MASON

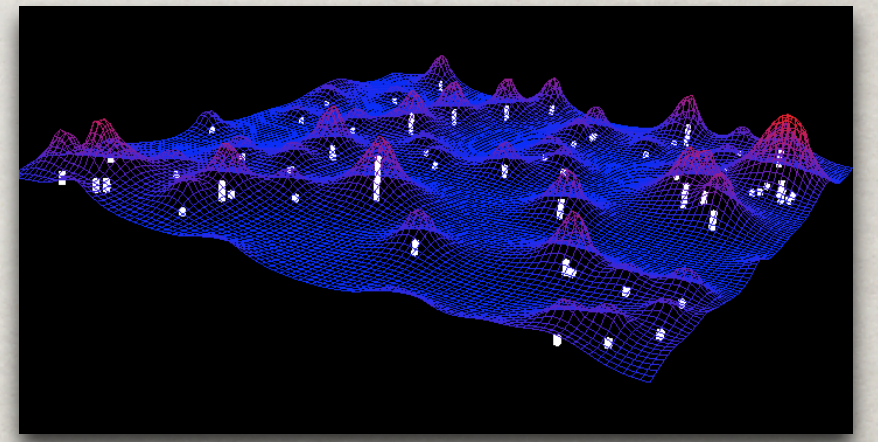
WCCS 2006 Tutorial

Sean Luke
George Mason University

MASON is a joint production of GMU's Computer Science Department and Center for Social Complexity

MASON

- **M**ulti **A**gent **S**imulation **O**f **N**eighborhoods... or **N**etworks... or something...



- Fast, portable, open source, general-purpose multi-agent core in Java, plus visualization tools and media tools

MASON Is Designed For...

- Highly Customized Experiments
- Experiments Entailing Many Model Runs
  - Parameter Optimization, Parameter Sweeps, Robustness Testing
- A Wide Range of Problem Areas
  - AI, Robotics, Physics, Biological and **Social Complexity Modeling**

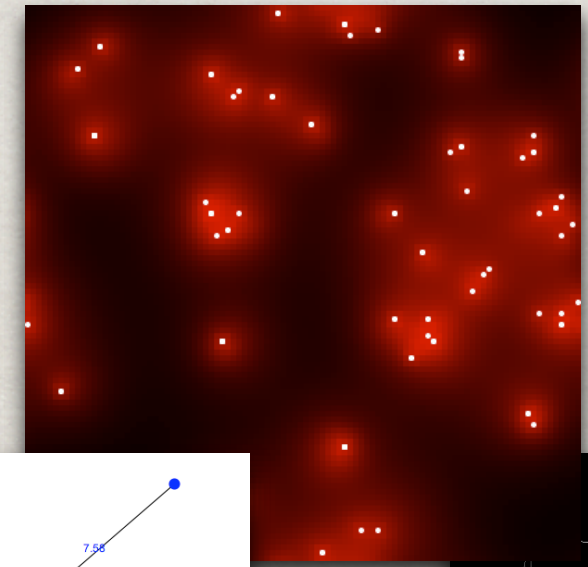
MASON Design Goals

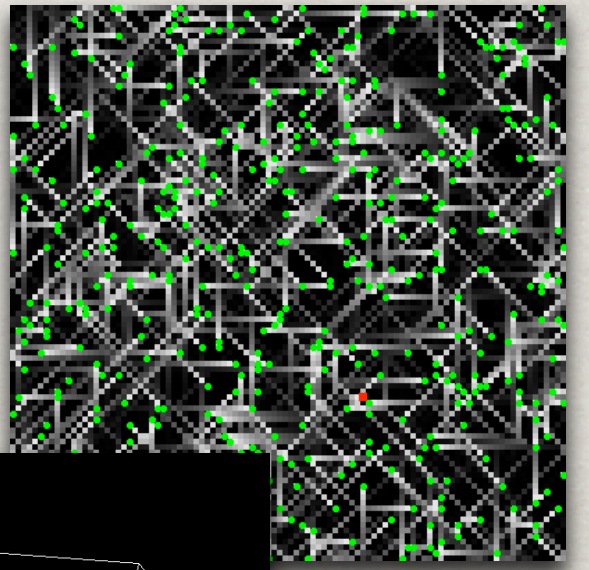
- Large numbers of parallel simulations
- Guaranteed duplicatable scientific results
- High degree of modularity and flexibility
- Small, easy to understand core model
- Separate visualization tools
- Checkpointing and recovery
- Fast
- Support high numbers of agents
- *Not a goal:* distributed simulation

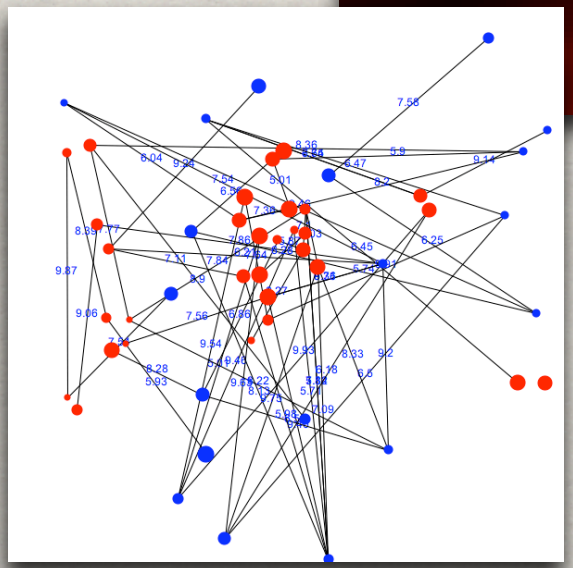
Built-in Fields in MASON

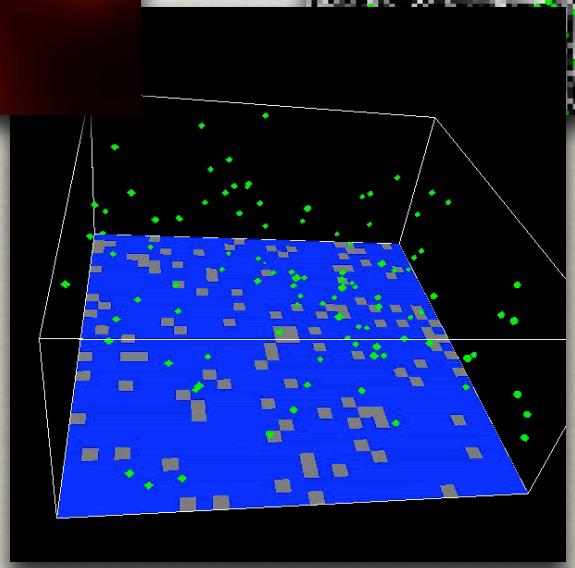
- Dense 2D, 2D hexagonal, and 3D grids of ints, doubles, and Objects (can be toroidal)
- Sparse 2D, 2D hexagonal, and 3D grids of Objects (can be toroidal)
- Continuous 2D or 3D space (can be toroidal)
- Networks (visualizable in 2D or 3D)
- Not Sufficient? Easy to make your own!

