A Simulation of the Economic Impact of Disaster Events

Independent Study Thesis

Presented in Partial Fulfillment of the Requirements for
the Degree Bachelor of Arts in Computer Science in the
Department of Mathematical & Computational Sciences at
The College of Wooster

by
Richie Pajak
The College of Wooster
2022

Advised by:

Heather Guarnera (Computer Science)

THE COLLEGE OF
WOOSTER

Abstract

The goal of this project is to determine the impact of natural disasters on an economy. Specifically, this project analyzes the changes in prices of goods caused by disaster events that impact the supply of goods within the economy. To accomplish this goal, an agent-based model of a small economy was built in Python to simulate the effects of a natural disaster on normal economic activity. The simulation features a collection of businesses who produce and sell goods along with economic actors known as agents that are able to purchase goods and work in businesses to earn money. In order to respond to the economic forces of supply and demand present in the economy, the businesses are able to dynamically change the prices of the goods they sell. These changes are then tracked and analyzed to determine if the changes seen in the model reflect what would be expected according to established economic theory. Results from this project show that the model is accurate in predicting price change trends in goods that are present in an economy that is impacted by natural disasters.

Acknowledgments

I would like to acknowledge my advisor, Dr. Heather Guarnera, for guidance through this process and assistance in writing and editing both the paper and software components of this project. Thank you for the constant advice, support, and encouragement that was needed to push through this project. I would like to thank Hope Carmody for her endless love and support throughout college. I would also like to thank all of the friends I have made at The College of Wooster that have helped me through my college career, namely, the Wooster Ultimate Frisbee team which has been a source of friendship, support, and joy since my first day on campus. Finally, I would like to thank my mom, dad, and brother for always being supportive of everything I do and for motivating me to accomplish everything I have both inside and outside the classroom.

Contents

CHAPTER 1

Introduction

With the onset of the Covid-19 pandemic, vast economic effects were seen from policies created to slow the spread of the virus. Most notably, many businesses not deemed "essential" were shut down. Within the first year of the outbreak, there were around 200,000 businesses in the U.S. that were shut down [9]. This forced closure of businesses reduced people's access to many different types of goods. This type of restriction has caused price increases in many goods [6]. Additionally, in order to reduce the amount of economic damage caused by the virus, the U.S. government has committed to spending 4.2 U.S. dollars on relief efforts [10]. Pandemics and other types of natural disasters can have drastic effects on the economies that are impacted by these types of events.

This project looks at the economic effects of disaster events that slow production and limit consumer access to goods. This project features an agent-based model of a small economy simulated within Python. This economy includes a collection of businesses who produce goods and consumers who purchase goods. The simulated economy can be hit by disaster like events that decrease the production of goods and restricts consumer access to those goods. Due to these events, businesses can change the prices of their goods to respond to the changes in supply and demand caused by the disaster events. This project investigates agent-based modeling and

how that technique can be used to model relevant economic theories that would apply to disaster events such as a pandemic, famine, drought, or tornado.

Results from the simulation show that the model accurately represents price changes that occur due to the onset of disaster events according to economic theory. Economic literature states that when the supply of a good is decreased, the price of that good will go up [2, 7]. The simulation shows this change in prices when disaster events negatively impact the supply of goods within the market.

Chapter 2 investigates agent-based modeling. This chapter examines the basic concepts of agent-based modeling and identify its weaknesses, strengths, and why it was chosen for this project. Chapter 3 looks at the economic theory that explains the economic effects that could be observed in an economy that is struck by a natural disaster. This chapter looks at various economic forces that can push prices of goods up and down and explains the market structures that are seen in this simulation. Chapter 4 discusses Pygame, a Python library that can be used to make two dimensional video games. Pygame includes key features that allow this project to exist as a visual agent-based model of an economy. Chapter 5 goes into the details of the implementation of the project. It explains the rules of the economy that is set up in the simulation and exactly how the agents of this model behave and make economic decisions. Chapter 6 analyses the results that have been collected from the simulation. It looks at various research questions that can be answered by this simulation and explains how the economic forces at play in the simulation shaped consumer and business choices.

CHAPTER 2

Agent-Based Modeling

The base methodology for this project is known as agent-based modeling. Agent-based modeling is a modeling technique used to create a model of a system using a collection of agents. An agent is an entity that represents a certain part of the system being modeled. A single model can have many agents of the same type or agents of different types. What distinguishes one agent from another is its properties, states, and behaviors [12, 8, 14, 5]. Agents have properties that represent certain aspects of the agent with values that go along with them. An agent's state is its current situation. This could be its location or current objective. Agents also have certain behaviors or a set of directions that instruct them on how to interact with other agents within the system. This could be agents of the same type as them or agents of a different type. The behavior of an agent will generally depend heavily on that agent's properties and current states. Under certain conditions, agents will behave in certain ways. In this way, agents are able to react to their environment and other agents around them. With a set of well defined behaviors, agents can be expected to perform a variety of tasks based on their current state. Agent-based modeling is able to model systems where each individual agent is slightly different from the others. Although these agents might share a set of instructions based on their properties, their properties will vary. Thus, each agent will take a different course of

action. Agent-based modeling is a strong method for simulating each agent making its own decisions and taking its unique actions based on its current situation. In addition, the action of one agent will impact the state of another, and therefore impact its course of action. No agent is able to make a decision in a vacuum. The agents are constantly interacting and creating changes that each other agent must adapt to. Each individual's actions then culminate into large, overarching trends. These trends can describe patterns within the agents' behavior. Generally, agents also have a graphical component that allows users to see the model of the system visually while it is in progress. As a large collection of agents interact over a period of time, trends can be seen and complex systems that the model is representing can be understood at a foundational level.

At its core, agent-based modeling is a way to fundamentally understand complex systems by observing its low level interactions. Agent-based modeling is a "bottom-up" method of modeling. The lowest level of interactions and events are simulated which create overall trends across the system. These aggregate effects are products of each individual agent being uniquely created and its behaviors and actions simulated. Having each agent individually simulated creates a level of detail and accuracy that makes agent-based modeling a strong methodology for modeling a system and determining the outcome of that system. By contrast, a "top-down" method of modeling might simulate an overall trend that is understood and from there infer what the individual components of that system might be doing to create the observed trends. Bottom-up systems allow for great levels of detail within a model.

2.1 Agent-Based Modeling vs Equation-based Modeling

Aside from agent-based modeling, one of the main ways to model a system is by equation-based modeling. The following section describes advantages of agent-based modeling over equation-based modeling.

One of the main differences between these two methods is that agent-based modeling gives discrete results while equation-based modeling gives continuous results [14]. This means that an equation-based system can give results that do not necessarily correlate with the real world. For example in an economy, an equation could predict that prices of a good will increase by a fraction of a cent. In practice, this result is difficult to correlate to the real world as cents are not split into fractions in normal economic practice. Discrete results are more applicable to the real world because they do not have this type of issue. Any conclusions that are generated from discrete results will make sense in the realm of possibilities that the model is simulating. An agent-based model of economic interactions would not see prices change by fractions of cents.

When building equation-based models, it is also important to have an understanding of what the overall result of the model will be before the model is generated. It is necessary to have a good understanding of the aggregate behavior of the model and test the model against a hypothesis that is determined based on the knowledge that is already present [14]. For this, it might be necessary to have knowledge in differential equations or other high-level math concepts. Agent-based modeling requires only the bare knowledge of how agents interact with each other. The end product of the model will become apparent as the model runs. This effect comes from agent-based modeling being a bottom-up design technique. As long as the

lowest levels of interactions are understood, the bigger picture is created by the model to then be interpreted.

This interpretation of an agent-based model also does not require high levels of understanding in the relevant fields. For an equation-based model it is likely that a lot of background knowledge and understanding is necessary to interpret how a model works, the equations that it uses, and the results that are produced. This is not necessary when interpreting agent-based models. While not many people have a high level of economic knowledge, most do understand how basic economic interactions work. Thus, they would be able to understand an agent-based model of an economy because it is built on these foundational, low level interactions. The model then produces the aggregate result of the interactions that are easy for untrained users to understand. In addition, an agent-based model is much easier to translate into a graphical representation, which is one of the goals of this project. Each agent can be represented on screen graphically and give users an intuitive representation of how the model is progressing while it is running. Being able to watch each individual part of the system come together to create a larger picture is one of the hallmarks of agent-based modeling.

Finally, in an agent-based model, each individual agent is simulated uniquely from each other agent. Because of this, it is possible to observe one individual agent and see its course of actions, decisions, and behaviors within the context of the model as it progresses. This creates a level of detail that is unattainable with equation-based modeling. Equation-based models show aggregate change and results to a system, not individual decisions and impacts.

Overall, agent-based modeling is the better choice for economic simulation because of the inherent ease of interpretation, visualization, and construction that is possible.

2.2 Agent-Based Modeling in the Context of Economic Simulation

Agent-based modeling is an incredible technique for answering the questions posed by this project. An economy is a complex and highly interconnected system of consumers and producers. These producers and consumers are examples of economic actors. An economic actor is any entity that is capable of making economic decisions. The most common economic actors in our society are people, but businesses, governments, and other organizations can be economic actors as well. Economic actors can be modeled using agent-based modeling and the outcomes of their decisions can be shown using this type of visual model.

This project simulates an economy using agent-based modeling to show the economic interactions between various people and businesses. With each agent having its own goals and behaviors being modeled constantly, a functioning economy is built from the bottom up. This agent-based model shows dynamic changes within the system over a period of time. For this project, as the economy changes over time and it reacts to economic impact events, the model demonstrates the effects of those events in real time. The results of these analyses make it easy for users to see and understand the causes and effects of the changes that occur within the economy due to these impactful events.

CHAPTER 3

Economic Theory

When discussing economic impacts, effects, and interactions caused by disasters and other events, there are several key concepts that need to be understood. These are basic building blocks that the field of economics is built from, and include the notions of supply and demand, the assumptions of rational decision making and perfect competition, compliments and substitutes, and the economic impact of natural disasters. All of these concepts are necessary to analyze an economy and the impacts that a natural disaster event could have on the economy.

3.1 Supply and Demand

Supply and demand are the main forces that determine prices. These two factors interact to create a system where buyers are looking to purchase a certain amount of goods for a particular price. The sellers within that system produce an amount of goods and sell them for the market price. The supply or demand of a good can change due to a wide variety of different factors. When these changes occur, the prices of goods can change. Supply and demand are some of the most fundamental tools used to analyze an economy and predict the changes it will see based on events that occur or actions taken by economic actors [2].

Demand is a measure of what quantity of a certain product people are willing to buy at a given price. There are many factors that influence this. The main factor that impacts the demand of a product is its price. At a low price, people are willing to purchase a large quantity of a product. At a high price, they will not want to buy as many. While price is the largest factor that determines demand, it is not the only one. Other factors include tastes and preferences, the prices of related goods, buyer income, population demographics, and buyer expectations. Tastes and preferences are simply buyers' attitudes and feelings toward certain products. This is the factor that advertisements look to change. Seeing an advertisement for a product might make you more inclined to purchase it. If a large number of people are affected in this way, the demand of that product will increase. Certain products have higher demand simply because people prefer them the most. Prices of related goods such as complements and substitutes to a good can change the demand of a good, as detailed below. Buyer income and population demographics directly impact the demand of goods in a market. As there are more people and more money for those people to spend, they demand higher quantities of the goods available to them. An additional impact on demand is buyer expectations. Buyer expectations are how a buyer expects a price to change in the future. If a buyer believes that the price will rise in the near future, they will demand more of a product now to take advantage of the lower price. If they expect the price to fall in the near future, they will demand less of a product now.

The demand for a product is the culmination of all of these factors. Each one affects the overall demand of a good. As these factors change, the demand for that product will as well. Shifts in demand are very common in economies and can heavily impact how people choose to spend their money.

Supply is equally important in understanding how economies function and change. Supply is the quantity of products that sellers are willing to make available

for purchase at a given price. Like demand, price is the most significant factor that determines supply. The amount of revenue that a seller can expect to make from a product will heavily influence what quantity that they will produce. Other factors include cost of production, technology changes, seller expectations, number of sellers, and natural events. Aside from price, production costs are one of the largest influences on supply as a high cost to produce a good will lead to fewer being supplied by the sellers. The main factors that influence production costs are labor costs and raw material costs. If either of these increase, it is likely that the supply of the good being produced will decrease. Technology changes influence supply by making it easier to produce larger quantities of product more easily. Technology can make the production of a good faster or less expensive. Either way, technological developments will increase the supply of goods by making production more efficient. Seller expectations influence the market similarly to buyer expectations. If a seller expects that the price of a good will rise in the future, they will sell less now to have more to sell at the higher price later. This leads to a decrease in supply. Conversely, if sellers believe that the price of a good will fall in the future, they will sell as many as they can before the price change occurs to take advantage of the higher price. This will lead to an increase in supply. The number of sellers in a market also influences the supply of a product. The more sellers there are, the higher the supply of that product. Finally, natural events will lead to changes in the supply of a good. Natural disasters such as droughts, hurricanes, or tornadoes all have the potential to damage equipment or products making it difficult or even impossible for businesses to produce goods. This will greatly decrease the supply of that good. Like demand, a change in any one of these factors will influence the overall supply.

Both supply and demand work together to create a market price for a good. Neither supply nor demand act independently. Both of these effects combined

create the market structure that we see. A change in supply has very little effect on the market without the demand of that product taking effect as well. If the supply of a product decreases, a seller will want to sell a lower quantity of products for the same price, but the consistent demand for the product will cause the seller to sell less for a higher price. The quantity demanded at the higher price will be lower than the quantity demanded at the original price. In the end, the decrease in supply without a change in demand causes a price increase. When analyzing an economy, neither supply nor demand can be considered individually. They both contribute to the overall market function.

3.2 Assumptions of Economic Agent Rational Decision Making and Perfect Competition

As economic agents, people tend to make the decisions that benefit themselves the most. This includes spending the least possible on goods that they want to purchase, or taking a job that will pay them the most. Economists, when studying trends and changes in an economy, assume that people make rational choices. It is assumed that people make the best decision for themselves based on the information that they are given. When analysing any economic market, it is important to know what kind of market is being analysed. For this project, the market structure is a conjunction of several perfect competition markets.

A perfectly competitive market is commonly defined by several assumptions [2]. The first of these is that firms produce identical goods. All of the goods being sold in a single market are the same and there is no inherent advantage to buying from one firm over another. Second of these assumptions is that there is a large number of buyers and sellers. While "large" is a relative term, this essentially means that no one firm has influence over the market as a whole. A single firm's decisions

will not impact the larger market. The third of these assumptions is the ease of entry and exit from the market. Firms are able to join and leave the market at will; there are no significant legal or capital barriers that would prevent a business from joining the market. The final assumption is perfect information. It is assumed that in a perfectly competitive market that all economic actors within that market, both consumers and producers, have perfect information over the products being sold and their prices. This allows consumers to make educated choices on where to purchase goods, and prevents any firm from gaining an advantage that is based in knowledge.

