Technical Guide for the Agent based model for the simulation of the relationship between crime and inequality

Student 1	Oishin Smith
ID	16401096
Student 2	Aaron Edgeworth
ID	13493068
Title	Agent Based model to simulate the effects of inequality on crime
Type	Technical Guide
Date finished	16/05/2020
Supervisor	Renaat Verbruggen

Introduction

Overview and Motivations

This application is a simulation tool for use to help with studies regarding economic and social factors that relate to the presence of criminal behaviour in a society. The simulation model is an Agent Based Model, or ABM. Agent based models are a very useful tool for assessing the emergence of behaviours of autonomous units, or “agents”, in a system. For our application, the system is a representation of a basic functioning society, and the agents are representations of the people that live within that society. Within the simulator the agents have behaviours that they are compelled to perform. Based on these behaviours, as well as factors of the system they act in, differing trends of behaviours begin to emerge. The relationship between inequality and crime was first brought to prominence by Gary Becker in 1968 with his paper “Crime and Punishment: An Economic Approach”. Recently studies and polling have emerged that appear to verify many of these arguments. Countries that experience very high levels of income and social inequality also have high levels of reported criminal activity. Our tool will hopefully be useful for the study of these effects and the relationship between them. The simulator may have applications in the fields of Social Sciences and Economics.

Glossary

ABM - an agent based model (ABM) is a class of computational model that is used in simulating the actions and interactions of autonomous agents within their environment.
Simulation - computational simulations are used to simulate an abstract model of a system. They’ve become a useful part in mathematical modelling of many systems such as economics and social sciences
Run - run refers to the simulation running
Step - steps are the equivalent to passing time in a simulation
Agent - an Agent is autonomous and will interact with their environment in various ways. An agent will be referred to in multiple ways in this technical guide. Names such as citizen and criminal could be used.
Sites - an Agent will interact with a site in specific ways

Technologies Used:

Java Development Kit
MASON api
Swing Java
JUnit
JFreeChart
iText
Java Media Framework

Research

Computer simulation is a popular methodological approach for researchers. Researchers have to make many assumptions about the exact cause and effects of the system under study. In this case our application is a tool that helps studies regarding economic and social factors that look into the presence of criminal behaviour in a society. This application takes information from various research papers and studies to support the implementation of many of the featuress.

An argument was put forward by Nobel prize-winning economist Gary Becker, that all crime is economic and all criminals are rational. In his paper “Crime and punishment: an economic approach” he said that places with larger wealth gaps between the poor and the rich have higher crime. Our simulation tool uses the gini coefficient to test this correlation between the amount of income inequality and crimes being committed [2].

The simulation uses stress and wealth, based on research, to deduce the likelihood of a person to commit a crime due to economic stress. Research has concluded that there is a relationship between economic crimes and financial stress. It was found that economic crimes are most likely related to financial stress, which means that these crimes are often committed due to attempts to resolve financial problems [1]. Research also suggests that the relationship between economic stress and crime is that economic stress motivates individuals affected by it to offend [3]. According to the UNODC report “Monitoring the impact of economic crisis on crime”, during periods of economic stress, the incidence of robbery may double [4].

Illness was another factor included within the simulation. Research shows that economic factors such as a person's wealth may result in individuals' stress levels increasing. If stress gets high enough it can influence the distribution and frequency of illness [5][6]. In the simulation, if an agent gets stressed enough they will have a higher chance of getting sick.

Unemployment is another factor relating to crime, specifically robbery and theft [7]. Gary Becker describes crime as “an individual decision made based on potential loss and gain” [2]. This supports the idea that unemployment can lead to higher crime within a society and the simulation bases some of its features from this research [7].

During the simulation, agents pay a global tax. This money goes towards helping agents in need. If an Agent is poor enough, they will receive social welfare payments and the reason this was implemented was to see the effects of social welfare on a society vs no social welfare in a society. According to research, welfare programs raised the living standards of the most destitute citizens [8]. It also created a sort of safety net for all members of society [9] and a good implementation of welfare was needed in the simulation based on this research.

Design

Crime World Console

The crime world console is where the settings and statistical info about the run is displayed. It has an About window that shows a general description of the simulator, it's goals and how it works. There is a console tab that gives access to various settings that can be altered as the simulation is running. The delay of steps-per-second, run seed automatic pausing on specific steps and more. There is a display window that allows the application to show varying displays if they were available. Currently there is only the display of the society and a graph display that shows initial statistics on a run. The inspector tab will show the data of a given object in the simulation if the user selects it in the display window. Double-clicking on an agent will present the variables of that specific agent and can allow them to be altered in real time.

Crime World Display

The crime world displays the display in which to view the simulation. It consists of a window showing the various items of the world like the agents, relevant sites, where the agents are and where they move to. It also shows the current frame or "step" the simulation is on to indicate how long it has been running for.

