1 The Blind Artist: Cooperative Evolutionary Art

Jeff Zia and Sam Mateosian

1.1 Introduction

Genetic programming is frequently used to "evolve" formulas which are too complex to design by any one human mind. Several individuals applied these same methods to the problem of creating visual images. The application of the process of evolution to the creation of art has generated incredible results. We hope to build upon the work of these individuals to extend the field of evolutionary art.

1.2 Basic Idea

We will develop a system by which images based on a genetic code will mutate and mate and by the force of human selection, evolve. Though this is not a new idea, it will serve as the foundation for our work.

The second portion of our work will be to create a networked environment in which many human participants provide the force of selection by which a community of individual images evolves. It is this cooperative aspect which we hope will provide new results and consequently room for further inquiry into the field of evolutionary art.

1.3 Scientific Inquiry

By simulating natural selection based on aesthetics, it is possible to reach a level of sophistication that seems to imply conscious design- this has been done before. However, in past experiments a single artist played the role of an animal breeder, or a God of sorts in that he has had the sole active guiding hand in the process. The artist has preconceived ideas of what he intends to create and consequently guides the images in that direction. What would happen, then, if we remove this guiding force? We plan to find out if it is still possible to generate art of a comparable level of sophistication.

We intend to use the input of many human commentators to determine the aesthetic fitness of individuals who belong to a community common to all participants. In doing so, no one individual participant can control the evolutionary direction of the community of images as a whole.

Because individual images now need to stand up to multiple factors of selection we expect that the resulting community of images will be more universally pleasing than the results of our predecessors.

1.4 Goals

1. By allowing multiple commentators, and thus represent multiple forces of selection, we hope to create a system of evolutionary art which more closely corresponds to the natural process of selection.

2. Learn a great deal more about artificial evolution, network interactions, and image manipulation.

3. Simply create art. As an additional test of aesthetic fitness, we will try to submit some of our generated results to the student art show and see if it could stand up to a more formal critique of "art."

1.5 Creating the Genetic Code

One of the primary decisions we had to make was how we were going to represent genetic code which could describe a visual image. Essentially there are two basic approaches to the problem. One involves describing a parameter set in which each parameter controls some aspect of the image being described. In this case the genetic code is simply the list of parameter values. To "grow" an image we simply take those values and plug them into a predetermined function.

The alternate approach involves genetic code which describes an image by including the function as well as the parameters. In this case we have a library of available operations which can be combined in any sequence. Each operation takes one or more variables, and in some cases other operations, as parameters. Once we have a symbolic expression of this sort it is given to an interpreter which then generates the resulting image. We chose to implement the second approach simply because it allows for an infinite number of possible combinations. There are only a finite number of combinations for any parameter set of static length. This, we felt, was too limiting. With symbolic expressions the possibilities are endless. To aid the evolution of this genetic material we will implement a number of methods by which these symbolic expressions can mutate and even mate. The exact details are too lengthy to go into at this point. For those interested in more specifics, we are essentially implementing the technique Karl Sims describes in his 1991 SIGGRAPH paper, "Artificial Evolution for Computer Graphics" (Computer Graphics, Volume 25, Number 4)

1.6 Implementation

We have chosen (with some amount of certainty) to use Java as the development platform for this project. The main drawback to this choice is the lack of built in imaging capabilities a more sophisticated language may have. That is not to say that it is not possible as we have seen some stunning examples of complex graphic effects implemented in Java. The advantages gained in distributability and platform independence outweigh the negative aspects.

1.7 Programming Tasks

One of the first problems we must solve is that of writing the actual code which does the work of all the various operations involved in the genetic code. If one operation is, for example, blur(Image img, int blurAmount), we must write the equivalent Java code which performs that operation on a given bitmap. Much of this work already exists in some form or another, hopefully this step will mostly entail commandeering preexisting code.

Additionally we intend to implement Sims' mechanisms for the generation, mutation of mating of symbolic expressions ourselves. Though, as new-comers to genetic programming, it will most likely prove highly challenging and time consuming, the learning experience will be a lucrative one.

The other major technical challenge lies in the creation of the networked environment in which we take the input of multiple users and apply it to a single common community of images.

The solution we have decided upon is based on the standard client/server model. On the server machine, we maintain a database of all individuals currently alive. When a user connects using our distributed client applet, the applet accesses the database and selects several individuals at random. The selected individuals are displayed by the client on the users screen at which time the user makes decisions as to the aesthetic fitness of the given images. The results of those decisions are then relayed back to the server, those images who receive enough votes against them die, thus the power of selection is exerted.

All mating and genetic mutations occur on the server. Mating preference is given to those individuals deemed more fit, a property which is determined again through user input. Additionally the server keeps track of the age of all individuals. Individuals of a certain age are destroyed in order to create the effect of natural death and maintain a stable population size.

The client/server solution is particularly well suited to this problem. It require few resources as the server need only maintain the genetic code for each individual while all image processing is done by the client applet. It will be fast as all interactions require only a very small bandwidth. The server is only sending a small text string to transfer the genetic code and is only receiving a minimal amount of input from any given client. Furthermore it is efficient in that it shares processing with the client. The client applet doesn't have to waste time generating the complex symbolic expressions, the server doesn't have to worry about generating, displaying or saving actual image files. All in all, a seemingly elegant solution.

In general the technical requirements of this project are great. One of the most interesting characteristics of this project is in the fact that it requires learning skills in so many different technical areas, from genetic programming, to mathematical image manipulation, to networking, to user interface design.

1.8 Testing

Once the program is in place, we will set up at least 3 testing environments for comparison.

One such environment will be contained, timed and observed. We will make use of at least one "control group", which will be hand picked based on some common feature (ex. Major, interest...etc.) and will participate in the process of evolving art for a set amount of time. The results of this trial will be compared with the results of other trials (again involving participants chosen by hand or at random) and with the results of trials performed in different environments. A second environment will be one void of any human participant. Because mating and mutating occur independent of user input (though it then lacks the discretion provided by selection, i.e. all individuals are equally likely to mate) we can simply allow the program to run for a given amount of time and then display the resulting community of images. A third environment is one which the client applet will be set up on several computers across campus, and any one at any given time could sit down and interact with the community of images.