With all of these assumptions in place it forces the firms within the market to be price takers. Within the market, the firms must sell their products at the market price. If they set their price higher than the market price, no one would purchase their goods because they could go to a different firm and get an identical product for a lower price. If they set their price lower than the market price, they will not be making as much profit as they could if they sold at market price.

Perfect competition is one of the most basic market structures that is seen within economics. Because this market structure has well defined rules and the outcomes for changes within this market are predictable, it is easy to use perfect competition as a basis for studying economic trends. For this reason, perfect competition is the market structure that is used in this study to study the economic impacts of natural disasters.

3.3 Complements and Substitutes

It is possible for goods to be related. Goods can be either substitutes for each other or complements to each other. For substitutes, consumers will generally buy one or the other. They may have a preference but largely, either would work for what

they need. A common example for substitutes is coffee and tea. While people may have a preference, they both serve as a hot drink for people to have in the morning. Many people would be satisfied with one or the other, however, they are unlikely to purchase both. For this reason, if the price of coffee dropped significantly, people would more commonly buy coffee. Thus, people would not want to buy as much tea. This is the relationship that substitutes have. When the price of one good drops, the demand for its substitutes drops as well. The inverse of this relationship is also true. When the price of one good increases, the demand for its substitute will increase.

Conversely, complements are goods that go well together. People often buy these goods together. A good example for complements would be peanut butter and jelly. One of the most common uses for these goods is to make a peanut butter and jelly sandwich. Although these goods can be used on their own, it is less common than using them together. If the price of peanut butter were to increase, people would purchase less peanut butter. Then, because they have less peanut butter, they will be making fewer peanut butter and jelly sandwiches. Therefore, they will need less jelly causing a decrease in the demand for jelly. The relationship for complements is that when the price of one complement increases, the demand for the other decreases. The inverse is true as well. When the price of one complement decreases, the demand for the other will increase.

3.4 Economic Impacts of Natural Disasters

When a natural disaster strikes an area, there can be a huge, widespread impact. There are both physical and mental effects that people of that area face. Some of the impacts caused are property damage, infrastructure damage, injury, mental health damage and more. These effects can greatly change people's economic choices out

of necessity, fear, and anticipation. These effects can be placed into two categories, direct and indirect effects.

Direct effects are any immediate damage caused by the disaster. This could be buildings collapsing from an earthquake, flooding in basements from a hurricane, or the loss of crops from a drought. Direct effects are measured by the costs of the damages that a natural disaster causes to an area, the cost to repair buildings, dry out basements, and replace crops that are lost.

Indirect effects are the economic changes that occur due to these direct impacts. If a hurricane strikes an industrial area of a city, property damage from winds and rain can be seen as direct effects. However, indirect effects will also be seen. Water damage, loss of power, or loss of workers can make it impossible to continue to produce goods at the same rate as before, if any are produced at all. The cost of the lost production is measured as the indirect effect of the hurricane. This decrease in production will be seen as a decrease in supply of those goods. However, if the demand for these goods stays the same or increases due to the conditions of the event, the price of the goods will increase. All of these changes to the market structure are indirect effects of the natural disaster.

One example of this indirect effect on the economy is gas prices in New Orleans after it was struck by Hurricane Katrina. In the weeks before the hurricane, the price of gasoline in the gulf coast region was around $2.50 per gallon. Due to the storm, the price rose to nearly $3 within a week [13, 4]. This rapid change was caused by power outages that prevented oil refineries in the area from producing gasoline. This increased price of gas is seen as an indirect cost of the hurricane. Another example of this price increase happening is in the iron ore industry. Data from the Emergency Events Database [11] shows that on average, after a disaster event occurs, iron ore prices increase 7.1 percent within six months of the disaster.

These increases can occur with a wide variety of goods and services depending

on the nature of the disaster and the location it impacts. Further effects of this supply loss will come from related goods. If the production of a substitute good is not impacted by the disaster, the demand for that product will greatly increase because while before people were buying a mix of the two substitutes, the good that was not impacted by the disaster is much more available and therefore more appealing for consumers. Similarly, the complements for goods impacted by disaster events will see a decrease in demand. Because people are buying less of one complement, they will buy less of the other and cause that decrease in demand. These impacts have been found in studies that look at price changes that are caused by disaster events.

There are two main causes found in case studies for price increases after disaster events. The first one is what has been detailed here. The economic forces of supply and demand force the price upward. The second is caused by malintentions. After disaster events, suppliers of goods intentionally set their prices much higher than the equilibrium price to take advantage of people in distress and willing to pay much higher than usual for goods. From data, it is difficult to tell the difference between these two effects [3]. In any given situation, it is likely that both of these effects play a partial role in causing the price to increase. It is difficult to tell what the portion of the price change is due to which factor.

All of the economic effects detailed here are seen in this project. Perfect competition is the market structure that is used to demonstrate the forces of supply and demand pushing the prices of products up and down. The complement and substitute effects on the demands of products are seen when disaster causes the supply of a product to decrease suddenly. These various economic tools of analysis are necessary in understanding the outcomes of the simulation that is at hand.

CHAPTER 4

Pygame

This project is entirely written in Python. The main Python library that is used for this project is Pygame [1]. Pygame provides functionality that allows users to create simple, two dimensional video games within python. Although this project is not a game in the traditional sense, there are a couple of key features provided by Pygame that are important for this project. These features include creating objects on screen, moving the objects around the screen, and detecting collisions between those objects. These are some of the most basic features of Pygame, and Pygame makes it very easy to accomplish these tasks.

4.1 Sprites

Within Pygame, *Sprite* is the basic class type used when creating an object that will be displayed on screen. Any object that is created that will be shown on screen with Pygame will extend the *Sprite* class. A class that makes up one of the most important attrivbutes of a *Sprite* is the *Rect* class. All *Sprites* in Pygame are rectangles of various sizes, and the *Rect* class represents this part of the *Sprite's* form. Any functions that would involve a *Sprite's* size, position, or movement will be called on the *Sprite's Rect*. For example, a *Sprite's* location could be set to a specific coordinate

using the following code, where `x` and `y` are integers that represent coordinates on the window.

```
Sprite.Rect.center = (x,y)
```

This would place the center of that *Sprite's Rect* at the given coordinate. Similarly, a *Sprite* can be moved by calling the `move` function on the *Sprite's Rect*.

```
Sprite.Rect.move(x, y)
```

In this case, `x` and `y` represent the distance in the x and y directions, left and right or up and down, for the *Sprite* to be moved. These values can be any integer, positive or negative. Some other useful functions included in the *Rect* class are `Rect.inflate` and `Rect.union`. The `inflate` function grows or shrinks the *Rect* based on a value. The `union` function uses one *Rect* as the calling object and takes another as a parameter. The function then makes a single large *Rect* from the two smaller ones.

Another important attribute for *Sprites* is the *Surface*. A *Surface* in Pygame represents an image or the texture of a *Sprite*. In its simpilest form, the texture for a *Sprite* can be a solid color. This is done using the `fill` function. One of the parameters of this function is the `color` parameter. This parameter is an `RGB` sequence that represents the desired color. For example, the following code will fill the Sprite's Surface entirely black.

```
Sprite.Surface.fill((0, 0, 0))
```

Note the two sets of parentheses around the `RGB` sequence. This is because the entire sequence is passed as a single parameter, rather than each number in the sequence being an individual parameter.

A function of the *Sprite* class itself is the `update` function. By default, this function does nothing, but it is intended that users of this function override it to include any functions or code that that a *Sprite* would execute every game tick. Then, when the game runs, `Sprite.update()` is called every game tick and the *Sprite* executes

its function accordingly. It becomes tedious, however, to call this function on each individual Sprite. This is one of the problems solved by Pygame's Group object. A Group is a data structure designed to hold *Sprites*. It is unordered and can be added to or removed from easily. The Group object has an `update` function. When this is called, it simply calls the `update` function of each *Sprite* within the Group. Another benefit of a Group is having a structure that allows a user to call any function on a large number of *Sprites* fairly easily. For example, suppose there is a Group called `businesses`. All of the *Sprites* in this Group represent a business that produces goods using the `produce` function. If all of the businesses needed to produce at the same time, this could be easily achieved with the `businesses` Group.

```
for x in businesses:
    x.produce()
```

This is the type of logic that is avoided using a Group's `update` function, however this cannot be done with other functions that a class might have.

One of the main uses of the *Sprite* class is collision detection. The ability to determine if two *Sprites* are touching each other is incredibly important for games and a project like this one. Pygame provides several different methods of detecting collision in both the *Sprite* class and the *Rect* class. Some of these functions simply check the collision between two specific *Sprites*. For example, `Sprite.collide_rect` function takes two *Sprites* as parameters and returns true if they are colliding and false if they are not. Another option is the `Sprite.spritecollide` function. This function takes a *Sprite* and a Group as parameters and returns a list of any *Sprites* in that Group that collide with the given *Sprite*. These functions requre that the *Sprites* have a defined *Rect* and use the collision functions provided in the *Rect* class. One such function is the *Rect's* `colliderect` function. Like Sprite's `collide_rect`, this

function tests for collision between two *Rects*. For this function, the first *Rect* being tested for collision is the calling object and the second *Rect* being tested is passed in as a parameter. As before, the function returns true if the *Rects* are colliding and false otherwise. There are similar functions `collidelist` and `collidedict` which test collision between the calling *Rect* and a list or dictionary of *Rects*. All of these functions provide very simple and straight forward collision detection between *Sprites*, which is one of the main reason Pygame was chosen for this project.

4.2 Visualization

A benefit of using Pygame for this project is the easy visualization that is inherent in Pygame. Pygame visuals are created using the `display` module. This module has many functions that allow users to create a window to act as the environment for their game. The other module that is used to create objects visually is the `draw` module. This module handles creating the objects on the display correctly.

This project draws objects onto the screen using these two modules in conjunction with each other. When a user calls the `blit` function on a Surface object, they pass in an objects Surface and its Rect. This will use the `draw` module to draw the passed in object onto the calling object's Surface. When the `display` for this project is initialized, the `display.setmode` function is called. This returns a Surface object to be used as the screen for the simulation. This Surface is used as the calling object any time the `blit` function is called. This draws the passed object's Surface and Rect onto the calling Surface, creating the effect of an object, or in this case a Sprite, being drawn onto the screen for users to see. Using these two modules, Pygame's built in functionality handles the visualization of objects being shown on a screen.

A simple example of drawing multiple Sprites, which all have `surf` and `rect`

variables representing their Surface and Rect objects, in the Group `all_sprites` onto the screen is shown here.

```
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
all_sprites = pygame.sprite.Group()

while True:
    for x in all_sprites:
        screen.blit(x.surf, x.rect)
    pygame.display.flip()
```

The while loop causes the screen and all objects in the `all_sprites` Group to be updated and redrawn each time the loop repeats. This creates a continuously moving image of objects on the screen.

4.3 Main Loop

This leads to the understanding of what is known as the main loop of the program. This loop runs continuously as the simulation runs to handle any processes that need to happen each "tick" or each time the loop repeats itself. These processes include updating the various sprites used in the program, redrawing the sprites, and updating the screen. There are many other processes that can occur within this loop depending on the needs of the specific project.

One other aspect that the main loop includes is a way to end the loop. Pygame includes functionality for reading in commands from the keyboard. Using this, each pass of the main loop detects if the escape key is pressed. If it is, the main loop is

exited and the program ends. The code used to check if the escape key is pressed and to therefore exit the main loop is shown here.

```
clock = pygame.time.Clock() # keep track of time


running = True
while running:
    for event in pygame.event.get():
        if event.type == KEYDOWN:
            if event.key == K_ESCAPE:
                running = False
        elif event.type == QUIT:
            running = False
    clock.tick(60) # Use 60 frames per second
```

This check uses Pygame's event system. Events of specific types can be called at any time within the program and handled using the for loop shown above.

Events are differentiated by their `type`. The `type` of the event will dictate how it is handled. In this case, the `KEYDOWN` and `QUIT` events are handled by setting the boolean variable `running` to false whcih causes the main loop to exit, so long as the key pressed to trigger the `KEYDOWN` event is the escape key denoted by `K_ESCAPE`. Because of this system, at any point in the program, a `QUIT` event could be called which would cause the program to end without the need for the escape key to be pressed. If a Pygame user wanted to incude more events in thier program and handle them, they would simply need to add the check for them as an if statement within the for loop. Pygame also supports custom events for users to create themselves.

Another aspect of the simulation that is controlled within the main loop is the animation's speed. This is done using Pygame's `Clock` object. This object belongs to the `time` module. The `Clock` object is an object that keeps track of time. With each game tick, or each pass of the main loop, the `Clock.tick` function is called. This function takes an optional parameter which represents the maximum framerate, or ticks per second, the program can run. It is the maximum number of times per second the screen can be updated. If this parameter is not given, there will be no maximum framerate. An example of this function being called in the main loop is shown here.

In the above example, the program would be limited to 60 frames per second. This is useful for adapting my simulation for its current need. When debugging or understanding how the parts of the simulation are behaving, it is useful to have the simulation run slowly and get a detailed look at what is occuring. However, when the simulation needs to run quickly and get through many rounds in a short time span, the framerate is left uncapped to save time.

CHAPTER 5

Implementation Details

This simulation uses Pygame to create a visual simulation of an economy and the people and businesses that make it up. It simulates the production, purchase and consumption of goods. It also simulates employment. Pygame's many features are used to visualize this simulation in an understandable way. The simulation is split into rounds or days. Each round the same steps are taken. Agents that are determined to be workers go to work for that day and produce products for the business where they are employed. Next, any agents that were not employed by the businesses become consumers for that day. They then select what product they would like to buy for the day and travel to the business that sells that product at the lowest price. Should that business run out of products, the agent will move to the business with the next lowest price until they are able to reach a business with products available and buy them if they have enough money. The worker at that business for the day receives all revenue the business earned for that day. After purchasing products, all consumers return to their homes. Then agents who were workers return to their homes and the process starts again. Depending on the number of products that go unsold in a business, that business will either increase or decrease the price of their business. If they sold all of their products, the price will increase to make additional profit. If not all products were sold, the

price will decrease to attract more customers. Through this process of increasing and decreasing price due to the supply and demand of the products available, the market for each good reaches equilibrium when the price is at a level where changes are no longer necessary.

5.1 Agents

The economic agents of the simulation represent people. These agents have the sole goal of earning money to purchase goods. These agents have complex decision making processes that make them rational actors within the economy.

5.1.1 Class Variables

The *Agent* class extends the Pygame *Sprite* object. Therefore, they have *Surface* and *Rect* objects as instance variables. Other instance variables that the *Agent* class contains are the variables `size`, `money`, `home`, `destination`, `at_home`, `consumer`, and `business_options`. These variables have uses that are described here.