Graphs window

The graphs window shows a user data about the state of the simulation at the very start of the run. It shows the user the income distribution at the start and the number of homeless people as well. This was designed to give a user a simpler way of seeing data relevant to the start of the simulation.

Play/Pause/Stop button

The play button can be pressed to start the simulation. Once the simulation starts all the agents and sites are rendered in and start moving. Each agent can be clicked and will continually display new data relating to each agent. Once the pause button is pressed, the simulation halts. The user will see that all agents have stopped moving and can press the play button to continue the simulation. If the stop button is pressed, the simulation stops and once the play button is pressed, a new simulation starts again.

Agents

Once a simulation has started, Agents will start moving around the sandbox and will go about their "daily" activities.

Sites: Depending on the time of the day, an agent will either work, roam or go home. When an agent goes to work they start earning income. The time when an agent goes to work varies, normally agents will start work at 9:00 and leave at 17:00 but some agents may start late or work in the afternoons or evenings.

Most Agents will also go home in the evenings (from 9pm onwards) to get some sleep and will reduce their stress as well. Agents may also be homeless or unemployed causing stress to those particular agents.

Needs: An agent's needs are constantly depleting as the simulation runs. An agent will have to replenish their needs once it gets low enough by going to a shop and paying money. This was done to simulate the needs of a real person.

Color: An agent's color will change depending on their stress level. If the agent is stressed enough, they will turn a red color and if the agent's stress is low then they will be a blue color. An agent can also become sick which will turn them green and the agent will have to pay extra money at the end of the day and become healthy.

Sites

Agents will interact with sites in different ways. The Work site is used by agents to earn money. They will typically go work from 9am to 5pm but this depends on the individual agent. The Shop is used by agents to replenish their needs and is open all day. Home is when an agent will go to sleep and reduce their stress.

Crime

An agent will go about their daily routine and earn money, sleep, roam etc. an agents stress will vary as the simulation progresses. An agent gets more stressed depending on the amount of money that they’re holding. This is due to the worry of not being able to pay their expenses or tax at the end of the day. Once an agents stress is high enough they may start committing crime and will in turn steal from other agents to get money.

Movement

Agents can roam around their environment or they go to specific places depending on their needs and the time of the day. An agent will also chase after another agent to simulate crime and will in turn rob money from the target agent. Movement can either be random or planned.

Time tracker

The simulation will constantly be tracking the time that it is live. The time is used by Agents to keep track of the day and each Agent will change their behaviour as the time changes. The simulation run time is also used to display the current time of the day on the top of the main CrimeWorld Window.

Disease

An Agent can become sick on any day. If an Agent becomes sick, they turn a greenish color and will have to pay extra money at the end of the day to recuperate.

Inspectors

Agents can be clicked to display a lot of information about them. Each agent will have different properties and the inspector will display the coordinates of the agent. Properties such as the citizen number, location, needs value, stress value, robed or robbed and wealth are displayed. These values give a user information about each Agent and every Agent will have different values.

Economic system

The economy will vary from getting better to getting worse. the better the economy gets the more jobs there are and the more income the agents will receive. This will affect crime and stress throughout the environment as well because there will be less of a worry for money and therefore less crime will happen. With a worsened or bad economy there will be less jobs and worst paying jobs available to the agents. Agents will not receive as many promotions and will struggle with paying for needs, taxes and expenses.

The economy was designed in such a way as to allow agents to gain varying amounts of income throughout the simulation. With a changing economy we are able to to calculate the gini coefficient and are also able to create larger wealth gaps between agents.

Graphs and statistics

The graphs and statistics window shows a user information about the Simulated world. When the play button is pressed, these graphs will be generated and will display for example, the income distribution and the amount of homeless people in the sandbox.

If you click on an individual agent, you have the option to chart different data about them.data like their wealth or their stress values going up and down.

3.0 Implementation & Sample code

Agents

We needed to come up with a good way of implementing agents, and in this case the project needed autonomous agents that would interact with their environment in various ways. All agents move freely and depending on the time of the day they'll do different things. Agents will also prioritize their needs and earning money so as to not gain stress and will commit crime if they have to. These agents are here to simulate real life people in real life situations to look into inequality within a society.

```
// generating and adding citizens into grid )
for (int i = 0; i < citizens; i++) {
     //createCitizensAttributes() creates the attributes for each generated citizen
       HashMap<String, String> citizenAttributes = createCitizenAttributes();
       Criminal criminal = new Criminal(i, this, steps, citizenAttributes);
```

```
        // Here we're adding the citizens to random locations
       Random random = new Random();
       int randomInt = random.nextInt(((100 - 0) + 1) + 0);
       int randomInt2 = random.nextInt(((100 - 0) + 1) + 0);
       yard.setObjectLocation(criminal, new Int2D(randomInt, randomInt2));

       // here Im adding the buddy relationships between citizens to get stress
       buddies.addNode(criminal);
       schedule.scheduleRepeating(criminal);

       allCriminals.add(criminal);
}
```

Sites

The reason we implemented sites was to give agents activities to do during the day. Each site will have different functionality and have different colors to represent the different sites. There are three different types of sites: Work, Home and Shop. An Agent will go to work to earn money, most agents will go home in the evenings to relax and sleep and agents will visit shops to replenish their needs. The implementation of the sites was necessary to simulate real buildings and activities that a normal person would do.

