THE STUDY OF MICROEVOLUTION USING AGENT-BASED MODELING

Matt Fifer

Computer Systems Research Project

2004-2005

Abstract

The goal of the project is to create a program that uses an agent-environment structure to imitate a very simple natural ecosystem: one that includes a single type of species that can move, reproduce, kill, etc. The "organisms" will contain genomes (libraries of genetic data) that can be passed from parents to offspring in a way similar to that of animal reproduction in nature. As the agents interact with each other, the ones with the characteristics most favorable to survival in the artificial ecosystem will produce more children, and over time, the mean characteristics of the system should start to gravitate towards the traits that would be most beneficial. This process, the optimization of physical traits of a single species through passing on heritable advantageous genes, is known as microevolution.

Purpose

One of the most controversial topics in science today is the debate of creationism vs. Darwinism. Advocates for creationism believe that the world was created according to the description detailed in the 1st chapter of the book of Genesis in the Bible. The Earth is approximately 6,000 years old, and it was created by God, followed by the creation of animals and finally the creation of humans, Adam and Eve. Darwin and his followers believe that from the moment the universe was created, all the objects in that universe have been in competition. Everything¿from the organisms that make up the global population, to the cells that make up those organisms, to the molecules that make up those cells has beaten all of its competitors in the struggle for resources commonly known as life. This project will attempt to model the day-to-day war between organisms of the same species. Organisms, or agents, that can move, kill, and reproduce will be created and placed in an ecosystem. Each agent will include a genome that codes for its various characteristics. Organisms that are more successful at surviving or more successful at reproducing will pass their genes to their children, making future generations better suited to the environment. The competition will continue, generation after generation, until the simulation terminates. If evolution has occurred, the characteristics of the population at the end of the simulation should be markedly different than at the beginning.

Background

Two of the main goals of this project are the study of microevolution and the effects of biological mechanisms on this process. Meiosis, the formation of gametes, controls how genes are passed from parents to their offspring. In the first stage of meiosis, prophase I, the strands of DNA floating around the nucleus of the cell are wrapped around histone proteins to form chromosomes. Chromosomes are easier to work with than the strands of chromatin, as they are packaged tightly into an "X" structure (two ">"s connected at the centromere). In the second phase, metaphase I, chromosomes pair up along the equator of the cell, with homologous chromosomes being directly across from each other. (Homologous chromosomes code for the same traits, but come from different parents, and thus code for different versions of the same trait.) The pairs of chromosomes, called tetrads, are connected and exchange genetic material. This process, called crossing over, results in both of the chromosomes being a combination of genes from the mother and the father. Whole genes swap places, not individual nucleotides. In the third phase, anaphase I, fibers from within the cell pull the pair apart. When the pairs are pulled apart, the two chromosomes are put on either side of the cell. Each pair is split randomly, so for each pair, there are two possible outcomes. For instance, the paternal chromosome can either move to the left or right side of the cell, with the maternal chromosome moving to the opposite end. In telophase I, the two sides of the cell split into two individual cells. Thus, for each cell undergoing meiosis, there are 2n possible gametes. With crossing over, there are almost an infinite number of combinations of genes in the gametes. This large number of combinations is the reason for the genetic biodiversity that exists in the world today, even among species. For example, there are 6 billion humans on the planet, and none of them is exactly the same as another one.

Procedure

This project will be implemented with a matrix of agents. The matrix, initialized with only empty spaces, will be seeded with organisms by an Ecosystem class. Each agent in the matrix will have a genome, which will determine how it interacts with the Ecosystem. During every step of the simulation, an organism will have a choice whether to 1. do nothing 2. move to an empty adjacent space 3. kill an organism in a surrounding space, or 4. reproduce with an organism in an adjacent space. The likelihood of the organism performing any of these tasks is determined by the organism¿s personal variables, which will be coded for by the organism¿s genome. While the simulation is running, the average characteristics of the population will be measured. In theory, the mean value of each of the traits (speed, agility, strength, etc.) should either increase with time or gravitate towards a particular, optimum value.

