```
import java.awt.*;
import java.awt.geom.*;
import java.awt.image.*;
import java.awt.event.*;
import java.util.*;

import javax.swing.*;
import javax.swing.Timer;

public class Simulation {

    public static void main(String[] args) {
        Interface content = new Interface("Pathing Simulation");

        content.update(content.getGraphics());
    }
}

class Interface extends JFrame {

    private JMenuBar UImenu;
    private JMenu Simulation, Units, Terrain, Settings;
    private JMenuItem toBeRecycled;
    ArrayList<Unit> simUnits = new ArrayList();
    ArrayList<Interactable> simTerrain = new ArrayList();

    int width, height;

    boolean addingUnit = false, addingUnits = false, removingUnit = false, removingUnits =
 false;

    //ArrayList<ArrayList<moveFlag>> grid = new ArrayList();
    moveFlag[][] grid;

    public Interface(String title) {

        super(title);

        setSize(700, 730);

        setLocation(0, 0);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);

        createBufferStrategy(3);

        JPanel temp = new JPanel(true);
        temp.setSize(getContentPane().getSize());
        temp.addMouseListener(new mouseInterface());
        setContentPane(temp);

        setRootPaneCheckingEnabled(true);
        //setLayout(new GridBagLayout());

        setJMenuBar(createMenus());

        paint(getGraphics());

        width = (int) (getContentPane().getSize().getWidth());

        height = (int) (getContentPane().getSize().getHeight());

        //ArrayList<moveFlag> stuff;
```

```
        System.out.println(width + " w   h " + height);

        grid = new moveFlag[width][height];

        paint(getGraphics());
        /*grid.ensureCapacity(WIDTH);

        for(int aa = 0; aa < width; aa++) {
            grid.set(aa ,stuff = new ArrayList());
            (grid.get(aa)).ensureCapacity(height);
            for(int bb = 0; bb < height; bb++) {
                (grid.get(aa)).set(bb, null);
            }
        }*/

        new Timer(10, new simTimer()).start();

        /*
        Component tempGlass = getGlassPane();
        tempGlass.addMouseListener(new mouseInterface());
        tempGlass.setEnabled(true);
        tempGlass.setVisible(true);
        setGlassPane(tempGlass);
        System.out.println(tempGlass.getClass() + "  " + tempGlass.getX() + "  " +
tempGlass.getY() + "  " + tempGlass.getHeight() + "  " + tempGlass.getWidth());
        */
        }

    private class simTimer implements ActionListener {
        public void actionPerformed(ActionEvent e)
        {
            step();
        }
        //this is the timer that runs the game
        //each cycle on the timer is .01 of a second
    }

    private class unitAdder implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            addingUnit = true;
            //System.out.println("testing");
        }
    }

    private class mouseInterface implements MouseListener {
        public void mouseClicked(MouseEvent e) {
            if(addingUnit) {
                //System.out.println(e.getX() + "    " + e.getY());
                //System.out.println(e.getXOnScreen() + "    " + e.getYOnScreen());
                createUnit(e.getX(), e.getY());
                addingUnit = false;
            }
        }

        public void mouseEntered(MouseEvent e) {

        }

        public void mouseExited(MouseEvent e) {

        }
```

```
    public void mousePressed(MouseEvent e) {

    }

    public void mouseReleased(MouseEvent e) {

    }
}

public void step() {
    /*System.out.println("BOO");
    System.out.println(grid);
    System.out.println();
    System.out.println(simUnits.toString());*/

    Unit tUnit;

    for (int d = 0; d < simUnits.size(); d++) {

        tUnit = simUnits.get(d);

        //System.out.println((tUnit.getPos())[0] + "     " + (tUnit.getPos())[1]);

        ((getContentPane()).getGraphics()).drawOval((tUnit.getPos())[0],
                (tUnit.getPos())[1], 5, 5);

        //g.setColor(Color.BLACK);
        //g.drawOval((tUnit.getPos())[0], (tUnit.getPos())[1], 5, 5);
    }

    update(getGraphics());
}

public void paints(Graphics g) {

    System.out.println("BOO");
    System.out.println(grid);
    System.out.println();
    System.out.println(simUnits.toString());

    Unit tUnit;

    for (int d = 0; d < simUnits.size(); d++) {

        tUnit = simUnits.get(d);

        g.setColor(Color.BLACK);
        g.drawOval((tUnit.getPos())[0], (tUnit.getPos())[1], 5, 5);
    }

    super.paint(g);
}

public void createUnit(int xp, int yp) {
    Unit u = new Unit(xp, yp);

    boolean temp = true;

    moveFlag[][] tempA = grid;

    for (int a = xp; a < xp + 5 && temp; a++) {
        for (int b = yp; b < yp + 5 && temp; b++) {
            temp = (boolean) (grid[a][b] == null);
```

```
                tempA[a][b] = (new moveFlag(u, u.getPos()));
            }
        }

        /*ArrayList<ArrayList<moveFlag>> tempA = grid;
        ArrayList<moveFlag> t;

        for(int a = xp; a < xp+5 && temp; a++) {
            for(int b = yp; b < yp+5 && temp; b++) {
                temp = ((grid.get(a)).get(b) == null);
                (tempA.get(a)).add(new moveFlag(u, u.getPos()));