Home is initialized here:

```
// initialise Home site and create site location
sitesHome = Home.sites;
sitesHome.field[Home.location().x][Home.location().y] = 1;
for (int i = Home.location().x - 3; i < Home.location().x + 3; i++)
{
       for (int j = Home.location().y - 2; j < Home.location().y + 2; j++)
              sitesHome.field[i][j] = 1;
}
```

Then setupSitesPortrayals generates the Home sites in the environment for the user to see

```
private void setupSitePortrayals()
{
       CrimeWorld crimeWorld = (CrimeWorld) state;

       homeSitesPortrayal.setField(crimeWorld.sitesHome);
       // setMap for every pixel around the location
       homeSitesPortrayal.setMap(new sim.util.gui.SimpleColorMap(0, 1, new Color(0,
0, 0, 0), new Color(255, 0, 0, 150)));
}
```

The sites interface that Home inherits from:

```
package sim.app.crime;

import sim.util.Int2D;

interface Site {

	public Int2D location();

	public String locationName();

	public void resources();

	public void whoIsInside();

	public void howManyInside();

	public void color();

	public void addPersonToSite(Criminal criminal, CrimeWorld crimeWorld);

	public void removePersonFromSite(Criminal criminal, CrimeWorld crimeWorld);

}
```

Crime

Implementing crime was difficult as we ran into several problems along the way. Crime was necessary to implement because the whole project revolved around inequality. The more unequal the society, the more crime was committed and factors such as the economic system, disease, stress and needs came into play. The more stressed a person became because of a worry of money the more likely they were to commit a crime.

The below code shows the agent scanning their neighborhood for the closest citizen to them and then picking that citizen as a target to rob.

```
private void chooseTarget(CrimeWorld crimeWorld)
{
	Bag nearbyCitizens = getNeighbors(myLocation.x, myLocation.y, myLocation,
crimeWorld, crimeWorld.yard, 5);
        if (nearbyCitizens != null)
                targetToRob = (Criminal) nearbyCitizens.get(0);
        else crimeWorld.yard.setObjectLocation(this, moveRandomly());
}
```

```
public Bag getNeighbors(int x, int y, Int2D myLocation, CrimeWorld crimeWorld, SparseGrid2D
yard, int j) {

        Bag objectsAtLocation = null;
        // scan area for neighbors
        int i = x - j;
        while (i <= x + j && targetCitizen == null) {
                int k = y - j;
                while (k <= y + j) {
                        objectsAtLocation = yard.getObjectsAtLocation(i, k);
                        if (objectsAtLocation != null && (i != x && k != y)) {
                                objectsAtLocation = yard.getObjectsAtLocation(i, k);
                                // return any neighbors that are found
                                return objectsAtLocation;
                        }
                        k = k + 1;
                }
                i = i + 1;
        }
        // this should only return nulls
        return objectsAtLocation;

}
```

Movement

Movement was another important implementation. Movement is either going to be random or there will be a purpose to it. In the case of a criminal, a criminal will move erratically or randomly around the sandbox and try to find a target citizen to rob.

Random movements for agents:

```
private Int2D moveRandomly() {
        int x = myLocation.x;
        int y = myLocation.y;

        Bag objectsAtLocation = crimeWorld.yard.getObjectsAtLocation(x, y);
        objectsAtLocation = getNeighbors(x, y, myLocation, crimeWorld,
crimeWorld.yard, 5);

        if(objectsAtLocation!=null) {
                criminalTarget =
crimeWorld.yard.getObjectLocation(objectsAtLocation.getValue(0));
                targetCitizen = (Criminal)
crimeWorld.yard.getObjectsAtLocation(criminalTarget).getValue(0);
                        targetCitizen.setRobbed(true);
                this.setRobbing(true);
                robbingTarget = targetCitizen.isBeingRobbed;
        }

        // make the target citizen stop moving and move the criminal towards the
target citizen
        if(targetCitizen != null && (targetCitizen.getLocation().x != x &&
targetCitizen.getLocation().y != y)) {
                //System.out.println(criminalTarget);
                return moveCriminalFromMyLocationToTargetLocation();
        }
        if(r.nextBoolean())
                x++;
        else x--;
        if(r.nextBoolean())
                y++;
        else y--;
        return new Int2D(x, y);

}
```