Algorithms

At its most basic level, the program written to model microevolution is an agent-environment program. The agents, or members of the Organism class, contain a genome and have abilities that are dependent upon the genome. Here is the declaration of the Organism class:

```
class Organism
{
public:
        Organism();                         //constructors
        Organism(int ident, int row2, int col2);
        Organism(Nucleotide* mDNA, Nucleotide* dDNA, int ident, bool malefemale, int row2, int col2);
        ~Organism();            //destructor
        void printGenome();
```

```
		void meiosis(Nucleotide* gamete);
		Organism* reproduce(Organism* mate, int ident, int r, int c);
		int Interact(Organism* neighbors, int nlen);  //determines actions during simulation steps
		void ChangePos(int row2, int col2);
		int GeneValue(bool parent, int chromnum, int gennum);  //assigns a gene a numeric value
		int Laziness();				//accessor functions
		int Rage();
		int SexDrive();
		int Activity();
		int DeathRate();
		int ClausIndex();
		int Age();
		int Speed();
		int Row();
		int Col();
		int PIN();
		bool Interacted();
		bool Gender();
		void setPos(int row2, int col2);
		void setInteracted(bool interacted);
	private:
		void randSpawn(Nucleotide* DNA, int size);				//randomly generates a genome
		Nucleotide *mom, *dad;								//genome
		int ID, row, col, laziness, rage, sexdrive, activity, deathrate, clausindex, speed;	//personal characteristics
		double age;
		bool male, doneStuff;
	};
```

The agents are managed by the environment class, known as Ecosystem. The Ecosystem contains a matrix of Organisms. Here is the declaration of the Ecosystem class:

```
	class Ecosystem
	{
	public:
		Ecosystem();				//constructors
		Ecosystem(double oseed);
		~Ecosystem();				//destructor
		void Run(int steps);	//the simulation
		void printMap();
		void print(int r, int c);
		void surrSpaces(Organism* neighbors, int r, int c, int &friends);  //the neighbors of any cell
	private:
		Organism ** Population;  //the matrix of Organisms
	};
```

The simulation runs for a predetermined number of steps within the Ecosystem class. During every step of the simulation, the environment class cycles through the matrix of agents, telling each one to interact with its neighbors. To aid in the interaction, the environment sends the agent an array of the neighbors that it can affect. Once the agent has changed (or not changed) the array of neighbors, it sends the array back to the environment which then updates the matrix of agents. Here is the code for the Organisms function which enables it to interact with its neighbors:

```
	int Organism::Interact(Organism* neighbors, int nlen) //returns 0 if the organism hasn't moved & 1 if it has
	{
		fout << row << " " << col << " ";
		if(!ID)//This Organism is not an organism
		{
			fout << "Not an organism, cannot interact!" << endl;
			return 0;
		}
		if(doneStuff)//This Organism has already interacted once this step
		{
			fout << "This organism has already interacted!" << endl;
			return 0;
		}
		doneStuff = true;
		int loop;
		for(loop = 0; loop < GENES * CHROMOSOMES * GENE_LENGTH; loop++)
		{
			if(rand() % RATE_MAX < MUTATION_RATE)
				mom[loop] = (Nucleotide)(rand() % 4);
			if(rand() % RATE_MAX < MUTATION_RATE)
```

```
                    dad[loop] = (Nucleotide)(rand() % 4);
}
age += 1 / (double)speed;
int x, friends = 0, empties, temp, temp2, males = 0, females, r1, c1, action, idle, repro, kill, move, die, spaces[8];
for(x = 0; x < nlen; x++)//get stats for neighbors
{
          if(neighbors[x].PIN())
          {
                    friends++;
                    if(neighbors[x].male)
                              males++;
          }
}
females = friends - males;
empties = nlen - friends;
if(friends / nlen > clausindex / 8) //The Organism is too crowded
{
          fout << ID << " Suffocated" << endl;
          return 1;
}
//This should be where you determine what the likelihood of each action is, depending on the organism's characteristics
idle = laziness;
repro = sexdrive;
kill = rage;
move = activity;
die = deathrate;
if(!friends) //no organisms in surrounding space
{
          kill = 0;
}
if((male && !females) || (!male && !males)) //no organisms of opposite gender in surrounding space
{
          repro = 0;
}
if(!empties) //no vacant spaces
{
          repro = 0;
          move = 0;
}
action = rand() % (move + repro + kill + idle + die);
if(action < move) //move
{
          temp = 0;
          for(x = 0; x < nlen; x++)
          {
                    if(!neighbors[x].PIN())
                              spaces[temp++] = x;
          }
          temp = spaces[rand() % temp];

          r1 = neighbors[temp].Row();
          c1 = neighbors[temp].Col();
          neighbors[temp] = *(this);
          neighbors[temp].setPos(r1, c1);
          fout << "Moved into vacant space " << neighbors[temp].Row() << " " << neighbors[temp].Col() << endl;
          return 1;
}

else if(action < move + repro) //reproduce
{
          temp = 0;
          for(x = 0; x < nlen; x++)
          {
                    if(!neighbors[x].PIN())
                              spaces[temp++] = x;
          }
          temp = spaces[rand() % temp]; //temp is the location where the child is gonna be

          temp2 = 0;
          for(x = 0; x < nlen; x++)
          {
                    if(neighbors[x].PIN() && ((neighbors[x].Gender() && !male) || (!neighbors[x].Gender() && male)))
```

```
                        spaces[temp2++] = x;
                }
                temp2 = spaces[rand() % temp2]; //temp2 is the location of the father or mother that is not this organism

                neighbors[temp] = *(reproduce(&neighbors[temp2], ++ocount, neighbors[temp].Row(), neighbors[temp].Col
                        ()));

                fout << "Reproduced with " << neighbors[temp2].PIN() << " into space " << neighbors[temp].Row() << " " <<
                        neighbors[temp].Col() << endl;
                return 0;
        }

        else if(action < move + repro + kill) //kill
        {
                temp = 0;
                for(x = 0; x < nlen; x++)
                {
                        if(neighbors[x].PIN())
                                spaces[temp++] = x;
                }
                temp = spaces[rand() % temp]; //temp is the location of the victim;

                neighbors[temp] = *(new Organism(0, neighbors[temp].Row(), neighbors[temp].Col()));

                fout << "Killed Organism " << neighbors[temp].PIN() << " in space " << neighbors[temp].Row() << " " <<
                        neighbors[temp].Col() << endl;
                return 0;
        }

        else if(action < move + repro + kill + die) //die
        {
                fout << ID << " Died" << endl;
                return 1;
        }

        else //do nothing
        {
                fout << "Did Nothing" << endl;
                return 0;
        }
}
```

