InterDraw – An Advanced Online, Interactive, Collaborative Art Program

by

Kristopher T. Babic

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

May 2000

APPROVED:

Professor Matthew O. Ward, Thesis Advisor

Professor Micha Hofri, Head of Department

This thesis is dedicated to my girlfriend Sarah and my parents. Their love and support helped me through all the trying times and have pushed me to reach within myself to accomplish my goals.

Abstract

InterDraw is an art program that facilitates the artistic collaboration of multiple users. The goal of this collaboration is the creation of one unique computer image that represents a combination of the ideas and images provided by each of the users. InterDraw extends the already collaborative nature of the World Wide Web through the use of the Java programming language, which provides InterDraw with its cross-platform capabilities. Previously, collaborative art-like programs have been developed for specific operating systems or environments. This limitation prohibits any collaboration with users from other operating systems or environments. InterDraw breaks this limitation by using the power of Java to provide program access from any computer with an Internet connection and a Java enabled browser. The InterDraw clients collaborate by transforming objects drawn by a user into compressed binary strings that are then transported over the Internet to a server application. This server maintains a database of artist contributions and updates all other InterDraw clients collaborating on the same image. These binary strings provide a reliable transmission format that allows the drawn objects to be recreated by the InterDraw clients. Through user testing, InterDraw has been shown to provide an effective and entertaining forum for the creation of collaborative, dynamic images.

I would like to thank Matthew Ward, my thesis advisor, for his insight, direction and support, Isabel Cruz for allowing me use her students to test this project and for filling in when Matt was not around, Mark Appleton for keeping me sane during the long hours and little sleep, and the WPI Computer Science Department for providing me with support and insight throughout the project.

Contents

List of Figures

List of Tables

1 Introduction

In 1965 Gordon Moore developed the idea (now known as Moore's Law) that the density of transistors on a hardware "chip" will double every 16-18 months. This law also implies that the price of a transistor should half every 18-20 months. In the last three decades of the 1900s this law proved to be accurate. What effect did this have on the computer industry? Computers that once filled a room and cost millions of dollars in the 1960s, now could fit into an area smaller than a suit case and cost as little as $500. Computing power that was once only available to the government or the very rich was now available to the average consumer. This trend led the way for computers to be used in exciting and creative ways.

Computer graphics was one of the areas that benefited most from the expanding power and decreasing prices in the computer industry. As computers became more powerful, using computers as tools to create images became a feasible option for people. The computer power that was needed for the development of applications such as Adobe Photoshop® soon became available. This and many other applications allowed people to use the computer to further their own creative ideas in the form of artistic endeavors.

Communication is another field that has been dramatically changed since the development of the computer. These changes came about due to the creation of the Internet, which allows people all over the world to easily communicate and share ideas with each other. The idea of the Internet was first conceived in the early 1960s. It was the height of the cold war and the Department of Defense (DoD) wanted to develop a form of communication that could survive a nuclear war. Under the DoD's Advanced Research Project Agency (ARPA) guidance, a network architecture (ARPANET) was developed. The ARPANET grew into a network of super-computers that promoted the sharing of data and research among researchers in the United States. A massive computer network grew out of this initial project and turned into what we now refer to as the Internet.

This project is focused on extending the power of the computer to further explore the extent that computers can influence the areas of art and communication. The art field must now explore the avenues opened up by the new levels of communication brought upon us by the computer age and allow the creation of new and exotic forms of art.

1.1 Project Goal and Objectives

The overall goal of this project was to create a program that would utilize the tools of today's technologically advanced communications network to enhance the ability of artists to communicate with each other to develop truly unique works of art.

This goal was met through the completion of the following objectives. The first objective was to develop an advanced multi-platform drawing application. This application has all the basic features of most drawing programs, such as line, free hand, and basic shape drawing capabilities. The application also has the unique feature of being able to incorporate animated drawing objects into the work of art. These animations may be anything from an image changing locations on the screen to the complete removal of an object from the image.

The second objective was to develop an extensive three-tiered client-server-database model that facilitated the client interaction that was necessary to meet the overall goal of the project. While the client allowed the user to visualize and modify the current state of the art work, the server and database maintained a stored representation of all works of art and of each client's contributions to them. This model created a method of real-time communication between the client drawing applications, thus generating the ability to maintain a multi-user work of art.

1.2 Why Collaborative Art

Collaboration is the merging of the talent, designs, thoughts and ideas of multiple individuals. From the days that man lived in caves and hunted together to find food, to today's technology oriented society, collaboration has always played an intricate part in society. In the days of the hunter/gatherer society people had to collaborate with each other in order to trap and catch their prey. Today, we still see forms of collaboration. Almost all projects, whether they are designing software or developing a stronger metal alloy, people have to share their ideas and work together to get the job done.

The world of art is not usually one of collaboration. It mostly consists of individuals who like to express their ideas and thoughts through paintings, drawings, and other artistic endeavors. So where does collaboration come into play? There are many reasons for the introduction of collaboration into the art world. It has been found that collaborative art endeavors helps people express themselves more openly than otherwise. Psychologists, to aid in mental and emotional growth of patients, have used collaborative art in this therapeutic way. Another reason for collaborative art is the end result of the collaborative process. The result is a product that could not have conceived or created by an individual. It is something that is truly unique. While the artists are working on the artwork, they will be able to become closer to the other artists by being able to see the emotions each is putting into their work.

Why collaborative art? Collaborative art is a unique aspect of the creative process and the development of tools to facilitate this process enhances its ability to reach more individuals. This project is designed to help give collaborative art a new forum to explore.

1.3 Where Idea Originated

The idea for this project evolved out of Professor Matthew Ward's interest in the concept of collaborative art. His interest in collaborate art was developed during an excursion to the SIGGRAPH 96 conference, which was held in New Orleans. The SIGGRAPH conference is one of the leading conferences in the area of computer graphics. During Professor Ward's visit, he discovered an exhibit that allowed multiple people to create dynamic pieces of art through the use of body movement inside of a room. The movements of the people were captured by motion sensors and then translated into various effects, which were then displayed onto a viewing screen. This collaboration between various people intrigued Professor Ward.

From this interest Professor Ward developed the idea of using the Internet as a medium for collaborative art. The Internet itself is already a collaboration of millions of people, so why not extend it further to include the collaboration of art.

2 Background

This chapter provides an in-depth look at the concepts and ideas that are needed to fully understand this project. In it the areas of computer art and collaboration are explained and explored.

2.1 Computer Art

What is computer art? In short computer art is anything where the computer is used to create something that is artistic in nature. This can range from using the computer to make small drawings, to using the computer to machine a block of metal into a sculpture. While these and many more are considered computer art, when most people think of computer art it is usually the creation of images that comes to mind. Examples of such images can be seen in Figure 2.1and Figure 2.2.

Figure 2.1: One Purpose, Many Plans by Bill Neidow; image from http://www.sausalitoartfest.org *used without permission

Figure 2.2: Trinity by Joan Blades; image from http://www.sausalitoartfest.ort *used without permission

Both of these images are great examples of computer art. These images were created through the use of complex computer imaging software and the creative minds of the artists. Computer art can also be seen in the graphical representations of mathematical formulae. One such example is the use of fractals. Without the computer fractals are simply things that are self-replicating. When a computer is used by someone to graphically display a fractal, new and wondrous images can be created, such as the one displayed in Figure 2.3.

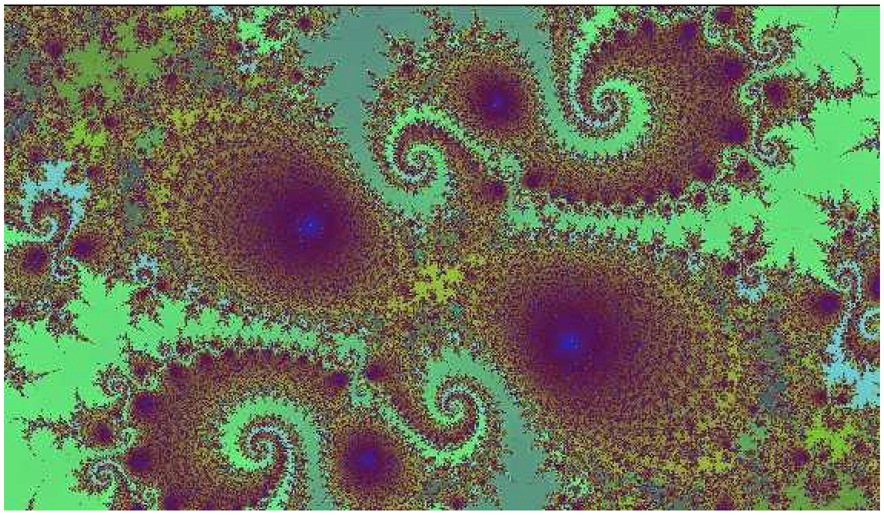


Figure 2.3: Fractal Peacock Image by Noel Giffin; image from http://spanky.triumf.ca *used without permission

These beautiful images could not be created without the aid of a computer. So what is computer art? It is impossible define. Computer art is whatever you make it out to be.

2.1.1 Definition of Art

What is the true definition of art? This question has been the topic of discussion by political leaders and philosophers alike. Can art be defined, though? To get a true look at art we must look at the origins of what we call art.

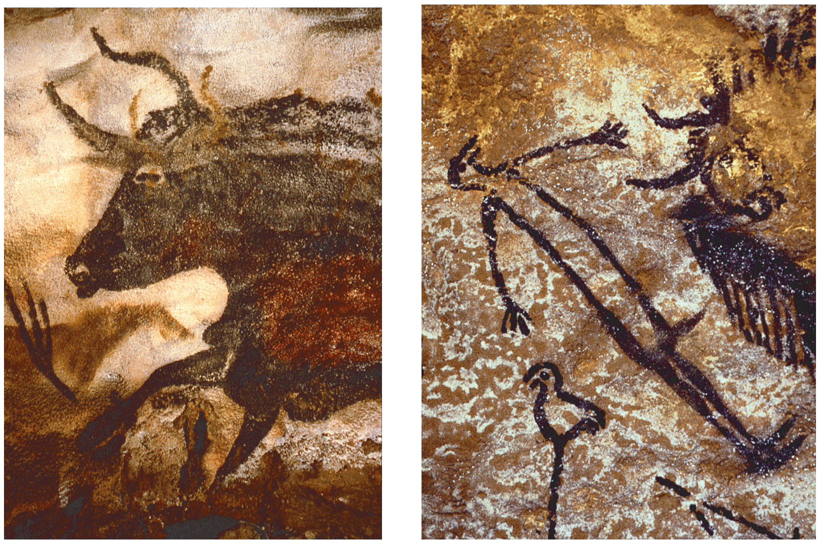


Figure 2.4: Images from Paleolithic Painted Cave of Lascaux, France; **images from http://www.culture.fr *used without permission**

Prehistoric man, approximately 20,000 – 25,000 years ago, began to draw rough images on the walls of caves with flint, stone and pigments made from plants. These were images of the things that were important to them. These were images of hunting and the animals that they hunted. As the years went by, the techniques used by these primitive artists became more advanced. The artists were able to find new pigments and methods of drawing. With these new methods the primitive artists were able to create more realistic images through the use of silhouettes and using different tones of color. Using these techniques they were able to reach a level of realism that was not achieved again until thousands of years later. The techniques that followed focused less on realism than on portraying images dedicated to their gods and rulers.

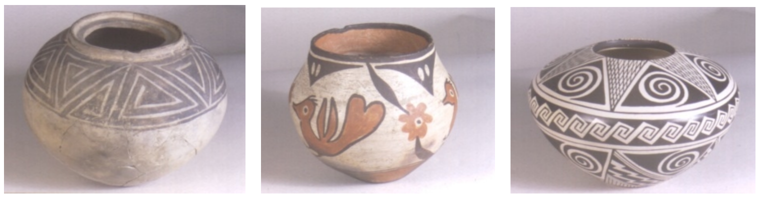


Figure 2.5: Pueblo Indian pottery from 1000 AD to 1970 AD; **images from http://www.cmnh.org *used without permission**

Later, when mankind moved out of the caves to build dwellings of their own, a new medium was discovered. This new medium was the discovery of methods to create pottery. Using this medium, artists were able to express themselves through the creating of shapes made out of clay and hardening them over a fire. When these shapes were hardened the artists found they were able to decorate the clay in the same manner as their predecessors did in the caves. Only these decorations were not life like, but stick-like representations of people and their surroundings. Not only did they discover they could decorate these pieces of hardened clay, but they could also use the clay to create items to decorate the home or to display a person's status.

Along with the Bronze Age came more media for artists to express themselves, and along with these media new methods to create art were discovered. These new artists could now create images, figures and objects out of bronze.

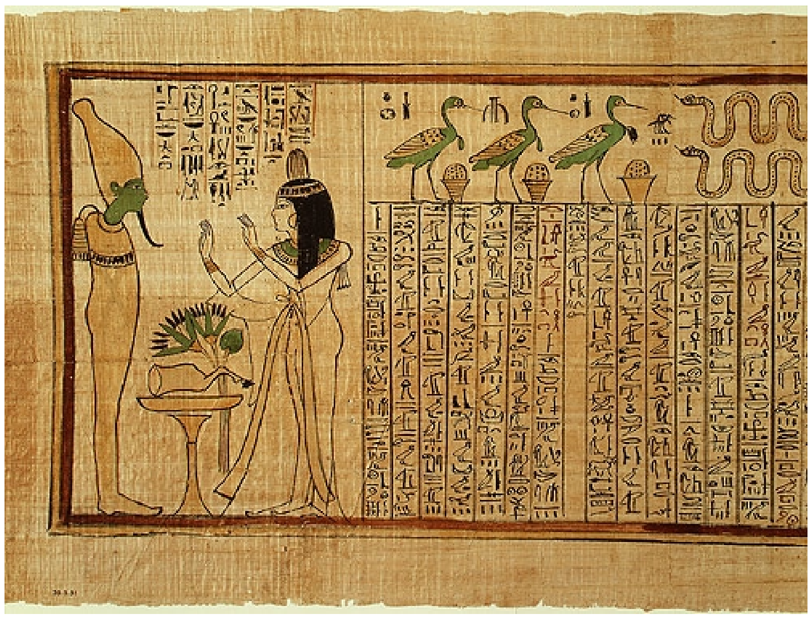


Figure 2.6: Section from the "Book of the Dead" of Nany, ca 1040-945 B.C; 21st Dynasty; Reigns of Psensenmes I-II; Third Intermediate Period; **image from http://www.metmuseum.org *used without permission**

The next stage of art arrived with the Egyptians. The Egyptians created art to immortalize both their gods and their rulers, whom they looked upon as gods. These images were much different from the art of the past. Their art emphasized completeness and not realistic representations. The artist's task was to preserve everything as completely as possible. How could they do that when they looked at things and saw that they could not see everything clearly from one angle? In an attempt to make everything they drew stand out with great clarity they drew everything as if it was being seen from an angle that best represented that object. This technique can be seen in their representations of the human figure. The head of a person is best seen in profile, while the upper torso is mostly seen from the front. The legs of the human were drawn as if seen from the inside of the leg, thus making it look as if the person had either two left feet or two right feet.

Next came the rise of the Greek Empire. Along with this rise came the rediscovery of realism. Much of their art was based on the realistic recreation of the human figure. They made these images to honor both their rulers and their gods, and as such they endeavored to make them works of beauty.