- `size`: Each agent is represented by a square. This variable is an integer that represents the side lenght of that square.
- `money`: This variable is an integer that represents the amount of money this agent currently has.
- `home`: This variable is a *Home* object that represents the home owned by this agent.
- `destination`: This variable is a *Rect* object that represents where this agent is currently moving toward.

- `at_home`: This variable is a boolean that represents if this agent is currently at its home.
- `consumer`: This variable is a boolean that represents if this agent is currently a consumer or worker.
- `business_options`: This is an list that represents the different business options an agent has to buy their desired product.

Finding a home for an agent is done using the `Group` system in Pygame. Whenever a new *Home* object is created, it is added to the `homes` group. Then when a new *Agent* is created, it calls the `find_empty_home` function. This function iterates through all the *Home* objects in the `homes` group and checks to see if each one is owned. If it is not, the *Home* becomes owned and is returned by the function. If no unowned homes are found, `None` is returned instead.

```
def find_empty_home():
    for x in homes:
        if not x.owned:
            x.owned = True
            return x
    return None
```

To facilitate rational economic decision making, agents have several different priorities or "prios" that they keep track of as well. These are stored as instance variables. The main variable that keeps track of these priorities is `curr_prio`, or current priority. This variable is a string that represents this agent's current objective. This string could be `'home'` or `'work'` if the agent is currently in the process of

going home or going to work at the business where they are employed. The string can also be the product type that they are currently looking to purchase. This is different than the `destination` variable in that `curr_prio` does not represent a literal location where it is trying to go, but rather it is a general concept of what this agent is trying to accomplish. This variable is largely responsible for determining the agents behavior, and it is checked every time the agent's `update` function is called to ensure that the agent is working to accomplish the correct goal. Other priorities that an agent has are `work_prio` and `prio_list`. `work_prio` is equal to the agent's `money` times negative 1. When the simulation is deciding which agents should act as workers for that day, it compares each agent's `work_prio` against all the others. The agents with the highest `work_prio`, and thus the least amount of money, are selected. `prio_list` is a dictionary that contains each product that is available for purchase in the simulation. The value for each entry in the dictionary is the amount of that product the agent has times negative 1. When an agent is deciding what product to buy each day, it compares the values in this dictionary. The largest value, and thus the product the agent has the lowest amount, is selected. At the start of each day, these "prio" values are used to determine what task the agent will perform that day. These values are crucial for the agent's decision making process.

The final task taken care of in the *Agent* constructor is setting the agent's initial position. The initial position of an agent is the same as its home. In order to position the agent on their home's location, the `center` of both objects' *Rect* is used.

```
self.rect.center = self.home.rect.center
```

This line of code sets the agent's own `center` to the home's. With this, the agent will be positioned on its home when it is created.

5.1.2 Update Function

The `update` function in Pygame is a function included for all *Sprite* classes. It is intended to be overwritten when a *Sprite* object is implemented. The function is inteded to be called every tick and should include behavioral code for the *Sprite*. For the *Agent* class, the `update` function performs two tasks. The first is determining exactly what object in the simulation is this agent's destination. That object's *Rect* is then set as the agent's `destination` class variable. The second part of the `update` function calls the `move` function, which moves the agent in the direction of the destination.

At any given point, the agent must decide what their current priority is. This is determined by the `curr_prio` variable. If the agent's priority is to go home, then the destination is simply set to its own home. If the agent's priority is to go to the business where it is employed and work for the day, the agent will first look through all of the available businesses and find one that is not being worked. Then, the agent will begin working at that business and set it as its destination. This process is shown below.

```
if self.curr_prio == 'home':
    self.destination = self.home
elif self.curr_prio == 'work':
    for x in businesses:
        if x.being_worked() is False:
            self.destination = x
            x.set_worked(True)
            break
```

If the agent is currently intending on purchasing a good from a business, its `curr_prio` will be the name of that product. In the main loop of the simulation, just before consumer agents begin moving toward the businesses they want to buy from, the `determine_business` function is called. This function iterates through all the businesses in the simulation, using the `businesses Group`, and adds the businesses that sell the product that the agent is looking to buy to the `business_options` list. This function is shown below.

```
def determine_business(self, businesses):
    for x in businesses:
        if x.product_type == self.curr_prio:
            self.business_options.append(x)
```

In the update function, all of the businesses in this list are compared and the one with the lowest price is selected as the `best_option`. This business is set as the agent's destination. When the agent reaches the business that is their destination, they attempt to buy products. If they are successful, the `business_options` list is cleared and the agent's `curr_prio` is set to `'home'`. If the agent is unable to buy products from that business, because they are out of stock, that business is removed from the list of possible businesses to buy from and the agent selects the next best one. If there are no businesses with products available for purchase, the agent has their priority set to home. This process is shown below.

```
    else:
        if len(self.business_options) != 0:
            min_price = float('inf')
```

```
                best_option = None
                for x in self.business_options:
                    if x.get_sell_price() < min_price:
                        if x.get_product_amount() > 0:
                            min_price = x.get_sell_price()
                            best_option = x
                if best_option is None:
                    best_option = self.home
                    self.curr_prio = 'home'
                self.destination = best_option
                if self.rect.colliderect(self.destination.rect):
                    if self.buy(self.destination):
                        self.curr_prio = 'home'
                        self.business_options.clear()
                    else:
                        self.business_options.pop(0)
            else:
                self.curr_prio = 'home'
```

Once a destination is determined for the agent, the `move` function is called which moves the agent in the direction of their destination. Then, there is a check to determine if the agent is at home and sets the `at_home` variable appropriately. This is shown below.

```
self.move()

        if self.rect.center == self.home.rect.center:
```

```
            self.at_home = True
        else:
            self.at_home = False
```

When the *Agent* arrives at the *Business* it intends to buy from, the `buy` function is called. This function is shown below.

```
def buy(self, business):
num_purchases = 4
success = False
for x in range(0, num_purchases):
    if self.money >= business.get_sell_price():
        if business.sell():
            success = True
            self.spend_money(business.get_sell_price())
            self.gain_product(business.product_type, 1)
return success
```

An *Agent* has a maximum number of goods that it is allowed to purchase from a *Business* on a single day. This is represented by the `num_purchases` variable. For each purchase the *Agent* is allowed to make, it checks to see if it has enough money to afford the good and calls the *Business'* `sell` function. This function returns `True` if the *Business* has goods in stock to sell. If both the *Agent* has enough money to afford the good, and the *Business* has a good available to sell, the sale is successful and the `success` variable is set to `True`. The `success` variable is then returned to show if the sale is successful or not. The *Agent* gains the good using the `gain_product` function.

This function takes the product type and amount to be added as parameters and updates the *Agent's* product amounts accordingly.

This update function holds the main decision making process for movement for the agents. Every time an agent needs to move, this function is called and the agent properly selects its destination and moves toward it. As such, it is one of the most central parts of the entire simulation.

5.2 Buildings

There are two types of buildings in this project, homes and businesses. Houses serve as a place for agents to be when they are not actively working or purchasing goods. Businesses are locations where agents are able to work to earn money and purchase products with the money they earn.

5.2.1 Homes

Homes are incredibly simple buildings as they do not have much complex functionality. The main purpose of a *Home* is to act as a placeholder location both visually and logically for agents to return to at the end of each day cycle. The most interesting part of a Home's functionality is placing it in the correct location on the screen. A home's position is determined using the `find_home_location()` function. This function is called any time a *Home* is created. This function is shown here.

```
def find_home_location():
    row = len(homes) // 5
    col = len(homes) % 5
    rowcoord = (row * 100) + 100
    colcoord = (col * 100) + 525
```

```
return colcoord, rowcoord
```

This function uses the number of *Home* objects already in the `homes` group. Using this number, the correct location on the screen for the *Home* to be placed is determined. The location, represented as `x` and `y` coordinates on the window, is returned and set as the position of the *Home's Rect*. In this case, each row of houses will be `100` pixels away from each other starting `100` pixels from the top of the window. Each of the columns will be `100` pixels apart starting `525` pixels away from the left side of the window.

Additionally, like *Agents*, *Homes* have a `size` variable that determines how large they are and a `rect.center` which represents their location. For a *Home*, this variable is set to the return value of the `find_home_location` function. A *Home* also has a `owned` variable which is a boolean that represents if this *Home* has an *Agent* assigned to it to prevent multiple *Agents* from being assigned to the same *Home*.

5.2.2 Businesses

Like *Homes*, *Businesses* serve as visual placeholders for *Agents* to travel to in order to produce goods, earn money and purchase goods. However, *Businesses* have a lot more to take care of behind the scenes, as they need to hold products and dynamically change the price of their products as the simulation proceeds. For this, they have a few more class variables. Those variables are described here.

- `product_type`: This variable determines what product type this *Business* produces and sells.
- `product_amount`: This variable keeps track of how much product the *Business* has in stock.

- `sell_price`: This variable represents the price of the good. This changes throughout the simulation due to the effects of supply and demand.
- `money`: This represents how much revenue a *Business* has made so far in that day. It will be equal to the `sell_price` multiplied by the quantity of products sold that day.
- `production_amount`: This variable represents the quantity of product a business produces and has available to sell on a given day. This is also the maximum amount of a product a business can hold at one time. Unsold product from a previous day does not carry over into the next day.
- `is_worked`: This boolean variable is `True` when this *Business* has a worker assigned to it and is `False` otherwise.
- `worker`: This variable is the *Agent* that is assigned as the worker for this business.
- `can_be_worked`: Disaster events can cause *Businesses* to be closed for a period of time. If this *Business* is affected by this type of disaster, this variable is False. This prevents this *Business* from having a worker or producing goods.

One of the most important tasks handled in the *Business* class is choosing the worker for the day. This is done with the `find_worker` function. This function is shown below.

```
def find_worker(self, agents):
    if self.can_be_worked:
        current_worker = None
        prio = float('-inf')
```

```
            for x in agents:
                if x.get_work_prio() > prio and x.get_curr_prio() != 'work':
                    current_worker = x
                    prio = x.get_work_prio()
            current_worker.set_worker()
            current_worker.set_curr_prio('work')
            return current_worker
        return None
```

This function finds the *Agent* with the highest `work_prio` that is not already working at a *Business*. The *Business* then hires that *Agent* to work for that day. This function is called for each *Business* at the start of each day cycle assuming the *Business* is able to hire workers and its `can_be_worked` variable is `True`.

Aside from hiring workers, the other main function of a *Business* is producing and selling products. Producing products is done with the `produce` function. This function is shown below.

```
def produce(self):
        if self.can_be_worked:
            self.product_amount = self.production_amount
```

Business inventory never exceeds its `production_amount`. This is all done only if the `can_be_worked` variable is `True`. Any time an *Agent* arrives at a *Business* and wants to buy a product, the `sell` function is called. This function is shown below.

```
def sell(self):
    if self.product_amount > 0 and self.can_be_worked:
        self.money = self.money + self.sell_price
        self.product_amount = self.product_amount - 1
        return True
    return False
```

As long as the *Business* is able to be worked and has products available to sell, the *Business* gains money equal to the cost of the good and loses one good. This function returns `True` if the sale is successful and `False` otherwise. At the end of the day when all consumer *Agents* have finished purchasing goods, the *Business* will give the profit from the day to the *Agent* that was working that business. This is done with the `give_profits` function. This function is shown below.

```
def give_profits(self):
    if self.can_be_worked:
        self.worker.gain_money(self.money)
        self.money = 0
```

This function causes the worker to gain an amount of money equal to the *Business's* `money` which represents the total profits for the day. The *Business's* money is then reset to prepare for a new day of business.

A *Business* must always adapt to the economic circumstances that they find themselves in. To do this, they change their price dynamically to respond to the forces of supply and demand in the simulation. A *Business* wants to sell all of its product at the highest price it can to make the most money possible. To accomplish

this, the *Business* will try to find the perfect price where *Agents* can barely afford their product. They increase their price if they find themselves selling all of their product and decrease their price if they find they have product left over at the end of the day. This happens at the end of each day with the `price_change` function. This function is shown below.

```
def price_change(self):
    if self.product_amount > 0:
        if self.sell_price > 1:
            self.sell_price = self.sell_price - 1
    else:
        self.sell_price = self.sell_price + 1
```

Though this function is logically simple, it has a large effect on the simulation overall. This function causes the *Business* to increase the price of its good if it sells all of its product, and increases it otherwise. However, the price of a good will never go below zero. Dynamically changing the price of the good with this method allows the *Business* to make the most money possible.

5.3 Products

In this simulation, there are two types of products that can be bought, food and water. Both of these products are needed by *Agents*. At the end of each day cycle, *Agents* lose one food and one water. When an *Agent* decides that it is looking to purchase a good on a given day, they will look to purchase whichever product, food or water, they have less of at that time.

Because an *Agent* will always buy the product they posess less of, the two goods act as compliments to each other. *Agents* prefer to have the same number of food and water at any given time. For this reason, if the price of one of the goods increases, the demand for the other good will decrease, and thus, the price for that good will decrease.

5.4 Daily Cycle

For each day, there is an order in which events occur. First, all *Agents* calculate their priorities for the day. Then for each *Business*, the worker with the highest 'work' priority that is not already a worker becomes a worker at that *Business*. An *Agent's* 'work' priority is based on the amount of money they have. Those with the least money have the highest 'work' priority. Therefore, the *Agents* with the least money become workers for the day. Next, all *Agents* that are designated as workers for the day travel to their *Business* of employment. When all workers arrive, all *Agents* that are consumers for the day travel to the *Business* that sells the product they are attempting to buy for the lowest price. If that *Business* has products available, the *Agent* will attempt to purchase up to 4 products. They will buy less than 4 if the *Business* has less than 4 to sell, or if the *Agent* cannot afford 4. If the *Business* the *Agent* travels to does not have any products in stock upon arrival, the *Agent* will go to the *Business* with the next best price on the good they want to buy. After finding a *Business* that they are able to buy from and successfully purchasing product, consumer *Agents* will travel to their *Home*. If an *Agent* is unable to buy a product, either because they cannot afford the products at any *Businesses* or because all *Businesses* are out of stock, they will return home. When all consumers are at *Home*, all worker *Agents* will travel to their *Homes* as well.

When all *Agents* are at their *Homes*, workers recieve wages for the day. Wages

are equal to the *Business's* revenue, which is the number of products sold multiplied by the price of the good. *Businesses* then change the price of their goods based on how many products they have remaining in stock and produce new goods.