1.9 Expected Results

We speculate that in grouping by common interests a pattern emerges from the final collection of images produced by that interest group. By running more trials using more hand-picked groups we hope to see separate and distinct patterns emerging from each group. At the same time we speculate that there will be visual elements common to all groups, representing what Steven Rooke refers to as an "archetype camera."

The unassisted run should get fairly random results, producing images which lack any properties of conscious design. Also, without a user to prune the path of growth, this process might be too slow to bring about change. That which we suspect will produce the most sophisticated images of all is the open-ended case, which allowed anyone to sit and participate. Since each image will have to survive through changing demands as many different users try the program. We will look for continuity among the end results of all trials. This way we would be able to determine if the elements of visual image which people find aesthetically pleasing do have a converging point.

1.10 Obstacles

Visual sophistication is a very difficult concept to define. As something could be very complex, but not very sophisticated (as a grid of squares on graphing paper). At the same time, an image could be a very simple form, but be composed in sophisticated manner (juxtaposition for example). How can we possible measure a "level of sophistication"?

Furthermore it is not simply the great hand-eye coordination of an artist which allows him or her to create art. We presuppose an intelligent designer when viewing a work of art, we try to decipher his or her "meaning" or "intent." Is it possible for art to exist for which there is no such designer?

These questions and many more will challenge our ability to draw conclusive results from such an experiment. This is not at all to say such an inquiry would be futile, it should provide numerous insights into human conceptions of art and what is visually pleasing. What will likely arise is a fertile ground for further inquiry.

1.11 Conclusion

It is our hope that through the creation of the system we have described we will open up pathways for further exploration of evolutionary art. We see the possibility for a great deal to be learned about the nature of those forms to which all of mankind seem drawn. There is also a good deal to be learned about the nature of evolutionary processes to take on the appearance of consciousness ones. Which gets us to the real point of this conclusion. From all of this, from the aspects of programming to the process of scientific inquiry, we hope simply to learn... *something*. We exist in a world which is constantly evolving, in order to survive we must be able to change ourselves, we must move and adapt... *we must learn*.

2 Ecological modeling

Jenn Nelson

2.1 Background

Interactions among organisms are of primary importance in theoretical ecology, particularly in community ecology. A community is basically a group of interacting populations of organisms. How populations interact determines how they evolve. A given population interacts with many others. The relative strength of these interactions therefore determines the relative influence on the population. However, populations need not interact directly to affect each other.

We may consider an indirect interaction between two organisms is as a chain of direct interactions. It is clear that in order for there to be an indirect interaction, groups of species (communities) must contain at least 3 species. For example, given two competing species of plants, a grass and a wildflower. The plants affect each other directly by diminishing the growth of the other. Assume now there is a cow that feeds only on the grass, and does not interact with the wildflower in any way. The cow can still affect the growth of the wildflower by feeding on the grass. By decreasing the amount of grass, the cow can indirectly help the wildflower. Thus the cow indirectly interacts with the wildflower through the direct interaction of eating the grass, and the competition between the grass and the wildflower.

There are several ways to quantify the direct interactions among groups of organisms, but it is much harder to identify and quantify the indirect interactions. Direct interaction strengths are summarized in what are called community matrices. A community matrix is matrix in which the entries represent the effect of the column species on the growth of the row species. Using the premise that an indirect interaction is a chain of linked direct interactions, we used these matrices to quantify indirect interaction strengths.

The relative importance of indirect interactions as compared to direct interactions in community dynamics is currently unknown. The purpose of this project is to analyze how the strength of the interaction chains varies among groups of species. It is possible for these strengths of these indirect interaction chains to vary with the size of the community considered and the length of chain considered. The length of the interaction chain is the number of direct interactions within the chain. Thus this project looks at how the strength of the indirect interactions vary.

2.2 Previous Work

This project is an extension of research done at Kansas State University during the summer of 1998, currently in preparation for publication. The previous work included the creation of an extremely crude program to process the matrices. The program sufficed to generate the data, however, in order make it easier to use (for anyone other than me) it will be necessary to redesign the interface.

In the original implementation of the program the chain strength was found by determining the minimal direct interaction strength within the interaction chain. Community matrices from many different literature studies were used as the data set for this analysis. Thus to facilitate the comparisons among the different studies, it was necessary to standardize the matrices prior to analysis. The current method was a simple re-centering of the data. As this is a controversial procedure, the method used to standardize the matrices is currently being reevaluated. An additional component of the investigation looked at the normality of the distribution of the interaction strengths. The analysis involved testing the strengths using Kolmogorov-Smirnov Goodness of Fit test.

2.3 Proposed Project

The first part of this project involves a simple alteration of the program to alter the method of computing the chain strength. As mentioned, the program currently determines the minimal chain strength. This is a very conservative estimate, as the strength of the influence most likely becomes diluted through the direct interactions of the chain. Thus the proposed change is to compute not the minimum of the direct interaction strengths, but rather the product. The second change to the underlying theory of the project is a possible change in the method of standardizing the matrices.

The original program was written very basically and crudely. It will be necessary to design and implement a workable user interface. The program needs to be able to output the data to a text file in order to move the information more easily to a spreadsheet. The program also needs an easier method to input the matrices to the program. This may involve a console window and most likely some form of a Graphical User Interface.

Another facet of this project is visualization of data. The output of the program is currently a very, very long list of numbers, followed by a few calculation results, such as the mean and variance, repeated several times. This obviously can be improved to facilitate analyzing the data, and navigating through the vast quantities of numerical output.