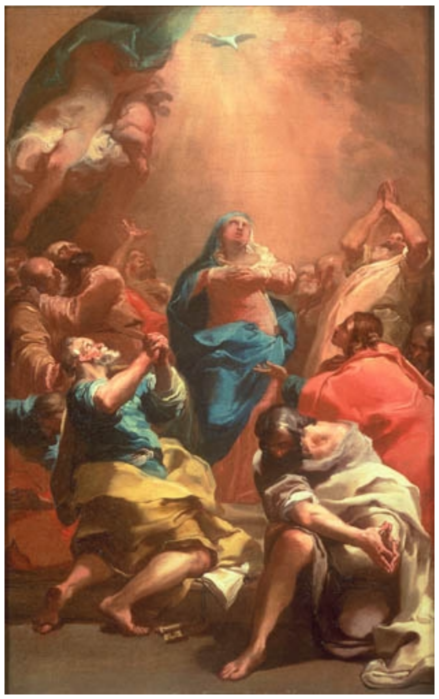


Figure 2.7: The Pentacost by Gaetano Gandolfi (1734 - 1802) Bologna, Italy; image from http://www.europeanpaintings.com *used without permission

The rise of the Catholic Church became one of the biggest influences of art. The Catholic Church persecuted anyone and anything that was against their teachings. These persecutions included many forms of artistic representation. The rise of the church also brought about what are considered the greatest works of art of all time, such as the Sistine Chapel. Even during these times, the form of art changed. It went from having the locations of holy people form geometric shapes in paintings (such as a triangle, where

Mary would be on top and others would be underneath) to the creation of images where there was no control of the formation of the work.

The next step in the history of art arrived with new forms of art that threw off the shackles of reality. Such artists as Picasso, Turner and Van Gogh created the art forms that we now know as surrealism, pointillism, cubism and impressionism.

Following these styles such artists as Andy Warhol developed the styles we now know as modern art. Warhol created pieces of art by taking images of people and things and simply composing their images onto a canvas in varying colors. Other artists create works or art that cannot be described with words.

We look at the long history of art and we try to see what makes any of it art. Is it art because we say it is, or is there something behind it? Is there some definition that can be written down and held in your hand? Morris Weitz makes this observation on art:

> "The problem of the nature of art is like that of the nature of games, at least in these respects: If we actually look and see what it is that call 'art', we will find no common properties-only strands of similarities. Knowing what art is is not apprehending some manifest or latent essence but being able to recognize, describe, and explain those things we call 'art' in virtue of these similarities."([Til84], p 26)

Weitz is comparing the definition of art to that of a game. At first it seems easy to come up with a definition of a game. The problem he brings up is when does a game stop being a game and become something else? If you change something in the game, is it then, by definition, still a game? There are many different variations of art. When does something stop being art and becomes something else and vice versa.

When a person looks at a big piece of metal that has been formed by an artist and doesn't see it as art, is it therefore no longer art? Leo Tolstóy in his essay, "What is Art?" says this about the subject:

> "The fact that I am accustomed to certain exclusive art and can understand it, but am unable to understand another still more exclusive art, does not give me a right to conclude that my art is the real, true art, and that the other one, which I do not understand, is an unreal, a bad, art. I can only conclude that art, becoming ever more and more exclusive, has become more and more incomprehensible to an ever-increasing number of people, and that in this, its progress towards greater and greater incomprehensibility, it has reached a point where it is understood by a very small number of the elect, and the number of these chosen people is becoming ever smaller and smaller."([Tol62], p 175)

Tolstóy is making the argument here that even though you may not understand the nature of the art itself, it does not mean that it is not art. Just because one does not understand something does not mean one has the right to condemn that which one does not understand. This issue brings up other problems as well. If something can be art without everyone thinking it is art, then what is art?

There are some that would say the definition of art is that art is the creation of anything of beauty. But what is beauty? Something that is beautiful to one may be seen as grotesque or even perverse by another. People have defined beauty as that which contains the ideals of God, while others define it as the recreation of goodness. Shaftesbury taught that "beauty is recognized by the mind only."([Tol62], p 94) Clearly this definition can be the only one, because the true definition of beauty is something that is personal to everyone and only that person can have a true understanding of their own definition of beauty.

B.R. Tilghman wrote about a definition of art, saying, "the very idea of a theory or definition is a confusing one."([Til84], p 197) From what we have discussed we can see that this statement is well founded. Art is something that is very hard to define. If you define something as art, you may be stating that something else is not art, when it should be.

Leo Tolstóy developed a better definition of art. He states that art is "an activity means of which one man, having experienced a feeling, intentionally transmits it to others."([Tol62], p vi) By using this definition Tolstóy is able to include almost everything that can be considered art, from a person digging a hole in a park to the Mona

Lisa. As long as the artist is able to transmit a feeling to someone else through his/her art, then he/she has created art.

This definition isn't perhaps the best one, though. E. H. Gombrich gave the best definition when he wrote, "There really is no such thing as Art. There are only artists."([Gom66], p 5) This definition is perhaps the closest definition we can possibly give to art. As long as one believes he/she is an artist and creates something they feel is art, then they have created art.

2.1.2 Is Computer Art Really Art?

As computers became more and more powerful, people started to recognize their ability to be used as a medium for art. But is anything created by a computer really art? The biggest opposition to this is that a computer is based on mathematical formulae and computation.[Spa99] How can art be created by such inhuman means? A computer is not what creates the art, though. The computer is only a tool that is used by an artist. It does not generally have the ability to create on its own. It instead must be molded and manipulated by a human user. This interaction is what makes the output of the computer "art."

If we were to condemn the use of a computer as a medium for art, must we not also have to condemn photography and even painting itself? There are many styles of art that are based on recreating nature. Since this is only the recreation of something created by nature, where can the human part apply? It applies in the same manner as with a computer. The human artist manipulates the medium with which he/she works to create something that has some meaning to that artist. Without that meaning the art is useless, but with it art can be created through the use of any medium, including the use of a computer.

2.2 *Idea of Collaboration*

According to Webster's Dictionary to collaborate is defined as follows:

> "col·lab·o·rate
> v. intr. col·lab·o·rat·ed, col·lab·o·rat·ing, col·lab·o·rates.
>
> 1. To work jointly with others or together esp. in an intellectual endeavor
> 2. To cooperate with or willingly assist an enemy of one's country and esp. an
> occupying force
> 3. To cooperate with an agency or instrumentality with which one is not
> immediately connected
>
> [Late Latin collaboratus, pp. of collaborare to labor together, fr. L com- + laborare to labor]" [Mer94]

I will be using the first definition for the purposes of this project. As this definition illustrates, collaboration is working together in a joint effort to some end. This end could be anything from developing a new software application, to helping a friend move into a new house.

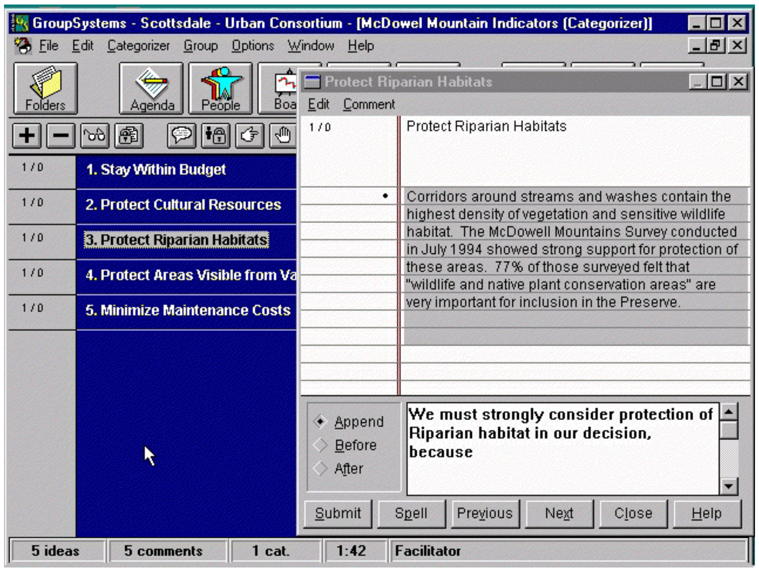
Collaboration allows us to achieve and do more than we would be able to do as an individual. In collaborative projects, the collaboration is the sharing of ideas, knowledge or data between the individuals or units involved with the project. All parts of the project must work as a whole to complete the overall objective.

Collaborative art is different in some aspects. While it does include the sharing of ideas and knowledge, it also reaches into areas that other collaborative ventures do not. Collaborative art also extends into the realm of being able to share emotions and feelings. The ability to provide this type of collaboration makes collaborative art truly unique, not only in its creations, but for what it provides to the people involved.

2.2.1 Collaborative Applications

One application of collaboration can be seen in many companies today. This application is web/tele-conferencing. Through the use of either the Internet or phone lines people are able to have conferences/meetings with each other regardless of their location. This method of communication has lead to the increased ability of people to work together in a more widespread environment.

Another application of collaboration is Active Response Geographical Information System (AR/GIS) Collaboration System. This system incorporates various geometrical decision tools into a network planning application to assist resource management teams in debating land issues, and the risks and impact a decision will make on the land. Users of this system are able to assess current land use status, develop objectives, and propose geographic scenarios. The information that is collected from the users is summarized and displayed graphically by the system. The users can then discuss these results, provide more comments and opinions, and share other relevant information. A screen shot of this application is shown in Figure 2.8.


GroupSystems - Scottsdale - Urban Consortium - [McDowel Mountain Indicators (Categorizer)]
File Edit Categorizer Group Options Window Help
Folders
Agenda
People
Boa
Protect Riparian Habitats
Edit Comment
Append
Before
After
Submit
Spell
Previous
Next
Close
Help
5 ideas
5 comments
1 cat.
1:42
Facilitator

1 / 0 1. Stay Within Budget
1 / 0 2. Protect Cultural Resources
1 / 0 3. Protect Riparian Habitats
1 / 0 4. Protect Areas Visible from Va
1 / 0 5. Minimize Maintenance Costs

1 / 0 Protect Riparian Habitats

Corridors around streams and washes contain the highest density of vegetation and sensitive wildlife habitat. The McDowell Mountains Survey conducted in July 1994 showed strong support for protection of these areas. 77% of those surveyed felt that "wildlife and native plant conservation areas" are very important for inclusion in the Preserve.

We must strongly consider protection of Riparian habitat in our decision, because

Figure 2.8: AR/GIS Collaborative Negotiation System; **image from http://www.ciesin.colostate.edu** ***used without permission**

One other example of an application of collaboration is the Collaborative Onsite Wearable System that is being developed by Human Computer Interaction (HCI) professors at Carnegie Mellon University. They are developing a system that will aid an aircraft mechanic in the diagnostics and repair of planes. The idea behind the project is to have the mechanic use a wearable computer to be able to perform onsite simulations and review past maintenance data to determine problems with the aircraft. If the aircraft mechanic is unable to determine the problem, the system allows him or her to establish and audio, video, data link with the help desk at another concourse. The mechanic can then interact with the lead mechanic there to discuss the current situation and solution.

2.2.2 Collaborative Art

Collaborative art is the creation of art through the collaboration of a number of individuals. All of these individuals contribute their own uniqueness to the final piece of art. Collaborative art is a unique form of art, because it is the only form that allows the feelings and emotions of more than one person and displays them in a work of art that one artist alone could not create.

Collaborative art has been used as a means of emotional release in group therapy sessions for years. Through the use of collaborative art the people in therapy are able to release their emotions while drawing on the strengths of the emotions of others. In the process they find new ways to allow their emotional well being to grow. An example of this is the Growing Through It art workshop.[Gro00] During the workshop people are put into groups and these groups work together to create both a "professional piece of art and a book." Mental growth accompanies this physical outcome to further benefit the participants, which is the true objective of the course.

Collaborative art has already found a place on the World Wide Web. One such example is the HypArt project.[Ros99] "'HypArt' stands for Hyper-Art and is the artistical equivalent to 'HyperText'. The idea is to create a single picture together with people from all around the world. HypArt features differently themed blank digital canvases; each divided into a series of squares. Artists the world over add their own artistic contribution to one of the subdivided areas."[Ros99] The developer of this site created it because he was fascinated by the idea of people all over the world being able to work together to create one image. An example of one such image is shown in Figure 2.9.

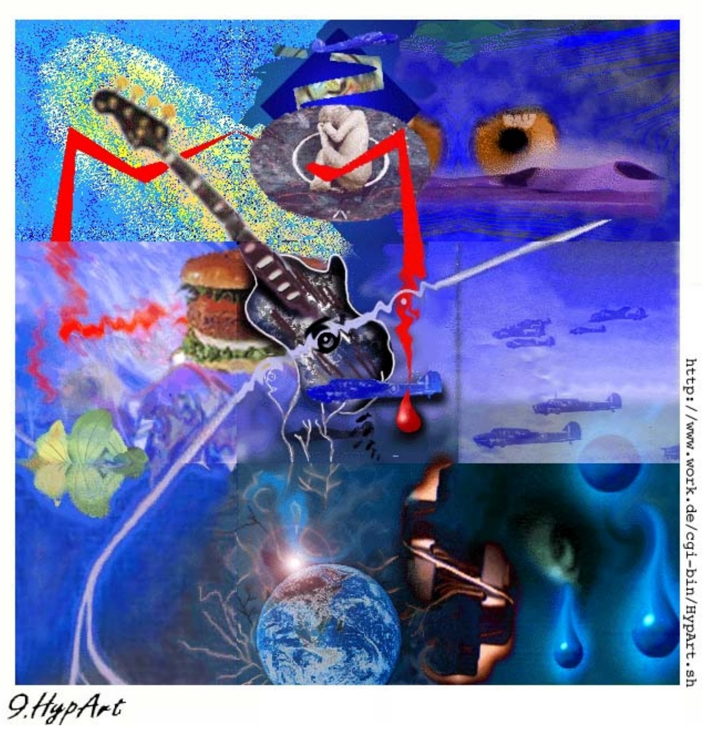


Figure 2.9: Blue - a HypArt creation; **image from www.work.de/cgi-bin/HypArt.sh *used without permission**

As evident in the name of the image, the theme of this HypArt creation was "blue." Klaus Rosenfeld developed the HypArt project in 1994. The very first project was started in June of 1994 and came to its successful completion in August of that year. The HypArt project started its 37th creation in March of 2000.

The Dreamview Collaborative Generations Projects is a great demonstration of collaborative art on the Internet. The idea behind this project is to create on initial image and to then allow people to modify that image to create a second-generation image. That image can than be modified to create a third generation and so on. The creator of the project described the reason behind the projects as follows:

> "All of the web (and life itself) is a collaborative project. Everyone learns from and picks up ideas from others. How many times have you looked at something -- an image, perhaps -- and said, 'I could do something with that,' or 'I would have done that differently,' or 'I'd like to take that image and just change this part and that part...'?" [Var00]

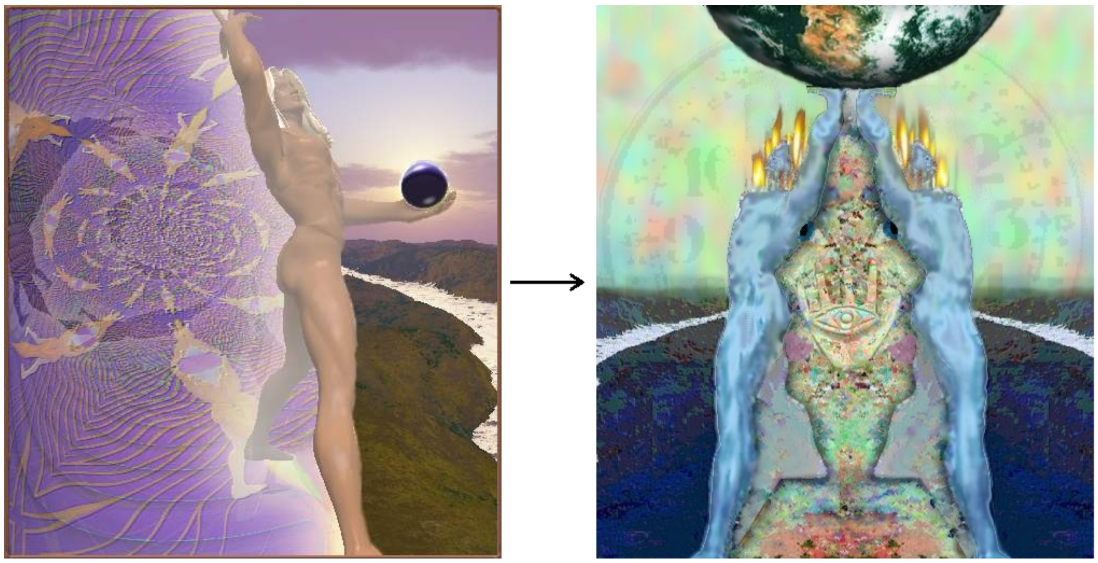


Figure 2.10: Generations Project: Generations 1 and 3; **images from http://www.cs.cmu.edu** ***used without permission**

Figure 2.10 displays the first and third generation images from this collaborative art project. It is easy to see the influences that the different artists were able to bring to the original image. As the project progresses the influences of the different artists will work the art piece into forms not imagined by the original artist.