Then, all necessary data is written to Excel and a new day cycle begins. A diagram showing the daily cycle process is shown in Figure 5.1

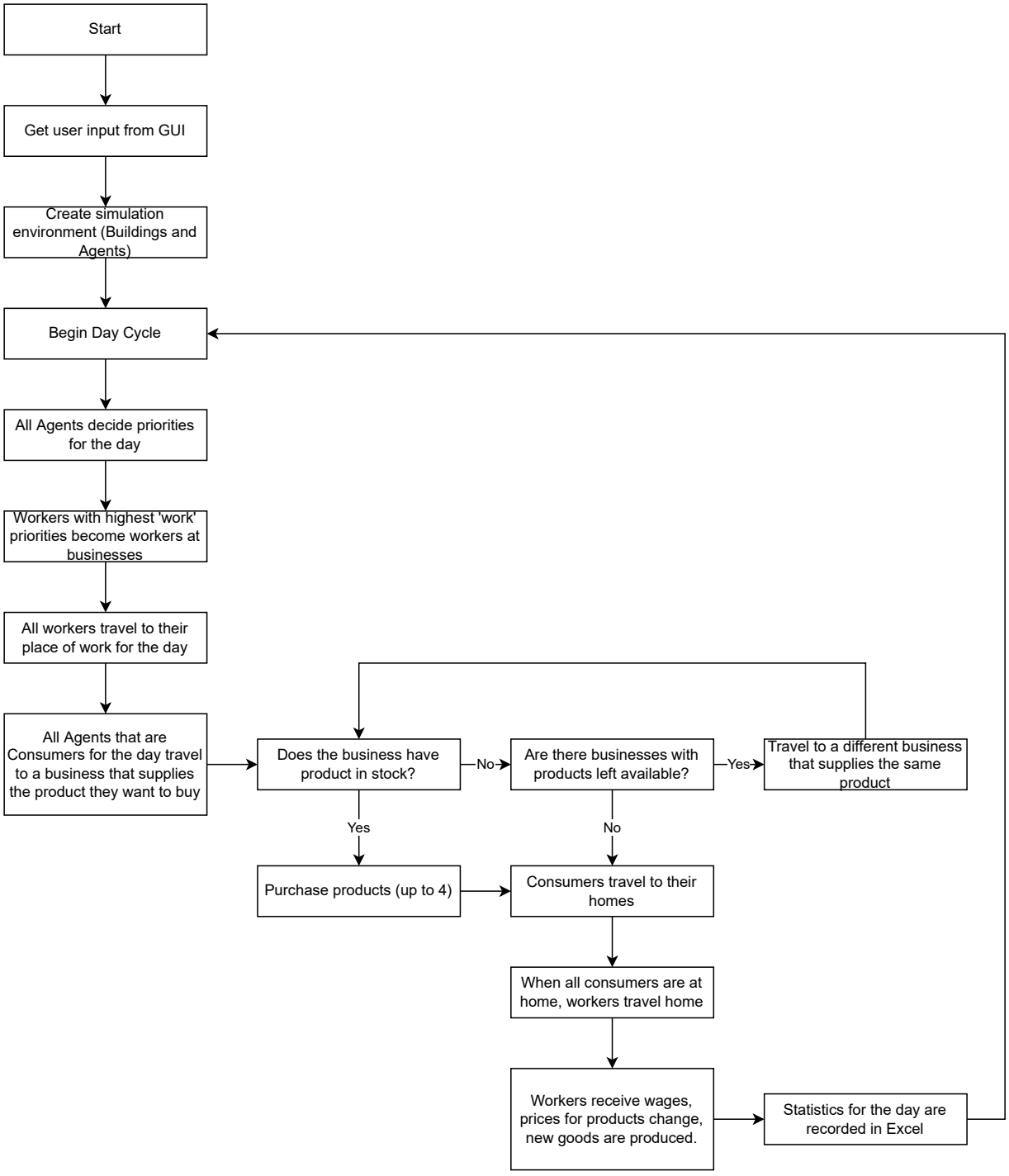

Figure 5.1: Simulation Daily Cycle

5.5 Impact Events

There are two types of Impact events that can effect the simulation, disaster events and relief events. Disaster events reduce *Business'* ability to produce while relief events provide help to *Agents*. For each different type of event, a start day and severity are specified by user input. For disasters, there is also an end day. The start and end day determine when an event will start and end. The severity of the event determine to what degree it impacts the simulation. This number will represent various different things based on the type of event. The different types of events are shown below.

Disaster Events

- Pandemic: A pandemic reduces the number of *Businesses* that are open. The severity of this event is the number of *Businesses* that will be closed for the duration of the event.
- Famine: A famine decreases food production. While this event is active, all *Businesses* that produce food produce less each day. The severity of this event is expressed as a percent decrease in production.
- Drought: A drought decreases water production. While this event is active, all *Businesses* that produce water produce less each day. The severity of this event is expressed as a percent decrease in production.
- Tornado: A tornado decreases the production of both food and water. While this event is active, all *Businesses* less each day. The severity of this event is expressed as a percent decrease in production.

Relief Events

- Stimulus: A stimulus event gives all *Agents* an equal amount of money. The severity of this event is the amount of money that is given.

- Aid: An aid event gives all *Agents* an equal amount of food and water. The severity of this event is the amount of food and water that is given.

The collection of user input is handled through the Python library Tkinter. Tkinter handles the creation of GUIs and the collection of input from them. This program uses two different windows. One handles the disaster events and the other handles the relief events. Both windows feature a dropdown menu to select the type of event and text boxes to specify start and end days along with event severities. This data is stored in a dictionary that has an entry for each data point. At the end of each day cycle, the current day number of the simulation is checked against the start and end days of the events for that run of the simulation. If there is a match, the proper processes are taken.

If the selected disaster event is a pandemic, when the disaster start day is reached, a number of *Businesses* equal to the disaster severity have their `can_be_worked` variables set to `False`. When this variable is false, the *Business* cannot produce goods, be worked by an *Agent*, or sell goods. When the end day of this disaster is reached, all *Businesses* have their `can_be_worked` variable set to *True*, allowing them to continue normal economic operations.

The disaster events famine, drought, and tornado all act very similarly. Famine impacts food production, drought impacts water production, and tornado impacts the production of both goods. When the start day of the event is reached, all *Businesses* that produce a good that is impacted have their `set_production_amount` function called. This function takes the severity of the disaster as a parameter. The severity for these disasters represent the percent decrease in production. The `set_production_amount` function is shown below.

```
def set_production_amount(self, amount):
    amount = float(amount) / 100
```

```
amount = 1 - amount
self.production_amount = int(self.production_amount * amount)
```

When the disaster end day is reached, all *Businesses* have their production amounts reset to the originial production level.

If the selected relief type is stimulus, when the event start day is reached, all *Agents* gain an amount of money equal to the relief amount. This is done using the *Agent* `gain_money` function.

If the selected relief is aid, when the event day is reached, all *Agents* gain an amount of food and water equal to the relief amount. This is done using the *Agent* `gain_product` function. There is no end day for relief events, as they are only a one time occurance.

5.6 Data Collection

Data collection for this simulation is done using the Pandas library. At the end of each day cycle, data points are collected from the simulation and written to csv files. Then, when the simulation ends, those csv files are converted to Excel spreadsheets using Pandas. There are two csv files that collect data, `simulationdata.csv` and `agentdata.csv`. `simulationdata.csv` stores data pertaining to the entire simulation, while `agentdata.csv` stores data for each individual agent. The specific data points that are tracked are shown below.

Simulation Data

- Average Food Price: The average price of food across all *Businesses* on that day.
- Average Water Price: The average price of water across all *Businesses* on that day.

Agent Data

- Food: The amount of food each *Agent* has on that day.
- Water: The amount of water each *Agent* has on that day.
- Money: The amount of money each *Agent* has on that day.

Writing to csv files is handled using Python's `csv` library. When the program starts, the headers for the csv files are written. These are labels for each column. For `simulationdata.csv` the header simply includes one row where the Average Food Price and Average Water Price columns are labeled. For `agentdata.csv`, the header is two rows. The first row includes the label for each *Agent*. Each *Agent* is numbered. Because there are three data points per *Agent*, every three columns are labeled with the appropriate *Agent* number. The second row labels the columns for food, water, and money of each *Agent* respectively.

In order to write to a file with the csv library, a *writer* object is created which has the function `writerow`. This function takes an iterable and writes it as a row of the file. At the end of each day cycle, the average price of food and water are written to `simulationdata.csv`, and the food, water, and money amounts for each *Agent* are written to `agentdata.csv`. This process is shown below.

```
with open('simulationdata.csv', 'a', newline='') as csvfile:
    csv_writer = csv.writer(csvfile, delimiter=',', lineterminator='\n')
    csv_writer.writerow((food_average, water_average))

row = []
for x in agents:
    row.append(x.money)
```

```
    row.append(x.food)
    row.append(x.water)
with open('agentdata.csv', 'a', newline='') as csvfile:
    csv_writer = csv.writer(csvfile, delimiter=',', lineterminator='\n')
    csv_writer.writerow(row)
```

Finally, when the simulation ends, Pandas is used to convert the csv files in Excel files. This is done using Pandas *DataFrames* and the `read_csv` and `to_excel` functions. A *DataFrame* is the object that Pandas uses to store data. Data in a *DataFrame* can be put into many different formats using Pandas functions. The process to read from a csv file and write to Excel is shown below.

```
filepath = os.path.dirname(os.path.abspath(__file__))
read_file = pd.read_csv(filepath + '/simulationdata.csv')
read_file.to_excel(filepath + '/simulationdata.xlsx', index=None, header=True)
read_file2 = pd.read_csv(filepath + '/agentdata.csv')
read_file2.to_excel(filepath + '/agentdata.xlsx', index=None, header=True)
```

This process of data collection automates putting important data from the simulation into Excel spreadsheets for analysis. An example of each csv file is shown below.

`selectiondata.csv`

average food price	average water price	average price
5.6	5.2	5.4
5.8	5.8	5.8

`agentdata.csv`

Agent_0 Money	Agent_0 Food	Agent_0 Water	Agent_1 Money	Agent_1 Food	Agent_1 Water
175	8	7	175	6	9

CHAPTER 6

Analysis of Simulation

This chapter discusses the results from many simulations including a variety of different disaster events and relief acts.

6.1 Results

The first set of tests included five simulation runs. The goal of these were to understand the effects of each disaster without relief. The first test used no disaster and no relief, and the next four each included one disaster, pandemic (close a number of *Businesses*), famine (reduce food production), drought (reduce water production), and tornado (reduce both food and water production). Each of these tests shows the impact of its respective disaster with no relief effect. The next ten simulations show the impacts of relief efforts. Tests 6 through 10 show each disaster option along with a stimulus relief (give all *Agents* an equal amount of money), and tests 11 through 15 test each disaster along with the aid relief (give all *Agents* an equal amount of food and water).

For these first 15 tests, all simulations were run for 200 days. Disasters began on day 60 and ended on day 80. All relief effects were introduced on day 70. The relief effects are incorporated halfway through the disaster events so that the effects of

the disaster can be compared equally with and without the relief. All disasters had a 40 percent effect on the production of goods. Stimulus relief gave all *Agents* 100 money and aid relief gave all *Agents* 15 food and 15 water. All *Agents* start with 100 money and between 5 and 10 food and water, chosen randomly. The price of goods at all *Businesses* starts at 5, and *Businesses* produce 15 of their good each day when not affected by a disaster event.

6.1.1 Disaster Impacts

The simulation with no disaster event shows the price of both food and water increasing steadily until the average price of both goods reaches an equilibrium level of around 17.2 to 17.4. At this point, the price stays consistent throughout the remaining days of the simulation. One important thing to note is that generally, the prices of food and water move in opposite directions. This is because *Agents* can only choose to buy one product at a time. When most *Agents* happen to choose one product over another, the price of that product will increase while the other decreases. The graph for this simulation is shown in Figure 6.1

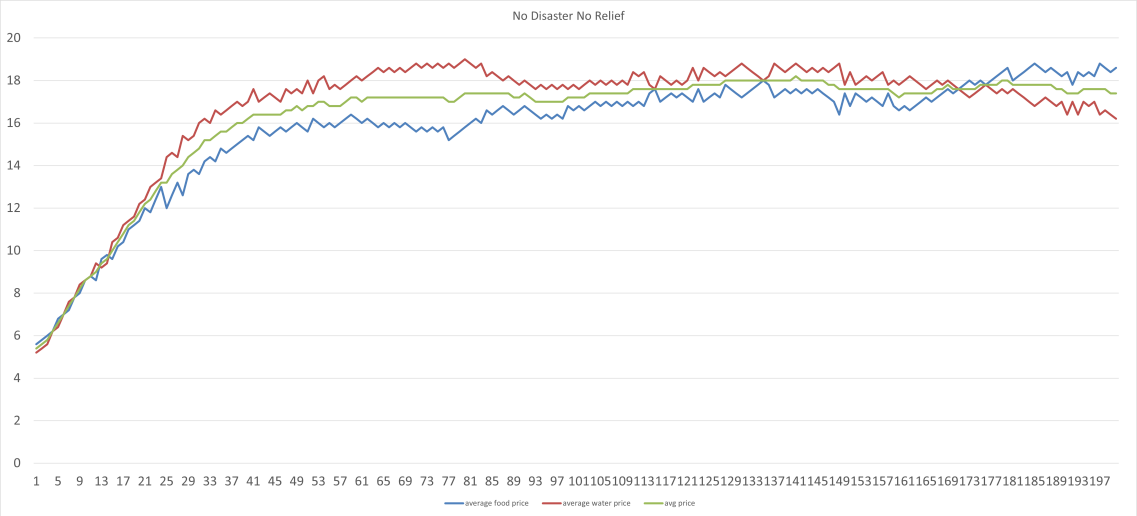

Figure 6.1: No Disaster No Relief

The simulation with the pandemic event had expected results. For the pandemic,

four of the ten *Businesses* were closed for the 20 day disaster period. When the disaster began, the average product price was 17.6. After the disaster began, the price average climbed steadily to 21 on day 75, where it stayed for the remainder of the disaster period. After the disaster ended, the prices dropped back to 17.8 by day 110. From there, the price stayed consistently between 18 and 17.4 for the rest of the simulation. This change in prices is seen because when the disaster begins, the total supply of both food and water is decreased significantly, but demand for these products stays the same. This pushes the prices upward. When there are fewer products available, at a normal price, the *Agents* are able to afford all the product that the *Businesses* are able to produce, so they buy all that is available. Because *Businesses* increase their price at the end of a day when the sell all of their products, all the *Businesses* increase their prices. This happens each day until the price of products is too high for *Agents* to be able to afford all the products that are produced. At this point, because only some *Agents* are able to afford the products, only some of the *Businesses* sell out of products. Those *Businesses* increase their price, while others that did not sell out decrease their price. Overall, this leads to no change in price. Thus, the price flattens out for a period of time. When the pandemic ends and all *Businesses* are fully operating, the price starts to decline. This is because the four *Businesses* that were closed during the pandemic still have their prices set to what they were before they were closed. Because *Agents* are able to choose to go to the stores with the best prices, all the stores that are newly reopened sell all their product. The *Businesses* with the higher pandemic prices do not sell many products and thus decrease their prices. This trend continues until the prices meet. Then, because the price is still too high for *Agents* to be able to afford to buy all the products available, the prices in all *Businesses* decrease until they reach the equilibrium price of 17.8. The graph for this simulation is shown in Figure 6.2