Disease

Disease was another important factor as it added in some chaos to the simulation. An agent had a set chance of getting sick every day, stress also affected the chances of an agent getting sick (research says that stressed people are more likely to get sick)

Once an agent goes above a certain stress value, they have an increased chance of getting sick. Their color will also change from red/blue to green.

```
private void contractingDisease(boolean isNewDay)
{
        if(this.stress > 70 && isNewDay)
       {
            boolean randomBoolean = Math.random() < 0.1;
            this.isSick = randomBoolean;
        }
        else if(this.stress <= 70 && isNewDay)
        {
            // theres a 0.05% chance of a person getting sick
            boolean randomBoolean = Math.random() < 0.05;
            this.isSick = randomBoolean;
        }
}
```

The following code shows how the agent is made green.

```
protected Color sickGreenColor = new Color(255,255,0);
public void draw(Object object, Graphics2D graphics, DrawInfo2D info)
{
       if(citizen.isSick)
       {
               paint = new Color(80, 220, 100);
       }
}
```

Time tracker

Time was another important feature that needed to be implemented. Users needed to see what time it was during the day to see how the agents reacted to time.

```
Steps steps = new Steps(crimeWorld);
Font font = new Font("SansSerif", Font.BOLD, 20);
public void draw(Object object, Graphics2D graphics, DrawInfo2D info)
                   {
      Rectangle2D bounds = new TextLayout("" + 11, font,
graphics.getFontRenderContext()).getBounds();

       graphics.drawString("" + steps.getTime() + ":00", (int)((info.clip.width -
bounds.getWidth()) / 2), (int)((10)));
}
```

Inspectors

Having the ability to click on an agent and inspecting different properties that they have such as their stress levels, needs meter and citizen name is an important implementation because it gives the user an idea of how unique each agent is.

The below code shows how inspectors were added to the Agents:

```
public Object getSimulationInspectedObject() {
        return state;
}
```

```
public Inspector getInspector()
{
        Inspector i = super.getInspector();
        i.setVolatile(true);
        return i;
}
```

Economic system

We thought that implementing an economic system was a good idea. The economy has been unpredictable lately and in fact scary. And in times of serious doubt and economic stress for many people we thought it would be interesting to add in a fluctuating economy. With this implemented, we could look into the gini coefficient that gave us a good insight into wealth gaps. It was also important to have a changing economy to simulate our current economy, where every so often things go bad and people start worrying about money a lot more.

The below code shows that the economy randomly changes every so often and affects the amount of jobs available to agents.

```
int jobsForHire = 20;
// new checks are done every day to change the economy. If jobsForHire is below 0,
// meaning the economy isn't doing so well, people have a chance of becoming
// unemployed and will also receive demotions (higher paid jobs become more scarce)
public void newDay()
{
        currentDay++;
        billDay(1);

        if (random.nextInt(5) == 0) {
                changeEconomy();
        }
        // Here is where the stress checks are done
        stressChange();

        if(jobsForHire > 0) {
                givePromotion();
                jobsForHire--;
        }

        if(jobsForHire < 0) {
                giveDemotion();
                jobsForHire++;
        }
}
// available promotions and jobs will vary depending on the current economy.
// the better the economy, the more promotions there are meaning more income
// for the agent population
private void changeEconomy()
{
        if(jobsForHire < 0) {
                if(!(random.nextInt(3) == 0))
                        jobsForHire = jobsForHire + 5;
                else
                        jobsForHire = jobsForHire - 5;
        }
        else {
                if(!(random.nextInt(3) == 0))
                        jobsForHire = jobsForHire - 5;
                Else

                        jobsForHire = jobsForHire + 5;
        }
}
```

Graphs and statistics

Graphs and Statistics are necessary for such simulations to allow the user to gather data about what's going on. WIth the following code, a user is able to see the amount of homeless people in the sandbox.