One of the questions looked at in the original manuscript/investigation of the project is the normality of the direct interaction strengths. To do this "boilerplate" Goodness of Fit tests of normality were used. An alternate approach, which may be utilized on an experimental basis, is instead performing a transformation on the data and then graphing it, to visually check the normality of the strengths. As part of this project, this alternate method will be tested.

The current weak point of the project is the possible lack of an empirically testable aspect of the project. It may be possible to compare what happens as species levels change in a very small community (perhaps only 3 species). The change in species levels could be determined by considering only direct interactions and then considering indirect interactions in conjunction with direct interactions. Then comparing the differences in the predictions of population level changes, however it may be hard to ascertain which model is more accurate due to lack of experimental data.

Thus, despite the potential difficulties of empirical testing of the model, this project has definite potential for software development, in both the reprogramming the methods of calculating the chain strengths, and designing a workable interface and file output system. The project also includes questions about how to manipulate a very large data set, and ways to display the vast quantities of data generated by the program.

2.4 Tentative Schedule

February 25: By this week, using JDK for the Macintosh, the fundamental changes to the program will be implemented. These fundamental changes are to the method used to determine the chain strength and to the method used to standardize the matrices. The community matrices will be run through the altered program to get a quick idea of how the changes affect the outcome of the data.

March 4: Conversion to CodeWarrior 4.0 on my PC. Work on redesigning the program so that data is output to a file.

March 11: Design a user interface.

March 18: Try to develop a more general (recursive?) permutation algorithm for the program.

April 1: Buffer week to make up for underestimating the amount of time necessary to complete previous weeks' goals. Review scientific literature on the subject.

April 8: Analysis of results and translation into English.

April 15: Normality tests of data.

April 22: Wrap-up.

April 29: Prepare poster and presentation of project.

May 6: Continuation of previous and actual presentation.

3 Evolution Modeling in a Java Development Environment

Michael Liedtke

3.1 Purpose

To test DNA mutation, replication, and environmental interaction to search for the mechanisms of speciation.

3.2 Scientific Inquiry

Is speciation driven by mutation? Is the diversity of life on Earth due to changes in genetic code? To what degree does the environment select genetic code? What effect does isolation have on natural selection? Why do some places with similar climates have widely divergent numbers of species? I am looking to gain a better understanding of how Earth came to have such a great bio-diversity by creating my own environment of artificial life and watching it grow and possibly evolve.

3.3 Programming Element

The research into the scientific questions raised in this proposal will be performed in a Java development environment. I plan on developing a base set of strings of genetic code which will be given guidelines to mutate. Theses rules of mutation will be derived from reality - the actual methods of mutation that scientists have observed in the laboratory. From this point, the strings of genetic code, DNA, will begin to replicate and mutate. They will be interacting with an environment - which will be set at levels ranging from static to highly dynamic - and evolving into new strings of DNA. All the mutation, replication, interaction, and results will be programming in and derived from the Java.

A graphic interface could be a didactic tool. Users could adjust the variables to see how the interactions of organisms propagate over lifecycles, and possibly watch evolution in action. However, this is a secondary consideration to the research, and will only be developed if the results of the experiments warrant it.

3.4 Goals

To learn more about evolutionary theory and seek possible concrete explanations for the diversity of life. To practice programming - working on a semester long project alone will help me improve my programming skills. The inherent interest in the project will motivate me to program and to discover new areas of programming.

If we can find mathematical relationships between the number of environmental factors and the number of species, we may be able to establish a method for cataloguing a species rich environment. For example, there are organizations which go into endangered land and do species counts. If a certain number is met, they declare it a global treasure and can often influence governments to help establish preserves. This is mere speculation, but it posits that we could arrive at better estimates of the number of species in a given area or the world, and help raise awareness to the apparent accelerated loss of species world-wide. Current estimates range over 100 million species; only 1.75 million species have been catalogued.

3.5 Specifics of Project

To commence the programming, I will create my genetic material, my genes. These will be represented by strings of the characters C, G, A, and T, the nucleotides bases of DNA. The specific sequences will be selectly randomly. These will replicate and mutate according to specifications gleaned from nature.

For the environment, I will create similar sequences. For the unchanging environment, these will be defined arbitrarily. For the randonly flucuating environment, they will change randomly. And for the interacting environment, they will be based on the current conditions of the actual genes. It may seem at first as if the genes and environment are virtually identical. However, the genes will be mutating as genes do, and the environment will not be mutating. It will be modelled after the actual behavior of actual environments as closely as possible. As for the interaction, this will be established by aligning the environmental sequences with the genes. If there is a match of environmental code found within a gene, then the property of that environmental condition will be applied to the genes which contain that sequence. Experiments will be run with the environment in the different states listed above, and comparisons will be made between the different tests run in the different environments.

These experiments will hopefully yield results which tell us something interesting about evolution, especially speciation. An example of an interesting result would be if we found clustering of genes into groups. Measuring speciation will be difficult. First, I must define what speciation means. My current guess is that I will look for groupings of similar sequences. If clustering is a result, and that clustering continues more and more - especially as the environment flucuates more - then we will have observed something which is analogous to speciation. Controlling the experiment too much is a big problem, so I will try to find the appropriate amount of randomness (usually more than one would guess at first glance). By confining my influence to local parameters, and if we get global patterns out of these local interactions, then we may have stumbled upon something really fascinating.

3.6 Calendar

I did a lot of preliminary research into evolution during January and February and I'm now planning out what I am going to program and how I will program this. The next week or two will consist of the final planning stages before the programming actually begins.

February: Finalize plans, begin programming. Set up environment of DNA strings.

March: Design mutation, replication, and interaction algorithms. Create environment that DNA lives in and the environmental factors. Then start the cycle of life. Let the DNA replicate over many generations. Let them mutate.

April: Bring everything together. Run the actual experiments and record data. Design output to print to graphs. Come up with data. Write reports.

May: Finish report. Make poster. Go mountain biking.