Examples

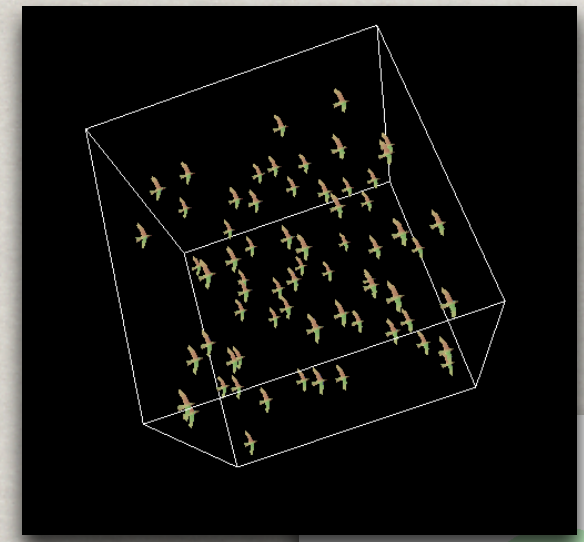


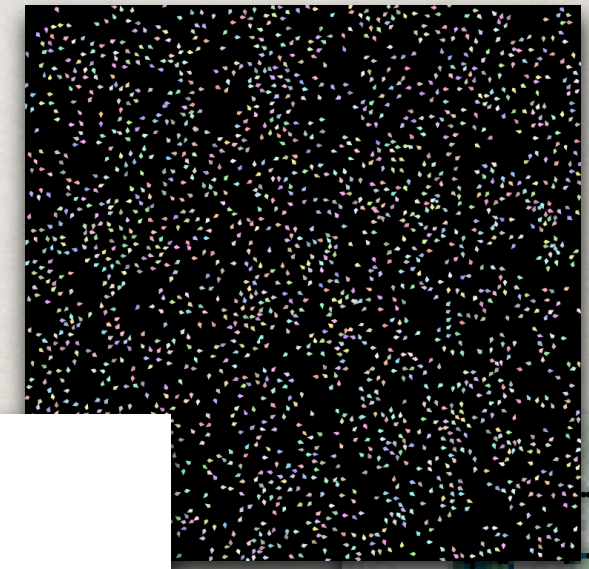


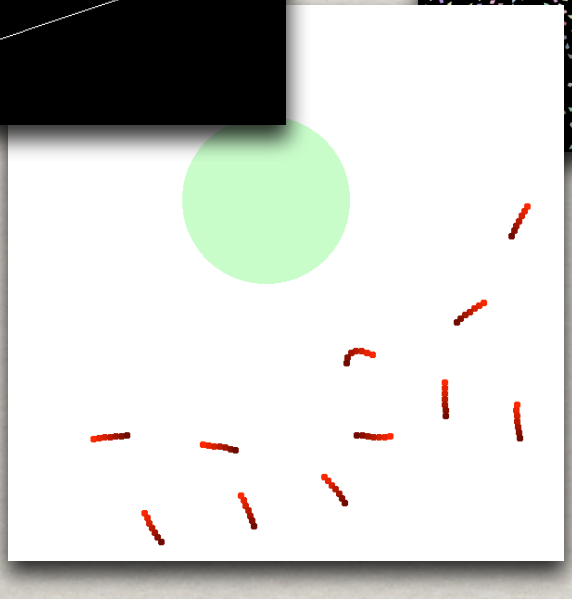


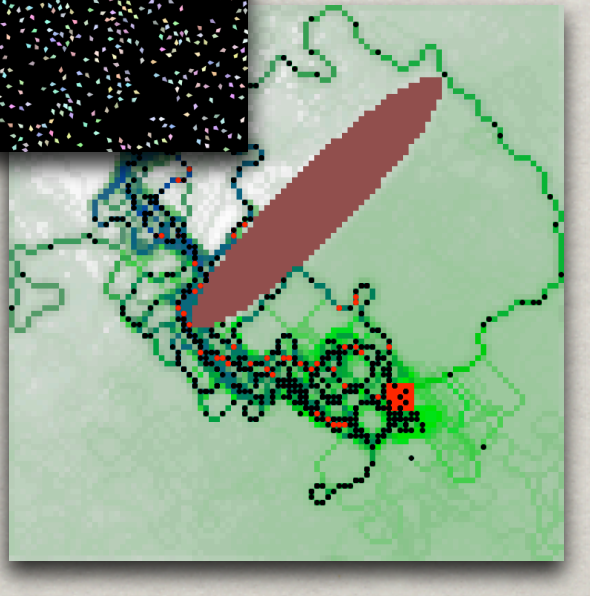


Examples

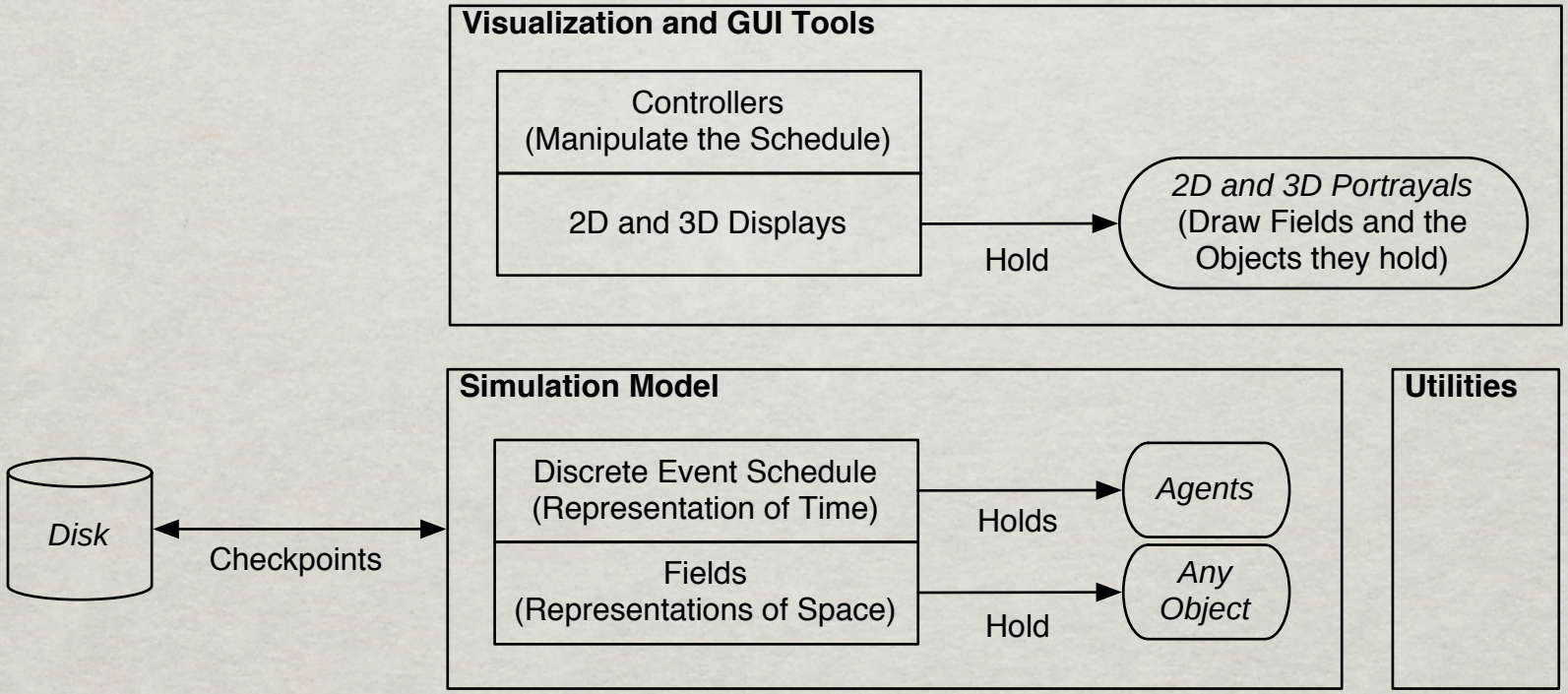




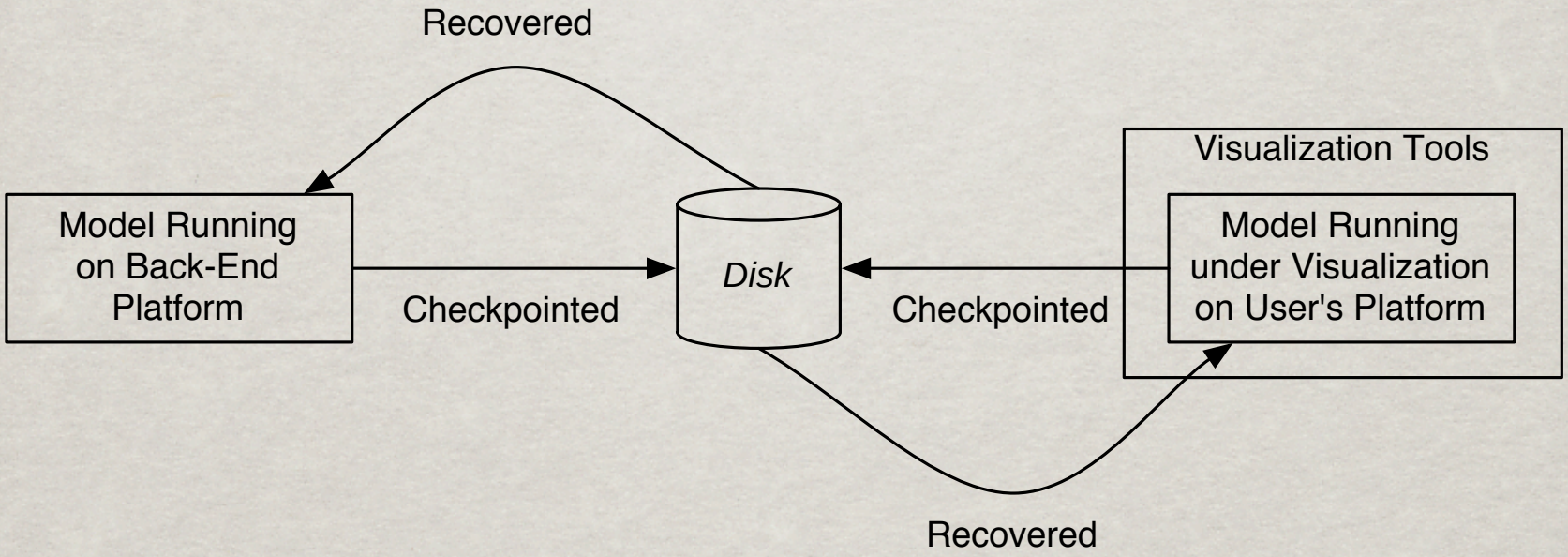




Layer Interactions



Checkpointing and Recovery



The Tutorial: A Basic Schelling Segregation Model

- Get MASON Running
- A Minimum Simulation
- Add a Field (a representation of space)
- Define our Agents
- Set up the Agents
- Build a Minimal GUI
- Add a Display
- Make the Agents Do Something

Time Permitting...

- Add Per-Agent Inspectors ("probes")
- Add a Model Inspector
- Add a Histogram
- Add a Time Series Chart
- Speed Up Drawing
- Checkpoint the Simulation
- More Demos!