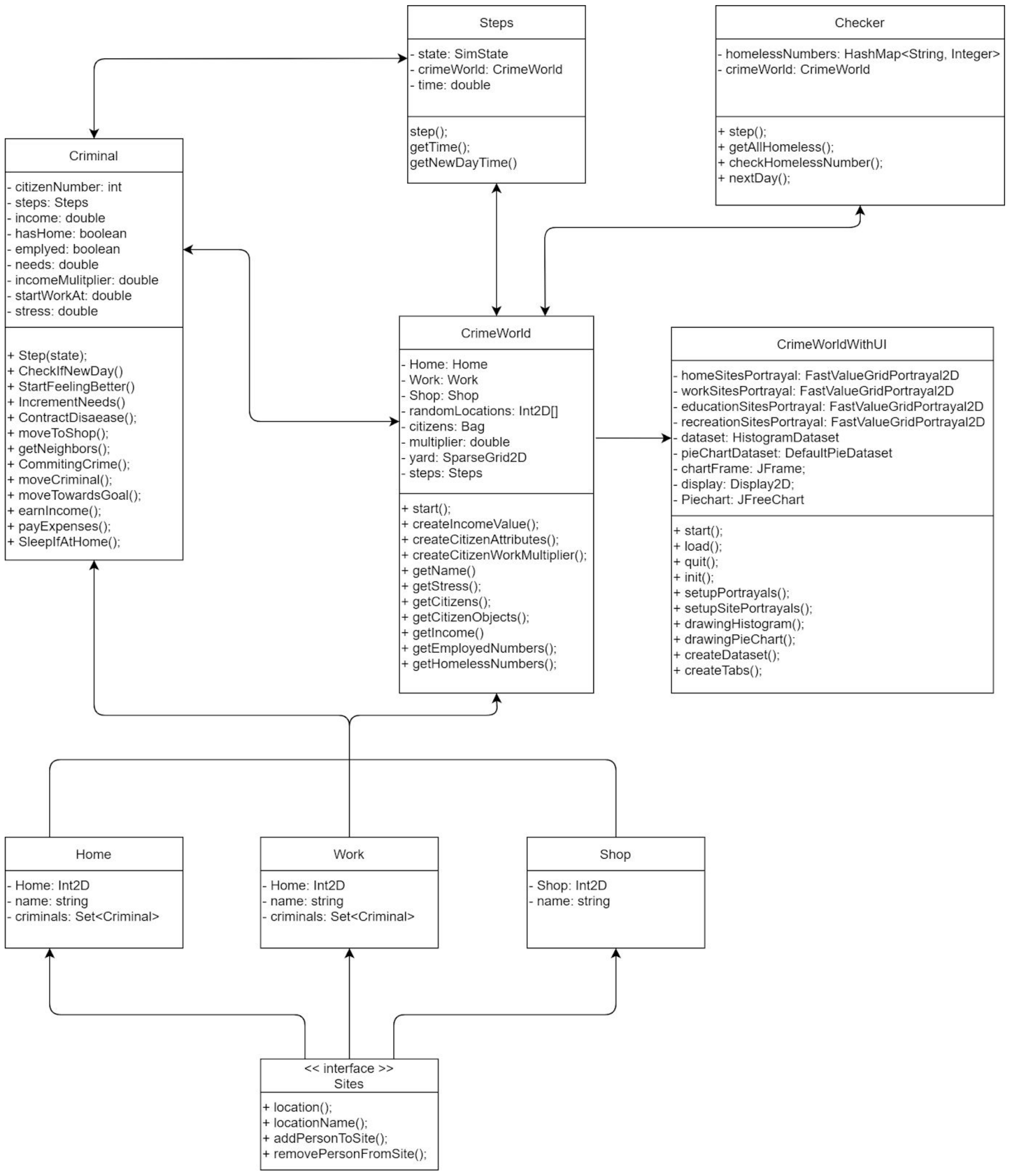
Here the homeless people are added into a hashmap and the hashmap is then used to calculate the amount of homeless people and to draw the graph.

```
private HashMap<String, Integer> createPieChartDataset()
{
        HashMap<String, Integer> homelessNumbers = new HashMap<String, Integer>();
```