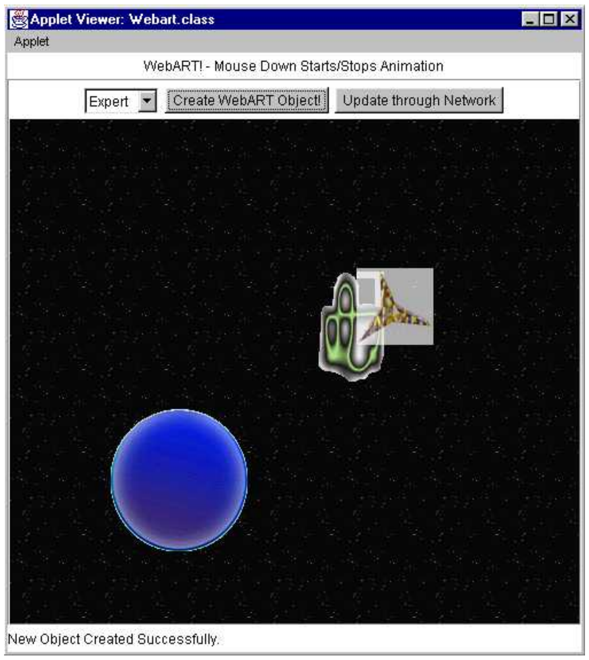
Another example is the Chain Art Project [Mit99] where a piece of art was sent around via e-mail and ftp and was modified by each person that it encountered to create the final image. Every person that worked on each individual piece downloaded the picture from a specified ftp site and then modified the picture using their own personal image editing tools. The image was then uploaded to the ftp site for the next person to download. By the end of the project over 130 images were completed through the efforts of over 200 participants.

These sites are just a few examples of collaborative art. There are many more examples both on the World Wide Web and in everyday life. All of these examples of collaborative art demonstrate the ability for people to combine their emotions into the creation of one unique work of art.

2.3 Related Work

WebArt and Microsoft NetMeeting® are some examples of related work in the area of the collaborative sharing of graphical information.

2.3.1 WebArt


Applet Viewer: Webart.class
Applet
WebART! - Mouse Down Starts/Stops Animation
Expert
Create WebART Object!
Update through Network

New Object Created Successfully.

Figure 2.11: Sample WebArt session

The only work we are aware of that has focused solely on the idea of web based interactive, collaborative art has been the development of WebArt.[Mas98] Andrew Mason developed this program as an MQP at WPI. WebArt was an interface that allowed users to develop simple sprites (small animated images) and to download them to the WebArt database. The user was then able to specify a sprite to use and a time-based function with which to specify the sprite's movements. This data would then be transferred to the database and submitted to the user programs.

While WebArt was working on a noble idea, it fell short in some aspects. It succeeded in its goal of providing a collaborative area in which multiple users could interact with each other on some type of digital canvas, but was limited in its potential to be a useful art tool. WebArt only allowed for the creation of small, animated sprites that moved in relatively strict patterns. This restriction to sprites and movement limited the ability to create art. It did not allow the users themselves to be able to exert a large amount of control over the development process, except to specify the sprite and a function. While this may be useful for perhaps an interactive gaming simulation or screen saver, it has limited utility as a tool in the development of art.

The WebArt project succeeded, however, in demonstrating that the development of an interactive, collaborative art program is both feasible and within the capability of current technology.

2.3.2 Microsoft NetMeeting

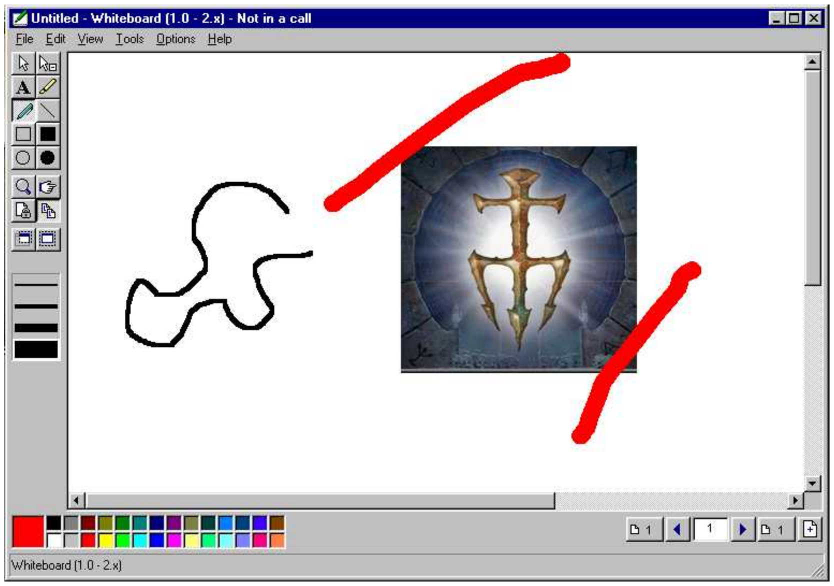

Untitled - Whiteboard (1.0 - 2.x) - Not in a call
File Edit View Tools Options Help
Whiteboard (1.0 - 2.x)

Figure 2.12: Sample NetMeeting Whiteboard session

Microsoft NetMeeting®, as far as we are aware, is the only commercial product that deals with the transfer of interactive images between multiple users. This program incorporates an interactive "white board." Users are able to draw using simple shapes, free hand and a highlighter brush. It also allows deletion of any of the objects created and basic cut and past features. The drawing program itself is a simplified version of Microsoft Paint.

One of the biggest downfalls to this application is its accessibility. To use Whiteboard the user must be running NetMeeting on a Windows operating system. This requirement limits the number of users who have access to the program, as the majority of high-powered graphic systems do not run the Windows operating system.

This application does show that the implementation of my project is possible. It is a simpler version than the application developed in this project, but it displays many of the basic characteristics that were incorporated into the final project.

3 Methodology

In this chapter we describe the methodology that was followed during the duration of the project. It describes what tasks were needed for the successful completion of the project and what their purpose was.

3.1 Project Tasks

1. Develop a simple online drawing program in Java

A simple online drawing program was the first step to reach my overall goal. This drawing program needed to include features that would allow a user to interact with a digital canvas. The features that were included in this simple interface were: drawing simple geometric shapes, free form brush strokes, and selecting colors from a simple color palette.

2. Develop network server and add network capabilities to drawing program

A network server needed to be created in order to provide a mean of client-to-client interaction. In providing this interaction the server needed to provide the clients with the ability to update information in the database and to retrieve information from the database. By providing these abilities the server was able to simulate a direct client-to-client interaction.

3. Create a global database

A global database needed to be created in order to maintain the state of the collaborative piece of art. This database needed to contain representations of the collaborative image(s). It was also a key element in providing the persistent storage for the collaborative images.

As with any shared data source, synchronization methods needed to be developed for accessing the database. These methods prevented any errors, such as race conditions, that may occur as a result of multiple clients simultaneously modifying information in the database.

4. Create dynamic attributes to drawing primitives

A method had to be devised for the interaction of the individual distributed drawing programs and the global database. The components of this interaction were devised in such a manner that allowed for the ability to update the global art and the individual pieces of art quickly.

5. Add animated features

Once a working interactive program was developed, animated features needed to be added to the program. These features include some or all of the following: animated sprites, animated sprites with tails, polar coordinate trails for sprites and particle systems, timed removal of items added to the digital canvas and adding time-varying attributes to sprites or other graphical primitives, such as color, size, position and shape.

6. Add advanced features to drawing program

Advanced features were added once all other features had been successfully implemented in the project. These features included, but were not limited to: allowing the user to select different images to modify, allow the user to create new images and a delete feature that allowed a user to remove any object he or she had added to the image.

7. Evaluation

The most important part of this project was deciding if it has been completed successfully or not. To be complete the project had to attain the first five goals. Along with completing these goals the resulting program had to be useful. The usefulness of this project was evaluated in the following ways.

1. Evaluating the number of users able to simultaneously use application
2. Evaluating the latency for updates under load
3. Having the program evaluated by a diverse range of users to see what features they find useful, what features they feel are missing and what features need to be improved upon.

4 Design

This chapter is dedicated to the organization of all design aspects that were needed in the development of this project.

4.1 Programming Language Selection

The first major design decision of this project was to decide which programming language was going to be used in the actual development process. Since this project was to be a networked application a few languages immediately came to mind. The final decision came down to the following four languages: C, C++, Visual Basic and Java.

The advantages of using either C or C++ were that I was much more familiar with the languages and in their networking abilities. They also have the ability to be used with a web page in the form of CGI (Common Gateway Interface) scripts. The biggest disadvantage to these two languages is the complexity of developing graphical applications with them. On any platform it is difficult to develop an application with a graphical interface using either of these languages. Another problem lies with the CGI scripts. CGI scripts are fundamentally server side applications and because of this feature user interaction is very limited. Another disadvantage to using C or C++ is that the developed application can only run on the platform on which it is developed. The reason behind this problem is that different operating systems use their own unique libraries. To move an application to a new operating system means rewriting the application code to work with the new operating system. This inability to switch platforms limits the number of users that will be able to use the application, which defeats the main purpose of the project, which is to create a collaborative art program that can be used by anyone.

The advantages of using Visual Basic are that it is an easy language to work with and to develop graphical applications. One disadvantage to using this programming language is its networking capabilities. Visual Basic does not have any local networking capabilities. To add network capabilities to Visual Basic, third party drivers or applications must be used. Similarly to C and C++, Visual Basic has the problem of not being able to change platforms. Visual Basic applications can only be used on Microsoft

Windows operating systems. This restriction limits the number of users that will be able to interact with the application.

Java turned out to have many advantages over the other programming languages that I looked at. One of the principal advantages is Java's platform independence. A Java application is run on top of a Java Virtual Machine (JVM). When a Java application is compiled, it is turned into intermediate byte code. This byte code is then interpreted by the JVM, which in turn runs the application. As long as an operating system has its own JVM that can interpret the Java byte code, a Java application can be transported to that operating system without having to recompile the Java code.

Another great benefit to Java is that that you have the ability to build both stand-alone applications and Java Applets. Java Applets are Java applications that run inside of a web browser. This feature is one of the most powerful features of Java. It takes the control of the Java program away from the operating system and instead places it on the web browser. Previously to the release of "Java 2", this benefit has some problems. Not all web browsers were capable of running the latest versions of Java and others would only run modified versions of Java. In June of 1999, Sun Microsystems released the first version of the Java 2 browser plug-in. This plug-in made Java, not only operating system independent but browser independent as well.

Java's graphical and networking abilities added to it being a good choice for a programming language. Java already has a variety of built-in image manipulation and drawing routines. These routines would simplify the overall work that would need to be done to complete the project. Java has some rather unique networking capabilities. The developers of Java have simplified networking to the use of a simple Socket class. To create a network socket in Java, you simply have to create a Socket object with a host name and a port number. Java does the rest.

Even with all of these advantages, Java did have some disadvantages, the primary one being that I had little experience with the programming language and its vast libraries. Another disadvantage was that Java was not as "stable" as some of the other languages. Java is still in the stages of being developed, and unlike C and C++, bugs are still being found within the low level Java code. These bugs can cause unexpected problems with your code.

After looking at the advantages and disadvantages of these languages I finally decided on the use of Java as my primary programming language. The benefit of being platform independent and being able to place an application within a web browser outweigh its disadvantages. The choice of which version to use was a simple one. I decided to use the most recent version of Java that came out at the beginning of the project. This version was "Java 2" or Java v1.2.2. The reason behind this decision was that the new version had fixed many of the bugs of previous version, was more stable, and made use of the new Java web browser plug-in.

4.2 Operating System Selection

This project is targeted at all of the "Java 2" enabled operating systems. During this projects duration, these included:

Windows NT/95/98/2000
Solaris
IRIX

Others operating systems may be tested in the future.

4.3 Database Selection

A database is required to maintain the collaborative images and to give the images a method of persistent storage. I took into consideration the use of three different databases. These databases were: Microsoft SQL Server, Microsoft Access and Oracle.

After evaluating these three databases, I determined that Oracle would be the best choice for this project. The reason behind this choice was that Microsoft SQL Server was too expensive for my personal use, and it was unavailable through academic sources. On the other hand, Worcester Polytechnic Institute had an Oracle server readily available for use and Oracle 8i Personal Server is available for download from Oracle for non-commercial use. Oracle is also a very reputable database and is know for being robust and easy to work with. Access is also available through the school and is easy to use. It was determined to be not as robust and reliable as the Oracle through preliminary testing of this project. Access would return errors as a result of sending basic queries to the database.

While it was decided that Oracle was to be used as the primary database for the project a Microsoft Access database would be kept as backup and for testing during the duration of the project. This decision was made because of the low memory requirements of Microsoft Access. It allowed for the multi-tasking of multiple applications, while Oracle's memory requirements do not.

4.4 Architecture Overview

This project is designed to implement a three-tiered architecture. This three-tiered structure is displayed in Figure 4.1. The first tier in the architecture consists of the client application. The client tier is responsible for all user interactions such as drawing and removing drawn items. This tier is also responsible for maintaining the visual representation of the current image.

The second tier is the server application. This tier has the responsibility of maintaining a virtual connection between the clients and the database tier. To provide a virtual connection to the database, the server maintains an active connection to the database tier as well as to any active clients. If a client requests data that is stored within the database, the server retrieves the information and then transmits it back to the client. Similarly, if the client requests that data is to be stored to the database, the server stores the information to the database.

The final tier is the database. This tier provides persistent storage for the client. All image data is stored in the database to be retrieved by a client request.

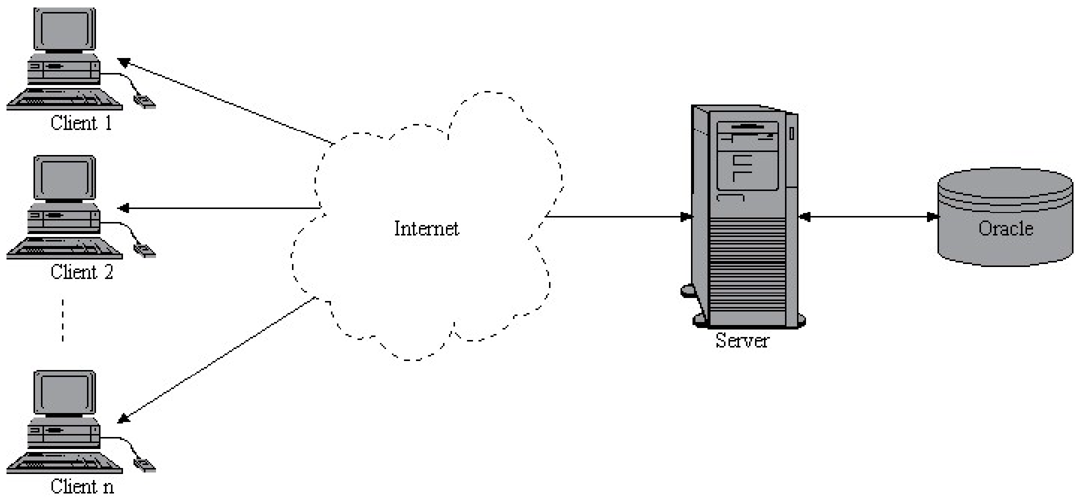


Figure 4.1: Flow chart of overview look at project

Figure 4.1 accurately displays the number relationships between the different tiers. The Server and Client tiers have a 1 to n relationship, where there can be any number of clients transacting with one server. There is also a 1 to 1 relationship between the Server and the Oracle database. This relationship restricts problems arising from multiple servers trying to access data from the database simultaneously.