4 Distributed Mathematical Computation

Peter Aykroyd and Dakila Clark

4.1 The Grid

Our project is based on a model of distributed computing often referred to as the grid. The commonly used metaphor for a computing grid is the a power grid. In a power grid, every person connected may not only consume power but is also able to sell power to the electric company by producing there own. Similarly, in a computing grid, each user attached to the system may use the resources of other computers and also may offer his own computer for use. In this way, when his machine is not in use locally it can be used by other grid consumers. Therefore, under such a model the computing resources of the world could be maximized.

Obviously, our project will not address the global scale that implementing such a grid would require. Or the social factors that will be necessary to bring the grid to pass. However, the software tools we develop will have some grid-like qualities in that the computers with our clients installed will use offer their idle CPU time to the project server which will then assign them work to be done.

4.2 Software Tools Development

Using the DCOM package available in Windows NT which provides distributed computing functionality, we will write both a client and a server which will be able to exchange information. The specific use of this in relation to our cooperative work with Rich Wolski at UCSD to find Ramsey numbers is that a server will be able to give the client a range of numbers to search. The client will then report back to the server to indicate whether or not a Ramsey number was found.

An important attribute of our software must be politeness with regards to the computerUs user. Therefore, it would not be appropriate for the client to search its range of numbers while the computer is in use. To this end, the client will only be active when it detects that the system is no longer in use. This is of the essence because users must be convinced that if they cooperate with the project by installing a client that it will not hamper than own work. Similarly, the client must be secure. No sane user would want to install software that could potentially put their own files at risk of being accessed due to a security bug. Therefore, we will not only make sure that both the server and client are secure but document it for reassurance of potential users. It is also our desire that the client and server be extendable. This is desirable so that we will be able to use our software to search for more than just Ramsey numbers. Indeed, software tools such as this have application with many scientific endeavors. Our implementation may either take advantage of a plug-in architecture which would allow the compiled client to learn what algorithm to perform on a given data set from it. Or, alternatively, may simply be written with a clear separation between the distributed computing tools and the code necessary for the specific scientific inquiry. Finally, in relation to our Ramsey numbers work, our client must be able to communicate with the UNIX servers running at UCSD which our the heart of that operation. This compatibility may or may not induce to code our own server with similar protocols to those of UCSD server software.

4.3 Ramsey Numbers

Ramsey numbers is really just an extension of the "pigeonhole principle". The nth Ramsey number Rn is defined as the smallest number k such that all complete bicolored graphs on k nodes possess a monocolored complete subgraph on n nodes. For example, it has already been proven that R3 = 6. To put this is into context, R3 = 6 is the smallest number of people that need to be at a party in order for 3 of the 6 to know each other, or 3 of the six to not know each other. We know this because, using graph theory, we can construct a graph with 6 vertices representing the 6 people and we can construct 3 red triangles and 3 blue triangles out of the 6 vertices. The only question now is whether or not we can do the same thing with less than 6 people. We notice that R3=5 doesn't work because we cannot construct 3 red triangles and 3 blue triangles. Hence we've just shown that 5 ¡ R3 ¡= 6. Therefore, R3 = 6.

Currently, the first four Ramsey numbers have been proven. The proofs for the others however have eluded mathematicians. For this project, we are concentrating our efforts on finding the 6th Ramsey number, which is known to be somewhere between 102 and 165. It's not feasible to search through all possible bicolored graphs for 102 nodes. However, by finding bicolored complete graphs having no monocolored complete subgraphs of 6 nodes, we can eliminate the existing lower bound and find new lower bounds, thus converging on the value of R6.

4.4 Future Exploration and Inquiry

Now the implications of finding Ramsey numbers through distributed computer resources appear to make our project extendable to other possible branches of combinatorial mathematics or number theory. For example, we can use distributed computer resources in finding primes (special primes: Mersenne and Twin), perfect numbers, or magic squares, although these are trivial, hackneyed exercises. Another possible venture for our project is to explore certain cryptology systems which are very closely related to number theory.

5 Lake modeling

Sean Hogan and Ryan Waller

5.1 Introduction

Our project is to develop a software tool to model the amount of phosphorous in the lakes in central Maine. According to Elaine Tiejen, author of the China Lake Syndrome the element phosphorous is the primary limiting factor determining the rate of plant growth in most Maine lakes. Phosphorous is a naturally occurring element and can be found in rocks and rainwater in low concentrations. However, much greater concentrations are found in eroded soil, road dust, fertilizers, septic wastes, and detergents. Tiejen states that phosphorous usually accounts for roughly one per five hundred parts of a typical freshwater plantUs total mass. A small increase in phosphorous can stimulate massive plant growth, usually in the form of algae blooms. Thus, a working software model of this process will be valuable to environmental scientists because they will have access to unlimited simulations. In addition, the software will allow the scientists to test the integrity of their model. In short, anomalies in the model will either provoke its revision or lead to new discovery.

The mathematical model we will be using was designed by Rockhall and Chupra, and was provided by Professor David Firmage. Professor Firmage implemented Rockhall and Chupra's model in the form of an Excel spreadsheet, but found that it was not an ideal implementation. Typically, the students in

Biology 493 use the model for their semester analyses of a nearby lake. The results of their analyses are then published and distributed to local government agencies, as well as state agencies involved with environmental protection. The new implementation we develop will hopefully be a vast improvement over the previous one. In addition, we hope the biology 493 students will have a more powerful tool to test results and run simulations with. The actual model we will be using entails coefficients and variables that will be accessible to the user to manipulate, but the actual equations of the model will be hidden from the user. All coefficients and variables will need to be checked for erroneous entries, such as negative areas or out of bound entries. Coefficients: There are approximately seventeen coefficients that are considered in professor Firmage's model, some include forested land, logged land, and roads near by. These coefficients have upper and lower bounds that will have to be determined individually by the user or entered in as a default setting.

