6-8-1999

Computers, Art and Smart Rooms: A Smart Picture Frame that Senses the Weather and Genetically Evolves Images

Marisa E. Kolodny
Dartmouth College

Dartmouth College Computer Science Technical Report: PCS-TR99-354

Computers, Art and Smart Rooms

A Smart Picture Frame that Senses the Weather and Genetically Evolves Images

Marisa Kolodny
Department of Computer Science
Dartmouth College

June 8, 1999

Senior Honor Thesis
Advisor: Daniela Rus

Abstract: By using sensors to sense the environment and genetic programming to evolve images, this thesis explores two methods for developing smart pictures that can be integrated with a living space. The system presented senses the weather and indoor conditions, displays current weather and forecast information retrieved from the web, and displays genetically evolved images. Sensing the weather not only provides the user with information they might find useful, but also allows the computer to gain a better understanding of the user which in turn allows the computer to respond more accurately. Genetic programming allows the computer to better respond to its environment by evolving fitter programs.

Table of Contents

1 Introduction

When imagining the future of computers, one imagines computers that are not simply controlled by the user, but computers that can respond to their environment and determine their actions on their own, computers that are autonomous and can do more than exactly what they are programmed to do. Computers of today are still mostly deaf and blind, they do not know more than the user tells them through the keyboard and mouse, and cannot do more than they were originally programmed to do. However, new methods are being developed to change this and provide computers with easier and superior ways of communicating with humans. Computers are learning to perceive what is happening around them as well as the implications of these facts. Various sensors, including video, microphones, infrared, and touch sensors, are allowing computers to see, hear and feel what is happening around them. In addition to being able to perceive their surroundings, computers are becoming more autonomous by learning to evolve more successful programs on their own. Computers can use a method analogous to Darwinian evolution to evolve new programs. Like in nature, it is the programs that are most fit, the programs that are most successful at achieving their goals, that evolve into new programs. Soon, computer systems will be like any other piece of furniture; they will serve their function without being told what to do. This thesis uses both these methods of creating more autonomous computers. The system presented uses sensors to perceive the weather outside and the microclimate inside, and uses genetic programming to evolve images. By sensing the environment around it and using genetic programming, this system explores two interesting methods for making computers more autonomous and responsive to the user.

Sensing the weather is beneficial because it provides the user with information they are interested in knowing, as can be seen by the plethora of radio stations, television stations and World Wide Web sites that provide this information. Computers can alert the user to close their car when it starts to rain outside or to leave early before the snowstorm gets worse. More importantly though, a system that senses inside and outside climate

information will know more about the user and will be better able to meet their needs. What people do and feel is greatly affected by the weather. The idea that human health has a connection with the daily and seasonal weather was first written about by the Greek physician Hippocrates in about 400 BCE [Lan86]. Since then, many researchers have studied the relationship and conclusions show that people's bodies and moods are effected by the weather. For example, bone and muscle pains, asthma, insomnia, headaches, anger, and depression have all shown signs of being related to the weather. Indoor temperature and lighting can effect people's moods, comfort and alertness which in turn effects how successful they are at what they are doing [Lan86]. Computers can be of more help to humans if they know the weather conditions and their effects. Research at the MIT Media Lab is focusing on developing innovative technologies that can recognize their users and understand what they are doing and feeling. They are concerned with "people being able to perceive systems, and to communicate naturally with other people and also with machines – even with grunts, groans, and gestures. [They] want it to be to the point where the machine feels almost a part of you" [Hed98].

Genetic programming is used to evolve the images that are displayed to the user. This allows the system to display a theoretically infinite number of images, many more than could ever be decided on in advance. Genetic programming attempts to answer the question "How can computers learn to solve problems without being explicitly programmed? In other words, how can computers be made to do what is needed to be done, without being told exactly how to do it?" [Koz94] Genetic Programming succeeds in answering this question by producing results in the form of computer programs as opposed to other specialized structures such as decision trees or production rules [Koz92]. A genetic program uses an algorithm that parallels Darwin's theory of evolution to effectively search the space of all possible computer programs and find the one that is most fit, the one that is most successful at accomplishing the given task. This allows for the discovery of solutions

the human programmer might not have thought of. Genetic Programming mimics nature's methods for successfully evolving organisms.