4.5 Client Design

The client is the main application in this project. It oversees the development, creation and removal of all elements in the collaborative drawing. It also provides the User Interface that allows the user to interact with the application.

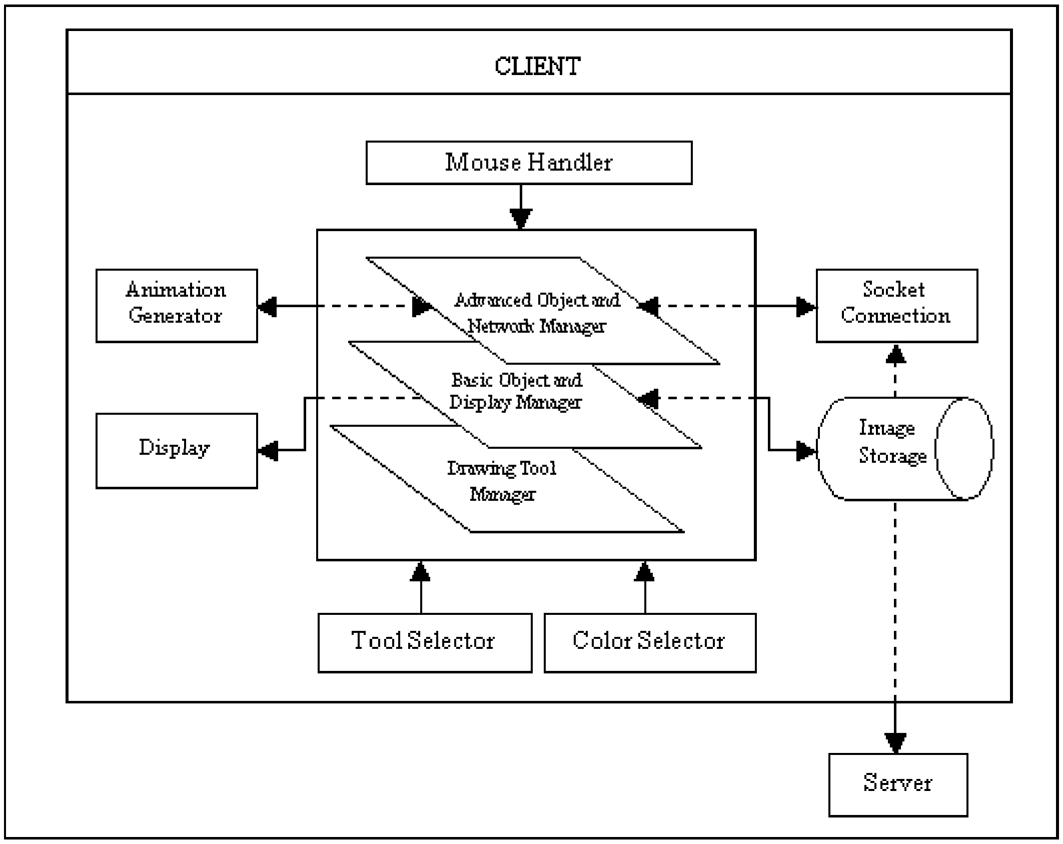


Figure 4.2: Client architecture

The core of the client application is split into three governing levels as shown in Figure 4.2. The lowest level is the Drawing Tool Manager is the most basic of the three levels. Its main tasks are to keep track of the active drawing tool and to respond to the drawing tool's actions. The middle level is the Basic Object and Display Manager. This level has the responsibility of managing the storage and display of all objects in the current image. The top level is the Advanced Object and Network Manager. Its responsibilities include the management of all animation objects and the socket connection to the server.

4.5.1 Drawing Tool Manager

The Drawing Tool Manager (DTM) is the simplest of the three governing levels in the client application. Its purpose is to oversee the storage and actions of the drawing tools. When a drawing tool is selected, the DTM deactivates the current drawing tool and updates it with the drawing tool that was selected. The DTM then processes and transmits the drawing tools actions.

4.5.2 Basic Object and Display Manager

The Basic Object and Display Manager (BODM) has two main tasks. The first task is to manage the displayed image and the second to manage the storage of all objects that make up the current image. The first task is broken down into two parts. The first part deals with refreshing the on-screen image. To provide an efficient method of refreshing the screen a double-buffered design is employed. In a double-buffered design, images are not drawn directly to the display. Instead the image is drawn to an off-screen buffer. When the image needs to be updated on the screen, the off-screen buffer is drawn to the screen. If double buffering is not used, the user may see flickering as different parts of the image are drawn to the screen. The flicking becomes more prominent if animation is added to the image.

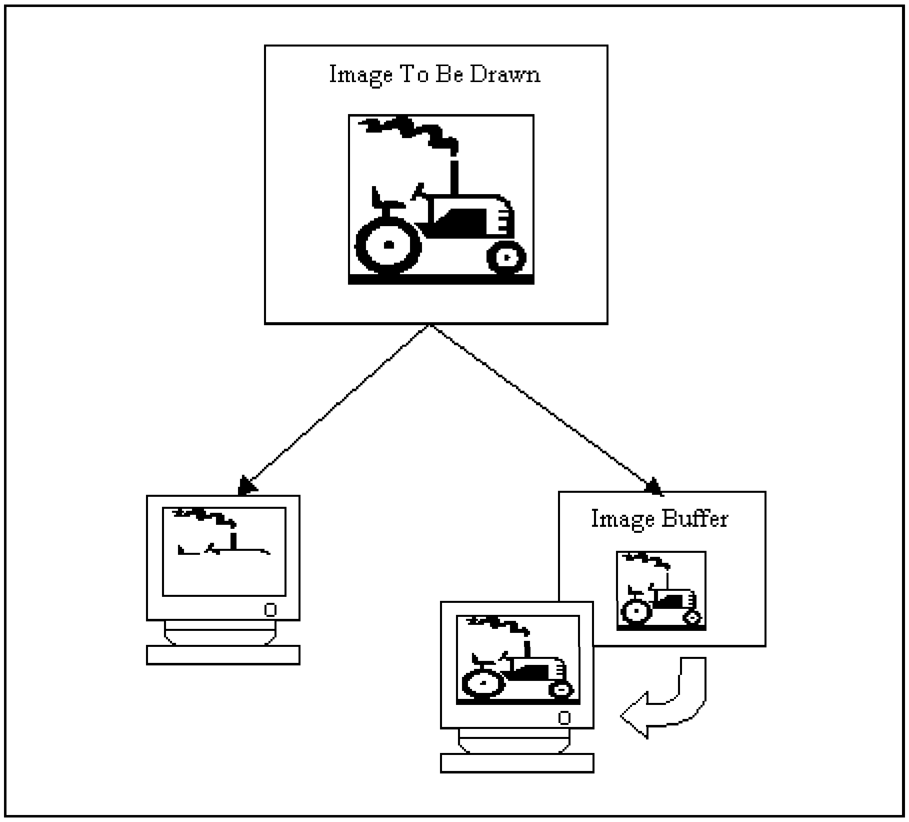


Figure 4.3: Double buffering example

Double buffering simplifies the refreshing of the displayed image by removing the time that would normally be needed to recreate the displayed image. The off-screen buffer already contains the displayed image, so it can be drawn directly to the screen without having to recreate the image. The example in Figure 4.3 illustrates the effects of using double buffering on a line art image. Lets say we are creating an image of a tractor by drawing individual horizontal lines. If we were to draw directly to the screen, the drawing process would be visible to the user and the I/O processing would be extensive. If the image needed to be refreshed for any reason, the image would have to be recreated, again adding extra waiting time. If we were to instead use an off-screen image in which to draw the image, we would be removing all of the I/O operations that the other method incorporates. Instead one I/O action would be needed to display the buffered off-screen image to the screen, which allows the user to see the image displayed at once, instead of line by line. When the image needs to be refreshed, the buffered image can once again be displayed to the screen, without having to recreate the image from scratch, thus saving the user from waiting for the image to be recreated.

To further increase efficiency the BOMD incorporates the idea of clipping. Clipping is the idea of limiting the modifiable area of the screen to some given subset called the clipping area. By limiting modifications to a small subset of the screen, updates can be done faster and more efficiently.

The second part of the first task is to incorporate new objects into the displayed image. This process is accomplished by rebuilding the image from all stored drawing objects. A new object is first inserted and then all objects are drawn to the off-screen buffer in chronological order of when they were created. The buffer is then displayed to the screen.

The second task of the BODM is the management of the storage of all drawing objects that are incorporated within the current image. The BODM uses two storage data structures. The first structure is used for the storage of all locally created objects and the second for storing all external drawing objects (e.g. objects downloaded from the server). While using one data structure for the storage of the objects would work, it would have added extra processing time to determine whether or not an object was created locally. Using two data structures, this determination is eliminated.

4.5.3 Advanced Object and Network Manager

The Advanced Object and Network Manager (AONM) is the highest-level manager in the client core. It adds the ability to maintain animation objects and all network capabilities such as uploading and downloading object data to and from the server.

The Animation Generator generates all of the parameters to create an animated object, leaving it up to the AONM to create the animation. Once the animation is created it is the AONM's responsibility to provide the animation object with animation capabilities. The responsibility also falls on the AONM to recreate an animation object from a stored state in the event that an animation object is downloaded from the server.

The name of the layer describes it as an Advanced Object manager. The lower two layers only manage objects and tools that relate to drawing images onto the screen. So the AONM must manage all non-drawing related tools, such as removing objects from an image. To allow the AONM this level of control it must have the ability to override the controls of the lower two levels.

The AONM also controls all network activity between the client application and the server. When an update is requested, the AONM first establishes a socket connection with the server. Second, all newly created drawing and animation objects are uploaded to the server. Finally, the AONM downloads any new object data from the server. All of the non-animated objects are passed to the lower layers for processing.

4.5.4 Tool Selector

The Tool Selector is a simple user interface module that allows a user to set the current active drawing tool. It passes the selected drawing tool information to the core layers for processing.

4.5.5 Color Selector

Similarly to the Tool Selector the Color Selector is a simple user interface module that allows a user to select the current drawing color. When a user selects a color, the Color Selector sends the color information to the core layers for processing.

4.5.6 Animation Generator

The main purpose of the Animation Generator is to generate all parameters and options used to create an animation object. The parameters and options are generated through an animation specific user interface that provides the user with access to the various animation parameters. When a user requests the creation of an animated object the Animation Generator sends the parameter and option data to the AONM layer where the data is used to create a new animation object.

4.5.7 Drawing Objects

The Drawing objects provide the user with the ability to control the development of the graphical images. There are two basic types of drawing objects: the animated objects and the non-animated objects. Using these objects the user is able to add lines, circles, animated cubes and much more to the collaborative image.

4.5.7.1 Non-Animated

The non-animated objects are static objects whose attributes do not change over time. An example of a non-animated object is a drawn line. Once the line is added to the image, it does not change its shape, color or location.

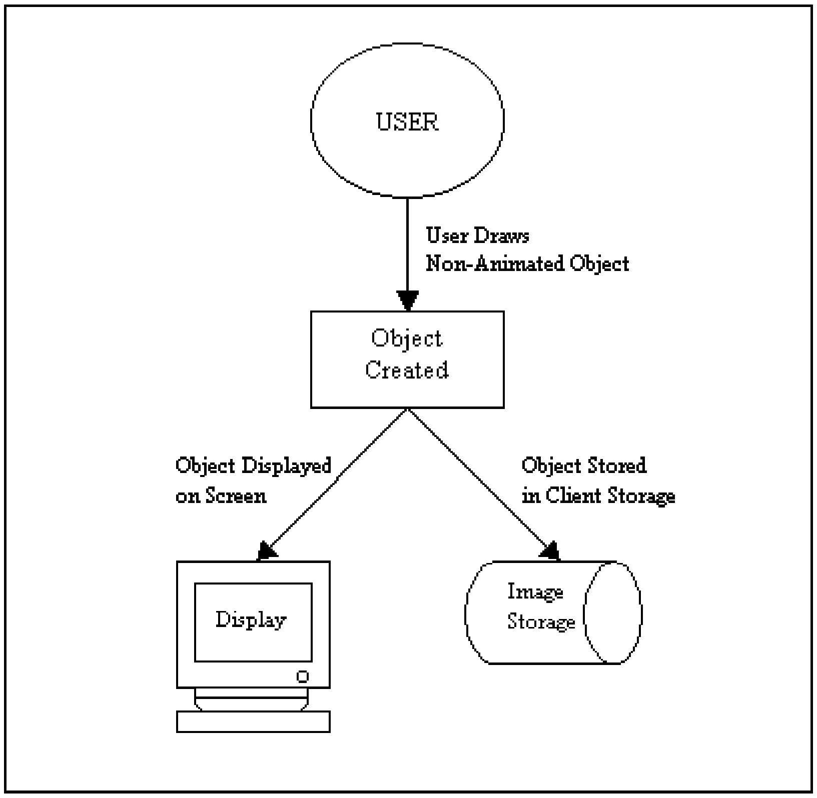


Figure 4.4: Flow chart of non-animated object creation

The design of the non-animated objects is relatively simple. Non-animated objects are "drawn" by the user onto the current image using a drawing tool. Drawing is defined as the means of using a mouse pointer to control the physical creation of an object in the image. When the user draws an object, the drawing tool creates the corresponding object. The object is then displayed to the screen and stored in the clients Image Storage buffer.

4.5.7.2 Animated

Animated objects are those objects whose attributes and/or locations change with time. An example of an animated object is a rotating cube. Once the rotating cube is added to the image, its image is continuously changing to give the effect of a 3-Dimensional rotating cube.

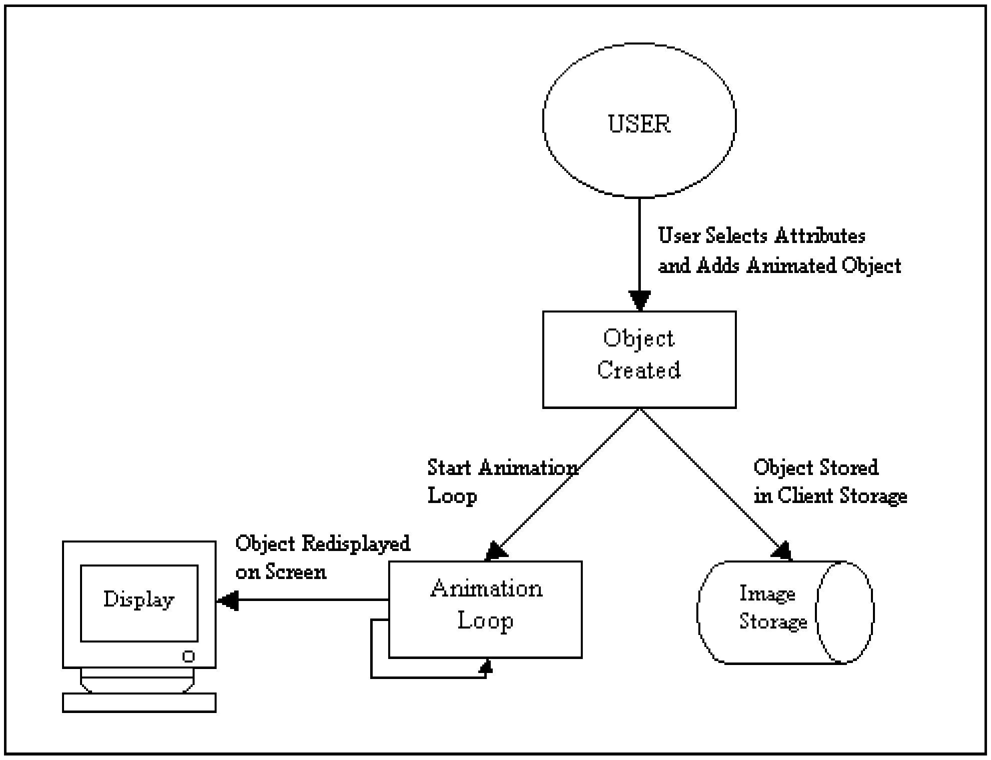

USER
User Selects Attributes and Adds Animated Object
Object Created
Start Animation Loop
Object Stored in Client Storage
Object Redisplayed on Screen
Display
Animation Loop
Image Storage

Figure 4.5: Flow chart of animated object creation

The design of an animated object is slightly more complex than a non-animated object. Unlike with a non-animated object, an animated object is not "drawn" to the screen by the user. Instead, the user sets or selects the different attributes of the animated object and then adds the object to the image. When the object is added to the image it is stored in the same manner as the non-animated objects, but instead of being displayed directly onto the screen an animation loop is started. The animation loop displays the animated object on the screen and after a timeout period it refreshes the displayed image of the animated object. This timed refresh gives the object the ability to be animated by continuously updating its appearance.