The famine showed an increase in price for food and a decrease in price for

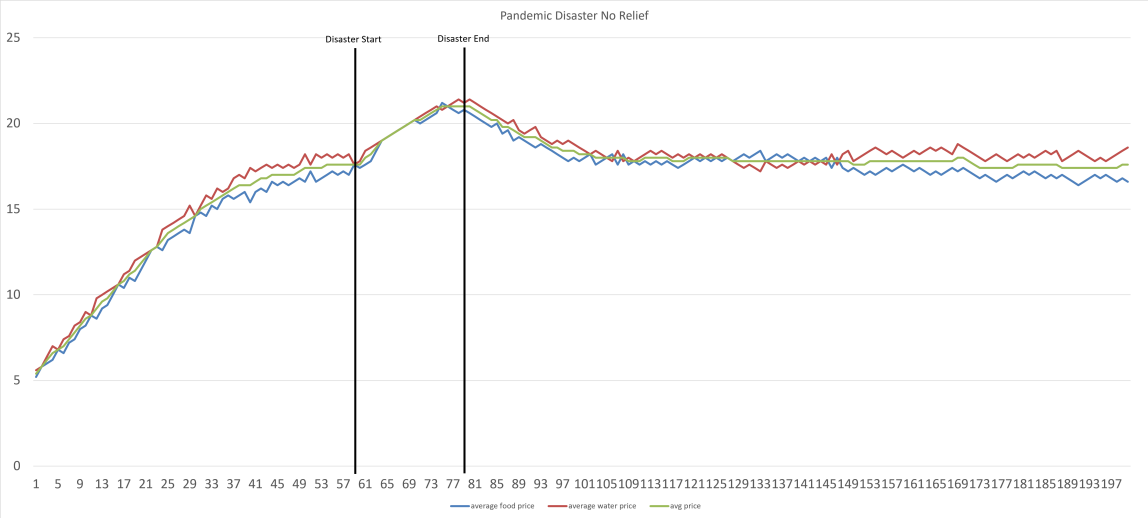

Figure 6.2: Pandemic Disaster No Relief

water. This disaster used a similar setup as the pandemic. During the disaster period, all *Businesses* that produce food produce 40 percent less than what they were producing before. When the disaster began, there was a sharp increase in price for food along with a gradual decrease in price for water. Before the disaster began, the price of food was 18.2, and the price of water was 16.6. By the end of the famine, the price of food was 30.2, and the price of water was 15.4. Then when the disaster ended, the price of food begins to decrease. The price of water continued to decrease but at a faster rate. After the disaster, both the price of food and the price of water decreased at a similar rate and flattened out at a similar point in time. Because of this, the prices never converged back to a similar price. The price of food stayed much higher than the price of water. The price of food leveled out to around 22.5, and the price of water flattened at around 6.5. This made the average price of all goods around 14.5, which was lower than the equilibrium price seen before the disaster or in the simulation with out any disaster. At the onset of the famine, the increase in food price is due to the lowered supply of food. When the production of food is decreased, all the food that is available is bought, and many *Agents* are

not able to buy as much food as they want to on any given day. Because of this, *Agents* spend more days attempting to purchase food. *Agents* base which product they want to buy solely on which product they currently have less of at the start of the day. If they are unsuccessful in purchasing this good, they simply go home empty handed. They do not attempt to purchase the other good. This causes the decrease in water price during the disaster. *Agents* are spending their days trying to buy food and mostly being unsuccessful in accomplishing this task. Thus, they spend fewer days looking to buy water, and the water *Businesses* do not sell out of thier product, so they decrease the price. Then, when the famine ends, the price of both goods decrease. This is because the *Agents* have to spend all of their available money on food. *Agents* continue to have less food than water for several days after the famine ends, so they continue to try to buy food. At this time, however, the food is available, and all the money they were unable to spend on food during the famine is now being spent. Because they spend all their money on food, *Agents* do not have money to spend on water. In the period shortly after the famine ends, there are two forces that push down the price of water. *Agents* are prefering to buy food during this time, and *Agents* do not have money to spend on water when they do want to buy it. Because there is now an additional force pushing the price down, it decreases at an accelerated rate. The graph for this simulation is shown in Figure 6.3

There is a very similar set of trends seen for the drought event, but with the food and water having opposite trends. Before the drought begins, the food price sits at 18 and the water price is at 17. Then, when the disaster begins, the water price increases quickly while the food price decreases slowly. The water price is 30.2 and the food price is 16.6 when the disaster ends. After the end of the disaster, the water price begins declining and the food price continues to decrease at a faster rate. The prices then level out with water being at 22, food being at 8.4, and the average being

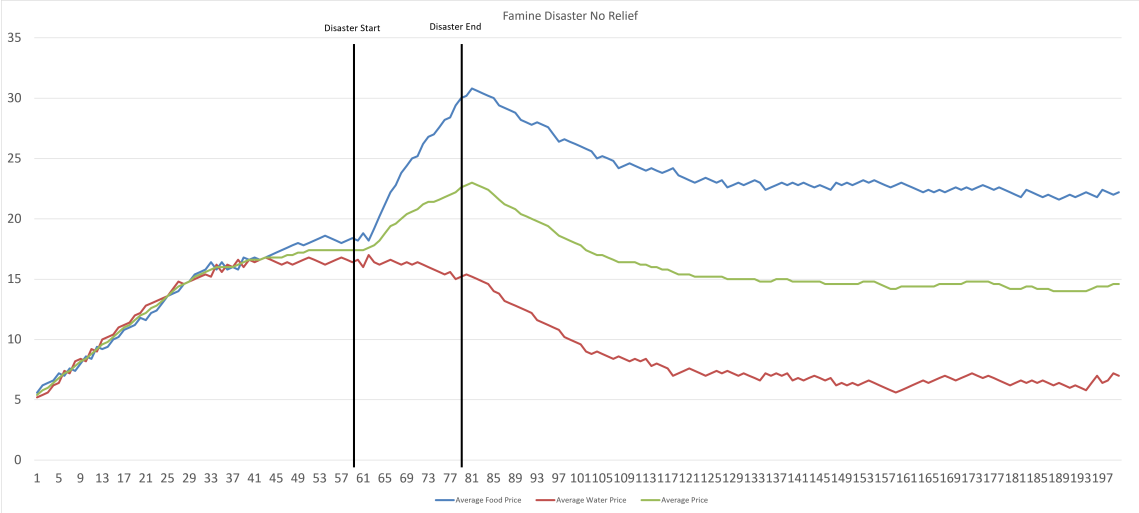

Figure 6.3: Famine Disaster No Relief

15.2. Again, this post disaster average price is lower than the prices seen before the onset of the disaster and with no disaster event at all. The graph for this simulation is shown in Figure 6.4

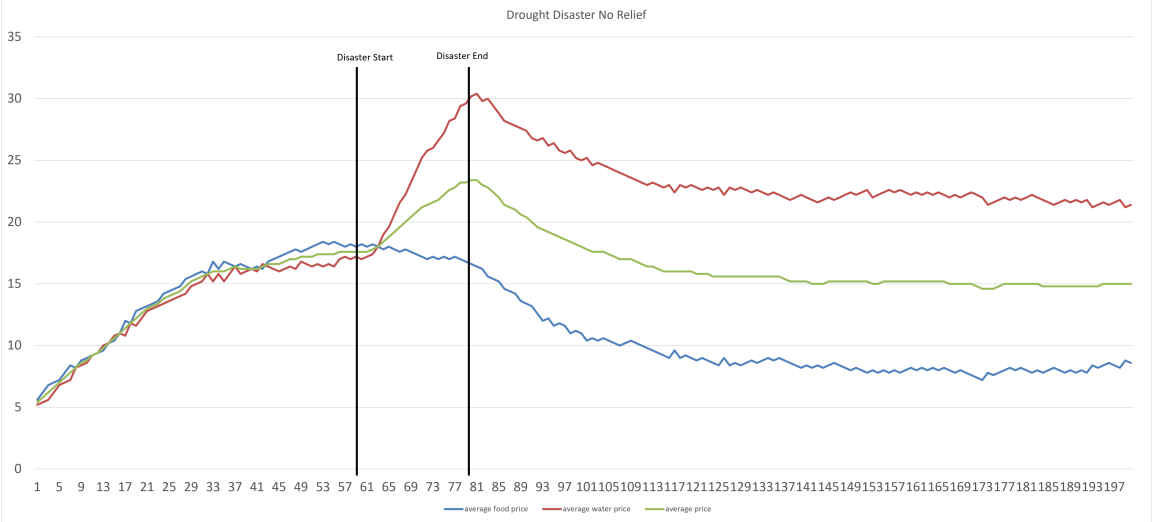

Figure 6.4: Drought Disaster No Relief

The tornado simulation shows results similar to the pandemic. This is expected because essentially, both disasters decrease the availability of both goods by 40 percent. Before the disaster starts, the average price of all goods is consistently 17.4.

Then after the disaster starts, the prices quickly increase. When the disaster ends, the average price of all goods is 28.4. After the disaster ends, the prices slowly decline until they reach the level that they were at before the disaster, 17.4. The prices remain steady at this level for the remainder of the simulation. The graph for this simulation is shown in Figure 6.5

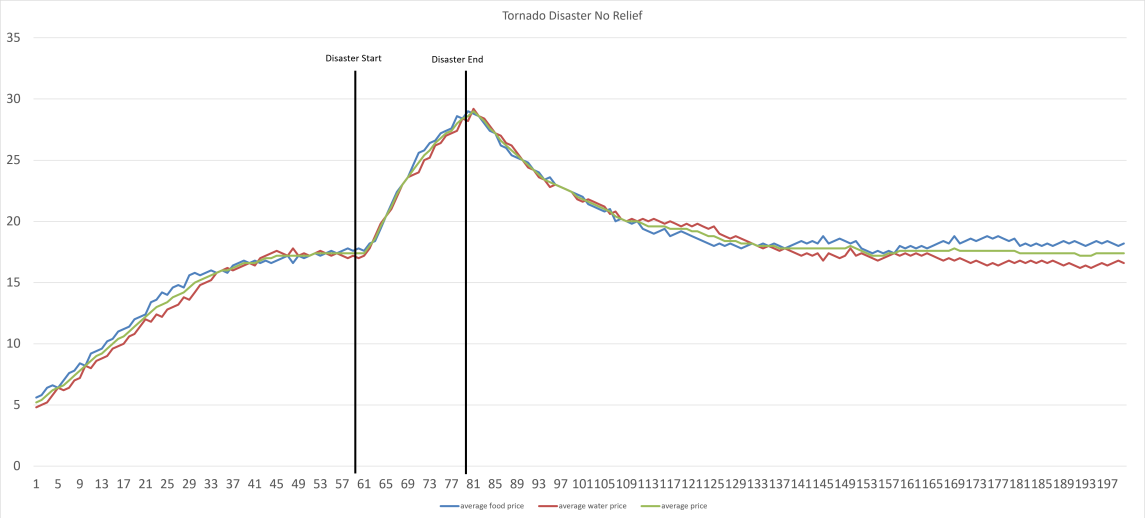

Figure 6.5: Tornado Disaster No Relief

6.1.2 Relief Impacts

Both relief options are intended to assist *Agents* in dealing with the effects of the disaster events. Both of the available relief options had unintended consequences. They both had minimal effect in assisting the *Agents*.

The intent of the stimulus relief is to assist *Agents* in being able to afford more expensive products that are created by the decrease in production from disaster events. However, because *Businesses* are always trying to sell at prices where *Agents* can barely afford to buy out all of their inventories, the increased money available for each *Agent* simply leads to the *Businesses* increasing their prices to account for the increase in available money. This effect is known as inflation. The total supply of

money available in the economy increases, so businesses raise their prices because the consumers are able to afford to pay the increased prices. This effect is seen in all simulations where the stimulus relief is present.

In the simulation with no disaster and stimulus relief, the average price of goods is about 17.4 before the relief comes into effect. This is the same price level that was seen in other simulations before disasters came into effect. After the stimulus was introduced, the average price of all goods began increasing and then flattened out at around 34.8. This new price level is exactly double the old price level. This makes sense as the total amount of money in the economy has doubled. All *Agents* start with 100 money, and they are given 100 money when the relief is introduced. The graph for this simulation is shown in Figure 6.6

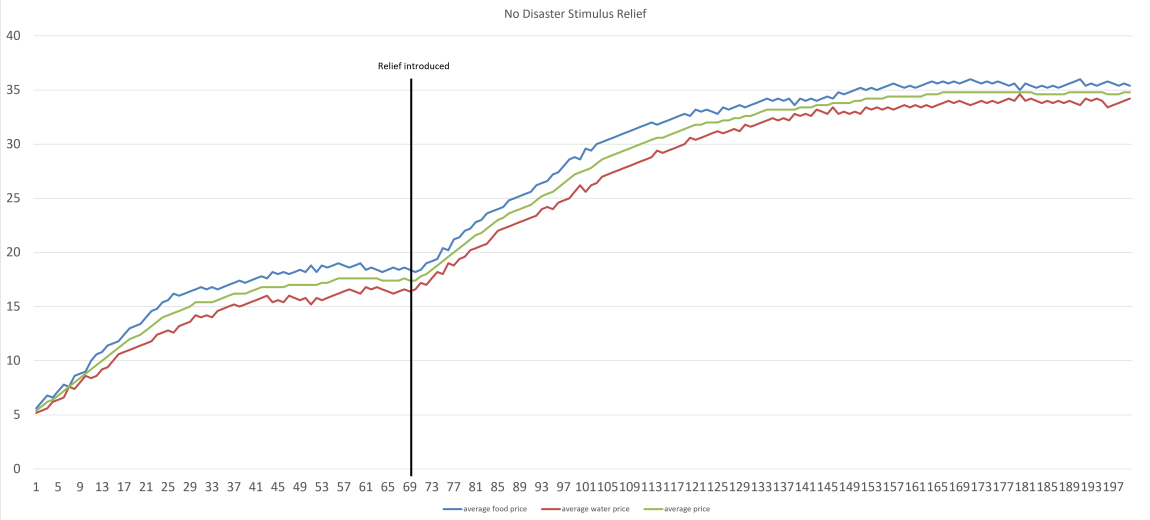

Figure 6.6: No Disaster Stimulus Relief

The simulation with the pandemic and stimulus showed a combined effect of both events. Before the disaster started, the price level was at 17.2. After the disaster was introduced, the average price level of all goods began increasing. On day 70, when the stimulus was introduced, the average price level of all goods was 20, which is a 10 day increase of 2.8. After the relief came into effect, the price levels

began increasing at a higher rate. 10 days later, on day 80 when the disaster ended, the average price level of all goods was 25.6, which is a 10 day increase of 5.6. That is double the increase that was seen from just the pandemic during the first 10 days of the disaster. After the disaster ended, the prices continued to rise but at a slower rate. The prices leveled out at 35 by the end of the 200 day simulation. The graph for this simulation is shown in Figure 6.7