> The organisms that are ill-suited for an environment die off, whereas the ones that are fit live to reproduce. Offspring are similar to their parents, so each new generation has organisms that are similar to the fit members of the previous generation. If the environment changes slowly, the species can gradually evolve along with it, but a sudden change in the environment is likely to wipe out a species. Occasionally, random mutations occur, and although most of these mean a quick death for the mutated individual, some mutations lead to new successful species. [RN95 p.619]

Genetic programming starts by creating a random set of programs made from the set of available functions and terminals specified as appropriate for the problem. For example, in our case the function set consists of basic image processing functions, the terminals are the original images given to the system and the random programs created are random combinations of the functions and terminals into a program that outputs an image. These programs are executed and evaluated to determine which ones are the most successful and are genetically bred using procedures for reproduction and recombination to form the next generation of programs. The evolution of images, based on work by Karl Sims, is the result of a collaboration between the computer that evolves the programs, and the user that evaluates them. "'You quickly evolve equations that you couldn't design or even understand', [Sims] says. Nevertheless, the process is simple enough that museum-goers can express their taste via interactive exhibits"[Gib93].

Together, these two ideas are used to create a 'smart' picture frame, a picture frame that can generate its display according to what it senses in the environment around it as well as on the Web. By using sensors and retrieving information from the Internet we can make the objects around us more pleasant and useful. Here, a picture frame can automatically project weather warnings and can generate images that will counteract any negative effects the weather might have. The picture frame could also project stock quotes, let you know what the kids are up to in the next room, and display the news as you drink your morning cup of coffee. One can imagine a system with similar sensors opening and closing your windows, adjusting the ventilation and clearing the snow from your driveway. Sun

Microsystems' new Jini codes will allow the objects and appliances around us to do just this. They will all be connected to the Internet as well as to each other. In the near future, your fridge will order more food when something expires, your bathroom mirror will display the news as you brush your teeth, and your car will know exactly where to take you based on the appointments in your date book and maps on the Internet [Lev99]. "Whatever the speed and shape of pervasive computing, the hope is that it will keep its main promise: integrating computers and the Internet so seamlessly into devices that we'll forget what's inside them. Instead, we'll concentrate on the tasks we want to perform in the first place" [Lev99].

2 System Description

The system developed for this thesis senses the climate inside (temperature and brightness) and outside (humidity, temperature and brightness). This information is supplemented by additional weather information retrieved from the World Wide Web. The system displays weather information to the user and uses the weather inputs it receives to genetically evolve images. The system uses a graphical user interface to interact with the user who can select one of two modes for the display. In the first mode, the system evaluates the images displayed through an internal function, while in the second mode it is the user who evaluates the images.

