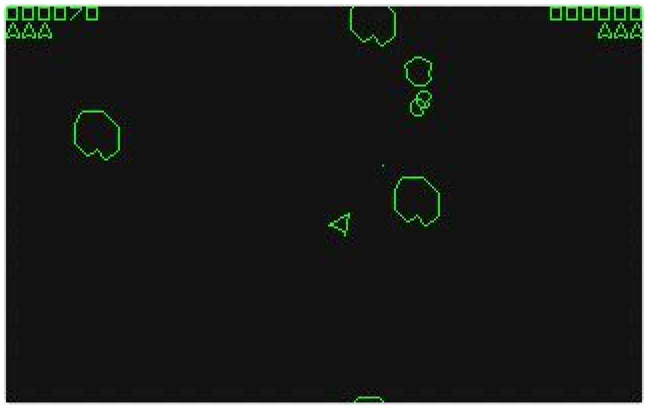

Image Source: http://www.atarimania.com/

Assignment

Overview:

Create a program that is similar to the classic "Asteroids" video game. (See the Wikipedia article at: http://en.wikipedia.org/wiki/Asteroids_(video_game) if you are unfamiliar with the game. You may also want to look at http://atari.com/arcade/arcade/asteroids-online.) The user's spaceship is a triangular outline that has engines that fire to move the ship forward (only). If the ship goes off one edge of the screen, it wraps around to the opposite edge. Also inhabiting the screen are many irregularly shaped asteroids, which the ship must dodge (or be destroyed!). The ship can fire bullets out the front of the ship to destroy asteroids. (In addition, there is an alien space ship that periodically appears and shoots back -- this is not a required part of the project, however!)

Specifics:

1. Available on Canvas are two classes:
   a) The *Game* class, which should not need to be modified. The class contains the methods needed for setting up the game window, adding key listeners, and playing the game. The method *play()* contains a continuous loop that redraws the screen and handles any keyboard input every 10th of a second. The *play()* method calls two methods in the *Asteroids* class: *paint()* and *handleKeyPress().*
   b) The *Asteroid* class, which extends the *Game* class. This class you will need to fill in some blanks. In particular, the two abstract methods specified in the *Game* class will need to be written -- *paint()* and *handleKeyPress()*.
      1. *paint()* is the method that draws the background, moves the ship/asteroids, and draws the ship/asteroids on the screen.
      2. *handleKeyPress()* is the method that processes the user's input. Pressing the up arrow will fire the ship's engines to move it forward, pressing left/right will rotate the ship, pressing the spacebar will fire the ship's guns, and pressing 'q' will quit the game.

2. You will need to write classes to represent the following aspects of the game:

   a) <u>The *Ship*.</u> The ship is triangular shaped, and begins life in the center of the screen, at rest.
      1. The ship can move: If the user fires the engines, the ship will increase its speed in the forward direction. If the user presses the left/right arrows, the ship will rotate counter-clockwise/clockwise. Firing the engines again will cause the ship to increase its speed in that new forward direction. If the ship goes off the left edge of the screen, it will reappear on the right edge. Likewise for the top and bottom edges. After the ship starts moving in one direction, it continues "coasting" without slowing down.
      2. The ship can collide with asteroids, and be destroyed.
      3. The ship can fire its guns in the forward direction, causing bullets to appear at the front of the ship and move across the screen with a constant speed.

   b) <u>The *Asteroids*.</u> An asteroid is a roughly circular shape with irregular outlines. At the start of the game, several asteroids (at least 5) appear at random locations on the screen, with random velocities.
      1. An asteroid moves at a constant speed across the screen, wrapping around from one side to the other.
      2. If an asteroid collides with the ship, the ship is destroyed.
      3. If an asteroid collides with a bullet, the asteroid and the bullet are both destroyed.
      4. We will not worry about what happens when two asteroids collide.

   c) <u>The *Bullets*</u>. A bullet is a small circular object fired from the front of the ship. It begins life with a velocity based on the direction that the ship is pointing.
      1. A bullet moves at a constant speed across the screen, wrapping around from one side to another.
      2. A bullet only "lives" for a certain time period (or distance moved) before disappearing.
      3. If a bullet collides with an asteroid ~~or the ship~~, both the asteroid~~/ship~~ and the bullet are destroyed.

   d) <u>The *Stars*</u>. A star is a very small dot that is a decoration behind the asteroids and ship. The stars are randomly distributed around the screen, with enough stars to make the background look nice. Stars do not move, and do not collide with anything.

3. Note that you will need to write classes for these different game objects. However, you'll find that many methods and fields are the same between the different objects. For example, both the ship and the asteroids move around and have (x,y) positions on the screen. Both the bullets and the stars are round shapes that have (x,y) positions on the screen. *You are required to group common methods/fields in parent classes!* No method or field should be duplicated between the ship/asteroid/bullet/stars.

Style:
As always, follow good programming style. Specifically:
- Have a comment before a function that describes what the function does.
- Use variable names that communicate their intended use.
- For each portion of your code, have a comment that describes what it's doing.
- *Do not duplicate methods or fields between objects -- if two objects both need a method/field, then that method/field belongs in the parent class of those objects!*

Tips:

- A few tips and techniques will be discussed in class -- be sure to take detailed notes!
- Handouts describing how to use the Java drawing classes will be distributed in lecture, and are available on Canvas.

Possibilities for Extra Credit:

If you receive an 80% or above on the Project, you may attempt to earn extra credit. However, there is a maximum of 20 possible extra credit points for this assignment.

- Adding a welcome screen, end-of-game screen (2 points)
- Adding a scoring method and displaying the score on the screen (4 points)
- Adding levels to the game (i.e. clear the screen, go up a level that is more challenging). (4-8 points)
- Adding the alien spaceship that shoots back (5-10 points)
- Adding dramatic animation of destruction of the ship or asteroids (5-10 points)
- When an asteroid is shot it shatters into smaller asteroids with diverging velocities (5-10 points)
- Additional features to the game, depending on difficulty (1-20 points)