Areas: The areas of the coefficient variables will need to be entered by the user some include the volume of the water inflow and the area of the lake surface. The areas are used to determine the per capita in years for the shoreline for seasonal use and year round use. The per capita measurement is determined by the calculation

```
(person/unit) *
(#of days (seasonal or year round)/365) *
(units seasonal or units year round)
```

The per capita is then used to calculate the area P loading with the calculation from the total mass loading.

Mass: the area multiplied by the coefficients determines the calculations of the individual masses. Once the mass calculations have been formulated then the high and low percentages of phosphorous content can be determined for each coefficient. Then the phosphorous content of the lake can be determined within a upper and lower bound.

We plan on implementing our project in the form of a Java applet. There are a number of reasons why we choose the Java programming language. First, the software can be stored and compiled on the web, which promotes accessibility for Colby students and faculty, as well as anyone interested in this subject matter outside of Mayflower Hill. Secondly, Java has an extensive graphics class that will enable us to create a graphics intense project. Third, Java offers a variety of powerful features that will allow us to produce the best software tool possible. Our project will present us with many design considerations over the course of the semester. Our first consideration is to create a fast and user friendly environment in which parameters can be changed easily, while being displayed or tracked. Other considerations are plotting multiple graphs or just single graphs. This project must posses all of the necessary attributes of a good software tool. It must be robust, readable, re-useable, and functional.

6 Macroeconomic Model Simulation

Eric Kostrowski

6.1 Introduction

During the time of the Great Depression John Keynes developed the greatest breakthrough in the field of macroeconomics, a model of the economy that logically explained the causal relationships between the major macro-variables. The basic model explains the workings of the goods and services market using GDP. It also describes the relationship between the financial markets and the goods market. Many economists have worked to extend the new Keynesian model to further detail. The extensions often occurred in order to explain recent economic activity that wasn't fully modeled in the Keynesian model.

This project will encompass the development of simulation for the new Keynesian model of the macroeconomy. The project will have the unique aspect of being able to accept extensions to the basic model in the form of "plug-ins." These "plug-ins," written by advanced users, will enable the application to handle more complex models. The application will use the basic model, and the addition of the option "plug-ins" to provide a GUI to the common user, by which they may simulate the economy.

6.2 The Basic Model

Major Variables:

Goods and Services Market:

Z : Demand for Goods and Services

C : Personal Consumption Expenditures

I : Gross Private Investment

G : Government Expenditures

T : Taxes

Y : Production (= Income)

Financial Markets:

H : Monetary Base

CU : Currency

R : Reserves Held by Banks

D : Checkable Deposits

i : Interest Rate

P : Price Level

M : Supply of Money

The basic will not deal with a dynamics of the model, i.e. it will assume that all reactions within the model occur instantaneously, and thus no inventories are held by firms. Also note that the basic economy is assumed to be closed, no imports or exports occur. These assumptions will be laxed within the model with through the use of "plug-ins."

6.3 Data Source

All variables above, and hundreds of other related ones, are available from the Bureau of Economic Analysis in their report, "Survey of Current Business." They are online, and provide this report and others at `http://www.bea.doc.gov/`

6.4 The Main Application

The main application will use a GUI to present the common user with a convenient interface to the model simulation. The user will first set up the model. They will do this by selecting the optional "plug-ins" that they wish to have in order to provide a level of detail specific to their purpose. Next they will select the starting data for the project. They will have the option of choosing from a number of historical data or of entering values in on their own.

Once the model is set up the user may put the simulation into action. The simulation will increment on a user defined time period (typically one quarter), calculating all aspects of the model at each step. The user can view any number of graphs from a selection of popular graphs (defined in the "plug-ins") or choose a custom graph. The application will have the capability of displaying a variety of different graph forms, including line, bar, and pie graphs.

The user can examine how the economy reacts to certain events by introducing certain scenarios into the simulation. These scenarios are defined within the "plug-in" system. The simulation sets a start point for each scenario as initiated by the user. The results of the scenario, the effect on every variable or aspect of the model can be examined on a dynamic basis.

6.5 The Plug-in System

The plug-in system represents the heart of the application. It enables the nearly limitless extension of the model. A plugin is written in a pseudo-subset of C. The script language will have the nearly the same syntax as C, however the language is extended with definitions of equations as specific types. The actually syntax and semantics of the script language are not fully developed as of yet. However, the main purpose of a plug-in script will be to: 1) define extensions of the model; 2) define pregenerated scenarios; 3) to define base graphs (those which need not be user defined).

The scripts will be based on an inheritance system. Each script must inherit from a more generally defined (definition of model is less detailed) script. The child script will have the ability to override certain definitions and functions. All scripts must eventually be inherited from the base model definition script.

The definition of resources (bitmaps, fonts, etc.) will be able to be done within the script. The resources could be used for any number of enhancements within the GUI. For instance a bitmap attached to a variable may be used label buttons referring to the variable. In order to manage files better, datafiles that contain multiple resources will be able to be created using a simple utility.

Scripts are read at run-time (as called upon by the user) and compiled to an internal representation of the model. Script files are not locked during operation of the main app. This enables the helper app to modify scripts during operation of the model. Scripts will not be recompiled once the model is in operation, the user will have to specify when recompilation of the scripts should be done.

6.6 Helper application

In order to ease the development of plugins, a helper application will be developed. It will use a simple GUI that will enable the more advanced user to visually develop extensions to the model. The program will enable the drag-and-drop creation of resource datafiles, as well as aid in the creation of the script files.

6.7 Possible capabilities

The development of an output system may be built into the program. It would enable the data resulting from the simulation of the model to be exported to specific file formats, such as an excel worksheet.

The ability to print graphs and charts may be added. This option will depend upon the constraints of time.

The fetching of script files from locations other than the local machine may also be added. External scripts could then be loaded over the network or internet.