To Run MASON Demos

- Install Java3D
- Move each **jar** file into the **mason** directory
- Double-click on the appropriate icon for your operating system in the **mason/start** directory:
  - OS X: **mason.command**
  - Linux: **mason.sh**
  - Windows: **mason.bat**

Setup To Build the WCSS Tutorial Examples

- Install Java3D
- The **mason** directory and each of the **jar files** in the "jar" directory must be added to your CLASSPATH.
- The WCSS tutorial code is in **mason/sim/app/wcss**
- The WCSS tutorial documentation is in **tutorialDocs/**tutorial

Schelling.java

```
package sim.app.wcss;

import sim.engine.*;
import ec.util.*;
import sim.util.*;

public class Schelling extends SimState
    {
    public Schelling(long seed)
        {
        super(new MersenneTwisterFast(seed), new Schedule());
        }

    /** Resets and starts a simulation */
    public void start()
        {
        super.start();  // clear out the schedule
        }

    public static void main(String[] args)
        {
        doLoop(Schelling.class, args);
        System.exit(0);
        }
    }
```

Output

```
MASON Version 12.  For further options, try adding ' -help' at end.
Job: 0 Seed: 1155246793094
Starting sim.app.wcss.Schelling
Exhausted
```

Schelling.java

```
package sim.app.wcss;

import sim.engine.*;
import ec.util.*;
import sim.util.*;
import sim.field.grid.*;

public class Schelling extends SimState
    {
    public int neighborhood = 1;
    public double threshold = 3;
    public int gridHeight = 100;
    public int gridWidth = 100;

    public ObjectGrid2D grid = new ObjectGrid2D(gridWidth, gridHeight);
    public Bag emptySpaces = new Bag();
```

• • •

Agent.java

```
package sim.app.wcss;
import sim.util.*;
import sim.engine.*;

public abstract class Agent implements Steppable
    {
    Int2D loc;

    public Agent(Int2D loc)
        {
        this.loc = loc;
        }

    public abstract boolean isInMyGroup(Agent obj);

    public void step( final SimState state )
        {
        }
    }
```

Red.java

```
package sim.app.wcss;
import sim.util.*;
import sim.engine.*;

public class Red extends Agent
    {
    public Red(Int2D loc) { super(loc); }

    public boolean isInMyGroup(Agent obj)
        {
        return (obj!=null && obj instanceof Red);
        }
    }
```

Blue.java

```
package sim.app.wcss;
import sim.util.*;
import sim.engine.*;

public class Blue extends Agent
    {
    public Blue(Int2D loc) { super(loc); }

    public boolean isInMyGroup(Agent obj)
        {
        return (obj!=null && obj instanceof Blue);
        }
    }
```

Schelling.java

```
public double redProbability = 0.333;
public double blueProbability = 0.333;

public void start()
    {
    super.start();

    // clear out the grid and empty space list
    emptySpaces = new Bag();
    grid = new ObjectGrid2D(gridWidth, gridHeight);  // first, all are null ("EMPTY")

    // add the agents to the grid and schedule them in the schedule
    for(int x=0 ; x<gridWidth ; x++)
        for(int y=0 ; y<gridHeight ; y++)
            {
            Steppable agent = null;
            double d = random.nextDouble();
            if (d < redProbability)
                {
                agent = new Red(new Int2D(x,y));
                schedule.scheduleRepeating(agent);
                }
            else if (d < redProbability + blueProbability)
                {
                agent = new Blue(new Int2D(x,y));
                schedule.scheduleRepeating(agent);
                }
            else  // add this location to empty spaces list
                {
                emptySpaces.add(new Int2D(x,y));
                }
            grid.set(x,y,agent);
            }
    }
```

Output

```
MASON Version 12.  For further options, try adding ' -help' at end.
Job: 0 Seed: 1155675734985
Starting sim.app.wcss.Schelling
Steps: 500 Time: 499 Rate: 412.20115
Steps: 1000 Time: 999 Rate: 425.17007
Steps: 1500 Time: 1499 Rate: 423.72881
Steps: 2000 Time: 1999 Rate: 424.44822
Quit
```

SchellingWithUI.java

```
package sim.app.wcss;
import sim.engine.*;
import sim.display.*;

public class SchellingWithUI extends GUIState
    {
    public SchellingWithUI() { super(new Schelling(System.currentTimeMillis())); }
    public SchellingWithUI(SimState state) { super(state); }

    public static String getName() { return "Schelling Segregation WCSS2006 Tutorial"; }

    public void init(Controller c)
        {
        super.init(c);
        }

    public void start()
        {
        super.start();
        }

    public void load(SimState state)
        {
        super.load(state);
        }

    public static void main(String[] args)
        {
        SchellingWithUI schUI = new SchellingWithUI();
        Console c = new Console(schUI);
        c.setVisible(true);
        }
    }
```

The Console


Schelling Segregation WCSS2006 Tutorial
File
About
Console
Displays
Inspectors
Schelling Segregation
WCSS 2006 Tutorial
A variation of Thomas Schelling's Segregation Model. We model two groups of people: Red and Blue. Each person prefers that there be at least threshold number of like-colored people living within neighborhood distance away from him. If this is not the case, he will get up and move to another location. In our version of the model, when a person wishes to move, he picks a random empty location to move to.
Time

SchellingWithUI.java

```
package sim.app.wcss;
import sim.engine.*;
import sim.display.*;

import sim.portrayal.*;
import sim.portrayal.grid.*;
import sim.portrayal.simple.*;
import java.awt.*;
import javax.swing.*;

public class SchellingWithUI extends GUIState
    {
    public SchellingWithUI() { super(new Schelling(System.currentTimeMillis())); }
    public SchellingWithUI(SimState state) { super(state); }

    public static String getName() { return "Schelling Segregation WCSS2006 Tutorial"; }
```

