Lab 10: Introduction to Computing

Practicing OOP

In this lab you will practice implementing and modifying the **`Ball`** object we designed in class. First, carefully study the **`Ball`** class:

```
class Ball {
  // Attributes/Properties of all Ball objects
  float x, y;     // location in sketch
  float radius;  // ball radius
  float dx = random(1, 3);
  float dy = random(1, 3);

  // Constuctors
  Ball() {
    y = random(height);
    x = random(width);
    radius = 25;
  } // end of Ball constructor

  Ball(float r) {
    x = width/2;
    y = random(height/2);
    radius = r;
  } // end of Ball constructor

  // behaviors
  void display() {
    push();
    fill(ballColor);
    noStroke();
    circle(x, y, 2*radius);
    pop();
  } // display()

  void move() {
    x = x + dx;
    y = y + dy;
    bounce();
  } // move()

  void bounce() {
    if (x > (width-radius)) {
      dx = -dx;
    }
    if (x < radius) {
      dx = -dx;
    }
```

```
    if (y > (height-radius)) {
      dy = -dy;
    }
    if (y < radius) {
      dy = -dy;
    }
   } // bounce()
} // end of class Ball
```

The class above can be used in the Sketch below:

```
// Example: How to use the Ball class
Ball b;
void setup() {
  size(800, 600);
  background(255);
  b = new Ball();
} // setup()

void draw() {
  background(255);
  b.move();
  b.display();
} // draw()
```

Task#1: Implement the sketch above and run it. You should see a black ball bouncing off the Display Window's walls.

Task#2: Modify the sketch to include a color attribute to the Ball object. The color of the ball should be a random rgb color in the constructors. Run the sketch several times to confirm the color. Next, modify the dx and dy attributes to be set to random values in the range [1.0..3.0).

Task#3: In the main sketch, define an array of ball objects. In setup() create N ball objects of random sizes. In draw() display as well as move all ball objects.

Detecting and adding collisions

Notice in the sketch from Task#3 that balls just go through each other. Let us add a way to detect any collision and have the balls move away due to it. One way to do this would be define a **`collide()`** method:

```
void collide(Ball b) {…}
```

When called on a **`Ball`** object, this method detects whether that object has collided with **`Ball b`**. The collision between two **`Ball`** objects can be detected if the distance between their center is less than the sum of their radii. Below, we define the following in the Ball class:

```
void collide(Ball b) {
  // Does this ball collide with ball b?
  if (distance(x, y, b.x, b.y) <= (radius+b.radius)) {
    dx = -dx;
    dy = -dy;
  }
} // collide()

float distance(float x1, float y1, float x2, float y2) {
  return sqrt((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 - y2));
} // distance()
```

Next, each time a ball is moved, we must see if it may have collided with another ball. This is done in the loop below.

```
// Check collisions
for (int i=0; i < balls.length; i++) {
  for (int j = 0; j < balls.length; j++) {
    if (i != j) {
       balls[i].collide(balls[j]);
    }
  }
}
```

Study the loop carefully to see how it works. It detects if a collision has occurred between any two balls.

Task#4: Add the above code to your sketch (at the end of the draw() function) and observe the behavior.