7 The Lunar Landing: Hoax or Scam?

Matthew Reale and Jared Woodward-Poor

7.1 Overview of the Lunar Prospector

In 1997, a satellite named Lunar Prospector was launched by the US to better study the Moon. Lunar Prospector has been collecting data from its orbit around the moon via five different scientific instruments: a magnetometer, an electron reflectometer, and alpha particle, gamma ray, and neutron spectrometers. The magnetometer is used to detect magnetic forces originating from the moon. It measures these forces along three different axes. The electron reflectometer works in conjunction with the magnetometer, and measures the strength of the magnetic fields at the moonUs surface. It does this by analyzing electrons reflected off of the moon. Together, the magnetometer and the electron reflectometer will give scientist a much greater understanding of the magnetic fields on and around the moon. The alpha particle spectrometer is used to detect gasses emanating from the moonUs surface by searching for radioactive alpha particles that are associated with these gasses. The gamma ray spectrometer is used to check for 10 different elements in the moon's crust. This data will help scientists know where to find useful metals in the future, and will shed light on the moonUs history. Finally, the neutron spectrometer is used to discover the presence of hydrogen as it exists in the form of water. In fact, the neutron spectrometer on board the Lunar Prospector recently found water, in the form of ice, at the polar caps of the moon.

Recently, the Prospector has entered a lower orbit for more intensive observation using all of the instruments. NASA has made much of the data available on the World Wide Web for download, and have also put some of this data into visualizations. These visualizations include multiple color line graphs, color map plots, three dimensional map plots, and color contour maps. There is also a way to see (approximately) where the satellite is at any given time. The web site that contains this information, as well as more information regarding the instruments aboard the Prospector is: `www.lunar.arc.nasa.gov`. We highly recommend you check this website out, as it is very well organized and contains lots of detailed information regarding the Lunar Prospector.

7.2 Our Objectives and Goals

We have several objectives for this project. Our first goal is to become familiar with this data set and its uses. We plan to spend the first few weeks of our project becoming familiar with the different groups of data, and how they are used. Since we have the ability to get raw data off the web, we need to figure out what exactly that data is. During these weeks we intend to write some code, probably in Java, that accesses the data sets and displays graphs and charts similar to the ones NASA uses on their web site. Once we are able to do this, we should be fairly comfortable with the data, and will be ready to try some new ideas.

One of these ideas involves filtering out background noise. Since the data is taken right from the Prospector and put on the web, there is a lot of background noise. The instruments are extremely sensitive, and thus pick up a lot of garbage that is not necessarily useful. This is apparent in the graphs that NASA has made available- they are very jagged and hard to read. We would like to come up with a way to cut down or perhaps eliminate this noise entirely. If we are able to do this, it would make the graphs much more readable and useful.

We would also like to come up with some different ways of viewing the data. NASA basically has one graph per instrument displaying information. The graph displays the data as a function of time. Therefore, the only way to connect that data with the actual surface of the moon is to check where the Prospector is, find out in what direction it is traveling, and then calculate the position of the data from that. We would like to find a way to graph the data directly as a function of the moonUs surface. Ideally, this would be in the form of a three dimensional graph. We would superimpose the raw data on an image of the moonUs surface from which it was taken. The closest NASA has come to this is to make available pictures of the entire moon, using colors to represent different data. For example, in terms of magnetic field, they used hot colors, such as red, to represent areas of high magnetism and cool colors, such as blue, to represent areas of low magnetism. However, this is only available for the entire moon, so it is not very useful if you want to zoom in one particular area. For example, NASAUs graphs could not be used to discover evidence of a golf ball or other remnants of a lunar landingI

This brings us to our third, and most ambitious, objective. Should we be able to get to a point where we can actually create three dimensional graphs and zoom in on certain sections of the moon, we would like to zoom in on the area where NASA supposedly put a man on the moon. Assuming there really was a lunar landing (and it was not filmed in Texas) NASA must have left some sign of their presence, and we would like to find hard evidence to back it up. However, considering there are literally billions of bits of data taken from all over the moon, this objective could be nearly impossible.

Even if we are not able to prove or disprove the lunar landing, we hope to at least come up with some new and creative ways of visualizing NASAUs data. Hopefully, our new visualization ideas will lead to some new extrapolations of the data, and we will be able to discover some interesting and enlightening ways of interpreting that data.

8 A Model for the Interaction Between Cells in the Nervous System

Summation of Synaptic Electrical Potentials

Mark C. Cattrell and Mike A. Bloom

8.1 Intro

Neuroscience can be considered an almost unexplored region of today's scientific frontier. Because of its vast complexity, the human brain and the functions of its individual cells continue to mystify scientists. It is our intention to spend this semester creating a model of behavior of the human brain as a function of several variables that affect its behavior. Our model will account for the effects of spatial and temporal variance as well as the permeability and capacitance of a neuron's membrane. After we create a robust program that accommodates for different values of these parameters, we will work with Professor James A. Murray of Colby's biology department to determine which other behaviors and variables we can approach.

8.2 Variables

When multiple electrical stimuli are influencing a neuron, the sum of their effects is called a summation. Summations occur differently when the distance that a potential must travel and the time between stimuli is not at a constant.

1. Spatial variance is the intrinsic variability of distance that separates synaptic interfaces from the position in the neuron at which summation takes place (the spike initiation zone, or SIE). When a message is sent to a specific cell in the nervous system, it travels in the form of a positive current of electricity entering the cell. Because a cell is not a perfect conductor, the potential entering a cell will naturally dissipate as it extends around the cell body. As the distance separating the input potential from the position in the cell that requires a threshold potential increases, the potential remaining in the cell becomes weaker.

2. Temporal summation occurs in a similar way: When a pulse reaches a cell, its strength dissipates over time inside the cell. But if the cell is stimulated again before the potential is allowed to return to its resting level, the net potential will be the sum of the two pulses.