The Organisms, during any simulation step, can either move, kill a neighbor, remain idle, reproduce, or die. The fourth option, reproduction, is the most relevant to the project. As explained before, organisms that are better at reproducing or surviving will pass their genes to future generations. The most critical function in reproduction is the meiosis function, which determines what traits are passed down to offspring. The process is completely random, but an organism with a “good” gene has about a 50% chance of passing that gene on to its child. Here is the meiosis function, which determines what genes each organism sends to its offspring:

```
void Organism::meiosis(Nucleotide *gamete)
{
        int x, genect, chromct, crossover;
        Nucleotide * chromo = new Nucleotide[GENES * GENE_LENGTH], *chromo2 = new Nucleotide[GENES *
                GENE_LENGTH];
        Nucleotide * gene = new Nucleotide[GENE_LENGTH], *gene2 = new Nucleotide[GENE_LENGTH];
        for(chromct = 0; chromct < CHROMOSOMES; chromct++)          //cycle through chromosomes in genome
        {
                for(genect = 0; genect < GENES; genect++)                   //cycle through genes in chromosome
                {
                        crossover = rand() % RATE_MAX;
                        (crossover < CROSSOVER_RATE) ? crossover = 1 : crossover = 0; //crossover if 1
                        for(x = 0; x < GENE_LENGTH; x++)          //cycle through nucleotides in gene
                        {
                                if(crossover) //paternal into 'gene,' maternal into 'gene2'
                                {
                                        gene[x] = dad[chromct * GENES * GENE_LENGTH + genect *
                                                GENE_LENGTH + x];
                                        gene2[x] = mom[chromct * GENES * GENE_LENGTH + genect *
                                                GENE_LENGTH + x];
                                }
```

```
                                        else        //maternal into 'gene,' paternal into 'gene2'
                                        {
                                                    gene[x] = mom[chromct * GENES * GENE_LENGTH + genect *
                                                              GENE_LENGTH + x];
                                                    gene2[x] = dad[chromct * GENES * GENE_LENGTH + genect *
                                                              GENE_LENGTH + x];
                                        }
                                        chromo[genect * GENE_LENGTH + x] = gene[x];
                                        chromo2[genect * GENE_LENGTH + x] = gene2[x];
                              }
                    }
                    crossover = rand() % 2;          //choose either chromo or chromo2
                    for(x = 0; x < GENES * GENE_LENGTH; x++)  //fill gamete with either chromo or chromo2
                    {
                              if(crossover) gamete[(chromct - 1) * GENES * GENE_LENGTH + genect * GENE_LENGTH + x]
                                        = chromo[x];
                              else gamete[(chromct - 1) * GENES * GENE_LENGTH + genect * GENE_LENGTH + x]  =
                                        chromo2[x];
                              //cout << gamete[chromct * GENES * GENE_LENGTH + genect * GENE_LENGTH + x] <<
flush;
                    }
          }
          delete chromo;                  //cleanup
          delete chromo2;
          delete gene;
          delete gene2;
}
```