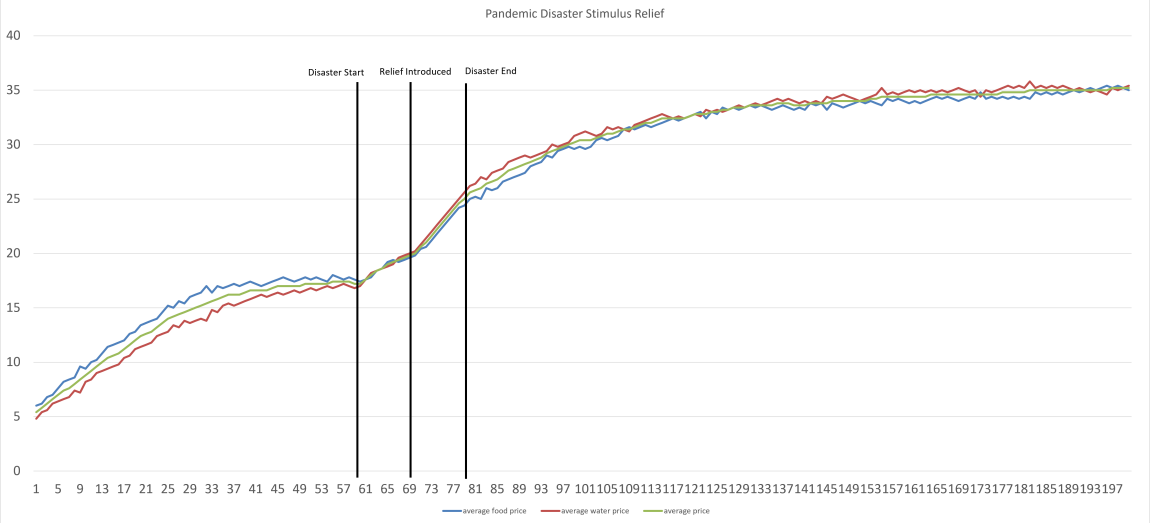

Figure 6.7: Pandemic Disaster Stimulus Relief

The famine simulation with the stimulus showed a large and fast increase in the price of food. Before the disaster began, the price of food was 17.8. On day 70, when the disaster was introduced, the average price of food was 24.6, which is a 10 day increase of 6.8. At the end of the disaster, on day 80, the price of food was 34.6. During the second half of the disaster, there was a 10 day increase of 10. For the average price of food to increase by 10 in 10 days, every *Business* selling food must have sold out of product and increased their price during that time. This is because the famine alone made *Agents* become short on food, so most would be looking to buy food during this 10 day window. Then because of the stimulus, they are able to afford the food, even at the increased price. So, all the food that is

available during this time is being bought. During this time, the price of water was mostly unchanged. This is because the decrease in supply of food pushed the price of water down while the inflation from the stimulus would push it up. Before the disaster took place, the price of water was 17. When the relief was introduced, the price of water was 16.8. Then, when the disaster ended, the price of water was 17. So, there was a net zero change in the price of water over the course of the famine. However, because of inflation, after the disaster ended, the price of water continued to increase until it leveled out to around 23. At this same time, the price of food ended up being around 41.4. This means that the average price level of all goods was around 32. Again the average price level of all goods post famine is about 2 lower than the averages that are seen before disaster events or after disaster events that impact both goods equally. The graph for this simulation is shown in Figure 6.8

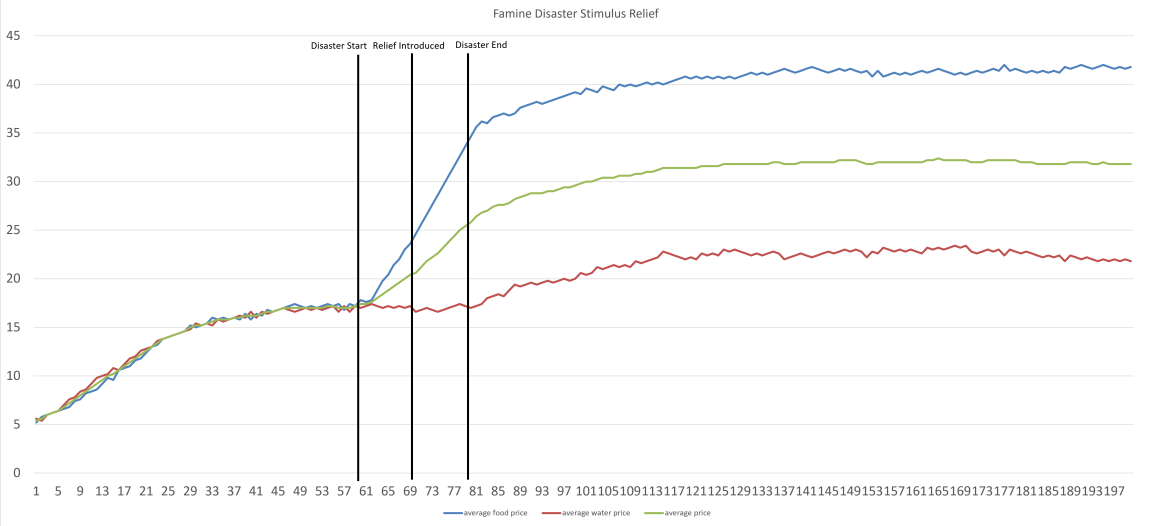

Figure 6.8: Famine Disaster Stimulus Relief

The drought simulation with the stimulus gave results very similar to the famine and stimulus simulation, but with the trends of food and water switched. The onset of the drought along with the stimulus led to a large increase in the price of water.

Before the disaster, the price of water was 18.2, and was 25.4 on day 70, when the stimulus was introduced. This is a 10 day increase of 7.2. Then on day 80, when the disaster ended, the average price of water was 35. The price increase over the second half of the disaster was 9.6. For this simulation, not all water *Businesses* sold completely out of stock each day, but it was very close. For the price of food, again the disaster event pushes the price of the good down while the inflation pushes the price up. The price of food was 16.2 before the start of the disaster and was 16.6 on day 70. By day 80, the price of food was 17. Overall, a 0.8 increase over the course of the disaster. After the disaster ended, the prices of both goods continued to increase. The average price of water ended up around 41.8 and the average price of food ended up around 24.6. The average price of all goods was around 33.2. Again the average price level of all goods after a disaster event that only impacts the production of one good is about 2 lower than the average price level that is seen before disaster events or after disaster events that impact both goods equally. The graph for this simulation is shown in Figure 6.9

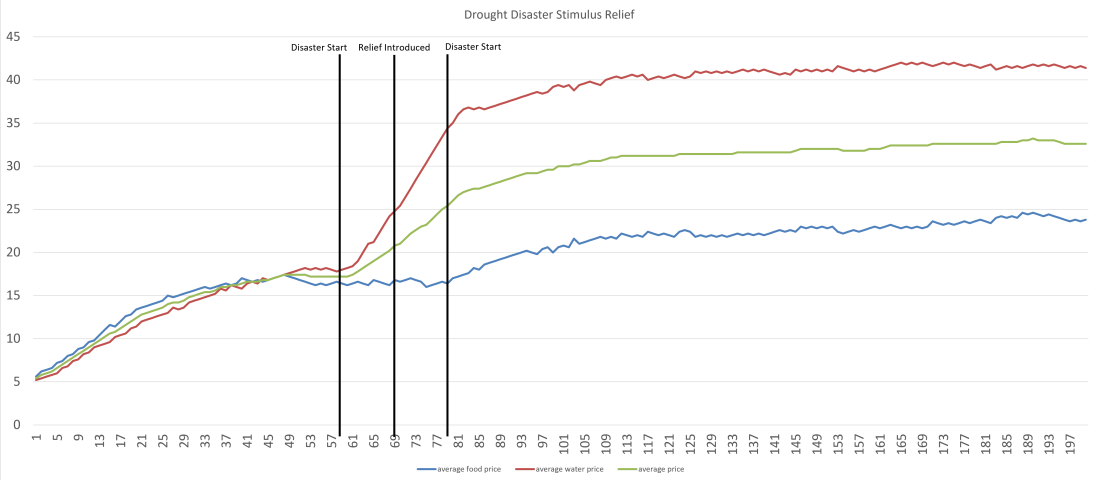

Figure 6.9: Drought Disaster Stimulus Relief

The simulation with a tornado event and stimulus relief showed large price increases in both goods. Before the start of the disaster, the average price of all

goods was 17.4. On day 70, the average price of all goods was 24.6, which is a 10 day increase of 7.2. On day 80, the end of the disaster event, the average price was 32.6, which is a 10 day increase of 8 across the second half of the disaster period. After the end of the disaster, the prices did not change much and the average price of all goods ended up at around 34.8 by the end of the simulation. The graph for this simulation is shown in Figure 6.10

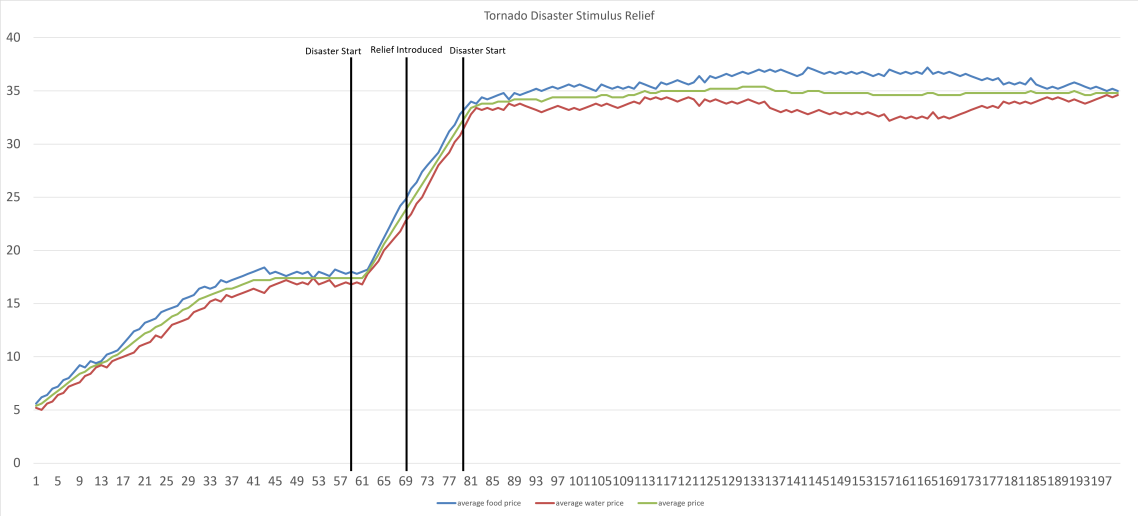

Figure 6.10: Tornado Disaster Stimulus Relief

The intent of the aid relief is to help *Agents* that struggle to access food or water during the course of a disaster. By giving all *Agents* an equal amount of food or water, they are less likely to run out during the course of the simulation. However, because there is no penalty for running out of either food or water, and no reward for having a certain amount in their possesion, being given an equal amount of both goods has no effect on an *Agent's* decision making process. *Agents* only look at which product they have less of when making a decision, so adding an equal amount to both does not change this. Because the aid relief event does not change *Agent* behavior, the results of simulation runs 11 through 15 are incredibly similar to the results of simulation runs 1 through 5.

The simulation with no disaster and aid relief shows results very similar to the

simulation with no disaster and no relief. When the simulation reached day 50, the price average price of all goods had flattened out. From day 50 through the rest of the simulation, the average price of all goods was between 17 and 18. The introduction of the aid relief showed no impact on the price of goods. The graph for this simulation is shown in Figure 6.11

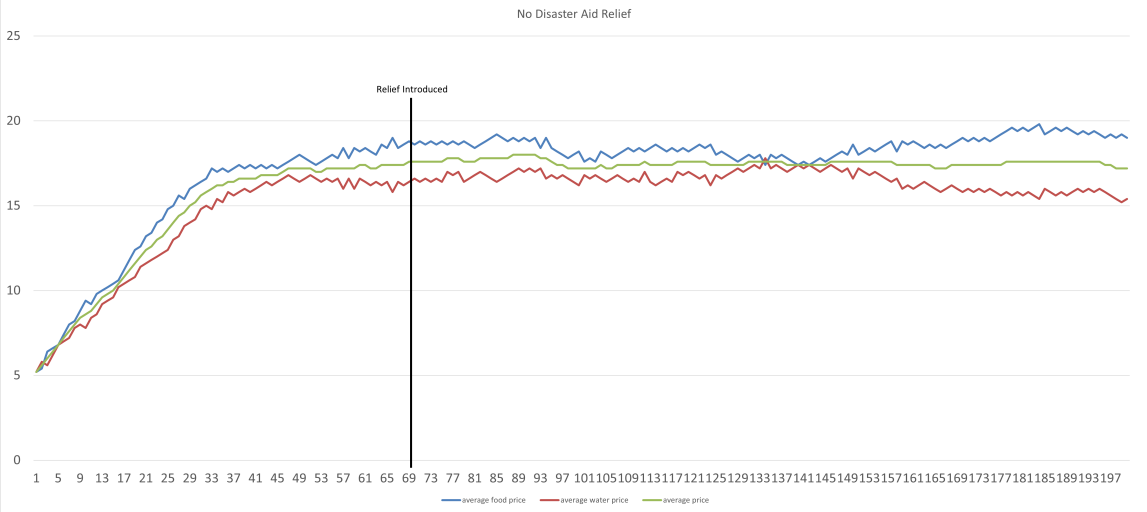

Figure 6.11: No Disaster Aid Relief

The simulation with a pandemic and aid relief ran very similarly to the simulation with a pandemic and no relief. When the disaster started, at day 60, the average price of all goods was 17.4. When the aid was introduced on day 70, the average price of all goods was 20. Then from day 75 to when the disaster ended on day 80, the average price of all goods was 20.8. Then, after the disaster, the prices dropped slowly until they flattened out at around 17.6. These results are not much different from the results seen on the simulation with pandemic and no relief. The aid relief is showing no impact on *Agent* behavior or price levels. The graph for this simulation is shown in Figure 6.12

The simulation with a famine disaster and aid relief also produced very similar trends to the famine and no aid simulation. When the disaster started, both the average price of food and water were 17.4. On day 70, when the relief was