3. The permeability of the cell membrane wall is another parameter which can affect the rate at which an electrical potential diminishes. This will inevitably be a difficult (though manageable) characteristic for us to model because the permeability of the cell membrane is affected by various factors. For instance, there are several kinds of "pumps" on the surface of the cell that actively move Na+ and K+ ions into and out of the cell. These pumps will almost all be completely closed when a stimulus occurs, but there are many other channels through which these ions and electrical charge can escape. Each neuron is unique regarding these characteristics, which requires that we create a dynamic program to accurately account for each intricacy.

8.3 The appearance of our program

As far as the aesthetics of the program go, we are planning on using a fairly extensive GUI. When the program is run, a picture of the neurons will appear on the screen. In the upper right hand corner of the screen, we will have an "EKG" type meter that displays the excitement and dissipation of the pulse.

We will also have arrow buttons (or something of that nature) that the user can manipulate in order to determine how close he wants to move one neuron toward the other (remember: the time it takes the second neuron to fire is a function of the distance between the stimulus and the SIE) There will also be similar arrow buttons to click to determine the resistance of the cytoplasm within the cell and the ease with which the pulse can escape.

Of course we will use the technical terms for these variables. All of these functions will be brought up on the screen when the user touches a button that brings down a drop-down window containing the choices. We will also use a GUI to cause the neuron to fire every time the user pushes the space bar. Each time the space bar is pushed, the user will be able to see the pulse moving toward the second neuron, and one will be able to see how close the potential in the cell is coming to the threshold on the EKG meter in the upper right hand corner.

When the pulse crosses the threshold, the target neuron will fire, causing the potential to rise much farther on the meter. The user will also be able to see the second neuron fire in the diagram.

8.4 Implementation

We will be writing our code in Java so that our finished product can be freely available via the world wide web. We will need to write several applets, learn the details of and create a graphical user interface, and write code that can be readily understood, but which will nevertheless dynamically model many aspects one of the most complicated processes known to man.

8.5 Schedule

Because of our previous experience with the Java programming environment, we feel that we can progress in this project fairly quickly and consistently. Over the next few weeks we will be focusing on the basic applet that will comprise our "EKG" component. We are confident that we will be able to have a graphic of a synapse fairly soon as well, depending on the complexity of the code that this would involve. By spring break we intend to have our GUI in working order, and we will have begun to implement the code for the actual physiological behaviors.

It is hard to extrapolate beyond Spring Break, but we are confident that our project will be proceeding smoothly and that we will be able to begin adding more involved features so that our software can be used as an educational tool.

This calendar is a rough estimate that we intend to follow, but we realize that there will likely be hurdles and setbacks along the way.

9 Ostracod Migration

Marylee Murphy and Jon Weinberg

9.1 Purpose

The project will be aimed at creating a tool to assist in Professor Sonya Skoog's research on migrations of Ostracoda in the Arctic Ocean.

9.2 Method

The first step in creating the tool will be to map the bathymetry (or depth contours) of the Arctic Ocean. The University of Hawaii has a publicly available and widely used Generic Mapping Tool (GMT) available on their website. Using this program and information from a publicly available world ocean database, the basemap will be created on which the rest of the project will build.

Next, the data that has been collected from cores in the Arctic Ocean will be plotted on the base map. Since the core data represents Ostracoda presences over time, first a number of maps representing the time intervals and Ostracoda occurences will be plotted. These maps will then be animated and the times between the known periods will be estimated to achieve smooth animation. Since there are 47 species of Ostracoda, the user will be able to choose which species they would like to view to eliminate clutter on the map.

If the above step finds that there are migrations of the Ostracoda, the next step will be to attempt to determine a cause for these migrations. A number of environmental factors will be investigated, most notabale temperature changes, currents, water masses, and any other variables that seem apropriate to include. These variables will be mapped and overlain on the basemap and the ostracod migrations and may also be colored and animated to show the passage of time. At this point the user will be able to choose which species to look at and which variables to observe. At this point a mathematical relationship may be looked for to compare the Ostracoda patterns to the variables.

9.3 The Scientific inquiry

Once the model is built, it will be possible to investigate any number of scientific questions. Are any species of ostracod native to certain water depths? What is the preferred temperature for which species? Are there definite migrations of the Ostracod populations? How does temperature affect the migrations? How does the water current affect the migration? Are all or any of these variables correlated with the migrations? Did different species migrate with respect to different variables? If a correlation can be established between the patterns and variables, it would be possible to make assumptions about arctic conditions millions of years ago.

9.4 Programming Aspect

The main programming language that will be utilized in this project will be Java. The Generic Mapping to will be the base software for this project that will be used to create any mapping portions of the project. The animation will most likely be a combination of maps created using GMT and Java programming to create smooth animation. The user interface will also be programmed in Java. Though GMT is written in C, there will be alterations that will be necessary to make to it that will be programmed in Java. Through this project java visualization, animation, creating user-friendly user interfaces, working with algorithms that can handle large data sets well to aid in creating smoothly animated images, and general data manipulation techniques will be explored.

9.5 The Master Plan

End of February: Download a working copy of GMT, find a roughly 1-minute-grid bathymetric database for the Arctic Ocean and plot the map. Begin the plotting of the Ostracod occurance data. Research all the variables we intend to look into at later points in the project.

March: Finish the bathymetric map with the overlain Ostracod occurances color coded. Begin to work on the animation, and work on integrating one or more of the variables. Begin the user interface.

April: Have the animation completed with several variables represented. Complete the user interface. Experiment with the program. Draw conclusions based on the behavior of the program. Make the poster and compose a presentation. Revise and refine.

9.6 Challenges

There are many parts of this project that are poised to create significant challenges. The entire project is going to be done in a Linux environment which is not fully operational at this writing. Installing GMT and a suitable Java Development Environment on the Linux machine could also pose problems. The main obstacle in this project will probably turn out to be facilitating the cooperation between the Java programs and the GMT. A main Java program will probably run the whole project, and it will have to invoke shell scripts and output post script files that were created by GMT.

10 Economic recovery modeling

John Doyle (and Tielman Van Vleck?)