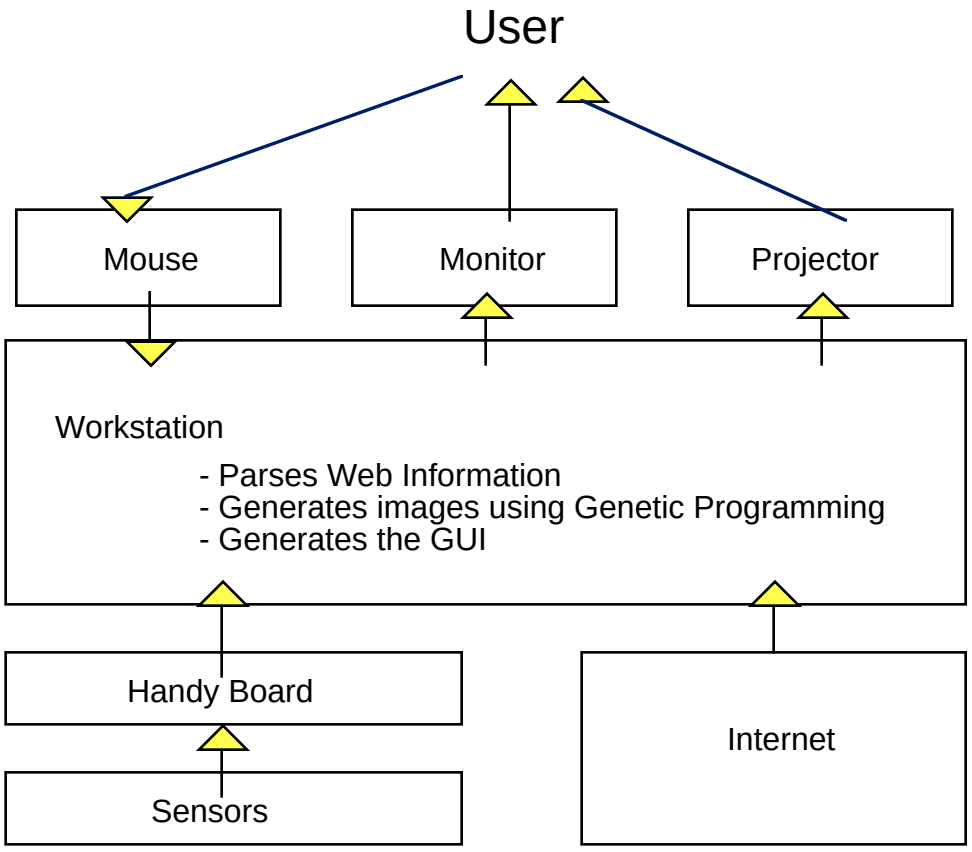

Figure 1: Block Diagram of the System

The main event loop of the system starts by getting the sensor values, retrieving the weather information from the Web, and creating the user interface. The system then waits for the user to start the genetic programming. The user can quit the system or switch

between the two modes at any time. Every time the user switches between the modes the weather information is retrieved from the Web again. When the genetic programming is started, the system gets the sensor values and creates an initial population of 8 programs that evolve images. For each program created, the image it evolves is displayed to the user, evaluated and assigned a fitness value. The initial population used is then used as a basis to create the next generation. The next generation also contains 8 programs that evolve images. The new images are evaluated and the following generation is created. This system repeats until 8 generations have been created, displayed and evaluated. At this time the image that received the best fitness value is displayed to the user and the system waits for the user to start the genetic program again.

2.1 The Sensor Interface

The main device used for the sensor interface is the Handy Board, originally developed for the MIT Robot Design project, and an extension board. It is a hand-held, battery-powered microcontroller board based on the Motorola 68HC11 microprocessor and includes ports for reading both analog and digital sensors, battery backed static RAM for on-board processing, as well as an LCD screen. The Handy Board runs Interactive C, a version of the C programming language. The sensors are read into the analog input ports on the Handy Board which does basic calibrating of the inputs and sends them over a serial port connection to the main workstation used. The sensor values are then used as variables in the evolution of the displayed images.

The following sensors were used with the system:

1) *National Semiconductor's LM34ACZ precision Fahrenheit temperature sensors* were used to measure both inside and outside temperatures. The sensor provided an output voltage linearly proportional to the temperature (+10.0 mV/°F). We needed to cut the pull-up resistor in the Handy Board out of the analog input circuit to maintain this linear output. The Handy Board will read an output voltage of 0V to 5V. As connected, with the sensor output going directly to the input port, the sensor has a

range of 5 to 300 degrees (50mV to 3V). The sensor can be expanded to cover its full range of -5 to 300 degrees (-50mV to 3V) if a pull down resistor is added. This output voltage can then be shifted and multiplied using optional amplifier chips and some resistors to increase accuracy and correspond with the full voltage range the Handy Board can read.

2) *EG&G Optoelectronics VT935G Photoconductive Cells* were used to measure the amount of light both inside and outside. The sensors were connected directly to analog input ports with outputs approaching 0 as the brightness measured increases.

3) *Honeywell's HIH-3605A integrated circuit humidity sensor* was used to measure the percent humidity outside. The sensor has a linear output voltage versus %RH (Relative Humidity). We needed to cut the pull-up resistor in the Handy Board out of the analog input circuit for this sensor as well. The following formulas were used in converting the output voltage to the true %RH:

$$Vout = Vsupply\ (0.0062\ (Sensor\ RH) + 0.16)\ @\ 25°C$$

$$True\ RH = (Sensor\ RH) / (1.093 - 0.0012T),\ T\ in\ °F$$

Additional sensors that could be added to the system to increase its understanding of the environment include a digital water sensor to detect precipitation, a pressure sensor to measure the barometric pressure and a CO2 sensor to measure the indoor air quality. Measurements of indoor air quality can serve to increase ventilation when measurements indicate poor air quality.