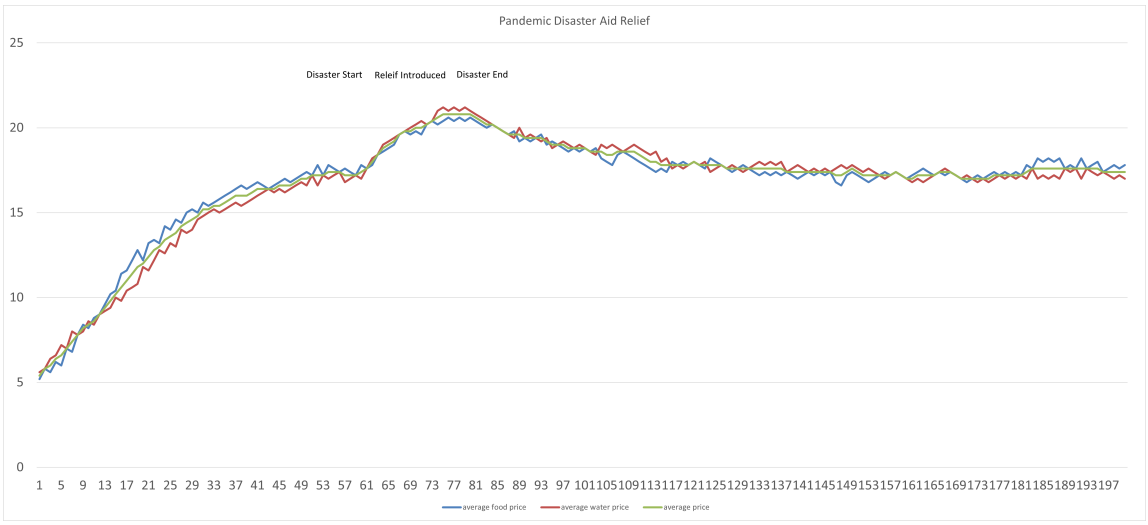

Figure 6.12: Pandemic Disaster Aid Relief

introduced, the price of food was 24.6, and the price of water was 17.4. Then at the end of the disaster period, the price of food was 31, and the price of water was 16.2. Then after the disaster, both prices dropped until they leveled out. The average price of food ended around 21.8, the average price of water ended around 8.6, and the average price of all goods ended around 15.2. Again, all of these results closely resemble the results of the simulation with no aid introduced. The graph for this simulation is shown in Figure 6.13

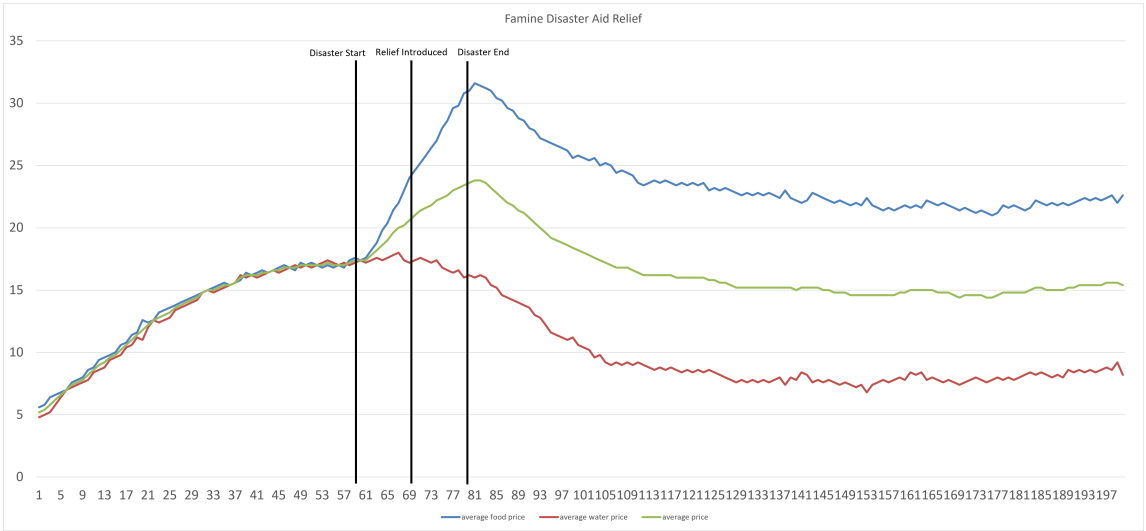

Figure 6.13: Famine Disaster Aid Relief

The simulation with a drought and aid relief showed very expected results. The results were similar to the previous drought simulation with no relief and to the simulations with a famine and no or aid relief, but with the products swapped. Before the start of the disaster, the price of food was 17.4, and the price of water was 17.8. Then on day 70, the price of food was still at 17.4, and the price of water was 25. Then when the disaster ended, the price of food was 15.8, and the price of water had increased to 31. Then, after the disaster ended, the prices for both goods dropped at similar rates until the average price of food was 4.2, the average price of water was 21, and the average price of all goods was 12.6. The graph for this simulation is shown in Figure 6.14

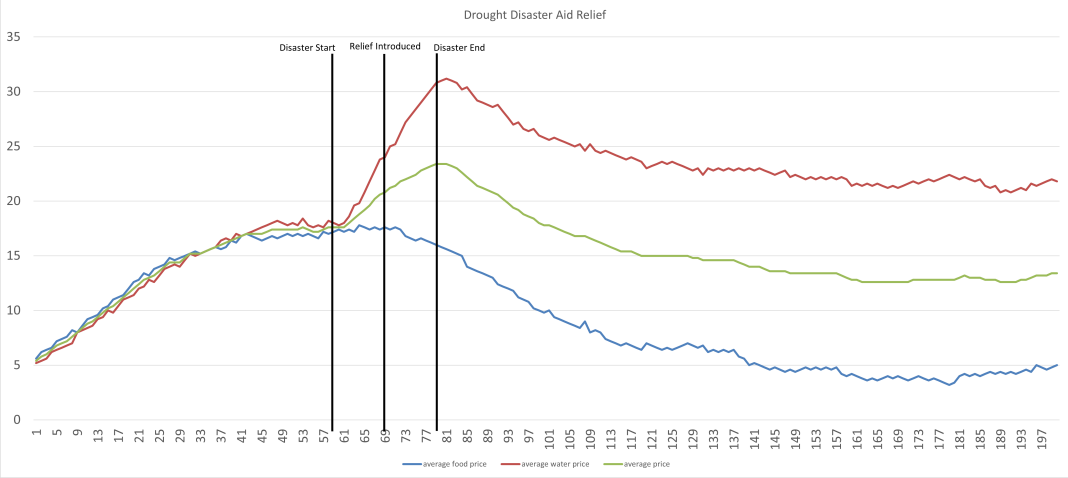

Figure 6.14: Drought Disaster Aid Relief

The tornado and aid simulation also showed expected results. When the disaster began, the average price of all goods was 17.2. Then, when the relief was introduced on day 70, the average price of all goods was 23.4. When the disaster ended, the average price of all goods was 28.8. After the disaster, the prices of all goods slowly dropped until they leveled out at around 17.8. The graph for this simulation is shown in Figure 6.15

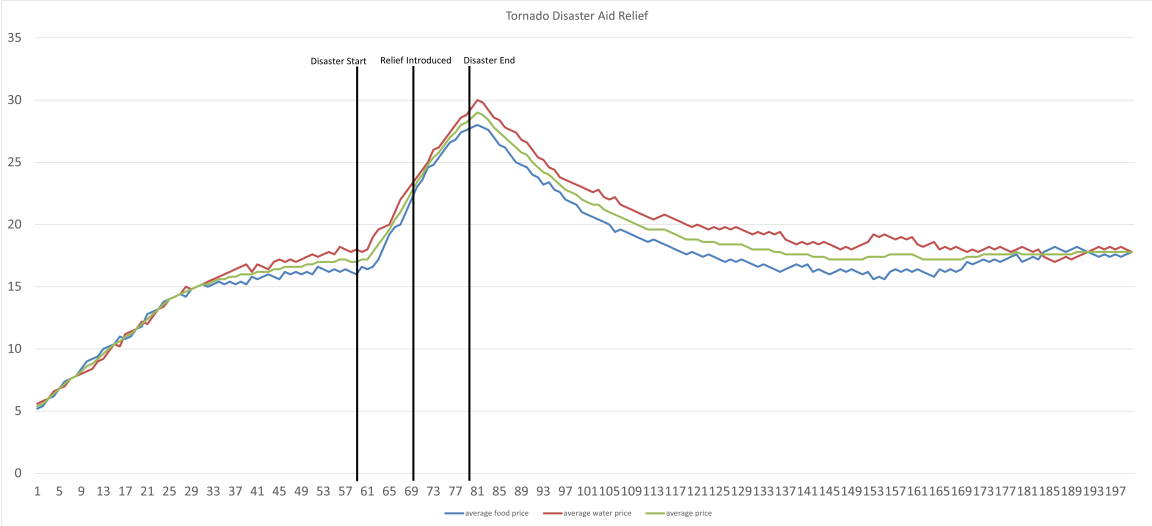

Figure 6.15: Tornado Disaster Aid Relief

6.2 Simulations with Increased Disaster Durations

One of the main questions that came from looking at data from the first 15 tests was how prices would change over a longer disaster period. To investigate this, four more simulations were run, one for each disaster. For these tests, the disasters started on day 60 as before, but they did not end until day 120. In total, the disasters lasted 60 days compared to the 20 days of the original tests. All other parameters of the simulations were the same as the previous tests, the total duration of the entire simulation was still 200 days, the production impact was 40 percent, and there were no relief events.

For the first of the long disaster simulations, a pandemic was introduced on day 60 and ended on day 120. Just before the disaster began, the average price of all goods was 17. When the disaster began, the prices of all goods began to increase until day 83, 23 days into the disaster. From day 83 through the rest of the disaster, the average price of all goods was mostly flat and stayed between 21.6 and 22.2. After the disaster ended, the prices slowly decreased until the end of the simulation

where the average price of all goods was consistently at 17.6. The graph for this simulation is shown in Figure 6.16

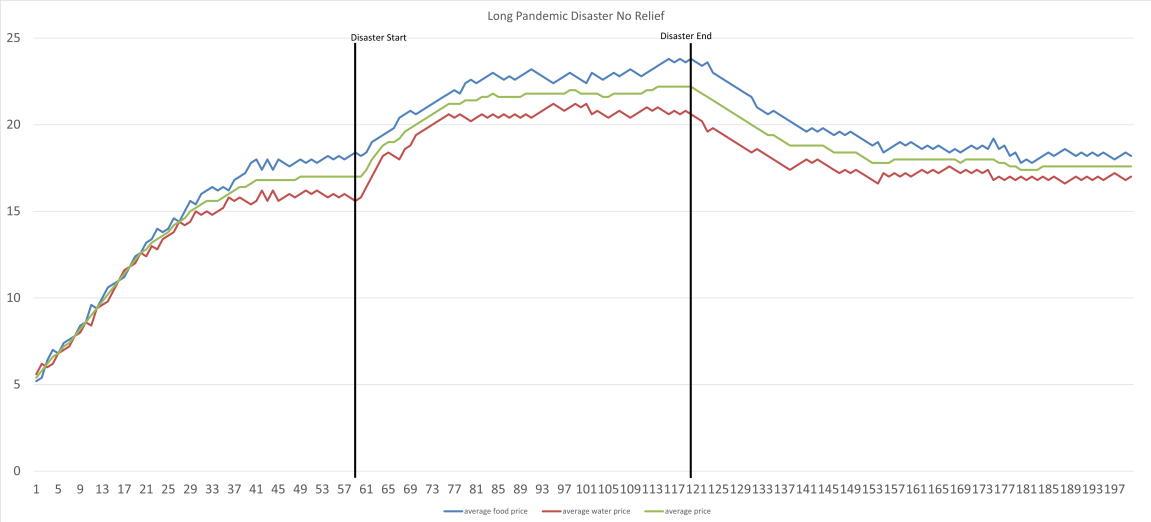

Figure 6.16: Long Pandemic Disaster No Relief

The long famine disaster exaggerated the trends that were seen in the original famine simulation. When the disaster started, the price of food was 18.2. During the course of the entire disaster, the price of food increased. It increased quickly at first and more slowly toward the end of the disaster. The price of food was at its highest point at the end of the disaster. That price was 41.4. After the disaster ended, the price of food began declining. Again, it declined quickly at first and then more slowly toward the end of the smulation to the point where it had almost flattened out at the end but not quite. At the end of the simulation, the price of food was 20.6. The price of water showed a very large decrease due to the famine. When the disaster started, the price of water was 16.6. For the first 20 days of the disaster, the price of water was fairly consistent, decreasing slightly. Then, starting on day 80, the price of water decreased at a fairly constant rate for the remainder of the disaster. The price of water reached 4.2. After the disaster ended, the price of water continued to decrease at a similar rate. On day 128, the price of water reached its

lowest point of the simulation, 1.2. At this point the price flattened out remarkably. The price of water stayed between 1.2 and 1.6 for almost the entire remainder of the simulation. On day 194, the price began increasing and reached a level of 2.2 by the end of the simulation. This remarkable flattening effect is due to the price floor instated on *Businesses*. Businesses are not able to set the price of their good below 1. The graph for this simulation is shown in Figure 6.17

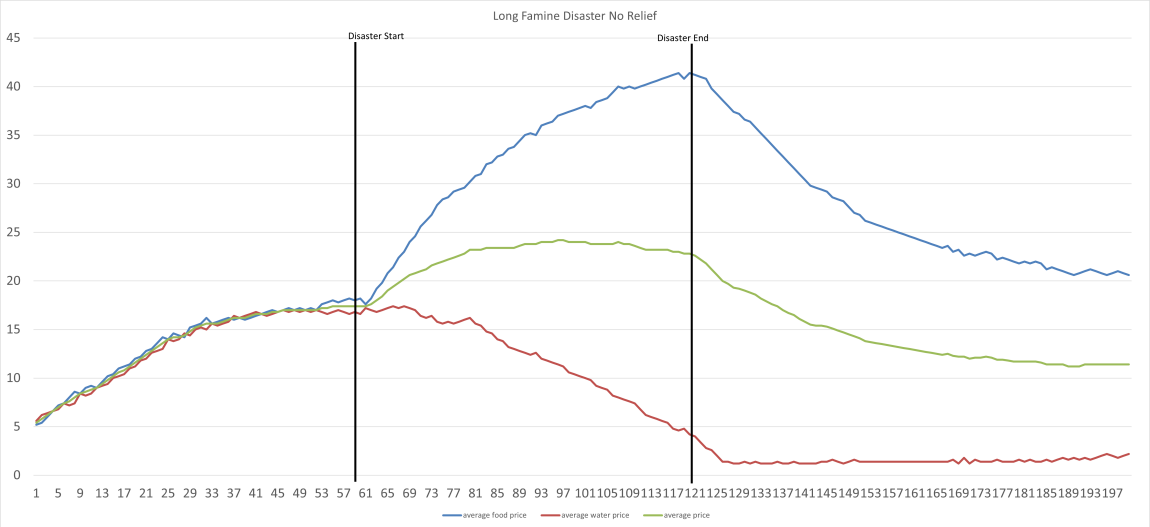

Figure 6.17: Long Famine Disaster No Relief

The long drought showed similar trends as the long famine. Once again, the trends from the original drought simulation are seen. When the disaster started, the price of water was 17.8. The price of water increased quickly after the disaster started. It continued to increase throughout the entire duration of the disaster, but more slowly as the disaster went on. When the disaster ended, the price of water was 40.2. After the disaster ended, the price of water quickly began decreasing. The price of water begins to flatten out as the simulation comes to an end, but it did not go on long enough to show if the price was truly flattening out. At the end of the simulation, the price of water was 20.6. The price of food was 17.4 when the disaster began. Over the course of the drought, the price of food decreased. The decrease

was slow at first but faster toward the end of the disaster. When the disaster ended, the price of food was 5, and the price began dropping much faster. Again, the price flattened out dramatically after it hit 1.2 and stayed between 1.2 and 1.8 from day 129 to day 184. After day 184, the price of food began increasing, but again, the simulation does not go on long enough to show how much of an increase would have happened or at what level the price would have leveled out given more days of the simulation. The price ended at a level of 3 after day 200. The graph for this simulation is shown in Figure 6.18