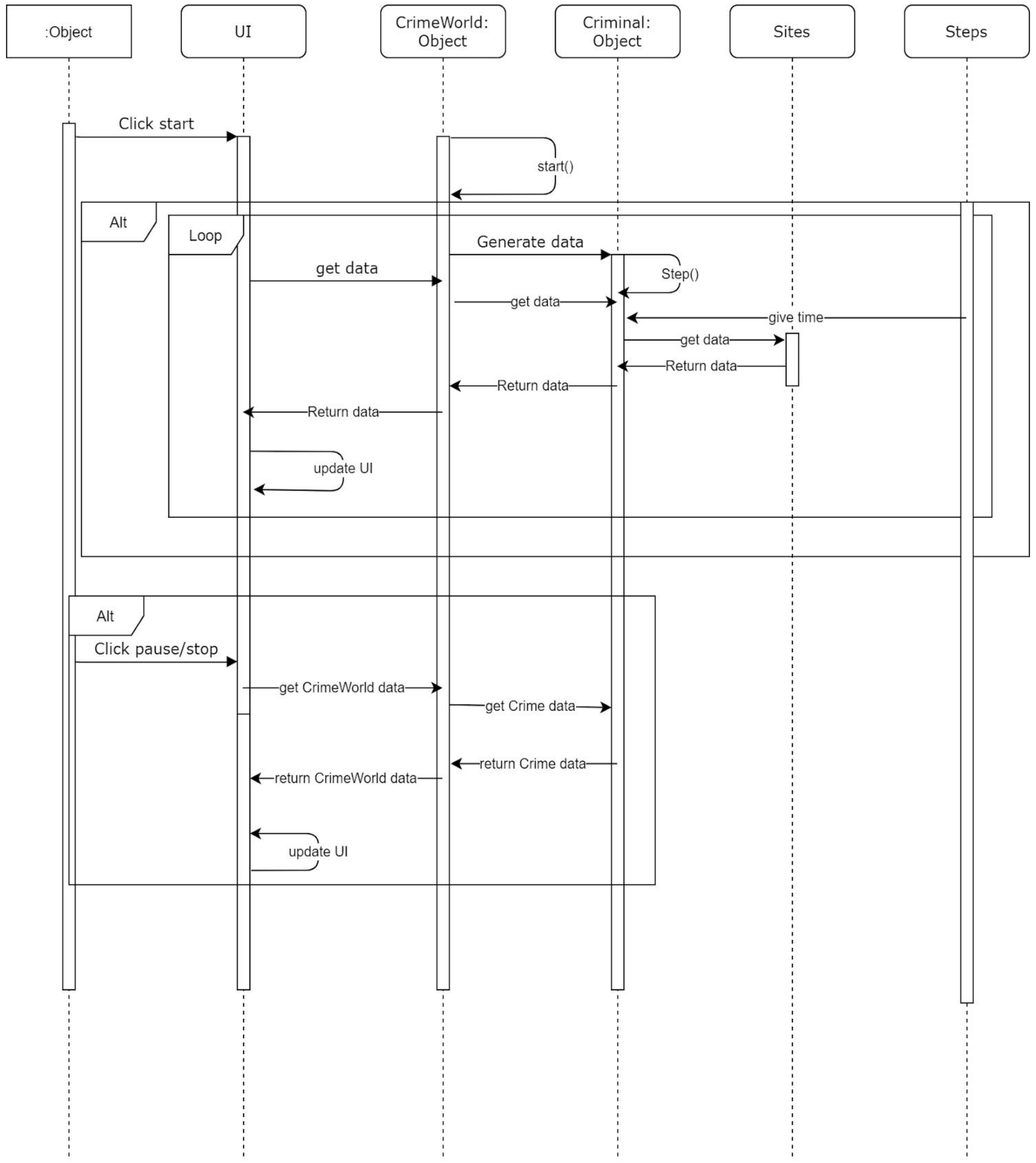
```
      for (int i = 0; i < CrimeWorld.getHomelessNumbers().length; i++) {
              System.out.println(CrimeWorld.getHomelessNumbers()[i]);
              if (CrimeWorld.getHomelessNumbers()[i] == false) {
                     if (!homelessNumbers.containsKey("is Homeless")) {
                            homelessNumbers.put("is Homeless", 1);
                     } else {
                            homelessNumbers.put("is Homeless",
homelessNumbers.get("is Homeless") + 1);
                            }
              } else {
                     if (!homelessNumbers.containsKey("is not Homeless")) {
                            homelessNumbers.put("is not Homeless", 1);
                     } else {
                            homelessNumbers.put("is not Homeless",
homelessNumbers.get("is not Homeless") + 1);
                     }
              }
       }

       pieChartDataset.setValue("Have a Home", homelessNumbers.get("is not
Homeless"));
       pieChartDataset.setValue("Homeless", homelessNumbers.get("is Homeless"));
       return homelessNumbers;

}
```

4.0 High level design

4.1 Class Diagram



4.2 Sequence diagram



5.0 Problems and resolutions

- **5.1 Mason outdated**

  Initially Mason was released in 2003. It was in development for 13 years. We found that much of the documentation and the API was difficult to understand and unclear, especially regarding the UI and its interaction with the Swing java libraries. Though there were resources provided the explanation for much of it was unclear. We found that a lot of time was spent searching through the manual to gain insights into how some basic functionality operated. There was also a lack of external resources as it is not a widely used library.

  There was a lengthy manual provided that after reading we found to provide understanding into the functionality of the program.

- **5.2 Graphs**

  How to develop dynamic graphs was one of the weaknesses of the documentation. Mason had some inbuilt graphing displays but for our purposes we needed to develop our own system to display the data dynamically. After some time we found that the layout of the class system and how the information was being fed to the SimState would make it very difficult. We eventually found a way to update the graphs based on a button press and relied on some of the inbuilt functionality to show the data points for the individual agents.

- **5.3 Lack of Documentation**

  In places the documentation tended to be quite bloated and vague. Due to the long development of the MASON library there were sections of the documentation that had not been updated to reflect the changes that were made between releases. This often caused confusion when relying on documentation to figure out issues. A few of the explanations in the documentation relied on methods that had been depreciated or updated in previous releases. As many ABMs are often made inhouse for large companies or research groups, there was not much external resources for overcoming these issues, and at parts required us to overwrite many methods to achieve the functionality explained in the manual.