4.5.8 User Interface

The User Interface for the client application is based on the idea of user familiarity. If the user interface of an application is similar to the interface of an application the user is familiar with, the user will have a much easier time learning to use the new application. Following this idea, I have based my interface design on the interface of the Microsoft

Paint application. A vast majority of users have either used or seen this application making it a good basis for the interface.

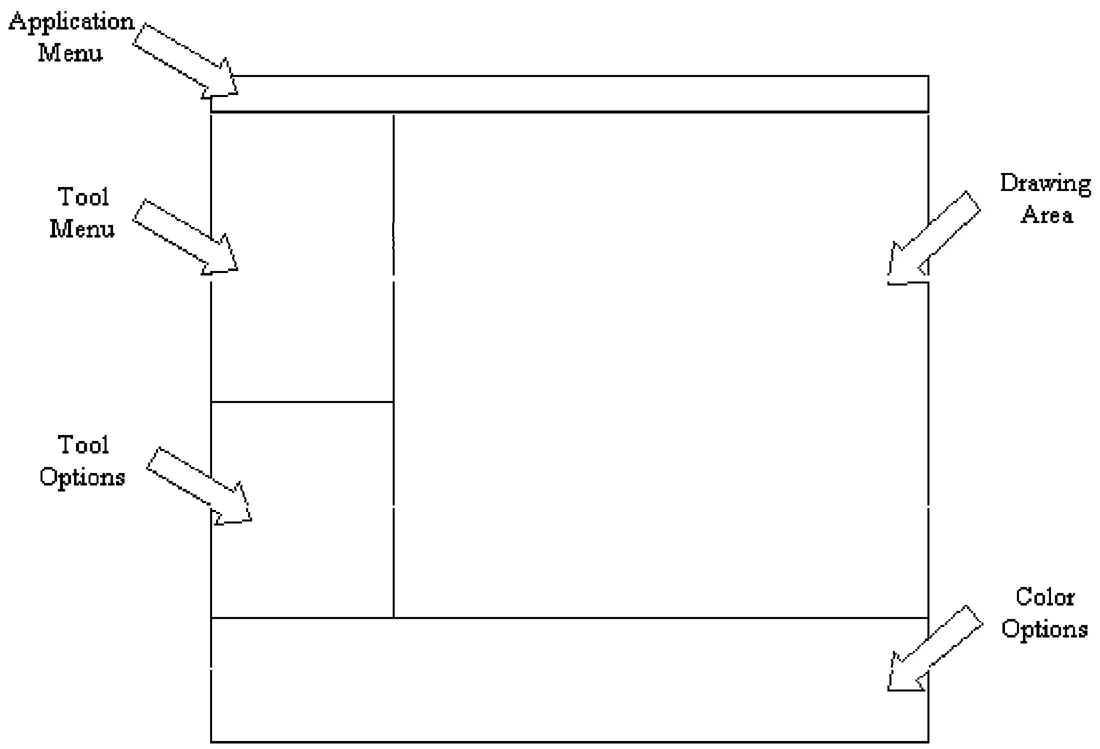


Figure 4.6: Microsoft paint user interface broken down to its essentials

Figure 4.6 illustrates the basic layout of the Microsoft Paint application. As with most applications the top section of the user interface is an application menu. This menu is normally a drop down menu, but it may also include optional toolbars. The left side of the application is dedicated to drawing tools. Most commercial drawing applications default to having the tool bar on the left side of the screen as well. In Paint the tool options lie directly below the tools themselves. This location makes the options easily accessible to the user. The bottom of the application contains options for changing the drawing color. The final area is the drawing portion of the user interface. As seen in the image, this is the largest portion of the user interface.

A benefit to using this style of interface is the accessibility of all options. The majority of the tools and options are included in either the tool or color option sections. This placement allows the user to quickly use and understand the application.

4.6 Server Design

The server provides the storage interface and inter-client communication for this project. Its main purpose is to wait for requests from the clients and to then perform any action that is needed, such as storing and retrieving image data.

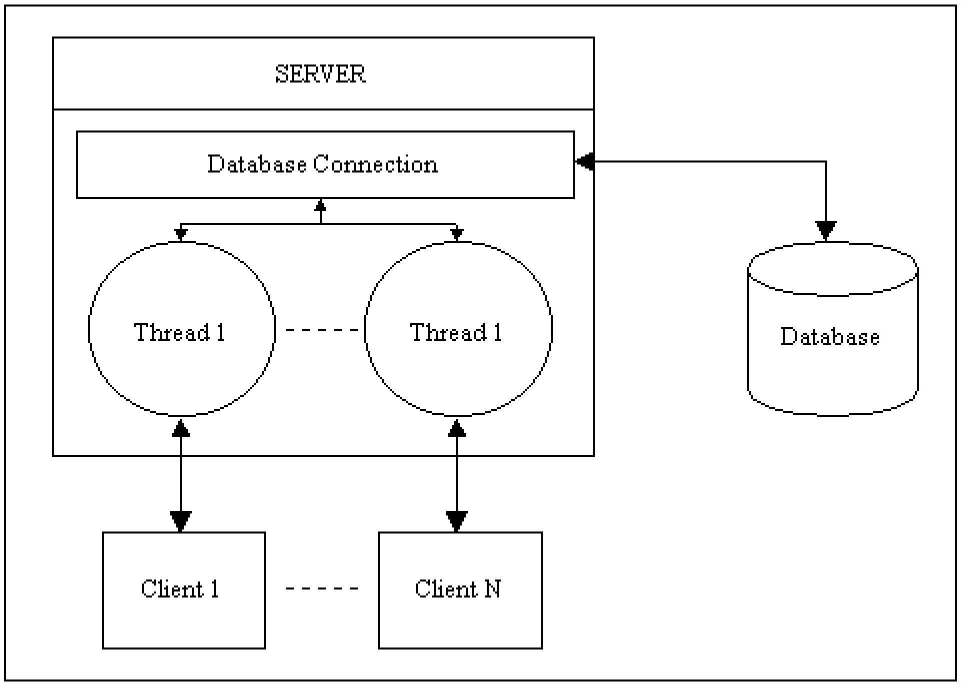


Figure 4.7: Server architecture

The server is designed as a multi-threaded server. As can be seen in Figure 4.7, a new thread is created to service each client that connects to the server. These threads have the ability to work in parallel with each other in order to process the requests of multiple clients simultaneously. While each thread work independently of each other, each one communicates to the database through the same database connection. This connection is the only connection that the threads have to the database. This single connection limits some of the problems, such as race conditions, that arise from multiple processes interacting with one data source.

4.6.1 Operations

The server performs a variety of operations that can be invoked by each client connected to the server. These operations are: the creation of a client ID, return of a time synchronization value, retrieval of the names of all editable images, and the updating and retrieval of image data. These operations give the clients the façade that they are interacting directly with the database, when it is the server that performs the actual interaction with the database.

1. Create a Client ID

The client ID is a unique identifier that the server gives to each client. This identifier allows the server and the other clients to differentiate between the data that each client adds to the collaborative image.

2. Return a Time Synchronization Value

The use of animated objects, whose locations depend on when they were created, generates the need for each client to maintain the same time regardless of time zone or computer time. The Time Synchronization Value (TSV) creates a global time synchronization between the server and the clients. Taking the difference between the time on the client and server generates the TSV.

3. Retrieve Editable Image Names

The server has the ability to retrieve all editable image names. This ability gives the client the capability of determining what the current images in the database are and to select one to modify.

4. Update / Retrieve Image Data

These operations provide the persistent storage abilities for the client. On an update the server will retrieve all information from the client and after validating the input, store it to the database. For retrieval operations, the server retrieves the specified image data from the database and then sends that data back to the client.

4.7 Database

Is there really a need for a database in this project? The answer is both yes and no. The project could be implemented without using any form of a database, with the server storing and retrieving information from internal data structures.

There are some rather imposing limitations on the database free scheme. The first limitation of the scheme would be the complexity of the server. To provide the data management, such as sorting and retrieving specific information, the server would have to become much larger and complex. This enlargement could lead to code that is harder to maintain and debug. A second limitation is the loss of reliable persistent storage. If the server was shutdown for any reason, the information within the internal data structures would be lost. Even if the server stored information to a file, the file could become corrupt or erased by the server crashing during a read or write routine.

The database scheme does have its limitations as well. The biggest one being the server has little or no control over the storage methods of the database. However the database scheme has advantages that outweigh its disadvantages. Using an external database removes all data management from the server. The server can easily store and retrieve specific information from the database without adding to the complexity and size of the server itself. Perhaps the foremost advantage to using a database is its ability to provide persistent storage. If the server goes down for any reason, the database will not be affected. The information that was stored in the database can then be easily retrieved when the server is reconnected.

The more important issue in the design of the database is the type of information that needs to be stored. The database needs to maintain information on when objects were added to the database and when they were removed from the collaborative images. This information provides the clients with the ability to accurately update the image that is being displayed. The database also needs to keep track of which image each object being added to the database is related to. While this information may be already contained within the object data, this additional storage will provide for faster and more efficient searching of the database for objects contained within a particular image. Along with all of this other information, the database also needs to store the individual object data. This information is what allows the clients to recreate each image from the stored information.

4.8 Communication

4.8.1 Client / Server

A client communicates with the server through the user of a TCP/IP connection. This connection provides an error free and reliable connection between the two. All communication between the client and server is done through the use of data frames. These frames encompass any commands and data that need to be transferred. The frames use character delimiters to specify the start and end of the frame and also the internal components. The frames also make use of character stuffing to prevent incorrect parsing of the communication frame.

4.8.2 Server / Database

The server connects to the Oracle database through the java Database Connectivity (JDBC) API. Sun Microsystems describes the JDBC technology as:

> "JDBC^{TM} technology is an API that lets you access virtually any tabular data source from the Java^{TM} programming language. It provides cross-DBMS connectivity to a wide range of SQL databases, and now, with the new JDBC API, it also provides access to other tabular data sources, such as spreadsheets or flat files. "[Sun00]

The JDBC API requires a driver to mediate between the JDBC technology and the database. In most cases this driver can be downloaded from the homepage of the database provider. The JDBC driver for the Oracle database is available directly from the Oracle website. The JDBC API provides access to the database through SQL queries. The application using JDBC technology is able to send queries to the database and receive results from those queries.

4.9 Evaluation

An effective evaluation is essential in providing an accurate look at the usefulness and quality of the InterDraw project. The evaluation is broken into two parts. The first part of the evaluation is user testing. User testing provides feedback on the usefulness and durability of the program under real world conditions. Another purpose is to facilitate in the discovery and repair of previously undiscovered bugs in the application.

The second part of the evaluation is a user survey. The survey provides feedback on the users and their views of the InterDraw program. The survey is split into three sections. The first section includes general demographic information about the user. The second part of the survey is questions that rate the effectiveness, quality and usefulness of the program. The final section is a user opinion section, where the user can give ideas on how to improve the program and what they liked or didn't like about it.

5 Implementation

This chapter describes an interactive collaborative art program. We have developed a program, InterDraw, that provides a collaborative environment where users can combine their artistic talents with others to develop unique graphical images.

5.1 Client Implementation

As described in the design section there are three layers that make up the core of the client application. These sections are the Drawing Tool Manager, the Basic Object and Display Manager and the Advanced Object and Network Manager. The majority of the client's functionality is implemented in these three layers. This section focuses on the implementation of these sections, while also touching on the implementation of the non-core areas of the client application.

5.1.1 Drawing Tool Manager

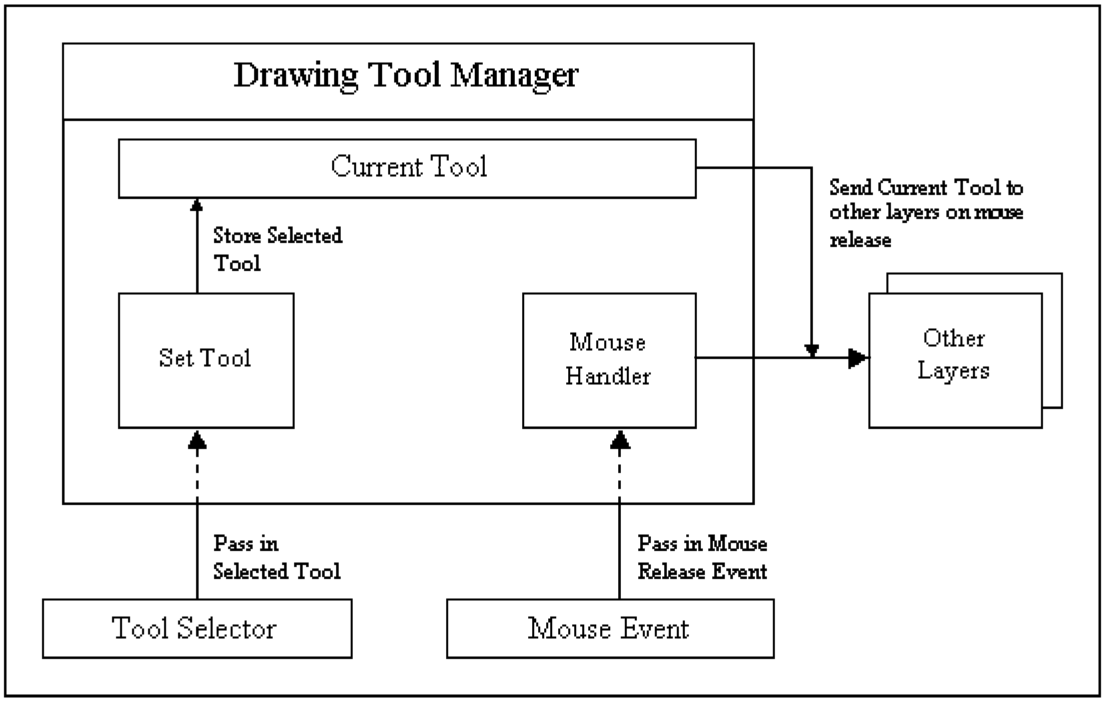


Figure 5.1: Flow layout of the Drawing Tool Manager

The Drawing Tool Manager (DTM) is the simplest of the three core layers. As described in the design, the DTM must store the current drawing tool and then monitor and send tool actions to the other layers. It turns out that this job is greatly simplified by having the tools manage most of their own actions (see later sections for more details).

The DTM has two major components: the Mouse Handler and the Set Tool component (see Figure 5.1). The objective of the Set Tool module is to update the Current Tool any new tool that is passed to the component. This tool comes from the Tool Selector and is generated by the user selecting a new tool to "draw" with. The selected tool is passed from the Tool Selector to the Set Tool component. When the new tool is received by the Set Tool component, the tool that is currently set as the Current Tool is set as inactive and the tool the component just received is set as active and stored as the Current Tool.

The second component of the DTM is the Mouse Handler. This component monitors all mouse events that occur in the application. Although all Mouse Events are received, the Mouse Release Event is the only event on which the Mouse Handler takes action. When the Current Tool is active, the receipt of a Mouse Release Event signifies the creation of a new object in the current image. On receiving the event, the DTM is prompted to send the Current Tool to the other layers for further processing.