• • •

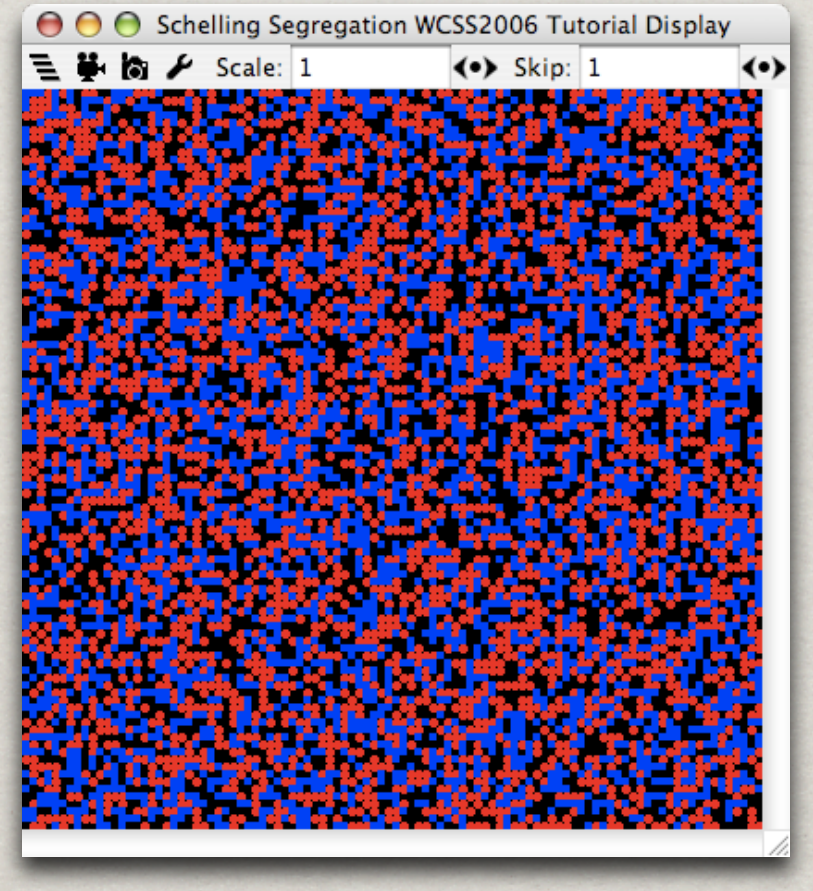
SchellingWithUI.java

```
public Display2D display;
public JFrame displayFrame;
ObjectGridPortrayal2D gridPortrayal = new ObjectGridPortrayal2D();

public void init(Controller c)
    {
    super.init(c);

    // Make the Display2D.  We'll have it display stuff later.
    Schelling sch = (Schelling)state;
    display = new Display2D(sch.gridWidth * 4, sch.gridHeight * 4,this,1);
    displayFrame = display.createFrame();
    c.registerFrame(displayFrame);   // register the frame so it appears in the "Display" list

    // attach the portrayals
    display.attach(gridPortrayal,"Agents");

    // specify the backdrop color  -- what gets painted behind the displays
    display.setBackdrop(Color.black);

    displayFrame.setVisible(true);
    }
```

SchellingWithUI.java

```
public void setupPortrayals()
    {
    // tell the portrayals what to portray and how to portray them
    gridPortrayal.setField(((Schelling)state).grid);
    gridPortrayal.setPortrayalForClass(Red.class, new OvalPortrayal2D(Color.red));
    gridPortrayal.setPortrayalForClass(Blue.class, new RectanglePortrayal2D(Color.blue));
    gridPortrayal.setPortrayalForNull(new SimplePortrayal2D());  // empty

    display.reset();    // reschedule the displayer
    display.repaint();  // redraw the display
    }

public void start()
    {
    super.start();
    setupPortrayals();  // set up our portrayals
    }

public void load(SimState state)
    {
    super.load(state);
    setupPortrayals();  // we now have new grids.  Set up the portrayals to reflect this
    }
```

The Display


Schelling Segregation WCSS2006 Tutorial Display
Scale: 1
Skip: 1

Agent.java

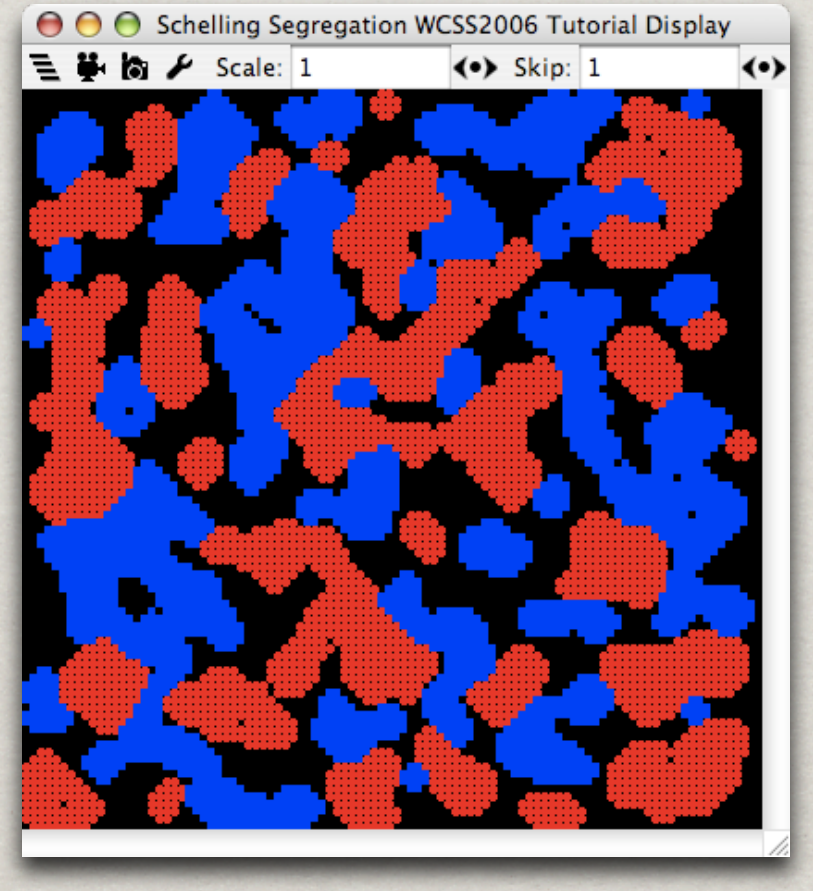
```
package sim.app.wcss;
import sim.util.*;
import sim.engine.*;

public abstract class Agent implements Steppable
    {
    Int2D loc;

    public Agent(Int2D loc)
        {
        this.loc = loc;
        }

    public abstract boolean isInMyGroup(Agent obj);

    IntBag neighborsX = new IntBag(9);
    IntBag neighborsY = new IntBag(9);
    double happiness = 0;

    public void step( final SimState state )
        {
        }
    }
```