- **5.4 Finding neighbors of Agent**

The getNeighbours function was one of the depreciated functions that we spoke about. Getting a collection of all the agents at differing distances was something that was important for the crime activity to work, as agents needed to search the nearby area for targets instead of picking ones at random, and the replacing function was not applicable. This required us to develop our own getNeighbours function and add the ability to search within a given space.

- **5.5 Vectors vs grid**

  Initially we began developing the simulation on a Continuous2D plane. This meaning that the agents would exist on a 2D plane in which and would move based on directional vectors. This caused many problems and was slowing down the simulation in an alarming way. So to optimise the speed at which the simulation was running at, we decided to use a SparseGrid2D. This means that agents can now only move up down left and right making things a lot more easier. It also made movement a lot more manageable and easier to understand.

- **5.6 Images in the UI**

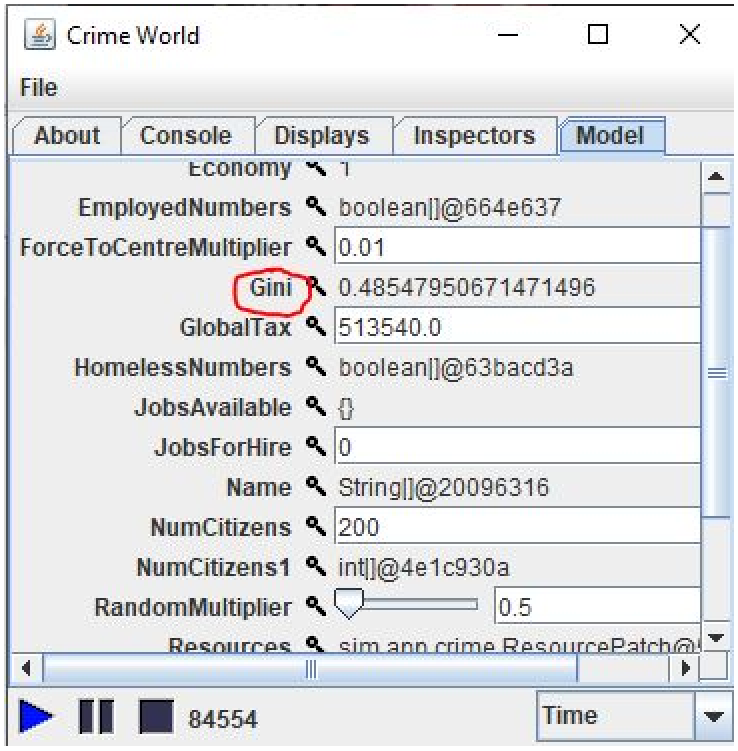
  Due to a huge lack of documentation on how to use some of Masons features, there was a big problem when trying to render in images (png, jpeg) for the sites. Originally we thought it would be a good idea to have an image of a factory for work, a house for Home and a shop building for the Shop site. In the end, we ran into a lot of problems and adding in images was just something that Mason did not allow. So to solve this problem we decided to create rectangles of objects of varying colors and it ended up working and displaying in a good way.

6.0 Results

It was interesting to see the results of the simulation. The gini coefficient starts off at 1.0 meaning that the wealth gap between the lowest earning and highest earning agent is the same. As the simulation progresses, a user will see the gini coefficient changing meaning that there's a bigger wealth gap between the lowest and highest earning agents.

It also affected crime rates. Depending on how well the economy is doing, the worse the economy the more crime happened and the better the economy the less crime there was. Agents' stress was also affected by the economy and played a vital role in crime. It was also seen that welfare had an important role in society. The poorer agents would receive social welfare payments meaning that they didn't have to worry about money when compared to simulations without social welfare.

If you let the simulation run long enough, you would end up with a low gini coefficient number that would gradually and slowly tend to 0.0 this meant that the wealth gap was getting bigger and bigger as time went on.