The functions and structures above are the most essential to the running of the program and the actual study of microevolution. At the end of each simulation step, the environment class records the statistics for the agents in the matrix and puts the numbers into a spreadsheet for analysis. The spreadsheet can be used to observe trends in the mean characteristics of the system over time.

Results

Using the spreadsheet created by the environment class, I was able to create charts that would help me analyze the evolution of the Organisms over the course of the simulation. The first time I ran the simulation, I set the program so that there was no mutation in the agent's genomes. Genes were strictly created at the outset of the program, and those genes were passed down to future generations. If microevolution were to take place, a gene that coded for a beneficial characteristic would have a higher chance of being passed down to a later generation. Without mutation, however, if one organism possessed a characteristic that was far superior to the comparable characteristics of other organisms, that gene should theoretically allow that organism to "dominate" the other organisms and pass its genetic material to many children, in effect exterminating the genes that code for less beneficial characteristics. For example, if an organism was created that had a 95% chance of reproducing in a given simulation step, it would quickly pass its genetic material to a lot of offspring, until its gene was the only one left coding for reproductive tendency, or libido.

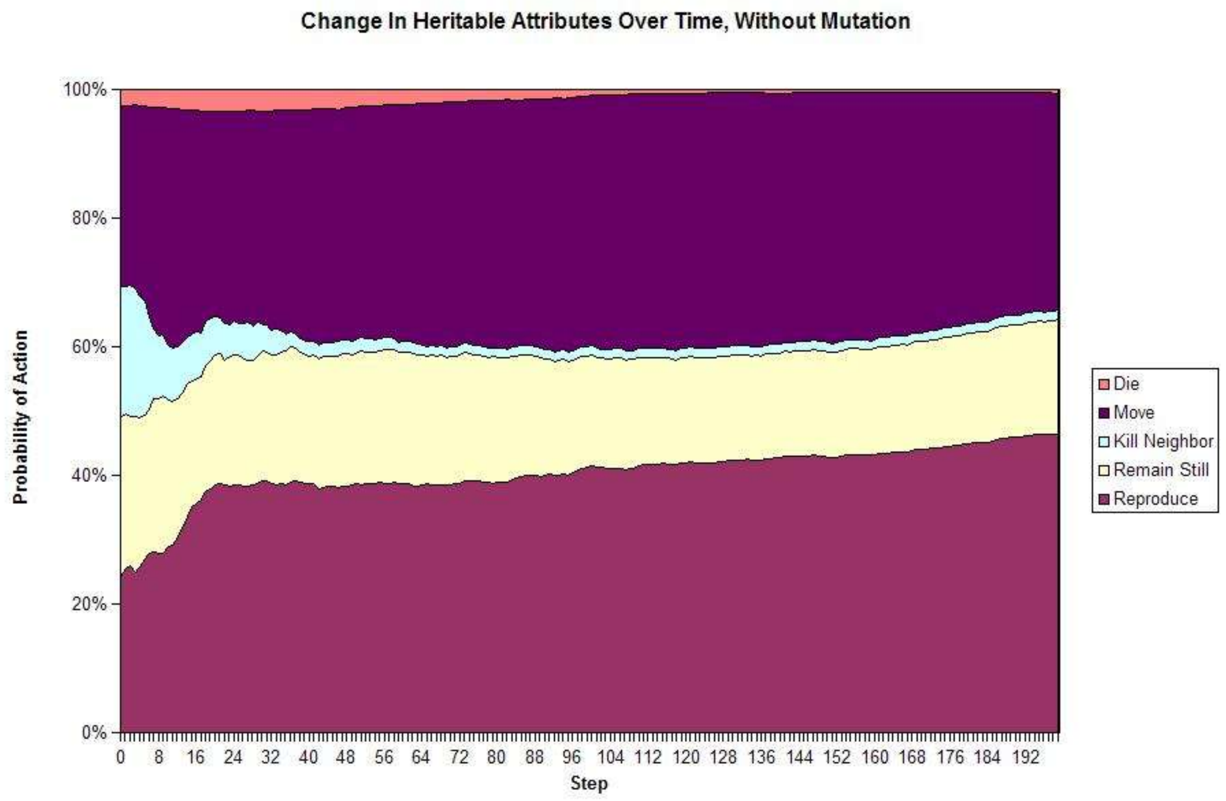

Figure 1

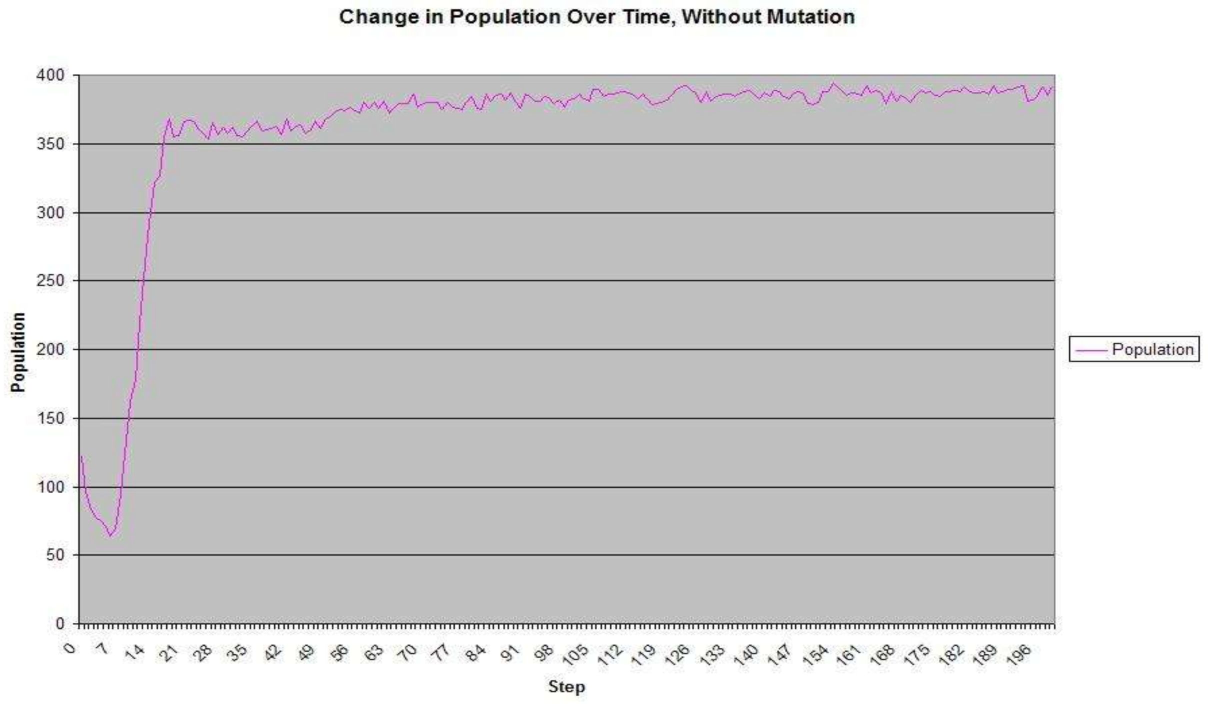

Figure 2

As you can see from Figure 1, the average tendency to reproduce increases during the simulation. The tendency to die decreases to almost nonexistence. The tendency to remain still, since it has relatively no effect on anything, stays almost constant. The tendency to move to adjacent spaces, thereby spreading one's genes throughout the ecosystem, increases to be almost as likely as reproduction. The tendency to kill one's neighbor decreases drastically, probably because it does not positively benefit the murdering organism. In Figure 2, we can see that the population seems to stabilize at about the same time as the average characteristics. This would suggest that there was a large amount of competition among the organisms early in the simulation, but the competition quieted down as one dominant set of genes took over the ecosystem.