2.2 The Internet Interface

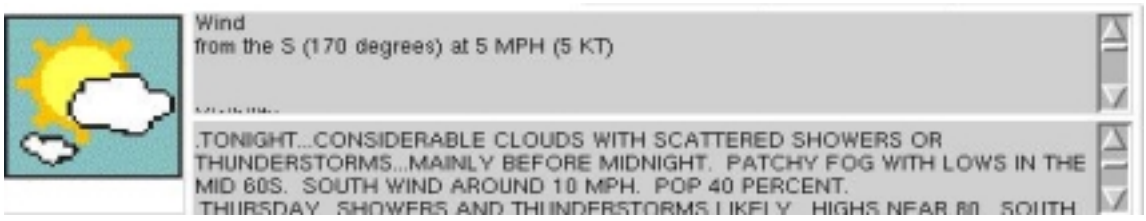

Figure 2: The Weather Update Section

In order to supplement the weather information from the sensors, the system uses Tcl/Tk's World Wide Web interface to retrieve information from two weather sites (http://tgsv7.nws.noaa.gov/weather/current/KLEB.html and http://iwin.nws.noaa.gov/iwin/nh/zone.html). There are additional Tcl procedures to parse this information and display it to the user. This part of the system will need to be maintained closely since the source is likely to need to be parsed differently whenever the host changes the structure of the page.