            }
        }*/

        if (temp) {
            simUnits.add(new Unit(xp, yp));
            grid = tempA;
        }

    }

    public void createUnits(int xp1, int xy1, int xp2, int yp2) {

    }

    public void removeUnit(moveFlag t) {
        simUnits.remove(t.mover);
    }

    public void checkUnitCollisions() {

    }

    private JMenuBar createMenus() {
        UImenu = new JMenuBar();
        // ---------------------------------------------------------------

        UImenu = new JMenuBar();

        Simulation = new JMenu("Simulation Controls");
        Simulation.setMnemonic(KeyEvent.VK_C);
        UImenu.add(Simulation);

        toBeRecycled = new JMenuItem("New Simulation", KeyEvent.VK_N);
        toBeRecycled.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N,
                ActionEvent.CTRL_MASK));
        Simulation.add(toBeRecycled);

        toBeRecycled = new JMenuItem("Open Simulation", KeyEvent.VK_O);
        toBeRecycled.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,
                ActionEvent.CTRL_MASK));
        Simulation.add(toBeRecycled);

        toBeRecycled = new JMenuItem("Save Simulation", KeyEvent.VK_S);
        toBeRecycled.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
                ActionEvent.CTRL_MASK));
        Simulation.add(toBeRecycled);

        toBeRecycled = new JMenuItem("Save Simulation As...", KeyEvent.VK_S);
        toBeRecycled.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
                ActionEvent.CTRL_MASK + ActionEvent.SHIFT_MASK));
        Simulation.add(toBeRecycled);
```

```
        Units = new JMenu("Units");
        Simulation.setMnemonic(KeyEvent.VK_U);
        UImenu.add(Units);

        toBeRecycled = new JMenuItem("Add Unit", KeyEvent.VK_U);
        toBeRecycled.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_U,
                ActionEvent.CTRL_MASK));
        toBeRecycled.addActionListener(new unitAdder());
        Units.add(toBeRecycled);

        toBeRecycled = new JMenuItem("Add Units", KeyEvent.VK_U);
        toBeRecycled.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_U,
                ActionEvent.CTRL_MASK + ActionEvent.SHIFT_MASK));
        Units.add(toBeRecycled);

        toBeRecycled = new JMenuItem("Remove Unit", KeyEvent.VK_R);
        toBeRecycled.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_R,
                ActionEvent.CTRL_MASK));
        Units.add(toBeRecycled);

        toBeRecycled = new JMenuItem("Remove Units", KeyEvent.VK_R);
        toBeRecycled.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_R,
                ActionEvent.CTRL_MASK + ActionEvent.SHIFT_MASK));
        Units.add(toBeRecycled);

        toBeRecycled = new JMenuItem("Select All Units", KeyEvent.VK_A);
        toBeRecycled.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_A,
                ActionEvent.CTRL_MASK));
        Units.add(toBeRecycled);

        // ----------------------------------------------------------------
        /*GridBagConstraints MenuBarConstraint = new GridBagConstraints(0, 0,
                super.getWidth(), super.getHeight(), 1, 0,
                GridBagConstraints.WEST, GridBagConstraints.BOTH, new Insets(0,
                        0, 0, 0), 0, 0);

        ((LayoutManager2) getLayout()).addLayoutComponent(UImenu,
                MenuBarConstraint);*/

        return UImenu;
    }
}

class Interactable {

    protected Shape includes;
    public boolean isSelected;
    protected int x, y;

    public Shape getShape() {
        return includes;
    }

    public boolean selected() {
        return isSelected;
    }

    public int[] getPos() {
        int[] temp = new int[2];
        temp[0] = x;
        temp[1] = y;
        return temp;
    }
```

```
    /*
     * public Interactable(boolean diff, int x, int y, int num, int wid, int
     * hei) { }
     *
     * private static Polygon generateRandomPolygon(int x, int y, int num, int
     * wid, int hei) { Point2D[] points = new Point2D[num];
     *
     * int tempx, tempy;
     *
     * Line2D templine1, templine2;
     *
     * for(int a = 0; a < (points.length); a++) { tempx =
     * (int)Math.rint(Math.random() * wid); tempy = (int)Math.rint(Math.random() *
     * hei); if() } }
     */
}

class Unit extends Interactable {

    private Path PATH;

    public Unit(int xp, int yp) {
        super.x = xp;
        super.y = yp;
        super.includes = new Ellipse2D.Double(x, y, 5, 5);
        super.isSelected = false;
    }

    public void move(int xp, int yp) {
        super.x += xp;
        super.y += yp;
    }

    public Path getPath() {
        return PATH;
    }
}

class moveFlag {

    public Unit mover;
    public int[] cords;

    public moveFlag(Unit u, int[] crds) {
        mover = u;
        cords = crds;
    }
}

class Path {

    ArrayList<moveFlag> flags = new ArrayList();
    moveFlag end;

    public Path(moveFlag start, moveFlag ends) {
        flags.add(start);
        end = ends;
    }

    public void getToPoint() {
        flags.remove(0);
    }
```

```
    public void changeRoute(int point, moveFlag np) {
        flags.set(point, np);
    }
}
```