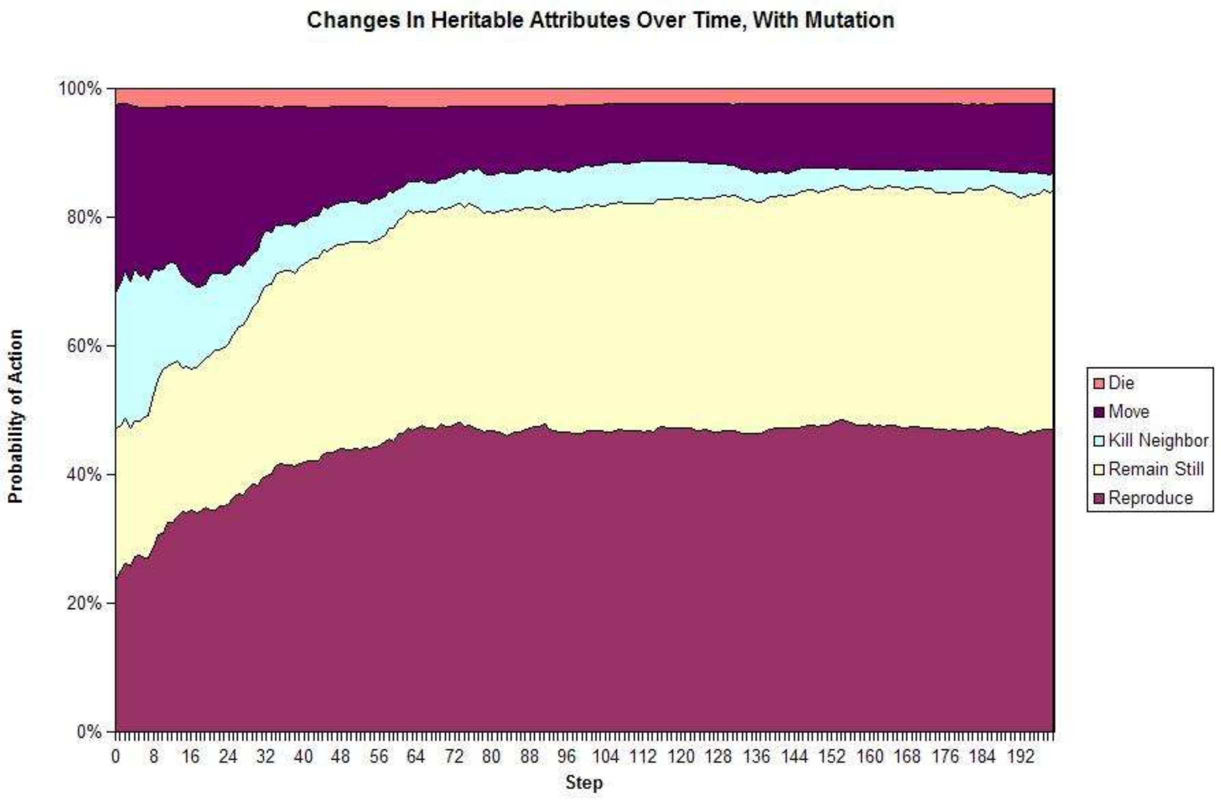

Figure 3

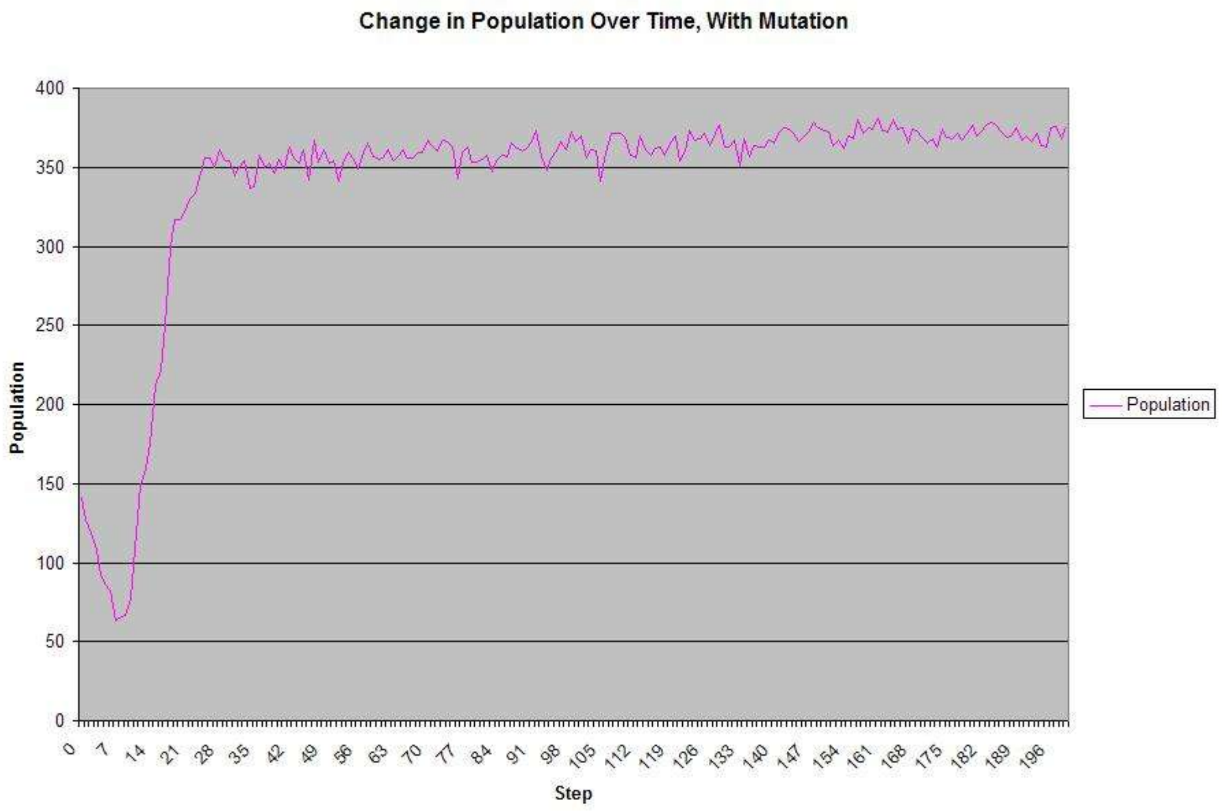

Figure 4

These figures show the results from the second run of the program, when mutation was turned on. As you can see, many of the same trends exist, with reproductive tendency skyrocketing and tendency to kill plummeting. Upon reevaluation, it seems that perhaps the tendencies to move and remain idle do not really affect an agent's ability survive, and thus their trends are more subject to fluctuations that occur in the beginning of the simulation. One thing to note about the mutation simulation is the larger degree of fluctuation in both characteristics and population. The population stabilizes at about the same number, but swings between simulation steps are more pronounced. In Figure 3, the stabilization that had occurred in Figure 1 is largely not present.

Conclusion

The goal of this project at the outset was to create a system that modeled trends and processes from the natural world, using the same mechanisms that occur in that natural world. While this project by no means definitively proves the correctness of Darwin's theory of evolution over the creationist theory, it demonstrates some of the basic principles that Darwin addressed in his book, The Origin of Species. Darwin addresses two distinct processes—natural selection and artificial selection. Artificial selection, or selective breeding, was not present in this project at all. There was no point in the program where the user was allowed to pick organisms that survived. Natural selection, though it is a stretch because nature was the inside of a computer, *was* simulated. Natural selection, described as the "survival of the fittest," is when an organism's characteristics enable it to survive and pass those traits to its offspring. In this program, "nature" was allowed to run its course, and at the end of the simulation, the organisms with the best combination of characteristics had triumphed over their predecessors. “Natural” selection occurred as predicted.

References

All of the information in this report was either taught last year in A.P. Biology last year and, to a small degree, Charles Darwin’s The Origin of Species. I created all of the code and all of the charts in this paper. For my next draft, I will be sure to include more outside information that I have found in the course of my research