• • •

Agent.java

```
public void step( final SimState state )
    {
    Schelling sch = (Schelling)state;
    if (sch.emptySpaces.size() == 0) return;  // nowhere to move to!

    // get all the places I can go.  This will be slow as we have to rely on grabbing neighbors.
    sch.grid.getNeighborsMaxDistance(loc.x,loc.y,sch.neighborhood,false,neighborsX,neighborsY);

    // compute happiness
    happiness = 0;
    int len = neighborsX.size();
    for(int i = 0; i < len; i++)
        {
        int x = neighborsX.get(i);
        int y = neighborsY.get(i);
        Agent agent = (Agent)sch.grid.get(x,y);
        if (agent != this && isInMyGroup(agent)) // if he's just like me, but he's NOT me
            // increment happiness by the linear distance to the guy
            happiness += 1.0/Math.sqrt((loc.x-x)*(loc.x-x) + (loc.y-y)*(loc.y-y));
        }
    if (happiness >= sch.threshold) return;  // I'm happy -- we're not moving

    sch.grid.set(loc.x,loc.y,null);        // Okay, so I'm unhappy.  First let's pull up roots.

    // Now where to go?  Pick a random spot from the empty spaces list.
    int newLocIndex = state.random.nextInt(sch.emptySpaces.size());

    // Swap the location in the list with my internal location
    Int2D newLoc = (Int2D)(sch.emptySpaces.get(newLocIndex));
    sch.emptySpaces.set(newLocIndex,loc);
    loc = newLoc;

    sch.grid.set(loc.x,loc.y,this);    // Last, put down roots again
    }
```

Display


Schelling Segregation WCSS2006 Tutorial Display
Scale: 1
Skip: 1

Output

```
MASON Version 12.  For further options, try adding ' -help' at end.
Job: 0 Seed: 1155675932470
Starting sim.app.wcss.Schelling
Steps: 250 Time: 249 Rate: 124.13108
Steps: 500 Time: 499 Rate: 126.83917
Steps: 750 Time: 749 Rate: 126.6464
Steps: 1000 Time: 999 Rate: 126.83917
Steps: 1250 Time: 1249 Rate: 126.19889
Steps: 1500 Time: 1499 Rate: 124.62612
Steps: 1750 Time: 1749 Rate: 120.36591
Steps: 2000 Time: 1999 Rate: 121.83236
Quit
```

Agent.java

```
package sim.app.wcss;
import sim.util.*;
import sim.engine.*;

public abstract class Agent implements Steppable, Valuable
    {
    Int2D loc;
    IntBag neighborsX = new IntBag(9);
    IntBag neighborsY = new IntBag(9);

    double happiness = 0;
    public double getHappiness() { return happiness; }
```

• • •

SchellingWithUI.java

```
package sim.app.wcss;
import sim.engine.*;
import sim.display.*;

import sim.portrayal.*;
import sim.portrayal.grid.*;
import sim.portrayal.simple.*;
import java.awt.*;
import javax.swing.*;

public class SchellingWithUI extends GUIState
    {
    public SchellingWithUI() { super(new Schelling(System.currentTimeMillis())); }
    public SchellingWithUI(SimState state) { super(state); }

    public static String getName() { return "Schelling Segregation WCSS2006 Tutorial"; }

    public Object getSimulationInspectedObject() { return state; }

    public Display2D display;
    public JFrame displayFrame;
    ObjectGridPortrayal2D gridPortrayal = new ObjectGridPortrayal2D();
```

• • •

Schelling.java

```
package sim.app.wcss;

import sim.engine.*;
import ec.util.*;
import sim.util.*;
import sim.field.grid.*;

public class Schelling extends SimState
    {
    public int neighborhood = 1;
    public double threshold = 3;
    public int gridHeight = 100;
    public int gridWidth = 100;

    public int getNeighborhood() { return neighborhood; }
    public void setNeighborhood(int val) { if (val > 0) neighborhood = val; }
    public double getThreshold() { return threshold; }
    public void setThreshold(double val) { if (val >= 0) threshold = val; }
    public double getRedProbability() { return redProbability; }
    public void setRedProbability(double val)
        { if (val >= 0 && val + blueProbability <= 1) redProbability = val; }
    public double getBlueProbability() { return blueProbability; }
    public void setBlueProbability(double val)
        { if (val >= 0 && val + redProbability <= 1) blueProbability = val; }
```