10.1 Overview

The purpose of the project is to create a model based on historical data that can predict how fast counties or states can recover from economic recession in comparison with one another. The software tool will accept large data sets sorted by either state or county (yet to be determined; will depend on available data) and generate an animated view of the data set over a 2D map of the US. The goal of the tool is to make it is wide ranging as possible, so that it can accept more than one type of data set and still be able to effectively map the data. The scientific question to be answered will focus on the reasons behind the ability of certain geographical regions to recover more quickly than others from recession. The hope is that the model may provide some insight into what types of efforts work on a local level to speed recovery from recession.

10.2 The Data

A couple of different sources for data have been located. The historical data for counties and states is available from the University of Oregon Government Data Sharing Project. This data will be used to generate the initial animated model. The predictions will be based on data concerning the composition of local economies, found at Stat-USA. Hopefully this data will provide some insight into why some counties or states can recover faster than others. Only state data for industry concentration is available; unless I can find it broken down further by county at another website, the project may predict only on a statewide basis. I will also consult with Randy Nelson to explore other options for locating relevant data.

10.3 The Software Tool

The tool will be created using Java (unless you think there may be something more useful!). Somehow we will have to create the boundaries for the color coding of the data. Whether state or county, I am hoping I can find some shortcut so we don't have to trace boundaries pixel by pixel for however many counties there are. Fifty states would probably be more reasonable. Aside from that, the tool will have to be able to understand the data set it is given, so we will have to consider the format of the data set. The tool will allow for a static frame to be generated for each period of time based on the available data. Users of the model will be able to input the magnitude and duration of the recession and also specify if certain industry or areas are particularly hard hit. Finally, the animation will take a number of static frames and string them together, and the user will be able to select how long the animation should run. The details of the tool will probably become clearer as the project progresses (in other words, I have no clue right now!!)

10.4 The Question

The concept of business cycles in economics is not new, nor is it particularly difficult to understand. However, the preictability of these cycles is very low. While this project will not answer the question of why, it may be able to give some insight into what factors aid in local recovery from recessions. By examining data on industry concentrations, income, and other possibilities that come along, I am hoping to come up with some answers as to why certain areas take longer than others to recover from recessions. If the size of the project does not fit the time frame, it can always be scaled back to include only a particular region. The question will also become clearer as we start to find some answers.

10.5 The Time Line

End of Feb All data collected, think about inputs and outputs for software

End of Mar Software tool completed to accept and map a frame or a few of historical data

End of Apr Software tool completed to allow prediction and animation

End of Sem Finish up, testing, get rid of bugs, etc...

11 Chaotic Risk: Determining Stock Volatility Using Fractals

John Doyle and Tielman Van Vleck

11.1 Introduction

The prediction of stock prices is an imperfect art. Prediction of human behavior will never be an exact science. Short of finding a way to predict where the price of a stock will go, the best way (other than insider trading) to profit from a stock is to predict it's current value and hope that others will realize this value at a later date bringing the value of the shares to what you had estimated. Estimating what people will pay for the stock in the future is extremely valuable information.

The value of a stock is the sum of all future dividends from the stock. The price individuals are willing to pay is a function of the present value of a stock plus the risk of the investment. Risk refers to the stock's volatility. Since risk is a disincentive to purchasing a stock, investors require a higher rate of return when making riskier investments. The goal of this project is to develop a model which will estimate the risk of an investment using chaos theories.

11.2 Background on Risk

The risk of a stock can be divided into two parts: systematic risk and unsystematic risk. Systematic risk comes from national disasters and changes in our economy which affect the entire country such as going to war or changes in the interest rate. These risks can not be avoided when buying American stocks. Unsystematic risks are those risks that affect only certain firms. Unsystematic risk may be nearly eliminated by dividing one's total investment between several firms. In finance circles, the systematic risk is reported as the beta coefficient (b) of the stock. Hundreds of unfortunate souls have dedicated their lives to estimating betas for various firms.

The Security Market Line is a financial model showing a positive linear relationship between the expected return on investments and their risk. This line traces back to the risk-free rate which is considered to be that of treasury-bills. If a stock lies above this line than it is less risky than investors believe and is therefore overvalued. Conversely, if it is found below the line the stock is undervalued because investors feel that the stock is less risky than it really is.

An accurate estimate of risk would give one the ability to identify under valued stocks which could be bought for less than their true value.

11.3 Estimation of Risk

While fractals can never be expected to predict a certain outcome, they can create graphs which look a great deal like actual plots of stock prices. Though the outcome will be quite random, the volatility depicted could be a useful estimator of risk. A model must be constructed to predicts possible trends for each stock. In order to estimate the risk of individual stocks, the model of possible outcomes will be run many times. The risk of the stock will be derived from the deviation in outcomes provided by this model.

There are two chaos models which we will be considering to estimate the risk of investments. The first, developed by Benoit Mandelbrot, uses his fractal theories to estimate volatility. This model is useful for estimating and displaying risk associated with a particular investment when at least four points are known. A significant drawback of Mandelbrot's model is its inability to predict future risk because it reflects only past performance of an investment. The second model uses phase curves to describe a recursive model which estimates future performance using parameter values derived from past performance.

11.4 Question

Is it possible to roughly predict the volatility of the apparently random behavior of stock prices using models based on chaos theory?

11.5 Tools

This model will probably be written in Java. It will start out with a minimal interface for input of initial variables and output of results. We hope to combine the fractal and phase curve models to explain both the present and future risk of an investment. Though this model should be applicable to any investment, we plan to focus primarily on use for stock valuation. The user will input a series of dates and corresponding values of the investment and the program will output both a graphical and numerical estimation of risk. We hope to determine estimations of the Dow, S&P 500, and other indices to provide the user with a standard to which they can compare their results. To estimate the accuracy of our model, we hope to compare some results of trials for real investments with the actual risk of these investments. Then we will extend the model to allow for future predictions of risk for all investments.