5.1.2 Basic Object and Display Manager

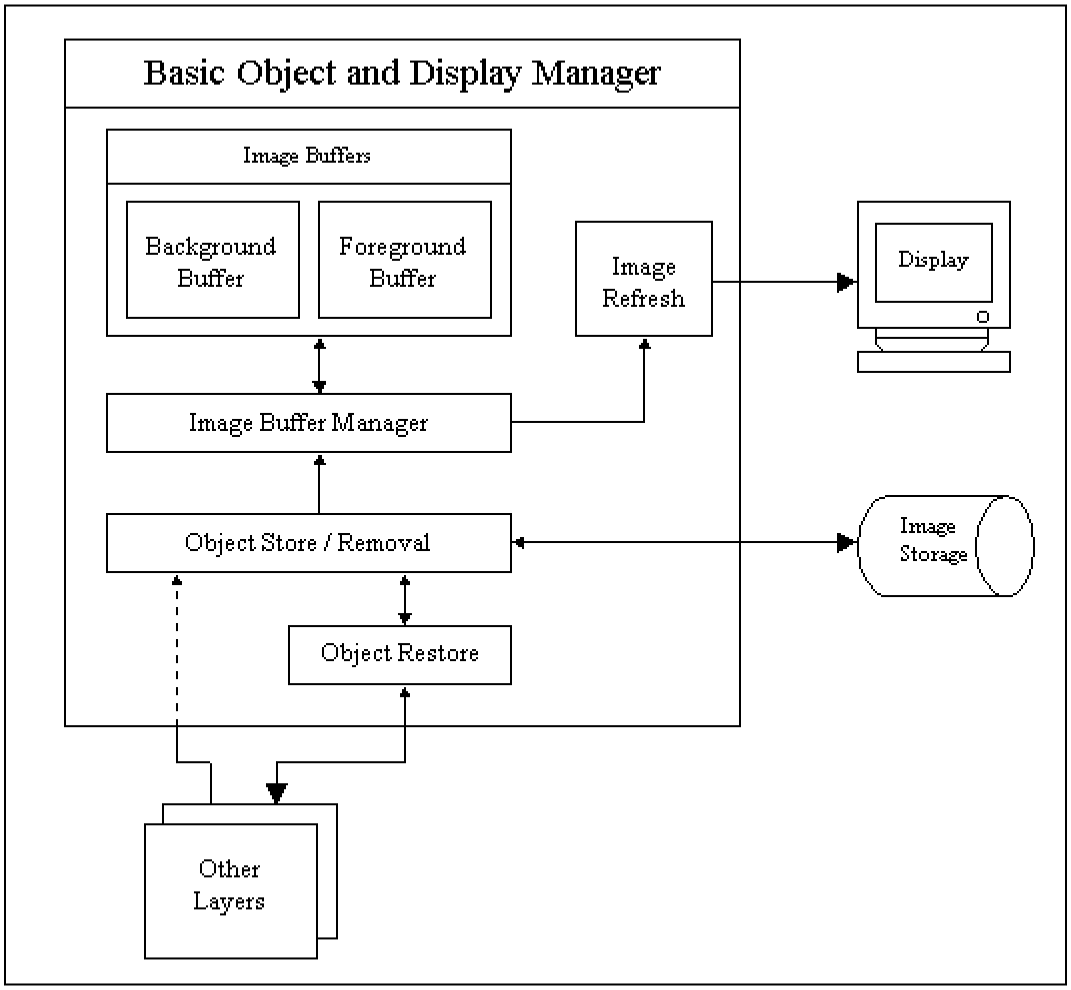


Figure 5.2: Flow chart of the Basic Object and Display Manager

As previously mentioned in the design section, the Basic Object and Display Manager (BODM) has two main tasks. The first task is to manage the displayed image and the second to manage the storage of all objects that make up the current image.

The BODM incorporates a double buffering scheme in its implementation. Two buffers are used: one for background graphics and another for foreground graphics (see Figure 5.2). The background buffer is used to store static or unchanging graphics. The foreground buffer is used for the display of temporary or dynamic graphics. The graphics in this buffer can be overwritten with the background buffer to remove unwanted graphics that appear in the foreground buffer. The reason for this buffer is to allow for the use of animated graphics. If an animated image was drawn to the background buffer, all of the underlying graphics would be lost, but if drawn to the foreground buffer, all the background graphics are maintained. Since the background graphics are being retained, the animated graphic can change location by simply overwriting the foreground buffer with the background buffer and redrawing the animated graphic in a different location in the foreground buffer.

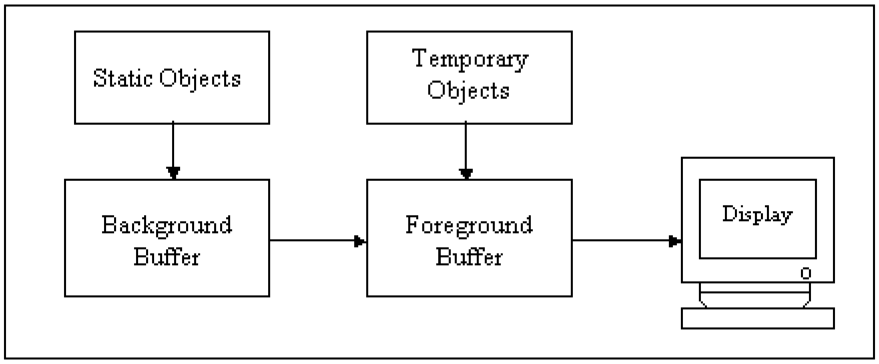


Figure 5.3: Flow chart of graphics from creation to being displayed on the screen

The management of the image buffers is provided by the Image Buffer Manager component of the BODM. Its primary objectives are to maintain the current state of each buffer and to update the display when needed. When a static object is created, it is initially drawn into the background buffer. The Image Buffer Manager then transfers the graphic data from the background buffer to the foreground buffer. When the image needs to be displayed, the foreground buffer is drawn onto the display. The foreground buffer can then be used as described in section 4.5.2. The reason behind the use of two buffers becomes apparent when temporary or dynamic graphics are introduced into the image. As mentioned earlier, all temporary graphics are drawn directly to the foreground buffer. When this graphic needs to be removed or changed, it can easily be "erased" by having the Image Buffer Manager overwrite the area in the foreground buffer encompassed by the graphic with the related area in the background buffer. It is very similar to a partial display refresh, where only part of the display needs to be redrawn. Instead of redisplaying the whole image, the Image Buffer Manager redraws only the image portions contained within the specified area.

The BODM also manages the storage, removal and the restoration of image objects. The storage of objects occur when the DTM signals the end of a drawing tool action, at the request of another layer, or after the after an image object is restored. The removal and restoration of an object is done only as the result of a request from another layer.

The storing of an object due to a DTM signal is the most common storage procedure. The BODM retrieves a storable representation of a newly generated drawing object from the Drawing Tool returned by the DTM signal. The storage representation format is described in section 5.5. This storable representation is then stored in the Image Storage, which implements a Linked List data structure. A Linked List data structure was used for its ability to easily change the order of the stored objects. Once the representation is stored, the DTM sends the graphical information of the object to the Image Buffer Manager for display.

Storage of an object on the request of another layer is similar to the last method described. In this event the drawing object is sent rather than the Drawing Tool that created the object. The BODM is able to retrieve the graphical information and storage representation directly from the drawing object, which are then handled in the same manner as above.

Object restoration is one of the most important features of the BODM. It gives the client application the ability to dynamically build an image from the stored representation of the drawing objects. When another layer requests that an object needs to be restored, the storage representation of the object is passed to the BODM. If the BODM does not recognize the drawing object represented by the storage representation (such is the case with an animated object), the storage representation is resent to the Advanced Object and Network Manager which has more advanced restoration capabilities. For known drawing objects, the BODM stores the object in Image Storage and recreates its graphical representation by using restoration properties of the related drawing object. The graphical representation is then sent to the Image Buffer Manager for display.

The final ability of the BODM is the removal of objects from the image. When another layer requests the removal of an object, either the object’s storage representation or its Object ID that is embedded within each storage representation (see section 5.5 for more information about the Object ID) must be passed to the BODM. The corresponding objects are then removed from Image Storage. If the object being removed is a static object, then the background buffer must be rebuilt by restoring all objects within the area previously occupied by the removed object.

5.1.3 Advanced Object and Network Manager

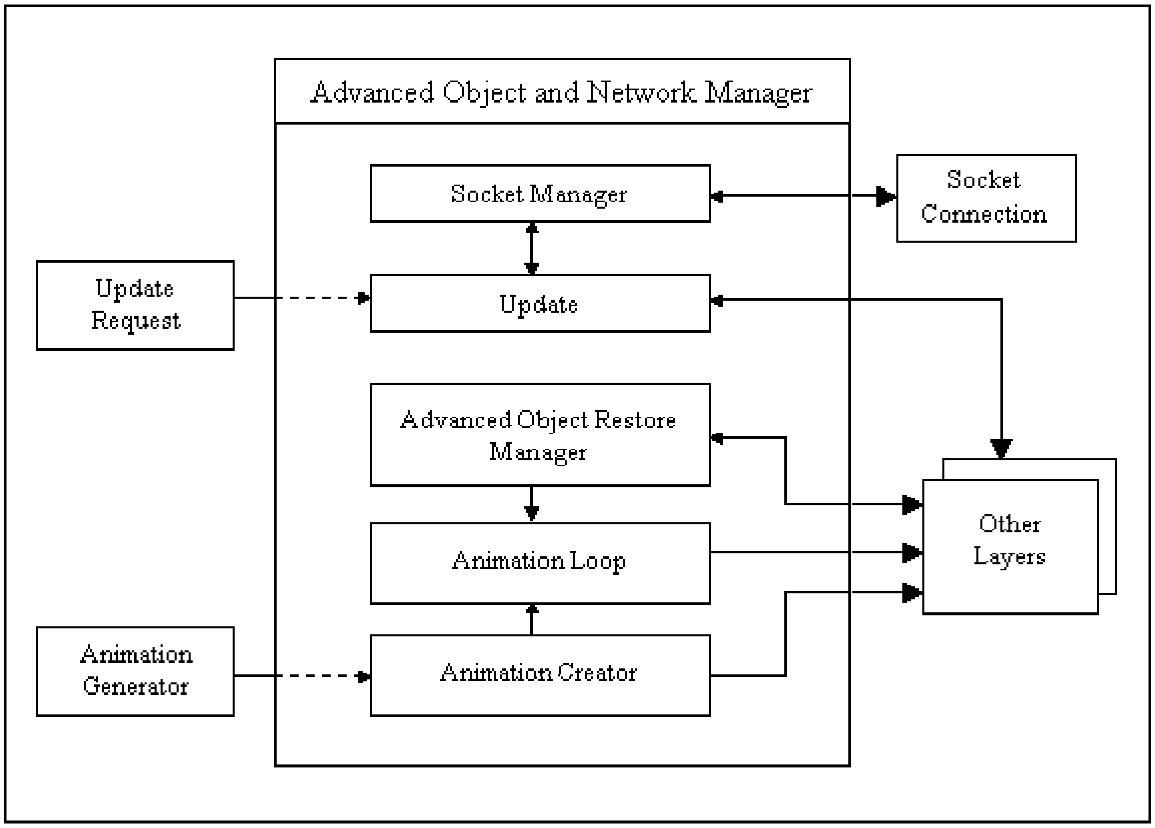


Figure 5.4: Flow chart of the Advanced Object and Network Manager

The Advanced Object and Network Manager (AONM) manages the creation, restoration and the animations of animation objects. It also provides the networking capabilities of uploading and downloading object data to and from the server.

As illustrated in Figure 5.4 the AONM is broken down into 5 component; the Socket Manager, the Advanced Object Restore Manager, the Animation Creator, an Animation Loop and finally an Update component. The rest of this section describes these individual components in more detail.

1. Socket Manager

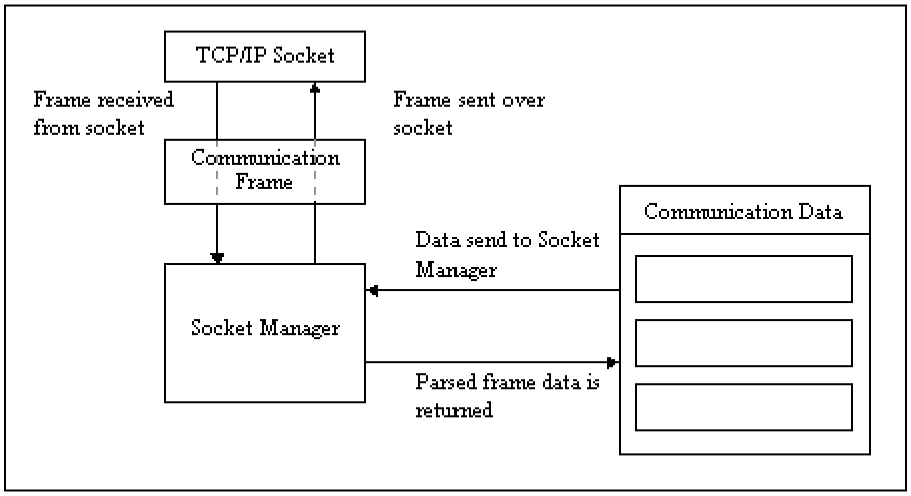


Figure 5.5: Network communication using the Socket Manager

The Socket Manager maintains a TCP/IP socket connection with the server and oversees all communication over that connection. When sending data over the socket connection the Socket Manager receives the object data, command type and other information needed to build the communication frame (see section 5.4.1 for information on the communication frame). When all required information is received the Socket Manager builds the communication frame and transmits it to the server over the socket connection. Similarly, when a communication frame is received from the socket connection the Socket Manager parses the frame and returns the individual data inside to the requesting component.

2. Advanced Object Restore Manager

The Advanced Object Restore Manager is very similar to the restore component of the BODM. The difference is in the type of objects that can be restored within the components. While the BODM's restore component can only restore basic object types (static), the Advanced Object Restore Manager can restore all animated objects and any new object types that may be added in the future.

When a request is made to restore an object, either from another layer or another module within the AONM, the restore ability of the drawing object is used to recreate the graphical representation of the object. Both the graphical representation and the storage representation are then sent to the BODM for storage and display. If the object being restored is an animated object, the Animation Loop takes over the animation of the object.

3. Animation Creator

The Animation Creator has the responsibility of creating animation objects. It receives parameter information from the Animation Generator that is needed to create each individual animation object. The parameter information contains specific information about the animation's class, such as its class name, constructor and individual parameters. This information is then used to dynamically construct the animation object.

When the animation object has been created, the graphical and storage representations of the objects are sent to the BODM for display and storage. Control of the animation object is then transferred to the Animation Loop, which takes care of the animation of the object.

4. Animation Loop

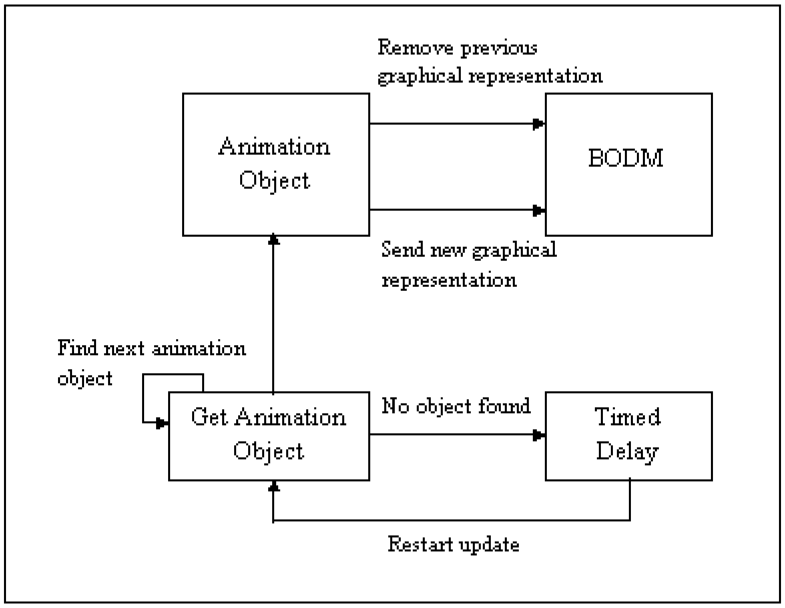

Remove previous graphical representation
Animation Object
BODM
Send new graphical representation
Find next animation object
Get Animation Object
No object found
Timed Delay
Restart update

Figure 5.6: Flow chart of animation loop

The Animation Loop is responsible for providing animation objects with animation capabilities. It provides these capabilities by periodically having the animation object refresh its image on the display. The first thing the Animation Loop does is to iterate through each animation object within the current image. For each animation object found, the Animation Loop tells the object to refresh its current image. This request prompts the animation object to send a request to the BODM to remove its previous graphical representation from the image and to update the image with its new image. When the Animation Loop has iterated through all of the animation objects in the current image, a time delay of 100 milliseconds is invoked. After the time delay expires the Animation Loop starts over. The 100 milliseconds time delay was derived from the testing of various time delays on the performance of the animated objects.