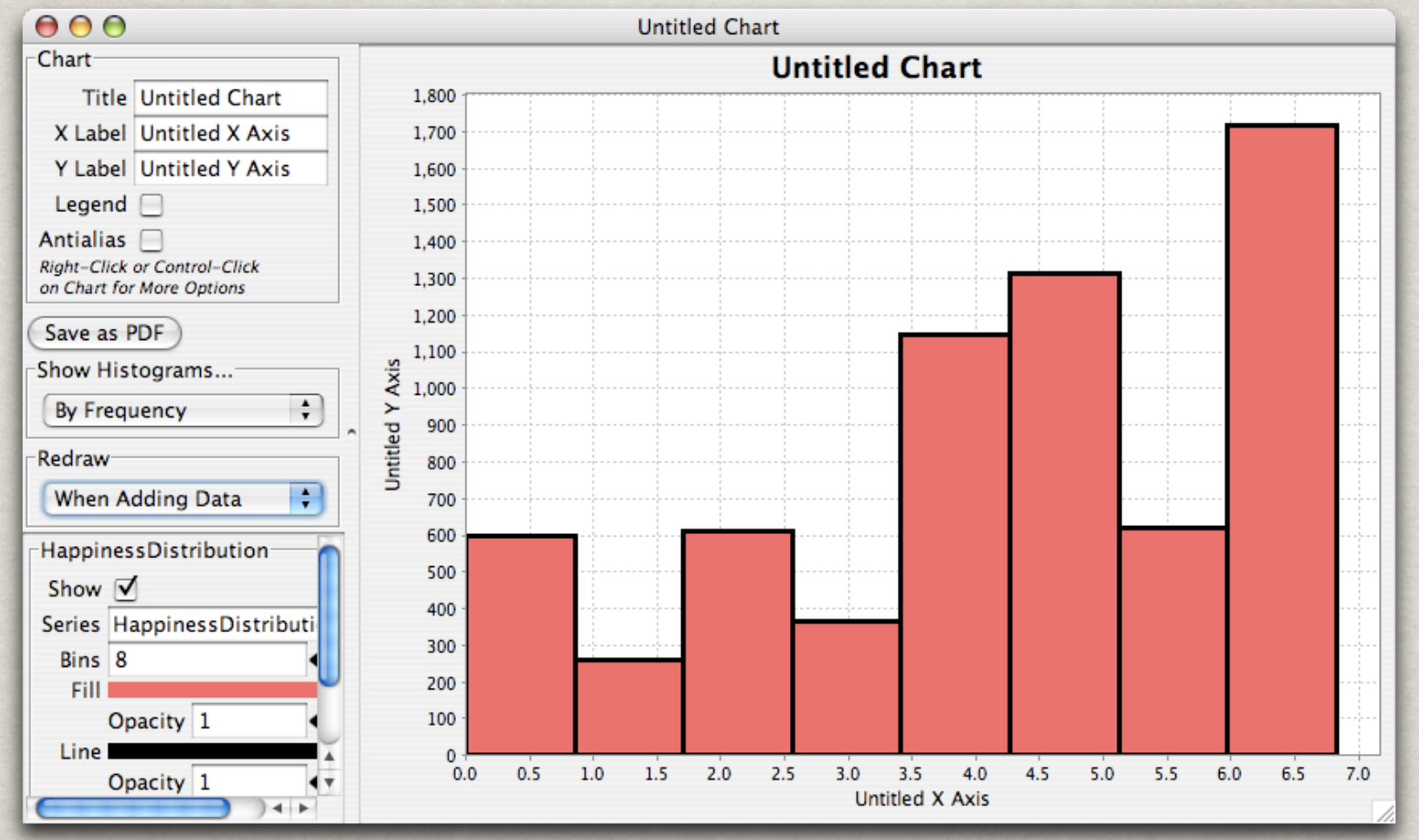
• • •

Schelling.java

```
package sim.app.wcss;

import sim.engine.*;
import ec.util.*;
import sim.util.*;
import sim.field.grid.*;

public class Schelling extends SimState
    {
    public DoubleBag getHappinessDistribution()
        {
        DoubleBag happinesses = new DoubleBag();
        for(int x=0;x < gridWidth;x++)
            for(int y=0;y < gridHeight;y++)
                if (grid.get(x,y)!=null)
                    happinesses.add(((Agent)(grid.get(x,y))).getHappiness());
        return happinesses;
        }
```

• • •

Histogram Display



Schelling.java

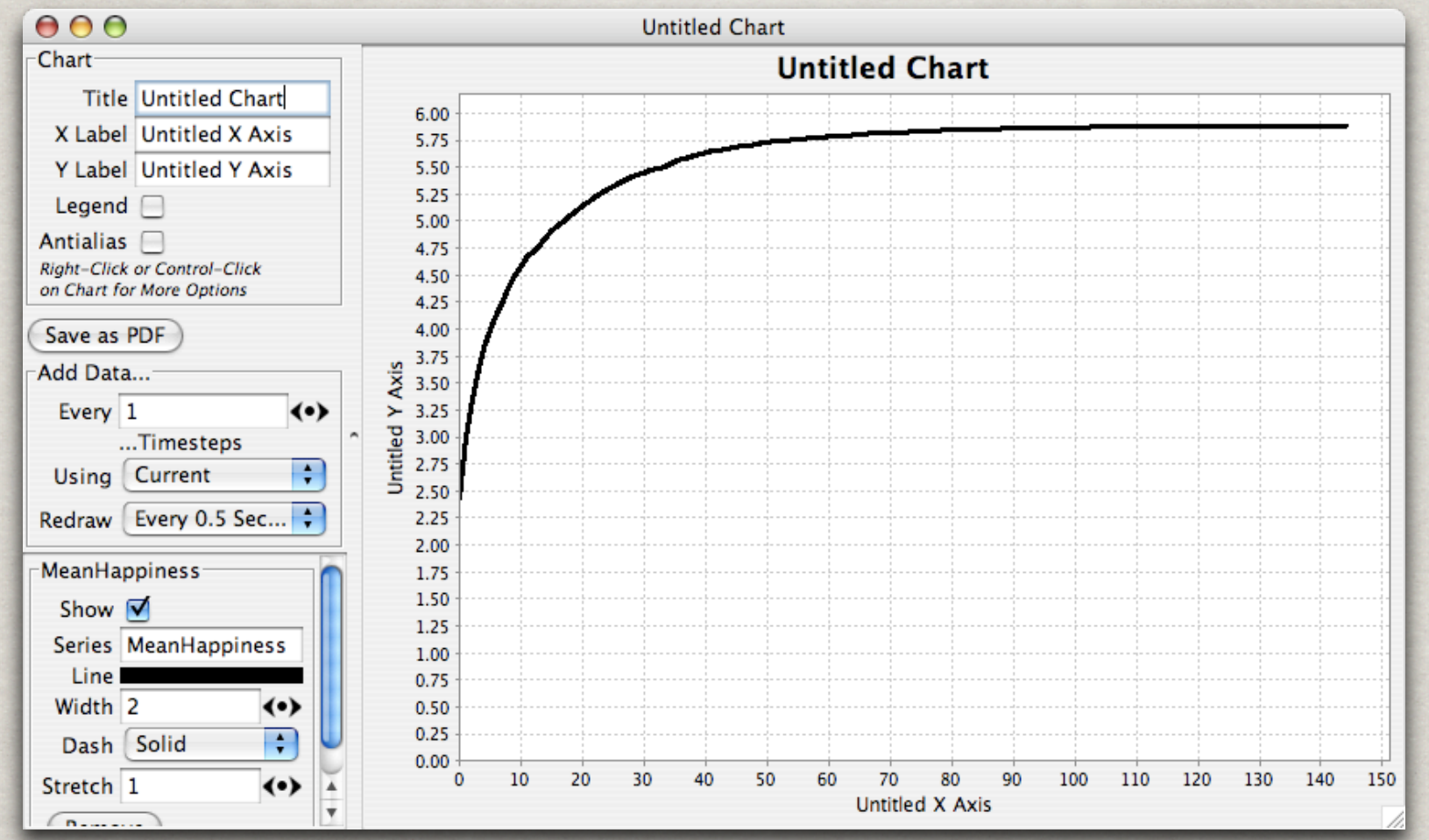
```
package sim.app.wcss;

import sim.engine.*;
import ec.util.*;
import sim.util.*;
import sim.field.grid.*;

public class Schelling extends SimState
    {
    public DoubleBag getHappinessDistribution()
        {
        DoubleBag happinesses = new DoubleBag();
        for(int x=0;x < gridWidth;x++)
            for(int y=0;y < gridHeight;y++)
                if (grid.get(x,y)!=null)
                    happinesses.add(((Agent)(grid.get(x,y))).getHappiness());
        return happinesses;
        }

    public double getMeanHappiness()
        {
        int count = 0;
        double totalHappiness = 0;

        for(int x=0;x < gridWidth;x++)
            for(int y=0;y < gridHeight;y++)
                if (grid.get(x,y)!=null)
                    {
                    count++;
                    totalHappiness += ((Agent)(grid.get(x,y))).getHappiness();
                    }
        if (count==0) return 0;
        else return totalHappiness / count;
        }
```

