ANIMATION IN JAVA – PART 1

We start off with the normal stuff for any program.

- first the package name
- then any imports
- then the name of our class
- and finally the public static void main line.

```
package graphics;

import hsa2.GraphicsConsole;
import java.awt.Color;

public class Fading {

	public static void main(String[] args)  {
		//and we start writing our program here
	}
} //end of class
```

However, graphics programs end up using a lot of classes, so it's typical to write the program using the constructor of the class instead of putting everything in main().

We'll learn about constructors at some point in the future … so don't worry about them now.

Notice the red word, that's the name of the class that you're making and it needs to be the same in all three places.

```
package graphics;

import hsa2.GraphicsConsole;
import java.awt.Color;

public class Fading {

	public static void main(String[] args)  {
		new Fading();
	}

	//this is called a constructor. It makes a Fading type object
	Fading() {

		// Our graphics program is now going to go here

	} //end of constructor

}
```

Run the program at this point. Nothing happens at all.

We now make a graphics console object called gc.
We also do whatever setup for the graphics that we need to do.

```
package graphics;
import hsa2.GraphicsConsole;
import java.awt.Color;

public class Fading {

        public static void main(String[] args)  {
                new Fading();
        }

        GraphicsConsole gc = new GraphicsConsole (800,600, "Spots");

        Fading() {
                //Setup section:
                gc.setAntiAlias(true);
                gc.setLocationRelativeTo(null); //centre window
                gc.setBackgroundColor(Color.GRAY);
//  $             gc.setBackgroundColor(new Color(120,120,120,10));
                gc.clear();
                gc.setColor(Color.RED);

        } //end of constructor
}
```

Run the program at this point.
You get a gray window in the middle of the screen.

Now we'll write the main animation loop.
Notice that it's a while loop and it continues on looping forever.
We also have to make a couple of variables that we'll be using.

```
package graphics;
import hsa2.GraphicsConsole;
import java.awt.Color;

public class Fading {

	public static void main(String[] args)  {
		new Fading();
	}

	GraphicsConsole gc = new GraphicsConsole (800,600, "Spots");

	Fading() {
		//Setup section:
		gc.setAntiAlias(true);
		gc.setLocationRelativeTo(null); //centre window
		gc.setBackgroundColor(Color.GRAY);
//  $		gc.setBackgroundColor(new Color(120,120,120,10));
		gc.clear();
		gc.setColor(Color.RED);

		//Variables:
		int size = 40;
		int sleepTime = 5; //in milliseconds

		//Main loop:
		while(true) {
			//make random numbers for location (based on 800,600)
			int rx = (int) ((Math.random()* 700)+50);
			int ry = (int) ((Math.random()* 500)+50);

			gc.fillOval (rx, ry, size, size);

			gc.sleep(sleepTime);
			/* the final thing in the loop must be "sleep".
			If it doesn't sleep the graphics may not get updated */
		}
	} //end of constructor
}
```

Run the program at this point (after typing in this page). The window starts filling up with red circles.

Note that we are making random numbers for the locations of x and y for the circles. These random numbers are setup specifically for a window size of 800x600 pixels.

Now we'll polish up the program …
First add some random colours.
NOTE: this is just a bit more code that goes into the main while loop. The whole program is not copied here.

```
            //Main loop:
            while(true) {
// $              gc.clear();
                   //make random numbers for location (based on 800,600)
                   int rx = (int) ((Math.random()* 700)+50);
                   int ry = (int) ((Math.random()* 500)+50);
                   gc.fillOval(rx, ry, size, size);

                   gc.sleep(sleepTime);

                   //make random colour for next ball
                   int r = (int) (Math.random()* 256);
                   int g = (int) (Math.random()* 256);
                   int b = (int) (Math.random()* 256);
                   gc.setColor(new Color(r,g,b));

// Here's a special way to make brighter random colours:
// gc.setColor(new Color(Color.HSBtoRGB((float)Math.random(), 1.0f, 1.0f)));
            }
```

Add in the green and blue lines. (Obviously, you have to add them in the correct place.) Then run the program.

Improvements

1. Find the two lines that have // $ at the beginning of them. Delete the // $
   a. This makes the background use a semitransparent dark gray colour.
   b. It also clears the screen with each loop through the while loop
   c. When it clears the screen, what it does is paint it with the semitransparent colour, so you can still see the previous circles on the screen. They just start fading.
2. Making random red, green, and blue values for the colours ends up with a lot of colours that are grayish.
   The second way to make random colours (which needs explanation) will result in brighter random colours.