1 Designing a Java Application

Today instead of learning new things, we will try to learn about putting together all we have learned to make a simple application. The eventual goal is to build an application that will simulate molecules moving in a two-dimensional rectangular box. Today we will see portions that implement the simulation with bouncing off the walls of the container only.

1.1 Vector2D

```
package gas;

public class Vector2D extends Object implements Cloneable {

    public double x;
    public double y;

    /** Creates new Vector2D */
    public Vector2D() {
    }

    public Vector2D(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double dotProduct(Vector2D v) {
        return this.x*v.x + this.y*v.y;
    }

    public double length() {
        return Math.sqrt(x*x + y*y);
    }

    public void multiply(double k) {
        x = k*x;
        y = k*y;
    }

}
```

1.2 Molecule

```
package gas;

public class Molecule extends Object {
    private double mass;
    private Vector2D position;
    private Vector2D momentum;


    /** Creates new Molecule */
    public Molecule() {
        mass = 1.0;
        position = new Vector2D(0.0, 0.0);
        momentum = new Vector2D(0.0, 0.0);
    }

    public Molecule(double mass, Vector2D position, Vector2D momentum) {
        this.mass = mass;
        this.position = position;
        this. momentum = momentum;
    }

    public void advance(double time) {
        position.x += momentum.x/mass * time;
        position.y += momentum.y/mass * time;
    }
```

```
    public Vector2D getPosition() {
        return position;
    }

    public Vector2D getMomentum() {
        return momentum;
    }

    public double getMass() {
        return mass;
    }

}
```

1.3 Box

```
package gas;

public class Box extends Object {

    public double width;
    public double height;

    /** Creates new Box */
    public Box() {
        width = 1.0;
        height = 1.0;
    }

    public Box(double width, double height) {
        this.width = width;
        this.height = height;
    }

    public Vector2D randomPosition() {
        Vector2D v = new Vector2D(Math.random()*width, Math.random()*height);
        return v;
    }

    public Vector2D randomMomentum(double maxMomentum) {
        double r = Math.random()*maxMomentum;
        double theta = Math.random()*Math.PI*2.0;


        Vector2D v = new Vector2D(r*Math.cos(theta), r*Math.sin(theta));

        return v;
    }

    public void reflect(Molecule m) {
        Vector2D pos = m.getPosition();
        Vector2D mom = m.getMomentum();


        if (pos.x < 0.0) {
            pos.x = -pos.x;
            mom.x = -mom.x;
        }

        if (pos.y < 0.0) {
            pos.y = -pos.y;
            mom.y = -mom.y;
        }

        if (pos.x > width) {
            pos.x = 2*width - pos.x;
            mom.x = -mom.x;
        }

        if (pos.y > height) {
            pos.y = 2*height - pos.y;
            mom.y = -mom.y;
        }
```

```
    }

}
```

1.4 Gas

```
package gas;

public class Gas extends Object {
    private Molecule[] molecules;
    private int numMolecules;
    private Box box;

    /** Creates new Gas */
    public Gas() {
        molecules = new Molecule[10];
        numMolecules = 0;
        box = new Box(1.0, 1.0);
    }

    public Gas(int maxMolecules) {
        molecules = new Molecule[maxMolecules];
        numMolecules = 0;
        box = new Box(1.0, 1.0);
    }

    public boolean addMolecule(Molecule m) {
        if (numMolecules >= molecules.length) {
            return false;
        } else if (m == null) {
            return false;
        } else {
            molecules[numMolecules] = m;
            numMolecules++;
            return true;
        }
    }

    public void advance(double time) {
        // Advance every molecule the given amount of time.
        // Also handle reflection from box boundaries.

        for (int i=0; i<numMolecules; i++) {
            molecules[i].advance(time);
            box.reflect(molecules[i]);
        }

    }

    public Molecule[] getMolecules() {
        return molecules;
    }

    public Box getBox() {
        return box;
    }

}
```

1.5 GasCanvas

```
package gas;

import java.awt.*;

public class GasCanvas extends java.awt.Canvas {
    private Gas gas;

    /** Creates new GasCanvas */
    public GasCanvas() {
    }
```

```
    public GasCanvas(Gas gas) {
        this.gas = gas;
    }

    public void setGas(Gas gas) {
        this.gas = gas;
    }

    public void paint(Graphics g) {
        if (gas != null) {
            double width = getWidth();
            double height = getHeight();
            double scale;


            Molecule[] molecules = gas.getMolecules();
            Box box = gas.getBox();

            scale = width/box.width;

            if (height/box.height < scale) {
                scale = height/box.height;
            }

            // Draw the box.
            g.drawRect(0, 0, (int)(box.width*scale) - 1,
                             (int)(box.height*scale) - 1);

            for (int i=0; i < molecules.length; i++) {
                Vector2D position = molecules[i].getPosition();

                g.fillOval((int)(scale*position.x) - 3,
                           (int)(scale*position.y) - 3, 6, 6);
            }

        }
    }


}
```

1.6 GasSimulation

```
package gas;

public class GasSimulation extends java.awt.Frame {

    private GasCanvas gasCanvas;
    private Gas gas;
    /** Creates new GasSimulation */
    public GasSimulation() {
        setSize(300, 300);
        setLocation(200, 200);

        gas = new Gas(100);
        Box box = gas.getBox();

        for (int i=0; i<100; i++) {
            Molecule m = new Molecule(1.0, box.randomPosition(),
                                      box.randomMomentum(3.0));
            gas.addMolecule(m);
        }

        gasCanvas = new GasCanvas(gas);
        add(gasCanvas);


    }

    public void step() {
        gas.advance(0.01);
        gasCanvas.repaint();
```

```
    }

    public static void main(String[] args) {
        GasSimulation simulation = new GasSimulation();
        simulation.setVisible(true);

        while (true) {
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                // Ignored
            }

            simulation.step();
        }

    }
}
```