Agent.java

```
public abstract class Agent implements Steppable, Valuable
    {
    Int2D loc;

    public Agent(Int2D loc)
        {
        this.loc = loc;
        }

    public abstract double doubleValue();
```

• • •

Time Series Chart



Red.java

```
package sim.app.wcss;
import sim.util.*;
import sim.engine.*;

public class Red extends Agent
    {
    public Red(Int2D loc) { super(loc); }

    public boolean isInMyGroup(Agent obj)
        {
        return (obj!=null && obj instanceof Red);
        }

    public double doubleValue() { return 1; }
   }
```

Blue.java

```
package sim.app.wcss;
import sim.util.*;
import sim.engine.*;

public class Blue extends Agent
    {
    public Blue(Int2D loc) { super(loc); }

    public boolean isInMyGroup(Agent obj)
        {
        return (obj!=null && obj instanceof Blue);
        }

    public double doubleValue() { return 2; }
   }
```

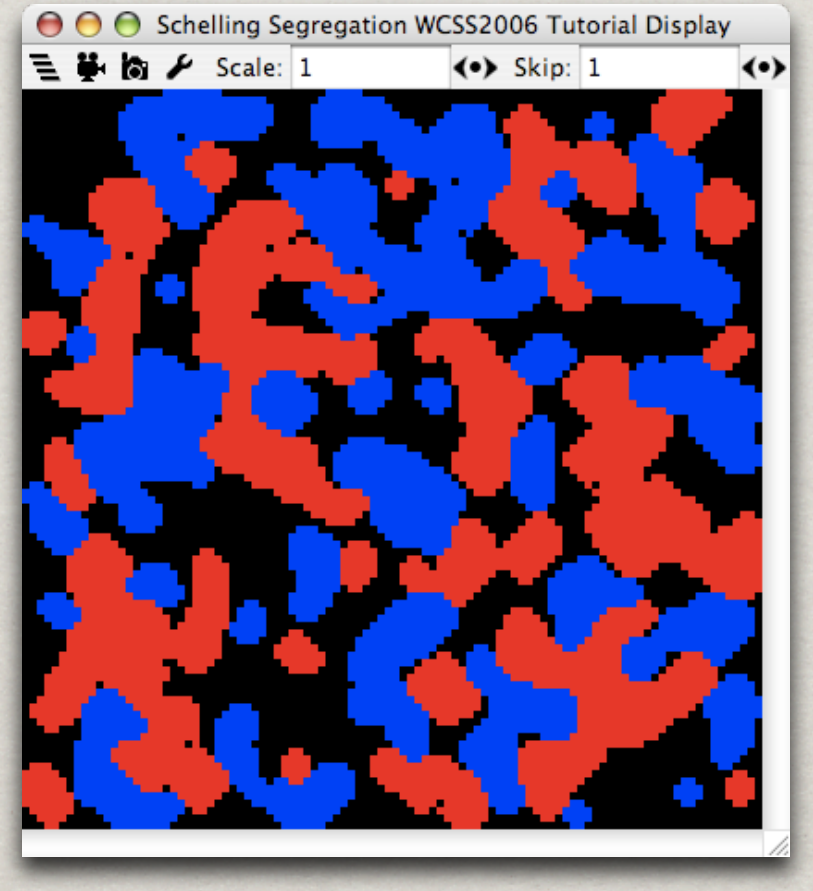
SchellingWithUI.java

• • •

```
public Display2D display;
public JFrame displayFrame;
FastObjectGridPortrayal2D gridPortrayal = new FastObjectGridPortrayal2D();

public void setupPortrayals()
    {
    // tell the portrayals what to portray and how to portray them
    gridPortrayal.setField(((Schelling)state).grid);
    gridPortrayal.setMap(new sim.util.gui.SimpleColorMap(
         new Color[] {new Color(0,0,0,0), Color.red, Color.blue}));

    display.reset();    // reschedule the displayer
    display.repaint();  // redraw the display
    }
```

• • •

Display (with Fast Portrayal)



Output

```
MASON Version 12.  For further options, try adding ' -help' at end.
Job: 0 Seed: 1155676001239
Starting sim.app.wcss.Schelling
Steps: 250 Time: 249 Rate: 117.53644
Steps: 500 Time: 499 Rate: 122.3092
Steps: 750 Time: 749 Rate: 120.71463
Steps: 1000 Time: 999 Rate: 121.24151
Checkpointing to file: 1000.0.Schelling.checkpoint
Steps: 1250 Time: 1249 Rate: 88.55827
Steps: 1500 Time: 1499 Rate: 124.06948
Steps: 1750 Time: 1749 Rate: 121.6545
Steps: 2000 Time: 1999 Rate: 123.57884
Checkpointing to file: 2000.0.Schelling.checkpoint
Steps: 2250 Time: 2249 Rate: 101.0101
Steps: 2500 Time: 2499 Rate: 123.88503
...
```

More on MASON

- MASON Home Page
  http://cs.gmu.edu/~eclab/projects/mason/
- Evolutionary Computation Laboratory
  Department of Computer Science
  http://cs.gmu.edu/~eclab/
- Center for Social Complexity
  http://socialcomplexity.gmu.edu