Crime World

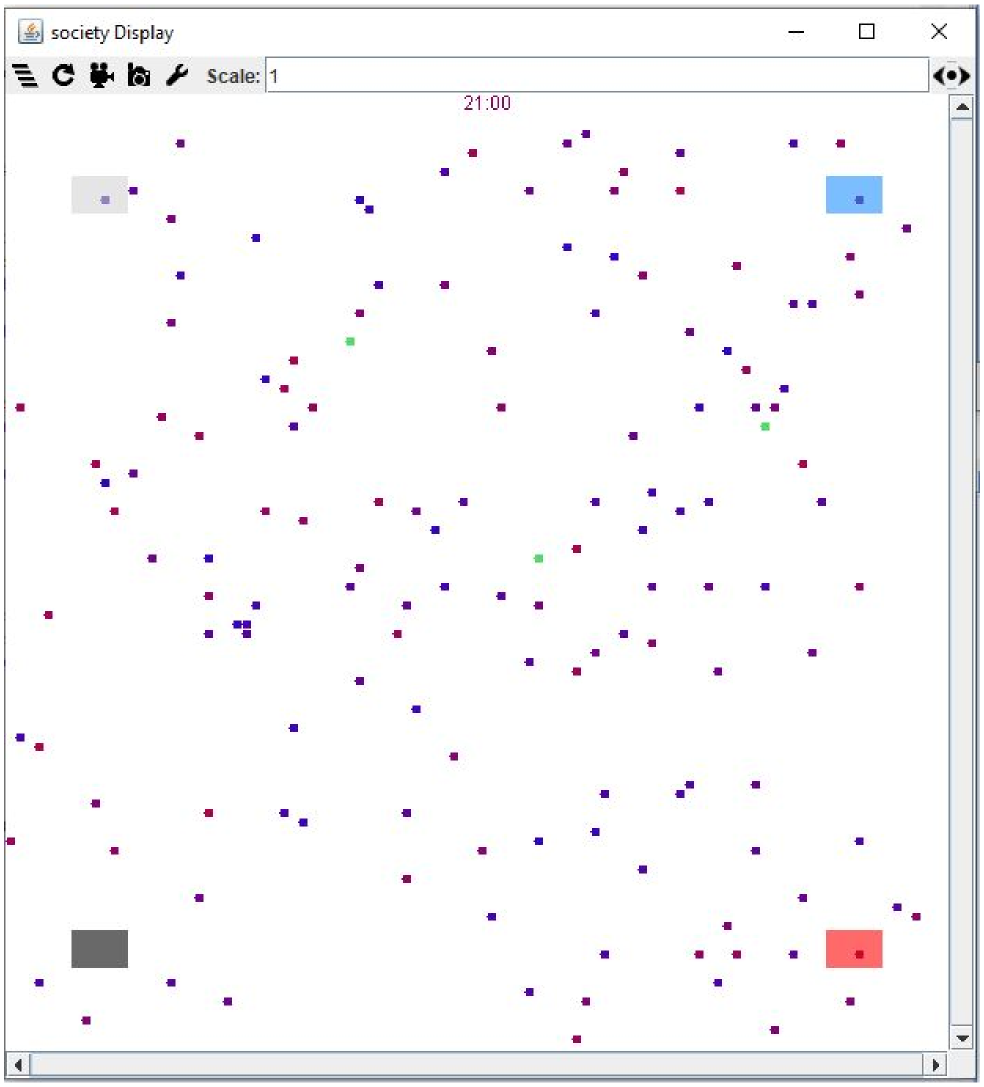
File

About | Console | Displays | Inspectors | Model

Economy	1
EmployedNumbers	boolean[]@664e637
ForceToCentreMultiplier	0.01
Gini	0.48547950671471496
GlobalTax	513540.0
HomelessNumbers	boolean[]@63bacd3a
JobsAvailable	{}
JobsForHire	0
Name	String[]@20096316
NumCitizens	200
NumCitizens1	int[]@4e1c930a
RandomMultiplier	0.5
Resources	sim.app.crime.ResourcePatch@

84554 Time

The resulting movement of each agent was also done in a good way. Movements such as the erratic movements of a criminal, specific movements and roaming was done in a good way. It really gave the simulation a dynamic feel to it.



Citations:

Felson, R., Osgood, D., Horney, J. and Wiernik, C., 2011. *Having A Bad Month: General Versus Specific Effects Of Stress On Crime*. [1]

Becker, G., n.d. *Crime And Punishment: An Economic Approach*. [S.l.]: [s.n.]. [2]

Weatherburn, D. and Lind, B., 1998. *Poverty, Parenting, Peers And Crime-Prone Neighbourhoods*. Canberra: Australian Institute of Criminology. [3]

Unodc.org. 2020. [online] Available at: <https://www.unodc.org/documents/data-and-analysis/statistics/crime/GIVAS_Final_Report.pdf> [4]

Zeitlin, L., 2020. *Organizational Downsizing And Stress-Related Illness*. [5]

Salleh, M., 2020. *Life Event, Stress And Illness*. [online] PubMed Central (PMC). Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3341916/> [6]

Mallubhotla, D., 2020. *An Analysis Of The Relationship Between Employment And Crime*. [online] Core.ac.uk. Available at: <https://core.ac.uk/download/pdf/59227413.pdf> [7]

Logue, J., 2020. *The Welfare State: Victim Of Its Success*. https://www.jstor.org/stable/20024635?seq=1. [8]

FOUARGE, D. and LAYTE, R., 2005. Welfare Regimes and Poverty Dynamics: The Duration and Recurrence of Poverty Spells in Europe. *Journal of Social Policy*, 34(3), pp.407-426. [9]