159.261 Lab 4
Asteroids – Part One

This lab is the first in a series in which we will develop a simple Asteroids style game that allows you to control a ship flying around the screen and shooting asteroids floating through space. Over the coming weeks we will add a number of additional features to this game including the use of sprites, animations, sound effects, menus etc. To start with, we will build the foundations of the game using simple graphics primitives and basic functionality.

This is also the first lab that will make use of our simple Java game engine.

1) Starting Code

This exercise will get you started with writing simple graphical programs in Java. You are given a starting program that creates a window and draws a string on the screen. You must write your own code to draw the smiley face.

1. Download the Java games engine `GameEngine.java` from the Stream site under *Course Resources*. You may also want to download the game engine manual for reference.
2. Make sure you can compile and run the lab starting code (you will need to compile both `GameEngine.java` and `Lab4.java`).
3. Test the example to make sure it works properly. You should be able to fly a spaceship (a triangle) around the screen by using the arrow keys.
4. Read through the starting code to make sure that you understand how this game works. Pay particular attention to the following functions – `updateSpaceship`, `drawSpaceship`, `keyPressed` and `keyReleased`.
5. Modify the code to stop the spaceship disappearing when it flies off the edge of the screen. You should modify the `updateSpaceship` function to detect when the spaceship has reached the edge of the screen and *warp* it to the other side. For example, if the spaceship reaches the left-hand side of the screen, you can add `500` to the ship's `x` position to make it appear on the right-hand side of the screen.

2) Adding a Laser

Add functionality to the game to allow the ship to fire a projectile (a laser). This involves three separate steps which all need to be implemented before the laser will work correctly (don't be surprised if nothing appears on the screen until you have completed part 3).

1. Add code to the `keyPressed` function to call the function `fireLaser()` whenever the user presses the space bar. Read the function `fireLaser()` to ensure you understand how it works.
2. Currently the function `drawLaser()` is empty. Write this function yourself to draw a line representing the laser (you will need to use `laserPositionX`, `laserPositionY` and `laserAngle`). Look at the `drawSpaceship()` function for an example of how to write this type of function. The laser should only be drawn on the screen if the boolean `laserActive` is set to true.

3. The function `updateLaser(dt)` is also empty. Implement the functionality needed to move the laser based on its velocity `laserVelocityX` and `laserVelocityY`. When the laser reaches the edge of the screen it should be destroyed by setting `laserActive` to false. Again, look at the `updateSpaceship` function for ideas on how to complete this.
4. Modify the function `fireLaser()` so that it only allows you to fire a laser if there is not one currently active.

3) Adding an Asteroid

The next step is to add an asteroid (represented as a circle) to the game that floats through space. This will be the main obstacle that the player needs to avoid.

1. In the function `init()` you should add code create an asteroid by calling the function `randomAsteroid()`. This will create an asteroid at a random position and with a random velocity.
2. Implement the functions necessary to update and draw the asteroid (it should just float through space). Whenever the asteroid reaches the edge of the screen, the update function should *warp* it back to the other side (just like our spaceship).

4) Collisions

The last step (for this lab) is to detect collision between the spaceship, the asteroid and the laser. The rules for these collisions should be:

- If the laser and the asteroid collide then they should both be destroyed and a new asteroid should be created (with a random position/velocity).
- If the asteroid and the spaceship collide then the game should end.
- Collisions between the laser and the spaceship can be ignored.

1. Write code to detect collisions between the laser, the asteroid and the spaceship. You may find the function `distance(x1, y1, x2, y2)` helpful (it returns the distance between the two points *(x1,y1)* and *(x2,y2)*).
2. Implement the response to these types of collisions (according to the rules above).
3. Test your game to make sure it works as intended.