2.3 The User Interface

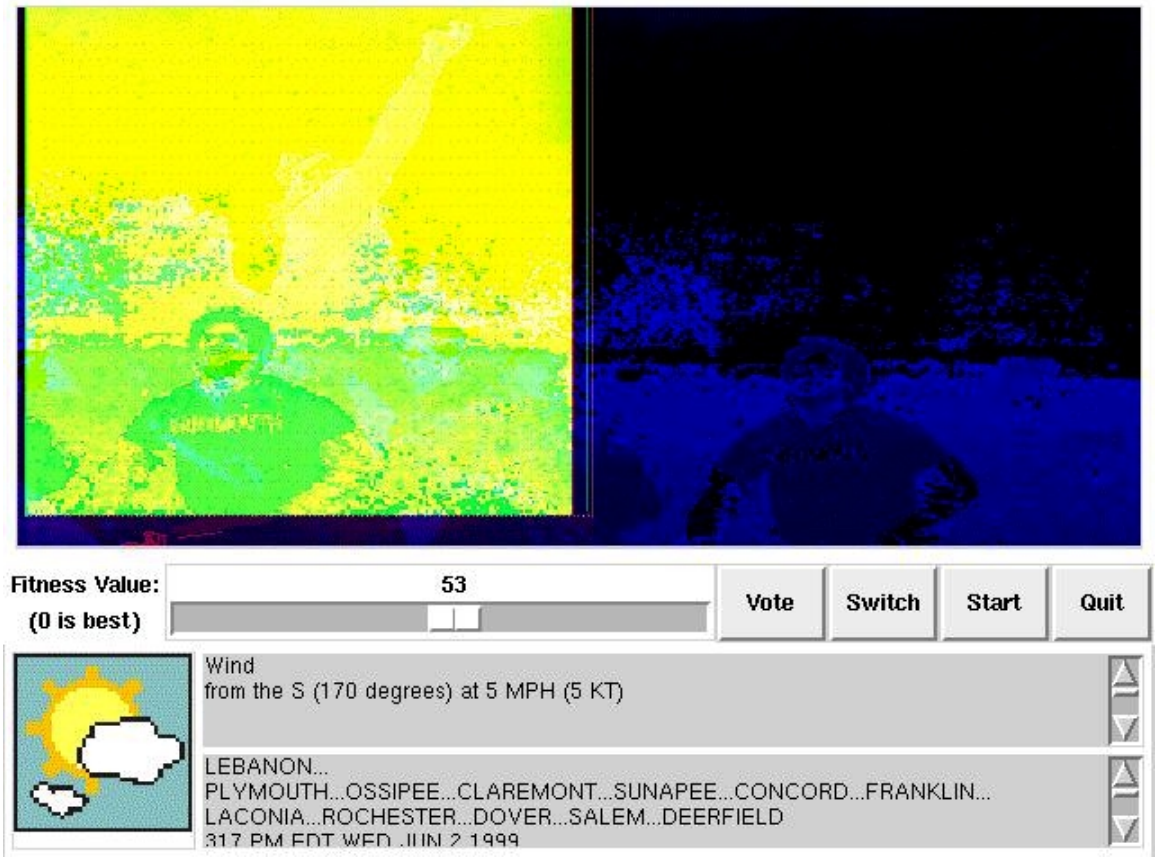

Figure 3: The User Interface

Tcl/Tk was used to implement the graphical user interface. At the start, the system opens a window that fills the entire screen. The program starts in the first mode in which the program itself evaluates the fitness of the images. The user can switch between this mode and the second mode in which the user determines the fitness values at any time. In

both modes, the main window is divided into three main parts. The top section displays the generated image to the user. The system displays the image produced by each genetic program evolved in each generation. The middle section is a button bar. There is a button to quit the system, start the genetic program and to switch between the two modes. When the program is in the second mode, in which the user evaluates the images, there is also a scale used to set the fitness of the image (0-100), and a vote button to indicate that the user has decided on a fitness value.

Figure 4: The Status Bar (First Mode)

Figure 5: The Status Bar (Second Mode)

The bottom section of the program is a weather update section. It displays information about the current weather as well as weather forecasts based on information it retrieves from the World Wide Web about the Lebanon, NH area.

2.4 The Projector

In order to get a large display area the *nSight LCD Monochrome Data Projector* is used. The projector has a VGA resolution of 640x480 and can display 16 shades of gray on a wall/screen up to 20 feet away. The projector is connected to the monitor output of the workstation and has an output to send the information to the monitor as well.

2.5 Genetic Programming to Evolve Images

The images displayed to the user are the results of programs that have evolved through genetic programming. The Genetic Programming C++ Class Library, developed by Adam Fraser and Thomas Weinbrenner, is used as a framework. The system starts by reading the original 12 photographs of Dartmouth, stored as ppm images, into objects where the individual pixels can easily be accessed and changed. These image objects also

store integer values taken from the sensor outputs at the time the image objects are created. These integer values are used by the functions that use non-image variables so the resulting image of a genetic program can be consistent each time it is evaluated. The different terminals and functions available for the system to use to build the population of programs are also set at this time. The set of terminals is simply the set of 12 original ppm images in the database. The function set consists of basic image processing functions, which all take one or two images as arguments, use the output from the sensors as additional variables, and return a new image. The following functions make up the function set:

Negate (returns the negative of the input image),
HReflect, VReflect, Reflect (returns a copy of the original image reflected across the horizontal, vertical or both axis),
HTrans, VTrans (returns a new image that is the result of translating the image horizontally or vertically),
Red, Green, Blue (returns a new image that has the red, green, and blue values increased respectively),
LightDark (returns a new image that is the result of increasing or decreasing the brightness of the input image),
Set (returns a new image which has been binarized),
Clip (returns a new image for which there are no pixels above a max value or below a min value),
Zoom (returns a new image that is a zoomed in or zoomed out version of the input image),
Add, Subtract (returns a new image that is the result of adding/subtracting the pixel values of the two images),
And, Or (returns a new image that is the result of logically anding/oring the two images together),
HMerge, VMerge (returns a new image that is the result of merging the left (top) half of the first image with the right (bottom) half of the second image)

Since each of the functions in the function set takes one or two images and returns an image, the result of any function can be any one of the arguments of any other function and can be returned as the final result of the program. This satisfies the *closure* requirement that "each of the functions in the function set should be able to accept, as its arguments, any value that may possibly be returned by any function in the function set and any value that may possibly be assumed by any terminal in the terminal set" [Koz94 p.35].

The Genetic Programming Kernel also allows for the use of automatically defined functions (ADFs). An ADF "is a function (i.e., subroutine, procedure, module) that is

dynamically evolved during a run of genetic programming and which may be called by a calling program (e.g., a main program) that is simultaneously being evolved"[Koz94]. Since ADFs allow a program to reuse pieces of code, they allow genetic programs to better solve problems that have regularities and patterns. This is similar to the way a computer programmer breaks a problem down into sub-problems, writes functions to solve each of the sub-problems, and then uses these functions to solve the overall problem. In this thesis, it is not necessary to repeatedly use a set of functions in order to come up with interesting result images. However, one ADF which takes two images as arguments was used. As a result, for each member of each generation, in addition to evolving a main program from the ADF function and the function and terminal set described above, the system also evolves an ADF function from the terminals Arg0 and Arg1 (it's two arguments) and the functions described above.