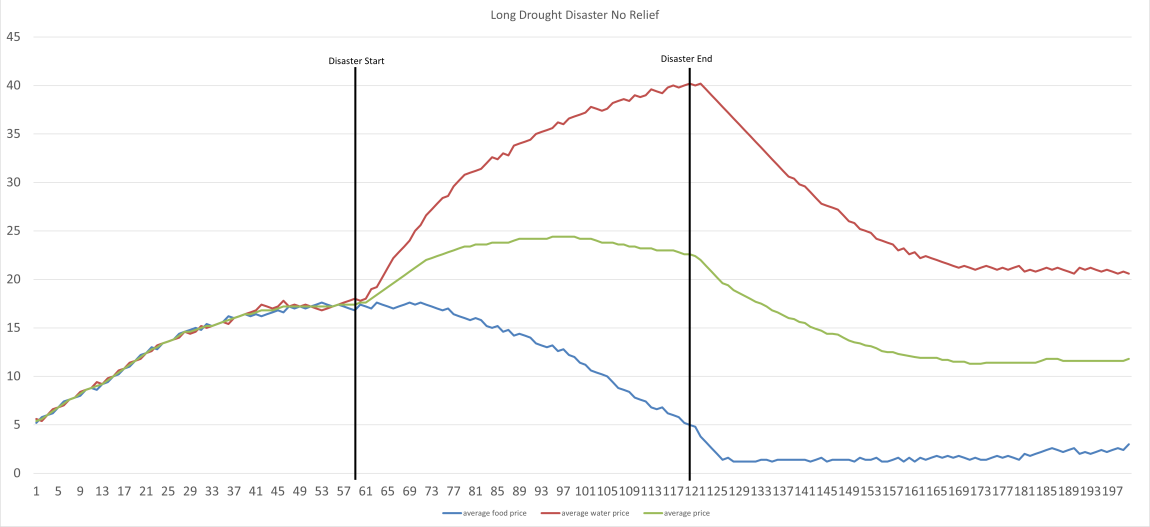

Figure 6.18: Long Drought Disaster No Relief

The long tornado simulation showed how high the prices will go when a tornado lasts for a long time. The average price of all goods was 17.2 when the disaster started. The prices increased quickly after the start of the disaster, and after day 96, began to flatten out. At day 96, the average price of all goods was 32.8. Between days 96 and 120, the average prices increased slightly, only reaching 34 by the end of the disaster period. Becasue of this, it is likely that if the disaster would have lasted even longer, the prices would not have gone much higher than 34. After the disaster ended, the prices quickly dropped at first and slowly toward the end of the

simulation. By day 185, the average price of all goods reached its lowest point, 17.4, and flattened out, staying between 17.4 and 17.8 for the rest of the simulation. The graph for this simulation is shown in Figure 6.19

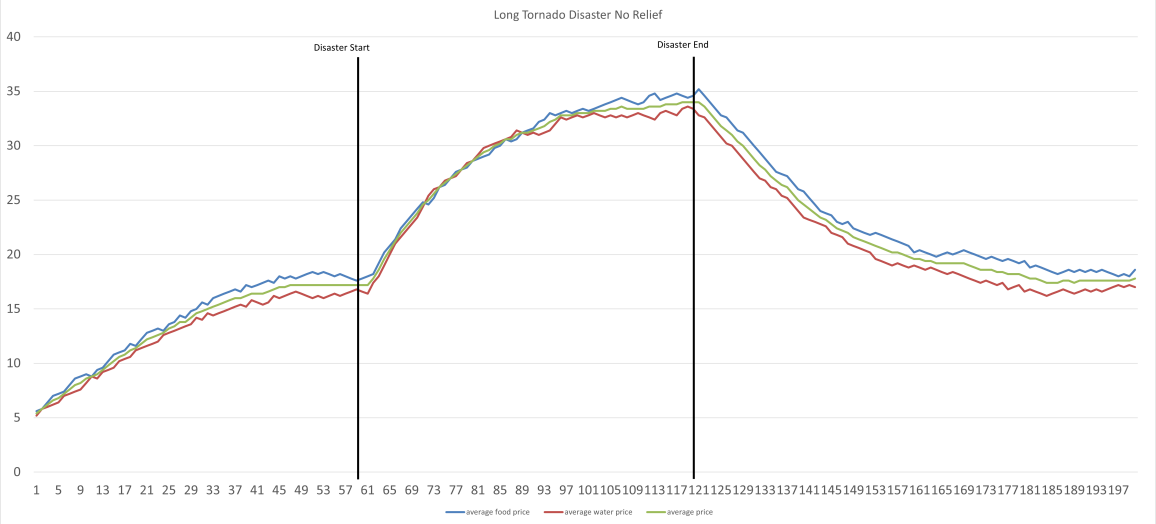

Figure 6.19: Long Tornado Disaster No Relief

6.3 Notable Observations

There are many questions that can be answered with the data that was collected from these 19 simulations. The first of these questions is why do prices of goods change? What causes prices to change? In this model, prices change on a small scale for each individual business at the end of every day cycle. If a *Business* sells all of its product, it will increase the price at that *Business*. If a *Business* does not sell all of its product, it will decrease its price. When looking at the charts in this chapter, the averages of all *Businesses* are seen. Any significant trends that are seen in these charts show a change due to economic forces that cause *Agents* to act in a certain way. For example, at the beginning of each simulation, the price of goods starts between 5 and 10. This is much lower than the equilibrium price of the goods, so at

the beginning of each simulation, the price increases until they reach around 17.4, which appears to be the equilibrium price for this simulation when not impacted by disaster or relief events. When the production for a good is decreased, this is a decrease in supply for that good. This causes the price of that good to increase. This is becasue there is less for the *Business* to sell, so they sell out of their product faster. Because the *Businesses* sell out for many days in a row, the price increases for many days in a row. This can also happen in the oppostie direction. When the supply of a good increases, such as when a disaster ends, there are many more products available. In order to be able to sell out of all the products that they have, *Businesses* must lower their prices. These changes can be seen in the graphs as the prices of products trend upward or downward.

These findings also help to establish the validity of the simulation within the constraints of the model Because the prices of goods change in a predictable manor due to supply changes, it can be determined that the model is working correctly and in line with the economic theory.

Supply changes can impact the price of related goods as well. The price change caused by the supply change also impacts the price of the related good. In this model, food and water are compliments to each other. For each unit of one good an *Agent* buys, they want to buy one unit of the other good. Because of this relationship, when the price of one good increases, the demand for the other good will decrease, meaning that the price for that good will also decrease. This effect is supported by economic theory and shown in the results of the model.

One additional question is why the prices of the two goods do not reconverge after a famine or drought push them in opposite directions. Because in all other scenarios, the prices stay relatively equal, it would make sense that when famine or drought events end, the prices would go back to being relatively similar as they were before the disaster. This is not the case because there are no economic forces

pushing the prices of these goods back together. In all other scenarios, the prices start at similar levels and are affected by the same forces. Because of this, the prices stay relatively the same throughout. However, when a famine or drought impacts the prices of goods, they separate. After the disaster ends, the prices have similar trends, but there is no force that pushes them back toward each other. The prices being the same in the other disasters is due to their simiar starting points and same economic forces being put on them.

Another idea that can be investigated by this data is the effectiveness of relief policies that a government could enact to combat the harsh effects of a disaster event. Based on the data collected by this study, it seems that monetary relief only works to cause inflation within the economy. Additionally, relief in the form of essential resources such as food or water do not help to combat the price changes caused by the disaster, but they do help to prevent people from running out of these resources, even if there is no benefit to this in the model.

CHAPTER 7

Conclusion

This project shows the potential impact of natural disasters on an economy. Using agent-based modeling techniques, and the Python library Pygame, the supply impact and price changes in goods produced in a small simulated economy were analyzed. It was found that when the production of a good is directly impacted by a disaster event, the market price of that good increases. When a disaster impacts two goods in an economy equally, the price of both goods increases at a similar rate and for the same amount of time. When the disaster ends, and the economy continues to produce at its previous production capabilities, the prices of goods return to the levels that were seen before the onset of the disaster. Additionally, any goods that are compliments to an impacted good see price decreases. When a disaster impacts only one good in the economy, goods that are compliments to that good see price decreases, especially after the disaster ends and people are able to spend their money on the restricted good. They become far less interested in buying the good that has been available to them the whole time. It was also found that government relief efforts to combat the impact of natural disasters can have unintended consequences or no effect at all. It is important that governments critically analyze a situation before action is taken in the wake of a natural disaster.

Overall, this project was successful in creating a working agent-based model of a

small economy. The trends that are seen in market prices of goods when faced with changes in supply and demand due to natural disasters is consistent with existing economic literature [2, 7].

7.1 Limitations

The simulation used in this project is clearly not truly indicative of economic processes that are seen in the complex world we live in. There are many ways in which the model used does not correlate well with the real world. Firstly, only businesses are impacted by the economic disasters, and they are only impacted in very specific ways. In reality, natural disasters come with many different types of damage. There are many different ways that a natural disaster might damage an economy that are not investigated in this project such as property damage or infrustructure damage that inhibits the use of roads or other means of transportation. Additionally, in this model, the impacts of natural disasters are distributed evenly. In the case of a real disaster, some homes or businesses will likely face much more damage or impact than others.

Another way that this model does not compare well to reality is the structure of the economy itself. In the model, there are only two different products. In reality, people have many different options of how to spend their money and what products to buy. In addition, in the real world, people are able to save their money and not choose to purchase anything if they don't want to. Because of the way economic actors make decisions in this model, people attempt to spend money on the product they have less of regardless of that products price.

In this model, there are no penalties or rewards for running out of food or water or amassing a certain amount. The people in this economy are simply trying to collect as much of each resource as they can with no end goal or penalty for failing.

In actuality, if a person were to run out of food or water, there would be serious health consequences. Additionally, if a person were able to purchase more food or water than they needed, they would begin to choose to buy other products, which is not possible in this simulation, as discussed.

Lastly, the change in production rates in this model happen instantly. One day, the businesses are producing at full capacity, and the next, they are reduced immediately to 60 percent productivity. In reality, in the wake of a disaster a business would likely shut down for several days in order to recover, then begin increasing productivity over a long period of time until full capacity is restored. This is dissimilar to the model where productivity is instantly reduced then restored over night.

7.2 Future Work

Because this model is imperfect, there is much more work that can be done to improve it. In order to make it more accurate to the real world, adding more product options for people to buy, and giving people the ability to not purchase any products if they do not want to would be a large step in the direction of building a model that can accurately reflect the world we live in and the economies that we participate in. Additionally to make the model more realistic, there should be penalties for having too little of a certain product. This would incentivize people to not run out of goods that they need. In the real world, we see health consequences and poverty traps associated with living without the resources needed to sustain a healthy life. This could be represented in the model to make it more accurate to our world.

One additional investigation would be to analyze inequality that is present within the model. Because some *Homes* are naturally further away from the *Businesses* than others, the *Agents* that live in these *Homes* are less likely to be able to purchase

products, especially if those products are impacted by natural disasters. Though the model implicitly includes inequality between *Agents*, it is not actively investigated. There is an economic measure of inequality known as the Gini Coefficient [2] that could be used with *Agent* data that is collected. The Gini Coefficient is used to describe income inequality present within a population. This value can be used to analyze inequality within the model and see if the disasters are more impactful to some *Agents* rather than others.

Finally, the simulation is uninteractive when it is running. It would be useful to see in real time how various aspects of the economy change as the simulation runs. Displaying relevant data and statistics about the simulation and highliting the current state of specific *Agents* or *Businesses* as the simulation progresses could help users understand the inner workings of the simulation and the economic forces at work a bit more.

References

1. Pygame documentation. URL https://www.pygame.org/docs. 16

2. *Principles of Economics*. University of Minnesota Libraries Publishing, 2016. 2, 8, 11, 69, 71

3. Supply and demand or price gouging? an ongoing debate. *Harvard Business School Online*, 2020. URL https://online.hbs.edu/blog/post/supply-and-demand-or-price-gouging-an-ongoing-debate. 15

4. U.S. Energy Information Administration. Weekly gulf coast regular all formations gasoline prices, 2022. data retrieved from U.S. Energy Information Administration, https://www.eia.gov/dnav/pet/hist/LeafHandler.ashx?n=PET&s=EMM_EPMR_PTE_R30_DPG&f=W. 14

5. Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *The Proceedings of the National Academy of Sciences*, 99, 2002. URL https://doi.org/10.1073/pnas.082080899. 3

6. Alberto Cavallo and Oleksiy Kryvtsov. How the pandemic has affected the economy, from empty shelves to higher prices. *PBS News Hour*, 2021. URL https://www.pbs.org/newshour/economy/how-the-pandemic-has-affected-the-economy-from-empty-shelves-to-higher-prices. 1

7. N. Gregory Mankiw. *Principles of economics*. South-Western College Pub, 4 edition, 2001. 2, 69

8. Marco Raberto, Silvano Cincotti, Sergio M. Focardi, and Michele Marchesi. Agent-based simulation of a financial market. *Physica A: Statistical Mechanics and its Applications*, 299:319–327, 2001. URL https://www.sciencedirect.com/science/article/pii/S0378437101003120. 3

9. Ruth Simon. Covid-19's toll on u.s. business? 200,000 extra closures in pandemic's first year. *The Wall Street Journal*, 2021. URL https://www.wsj.com/articles/covid-19s-toll-on-u-s-business-200-000-extra-closures-in-pandemics-first-year-1

10. USA Spending. The federal response of covid-19, 2022. URL https://www.usaspending.gov/disaster/covid-19?publicLaw=all. 1

11. Kaori Tembata. Natural disaster shocks and raw material prices in the steel industry. *Growth Mechanisms and Sustainability*, 2021. URL https://link.springer.com/chapter/10.1007/978-981-16-2486-5_5. 14

12. Leigh Tesfatsion. Agent-based computational economics. *Economics Department at The University of Iowa*, 2003. URL http://www2.econ.iastate.edu/tesfatsi/acewp1.pdf. 3

13. Martha C. White. Gas prices surged after hurricane katrina. but experts say ida will have limited impact at the pump. *NBC News*, 2021. URL https://www.nbcnews.com/business/business-news/gas-prices-surged-after-hurricane-katrina-experts-say-ida-will-n1278060. 14

14. Uri Wilensky and William Rand. *An Introduction to Agent-Based Modeling*. MIT Press, 2015. 3, 5