It is possible to have each animation object maintain its own animation abilities. The problem with this course of action is the amount of memory required. Each of the animation objects in the image would have to have its own timing thread running in the background. As more objects are added, the number of threads increases. Over time, this could cause the application to slow considerable. The implementation I chose does not have this problem, as there is only one thread updating all of the animated objects. The downside of using this implementation is that the refresh rate of the animated objects may become larger as more animated objects are added to the image.

5. Update Component

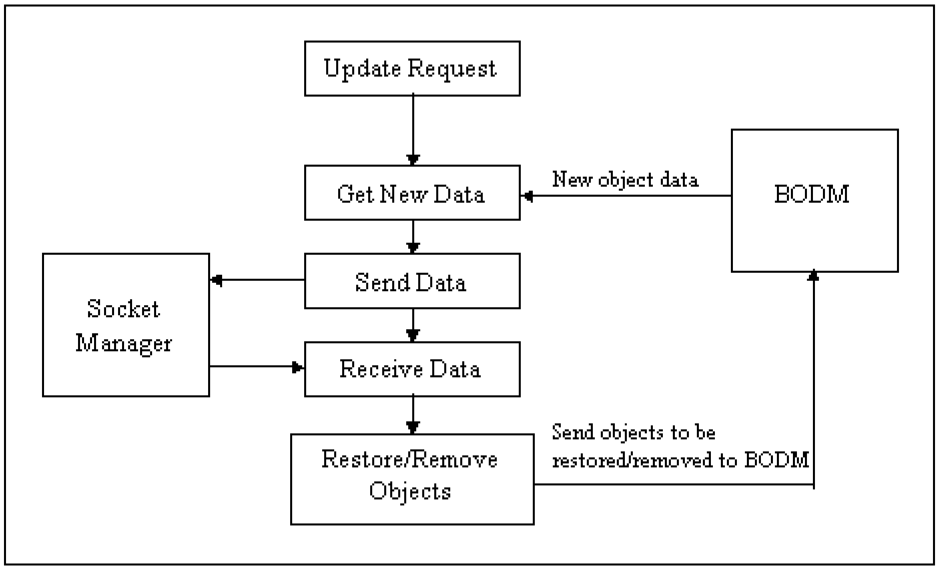


Figure 5.7: Flow chart of image update in Update Component

The Update Component is responsible for instigating the sending of new image data to the server and for updating the image with new image data received from the server. Updates are processed whenever an update request is received. An update request can be the result of user interaction with the program and specifically requesting an update, or the request can come from a background thread running in the background that periodically updates the image.

When an update request is received, the Update Component retrieves, from the BODM, object data that has been added to the image since the last update. The Update Component then passes the data on to the Socket Manager for sending to the server. Next, data is received from the server through the Socket Manager. If there are objects to be removed or restored, they are sent to the BODM for processing.

5.1.4 Tool Selector

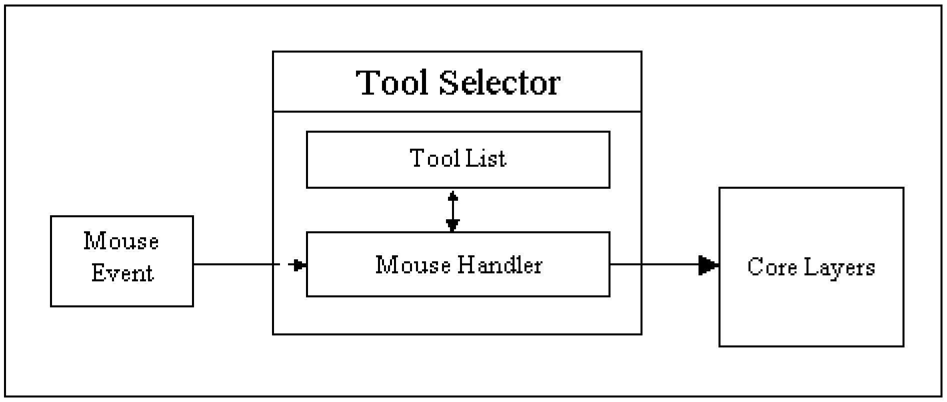


Figure 5.8: Flow chart of Tool Selector

The Tool Selector is a simple user interface module that allows a user to set the current active drawing tool. It is composed of two components: a Tool List and a Mouse Handler. The tool list is a listing of all tools and their selectable buttons. The Mouse Handler processes all Mouse Events. When a Mouse Release event is processed, the Tool Selector checks to see if one of the tools in the Tool List has been selected. If the location of the Mouse Release event is within the area encompassed by one of the tool's selectable buttons, then that was selected. The selected tool is then sent to the core layers of the client application for processing.

5.1.5 Color Selector

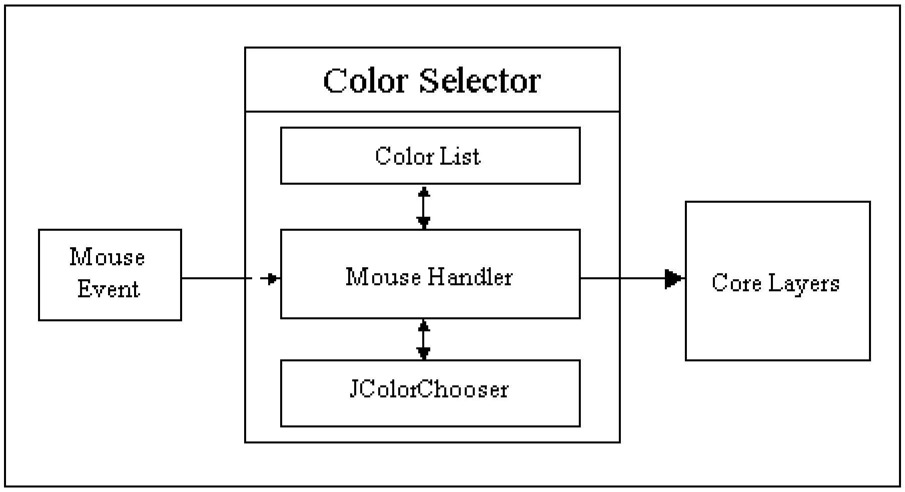


Figure 5.9: Flow chart for Color Selector

Similarly to the Tool Selector the Color Selector is a simple user interface module that allows a user to select the current drawing color. As illustrated in Figure 5.9, the Color Selector is composed of 3 primary modules: a Color List, a Mouse Handler and JColorChooser. The Color List is a list of basic colors and their selectable interfaces. The Mouse Handler is similar to the Mouse Handler component of the Tool Selector. When the Mouse Handler receives a Mouse Release event, the color list is checked to see if a color has been selected. This process for checking selection is the same as described in the Tool Selection Mouse Handler. If a color has been selected, it is sent to the core layers of the client application for processing.

JColorChooser is a built-in Java Swing class for a complex color selection dialog that allows a user to create his or her own color by varying different color parameters. The Color Selector creates a selectable interface for the JColorChooser. If the user selects this interface, the JColorChooser dialog is created and displayed to the user. When the user selects a color, that color is then sent to the core layers for processing.

5.1.6 Animation Generator

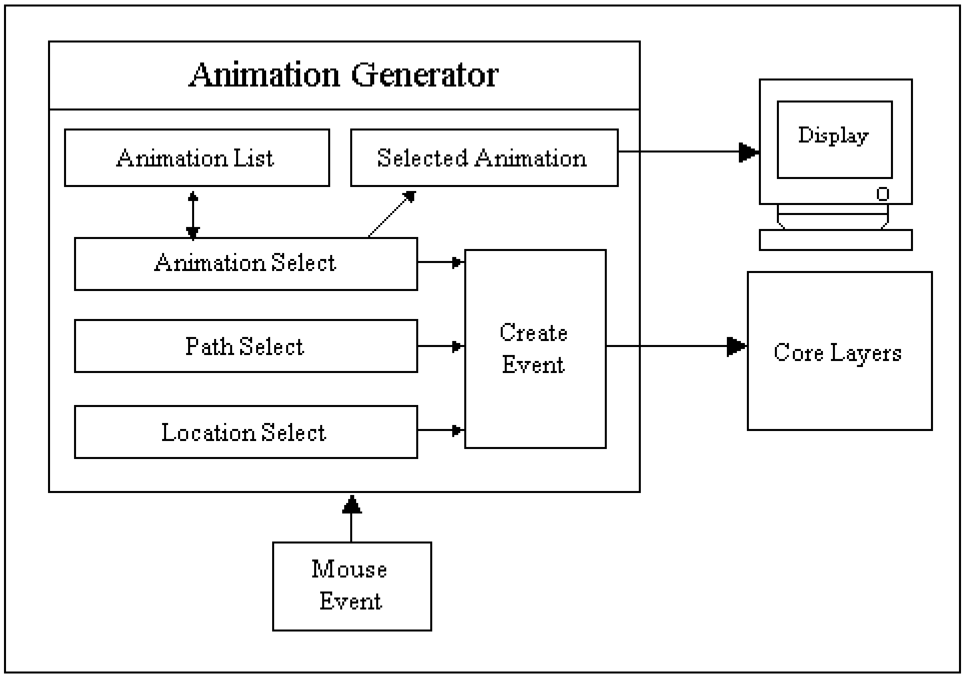


Figure 5.10: Flow chart for Animation Generator

The main purpose of the Animation Generator is to generate all parameters and options used to create an animation object. The Animation Generator uses three main components and a Create Event to generate the parameters and options needed for the creation of an animated object.

1. Animation Select

The Animation Select is a simple user interface that allows the user to select an animated object from a list of animated objects. The list is a simple 1-Dimensional array that contains different active animated objects. The use of an array allows the Animated Select to easily keep track of the selected animated object by its location in the array.

The interface generates two events: a NEXT event and a PREVIOUS event. These events are generated through the selection of Next ('>') and Previous ('<') selectable buttons.

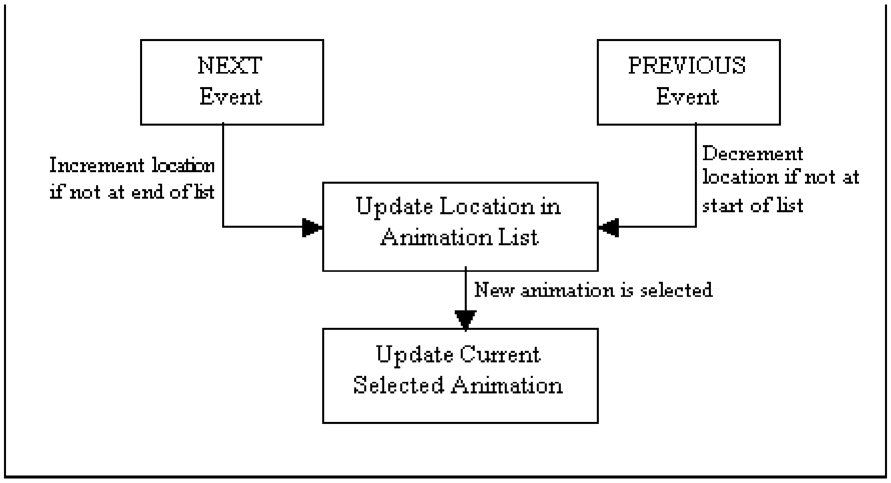

NEXT Event
PREVIOUS Event
Increment location if not at end of list
Decrement location if not at start of list
Update Location in Animation List
New animation is selected
Update Current Selected Animation

Figure 5.11: Flow chart of animated object selection

When a NEXT event occurs the Animation Select checks to see if the currently selected animated object is at the end of the selection array. If not, then the location is incremented and the next animated object is selected. When a new animation object is selected, that object is stored as the currently selected animation. The selected animation is added to the Animation Generator's display to allow the user a visual look at the animated object. The Animation Generator displays the animated object as described in the previous section, with one difference. Instead of using a double buffer implementation, the animated object is drawn directly to the screen. As the animated object is constantly changing, this simplification speeds up the display time of the animated object.

A PREVIOUS event is handled in the same way as a NEXT event, with the exception that the location is decremented instead of incremented.

2. Path Select

The Path Select creates a dialog that allows a user to select the path that the animation object will follow. For more detailed information on animation paths see section 5.1.7.2. Each path has a set of parameters that are selected by the user. For example, if the animation path were a line, the parameters are the direction of the line and the speed of which the animation object follows the path. The user can then choose to select the path. When this occurs, the selected path is returned to the Animation Generator to await further processing.

3. Location Select

Location Select is the simplest of the Animation Generator modules. Its responsibility is to select the starting location of the animated object on the image display area. When the user activates the Location Select, the Mouse Handler of the main drawing area is overwritten with the Mouse Handler of the Location Select. When the user uses the mouse to click on the main drawing screen, the Mouse Event for that action is received by the Location Select. The location of the Mouse Event is then returned by the Location Select as the starting location for the animated object.

4. Create Event

Create Event is the event that occurs when a user chooses to insert the selected animation object into the current image. When this event occurs, the values returned by the Location Select, Path Select and Animation Select are used to create class parameters for the selected animation object. These parameters are then passed to the core layers of the client application, which will then create the selected animation object.

5.1.7 Drawing Objects

As stated in the design section, there are two types of Drawing Objects; animated and non-animated. Although these types are different in nature, they contain the same basic functionality as shown in Figure 5.12.

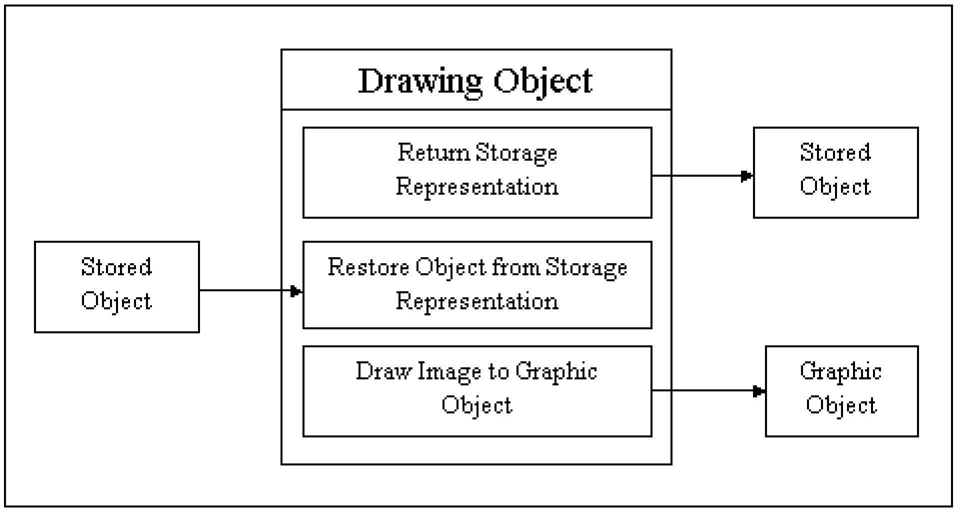

Drawing Object
Return Storage Representation
Restore Object from Storage Representation
Draw Image to Graphic Object
Stored Object
Stored Object
Graphic Object

Figure 5.12: Illustration of the basic functionality incorporated in all Drawing Objects

All Drawing Objects have the functionality of returning a storage representation of the drawing object, restoring the object from a stored representation and drawing its graphical image to a Graphic object. The StoredObject representation of the Drawing Objects is described in section 5.5. The ability of the Drawing Object to generate a stored representation and to then restore itself from that representation is one of the most important functions of a Drawing Object. Without this functionality it would be impossible to recreate an object without knowing everything about the object. This limitation would make it difficult to add new types of Drawing Objects without having to rewrite all restoration functions in the client application.