After setting the allowable functions and terminals, the system is ready to create the initial population of random genetic programs. The generated programs are represented as parse trees in which each node is either a function or a terminal. This system creates the initial population of 8 genetic programs that evolve images using ramped creation in which the genetic programs produced have increasing depths up to a maximum of depth 6. Each of these programs is then run, evaluated and assigned a fitness value. The standardized fitness is a positive number, with a smaller number indicating a better fitness. The *Evaluate* function used depends on the system mode. In the first mode, the system calculates the fitness of the genetic programs evolved by finding the average pixel value of the result image (a number between 0 and 255) and comparing this to the average value of the photocell sensors, stored in the image object. The closer these two numbers are, the fitter the genetic program that evolved the image. In the second mode the *Evaluate* function gets the fitness value assigned to an image by the user. In both cases, the fitness function also takes the complexity of the genetic program into account so as not to evolve programs that are to complex to evaluate.

The most successful programs created are genetically bred using reproduction and recombination to form the next generation of programs. Reproduction causes the most fit computer program to survive by being copied as is to the next generation. The other programs in the new generation are generated using recombination. Recombination uses the crossover method, which takes two programs and produces two new programs by selecting a point on each program and swapping the sub-tree below the selected point on the first program with that of the second program. Following the creation of the new population, a portion of the new population will be mutated. There are two kinds of mutation possible. The first kind of mutation is swap mutation in which a function or terminal in the genetic program is replaced by a function or terminal chosen at random. Only nodes that require the same number of children are exchanged. The second kind of mutation is shrink mutation in which a gene is chosen to take the place of its parent. The parent and all its other children are consequently deleted. This new generation of images can then be evaluated and the process continues. The genetic program will end after 8 generations and the user can start the process all over again.

It was decided to run the genetic programming system for 8 generations of size 8 after testing the system with many combinations of population sizes and number of generations. The quality and diversity of images evolved seemed to increase as the population sizes and number of generations increases. However, the limiting factors in this process are time and memory and the workstation used could not handle larger population sizes or larger numbers of generations consistently. The system could be improved by using a computer with a fast processor and large amount of memory so that more genetic programs can be created and evaluated.