2 Improving the Java Application

This time, we have an improved version of the gas simulation program from last time. This time, molecules collide as well! Here's a listing of the code where the program has been modified:

2.1 Vector2D

```
public Vector2D subtract(Vector2D v) {
    Vector2D diff;

    diff = new Vector2D(this.x - v.x, this.y - v.y);

    return diff;
}

public Vector2D add(Vector2D v) {
    Vector2D sum;

    sum =  new Vector2D(this.x + v.x, this.y + v.y);

    return sum;

}

public Vector2D multiply(double k) {
    Vector2D product;

    product = new Vector2D(k*this.x, k*this.y);

    return product;
}
```

2.2 Molecule

```
private double radius;

public Molecule() {
    mass = 1.0;
    radius = 0.01;
    position = new Vector2D(0.0, 0.0);
    momentum = new Vector2D(0.0, 0.0);
}

public Molecule(double mass, double radius, Vector2D position,
             Vector2D momentum) {
    this.mass = mass;
    this.radius = radius;
    this.position = position;
    this. momentum = momentum;
}

public double getEnergy() {
    return momentum.length() * momentum.length()/(2.0*mass);
}

public double getRadius() {
    return radius;
}

public void collide(Molecule other) {
    Vector2D r = other.position.subtract(this.position);

    if (r.length() <= (this.radius + other.radius)) {
        // We are in close proximity.
        // Now we need a unit vector in the direction of r.

        double l = r.length();
        r = r.multiply(1.0/l);

        // q is the magnitude of the momentum transfer.
        double q = 2.0/(this.mass+other.mass)
                   *other.momentum.multiply(this.mass)
                   .subtract(this.momentum.multiply(other.mass))
                   .dotProduct(r);

        // We are still approaching if and only if this is negative.

        if (q < 0.0) {

            this.momentum = this.momentum.add(r);
            other.momentum = other.momentum.subtract(r);
```

```
        }

    }

}
```

2.3 Gas

```
public void advance(double time) {
    // Advance every molecule the given amount of time.
    // Also handle reflection from box boundaries.

    for (int i=0; i<numMolecules; i++) {
        if (molecules[i] != null) {

            molecules[i].advance(time);

            for (int j=i+1; j<numMolecules; j++) {

                molecules[i].collide(molecules[j]);

            }

            box.reflect(molecules[i]);
        }
    }

}

public double getEnergy() {
    double energy=0.0;

    for (int i=0; i<numMolecules; i++) {
        if (molecules[i] != null) {
            energy += molecules[i].getEnergy();
        }
    }

    return energy;
}
```

2.4 GasCanvas

```
public class GasCanvas extends java.awt.Canvas
                   implements java.awt.event.ComponentListener {

    private Image buffer;

    public void update(Graphics g) {
        paint(g);
```

```
    }

    public void paint(Graphics g) {

        if (gas != null) {
            double width = getWidth();
            double height = getHeight();
            double scale;

            if (buffer == null) {
                buffer = createImage(getWidth(), getHeight());
            }

            Graphics h = buffer.getGraphics();

            Molecule[] molecules = gas.getMolecules();
            Box box = gas.getBox();

            scale = width/box.width;

            if (height/box.height < scale) {
                scale = height/box.height;
            }

            h.setColor(getBackground());
            h.fillRect(0, 0, (int)width, (int)height);
            h.setColor(getForeground());
            // Draw the box.
            h.drawRect(0, 0, (int)(box.width*scale) - 1, (int)(box.height*scale) -1);

            for (int i=0; i < molecules.length; i++) {
                Vector2D position = molecules[i].getPosition();
                double radius = molecules[i].getRadius();

                h.fillOval((int)(scale*position.x) - (int)(scale*radius),
                           (int)(scale*position.y) - (int)(scale*radius),
                           (int)(2.0*scale*radius), (int)(2.0*scale*radius));
            }

            h.dispose();
            g.drawImage(buffer, 0, 0, null);

        }
    }

    public void componentShown(java.awt.event.ComponentEvent evt) {
    }

    public void componentResized(java.awt.event.ComponentEvent evt) {
        buffer = createImage(getWidth(), getHeight());
```

```
    }

    public void componentHidden(java.awt.event.ComponentEvent evt) {
    }

    public void componentMoved(java.awt.event.ComponentEvent evt) {
    }
```

2.5 GasSimulation

```
        for (int i=0; i<70; i++) {
            Molecule m = new Molecule(1.0, 0.03, box.randomPosition(),
                                      box.randomMomentum(3.0));
            gas.addMolecule(m);
        }

        for (int i=0; i<30; i++) {
            Molecule m = new Molecule(5.0, 0.05, box.randomPosition(),
                                      box.randomMomentum(3.0));
            gas.addMolecule(m);
        }

        gasCanvas = new GasCanvas(gas);
        add(gasCanvas);

    public void step() {

        gas.advance(0.01);
        gasCanvas.repaint();

        if ((steps++ % 100) == 0) {
            System.out.println("Energy = " + gas.getEnergy());
        }
    }

    public void windowDeactivated(java.awt.event.WindowEvent evt) {
    }
    public void windowClosed(java.awt.event.WindowEvent evt) {
    }
    public void windowDeiconified(java.awt.event.WindowEvent evt) {
    }
    public void windowOpened(java.awt.event.WindowEvent evt) {
    }
    public void windowIconified(java.awt.event.WindowEvent evt) {
    }
    public void windowClosing(java.awt.event.WindowEvent evt) {
        System.exit(0);
    }
    public void windowActivated(java.awt.event.WindowEvent evt) {
```

```
        }
}
```