3 Sample Result Images

Figure 6: Image Evolved through Genetic Programming

Figure 7: Image Evolved through Genetic Programming

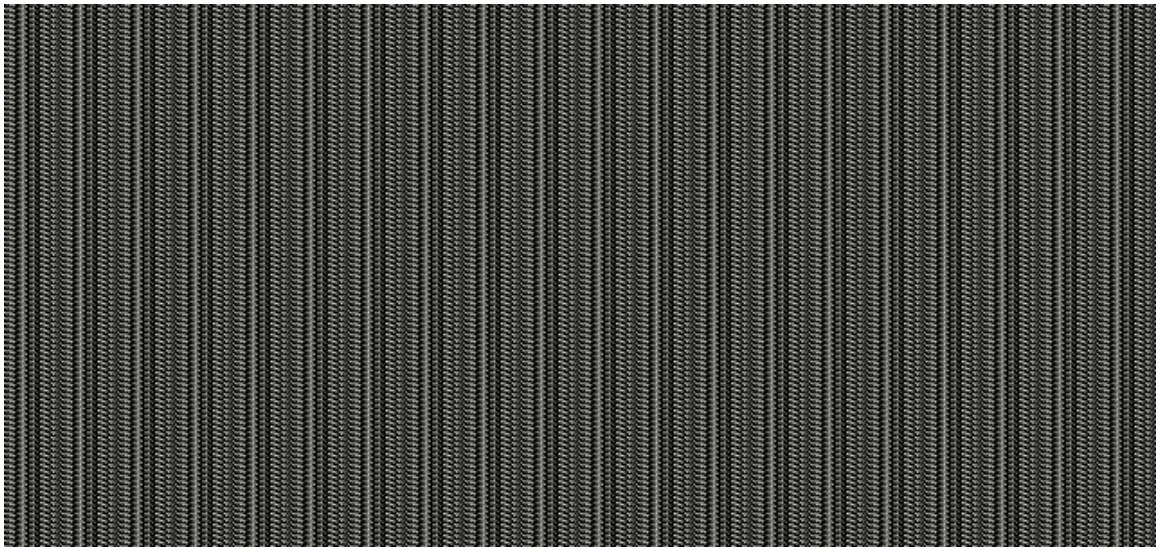

Figure 8: Image Evolved through Genetic Programming

Figure 9: Image Evolved through Genetic Programming

4 Conclusions and Future Work

This thesis explores two interesting advancements in computers: the use of sensors so a computer can determine on its own what is happening in the environment, and the use of genetic algorithms to develop effective computer programs on its own. The two ideas together will be helpful in creating a new generation of computers that can 'think' for themselves, computers that can choose their own reactions and come up with solutions they were never programmed to have. Although the system has not yet been subject to user tests, I think it can be very successful.

Although not yet implemented, it is possible to extend this project to display images to counteract the effects of the weather, to be soothing during conditions that tend to lead to stress and anger and to be stimulating when the people in the room are likely to be less alert. It could display images that are warming when it is cold and images that suggest happiness during weather that typically leads to depression. The system could also hypothesize about the alertness level or mood of the user and adjust its actions accordingly. Knowing the weather and its general effects will help computers to better understand the intentions of their users which will make the communication between the computer and the user more effective.

Additional developments to this system could include eliminating the user's mouse interaction with the program. A system with voice recognition and voice outputs would allow the user to enjoy the larger projected images in both modes and provide a much more natural user experience. Outputs could include instructions on what to do and a description of the system and what it is in the process of doing, as well as weather alerts and sounds to go along with the images. Voice recognition could be used to switch modes, start and stop the image evolution, and record the fitness values assigned by the user. Rotating weather updates would eliminate the other mouse interaction required by the user. The system could also be expanded to output information about the image evolution process and the equations used to generate the image. This would allow the system to be used as an educational tool

for learning about genetic programming as well as evolution. These additions would improve the educational as well as aesthetic value of a system that is already successful as a work of art and computer research.

5 Acknowledgements

I would like to thank my advisor, Daniela Rus, for her help and support throughout my thesis. I would also like to thank Keith Kotay for helping me solve many of the problems I encountered along the way, and my friends and family for their continued encouragement.

6 References

[Hed98] HEDBERG, Sara Reese, “MIT Media Lab’s quest for perceptive computers”, IEEE Intelligent Systems & Their Applications, v.13 no.4 (July/Aug. ’98) pp.5-8

[Gib93] GIBBS, W. Wayt, “Creative Evolution”, Scientific American, v.269 (Oct. ’93), pp.20-21

[Koz92] KOZA, John R., Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press, Cambridge, MA 1992

[Koz94] KOZA, John R., Genetic Programming II: Automatic Discovery of Reusable Programs, The MIT Press, Cambridge, MA 1994

[Lan86] LANDSBERG, Helmut Erich, “Weather, Climate and You”, Weatherwise, v. 39 (Oct. ’86), pp.248-53

[Lev99] LEVY, Steven, “The New Digital Galaxy”, Newsweek, May 31, 1999

[Lip94] LIPKIN, Richard, “Simulated creatures evolve and learn”, Science News, v.146 (July 23, ’94), p.63

[Mar98] MARTIN, Fred G., Robotic Explorations: An Introduction to Engineering through Design, Draft Edition

[Pen96] PENTLAND, Alex, *Smart Rooms*, Scientific American, v.274 (Apr., 1996), pp.68-72+

[RN95] RUSSELL, Stuart, NORVIG, Peter, Artificial Intelligence: A Modern Approach, Prentice Hall, Englewood Cliffs, NJ, 1995

[Sim91] SIMS, Karl, *Artificial Evolution for Computer Graphics*, Computer Graphics, Volume 25, Number 4 (July 1991), pp.319 - 328

[Sim94] SIMS, Karl, *Evolving 3D Morphology and Behavior by Competition*, Artificial Life IV Proceedings, ed. by R. Brooks & P. Maes, MIT Press, 1994, pp28-39