The ability to draw to a Graphic object is another key functionality of a Drawing Object. A Graphic object is an abstract class used in Java as an interface to graphical objects. The Graphic object contains methods for the direct manipulation of graphical objects. The ability to draw to a Graphic object is key, because it provides the Drawing Object with the ability to draw to an image buffer, directly to the screen or to any other object that can provide a Graphic object to the Drawing Object.

5.1.7.1 Non-Animated

Non-animated objects are static objects whose graphical representation does not change with time. Non-animated drawing objects come in two forms: free drawing objects and rubberband drawing objects.

1. Free Drawing Objects

Free Drawing Objects are used to represent drawing tools that require free form drawing abilities, such as airbrush painting and free hand drawing. In order to preserve the complex free form patterns, each point in the free form drawing must be maintained by the Free Drawing Object.

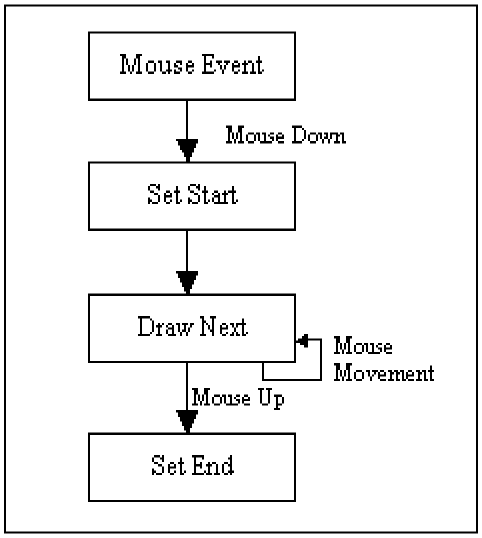


Figure 5.13: Overview of the Free Drawing Object drawing process

Figure 5.13 illustrates how a Free Drawing Object generates its graphical representation. The free form drawing of the Free Drawing Object is based on the receipt of Mouse Events. When a Mouse Down event is received by the Free Drawing Object, that location is marked as the starting location for the object. For each consecutive Mouse Movement event following the Mouse Down event a call is made to draw a path from the previous location to the new location returned in the Mouse Movement event. How this path is drawn is up to each implementation of a Free Drawing Object. In the case of free hand drawing, a simple line is drawn between the two points, but in airbrush painting a random scattering of points are drawn within a certain radius of the new point, giving it an airbrushed look.

When the Free Drawing Object receives a Mouse Up or Mouse Release event, one last path is created between the current and previous points. Following the completion of the drawing, the Free Drawing Object calculates the bounds of the new object and stores its creation time. All of this information will be used to create the object's storage representation when needed.

2. Rubberband Drawing Objects

Rubberband Objects are used to represent drawing objects that can be seen changing shape, like a rubber band, as the user draws them onto the screen. Examples of such objects are line, ellipse and rectangle drawings.

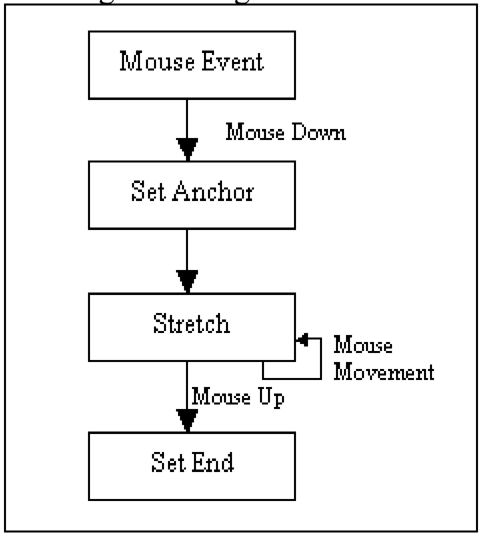


Figure 5.14: Overview of the Rubberband Drawing Object drawing process

Figure 5.14 illustrates how a Rubberband Object generates its graphical representation. The Rubberband object works very much like the Free Drawing Object with some minor, but important difference. As did the Free Drawing Object, the Rubberband Object is also based on the receipt of Mouse Events. When a Mouse Down event is received by the Rubberband Object, that location is marked as the anchor point for the rubberbanding image. For each consecutive Mouse Movement event following the Mouse Down event a call is made to stretch the object from its past location to its current one.

The stretching of the Rubberband Object is what sets it apart from the Free Drawing Object. To create the look of the object being stretched, the previous drawing of the object must be removed and a new one must be drawn. The removal of the previous drawing is implemented through the use of an XOR drawing state. Using this drawing state, if you overlap one image with another of the same color, the colors cancel each other out where the two images overlap. So, in order to erase the previous drawing of the Rubberband Object, the Rubberband Object is drawn in its previous location while in the XOR drawing state. This action effectively erases the previous image. The drawing state is then changed back to its normal drawing state and the Rubberband Object is drawn using the bounds between the anchor point and the new point retrieved by the Mouse Movement event. As with the Free Drawing Object what gets drawn depends on the implementation of the Rubberband Object. For example the rubberbanding object could be a line, rectangle or ellipse.

When the Rubberband Object receives a Mouse Up or Mouse Release event, the last location is once again removed and the final location of the object is drawn. Following the completion of the drawing, the Rubberband Object calculates the bounds of the new object and stores its creation time. All of this information will be used to create the object's storage representation when it is requested.

5.1.7.2 Animated

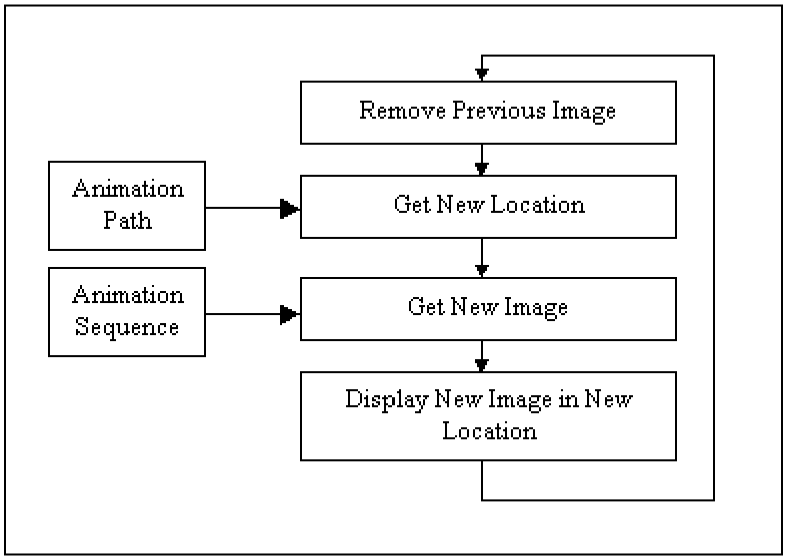


Figure 5.15: Flow chart of the animation process of an Animated object

Animation objects incorporate two types of animation. The first type of animation is the ability of the objects to change their location on the image. The second type is the changing of the image displayed by the Animation object. To provide for these abilities the objects are composed of two main components: an Animation Path and an Animation Sequence. The Animation Path is used to generate the current screen location of the Animation object, and the Animation Sequence is used to generate the current image being displayed for the Animation object.

Figure 5.16 illustrates the animation process of an Animated object. The first step in the animation process is to remove the previous drawn image of the object. The removal of the previous image is implemented by sending a request to refresh the area around the Animated object's previous location. In the case of the Basic Object and Data Storage layer of the client, this request would result in the background buffer being drawn to the foreground buffer. The next step is to retrieve the object's new location, which is easily retrieved from the Animation Path. The new image, if any, is also retrieved from the Animation Sequence. The new image is then drawn to the drawing area containing the Animated object.

1. Animation Sequence

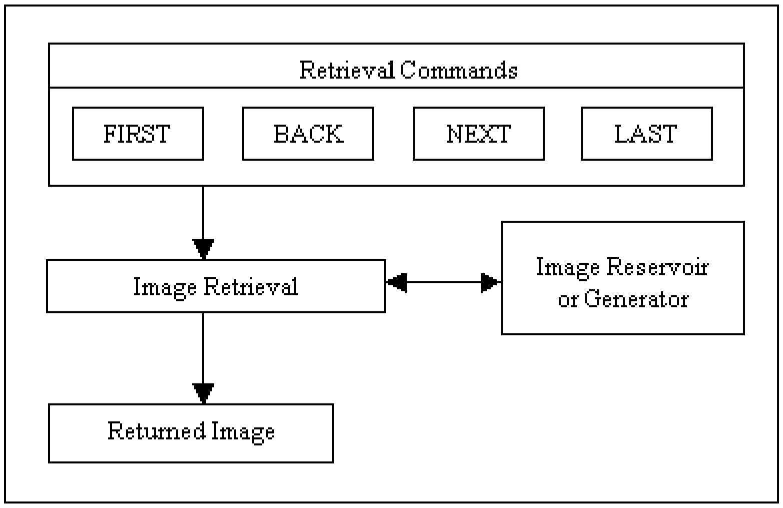


Figure 5.16: Image retrieval process of Animation Sequence object

The Animation Sequence is responsible for returning a sequence of images that compose an Animated object's animation. When an image is requested, an image is either retrieved from an internal image reservoir or created by an image generator. An image reservoir is used to create sequences of images that may be repeated many times. An image generator, on the other hand, provides the unique ability to create dynamically changing sequences of images. The implementation of the image generator is unique to each Animation Sequence, where it could be anything from the generation of random lines, to the creation of complex graphical images.

An Animated object interacts with the Sequence object though a series of image retrieval commands. These commands include the retrieval of the first, last, next and previous images. In the case where an image generator is being used, it may not be possible for the generator to recreate previous images, and as the image is dynamic a last image cannot be returned. When an image reservoir is used, these commands provide some control over the images that are returned to the Animated object.

2. Animation Path

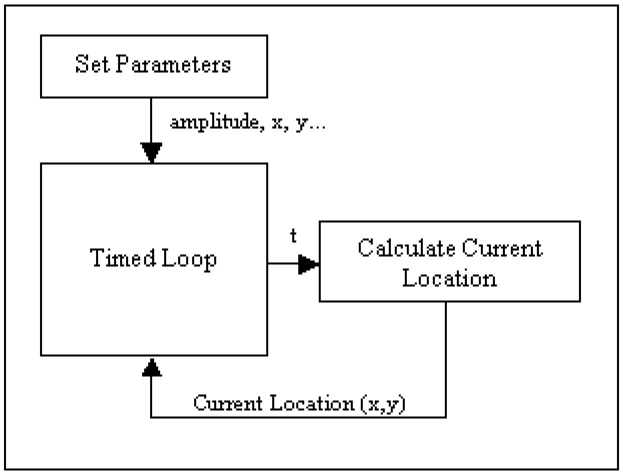


Figure 5.17: Animation Path process loop

The Animation Path is responsible for returning the current screen location of the Animated object. The Animated Path uses a sequence of time-based formulas to calculate the current location of the object. The process of calculating the location is illustrated in Figure 5.17. When the Animation Path is created, it is passed a starting location and various path dependant parameters. The Animation Path keeps track of time since it was created and passes that value to a time dependent formula that calculates the current location of the Animated object passed on how much time has passed since its creation. The formula used in the calculation of the location is dependent on the implementation of the Animation Path.

To keep the location calculated within view at all times, a few strategies were tried. The first strategy was to have the path bounce off of the borders of the drawing area where the path intersects a border. While this strategy works well for simple paths, for more complex paths the calculations become less feasible. A much simpler method was derived to work for both simple and complex paths. This method uses a wraparound technique to keep the location within the drawing area. When the location goes off one side of the image, it relocates to the other side of the drawing area as if all the borders of the drawing area are connected.

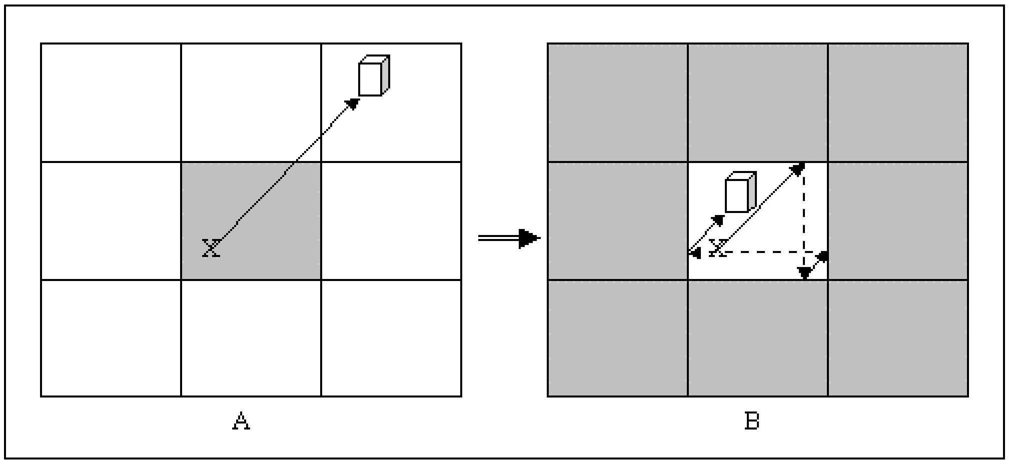


Figure 5.18: Example of before and after wraparound technique is applied to a line path

The shaded box of illustration A in Figure 5.18 represents the drawing area of the client application. The white boxes are projections of the drawing area into the surrounding regions. In illustration B this color scheme is reversed. The solid arrows in both images represent the path returned by the Animation Path. The dotted arrows represent the object wrapping around the drawing area and are not returned locations of the Animation Path.

The path in illustration A represents a non-wraparound path. As shown in the illustration the cube is no longer visible within the drawing area. In illustration B a wraparound path is demonstrated. When the cube reached the top border of the drawing area, it was projected down to the bottom of the drawing area. It then intersected with the right border of the drawing area and projected onto the left side of the drawing area. Notice that the final location of the cube is in the same relative area in the drawing area in illustration B as the projected drawing area in illustration B. This property of the wraparound scheme simplifies the calculations required to calculate the current location of the path.

The wraparound x and y values are calculated as follows:

```
$X_{wrapped} = X_{non\text{-}wrapped} \bmod DrawingArea_{width}$
$Y_{wrapped} = Y_{non\text{-}wrapped} \bmod DrawingArea_{height}$
```

To handle cases where either Xwrapped or Ywrapped is negative, the values are processed further:

```
if ($X_{wrapped} < 0$)
   $X_{wrapped} = DrawingArea_{width} + X_{wrapped}$
if ($Y_{wrapped} < 0$)
   $Y_{wrapped} = DrawingArea_{height} + Y_{wrapped}$
```

Three basic paths were implemented for this project: a line or vector path, an elliptical path and a sine wave path. Each path is discussed below

Line Path

$$\boxed{\begin{aligned} X(t) &= X_{initial} + V_X * t \\ Y(t) &= Y_{initial} + V_Y * t \end{aligned}}$$

Figure 5.19: Line Path location formula

A Line Path is a simple path based on a straight line or vector. The path starts at a starting location and moves away from the starting location at a constants speed and direction. A direction vector (V) is used to control the direction and speed of the path's movement. A larger X component in V will increase the speed that the path moves in the X direction and the same applies for the Y component of V.

Elliptical Path

$$\boxed{\begin{aligned} X(t) &= X_{origin} + X_{radius} * \sin(\Delta angle * t) \\ Y(t) &= Y_{origin} + Y_{radius} * \cos(\Delta angle * t) \end{aligned}}$$

Figure 5.20: Elliptical Path location formula

An Elliptical Path creates an elliptical orbit around a center origin location (X_{origin}, Y_{origin}). The elliptical orbit has an X_{radius} and Y_{radius} that control the height and width of the obit. The speed of the orbit is controlled by the change in angle (Δangle) at each time increment (*t*). As Δangle increases in size, the rate of obit increases as well.

Sine Wave Path

```
X(t) = X_initial + Δangle*t
Y(t) = Y_initial + amplitude*sin(frequency*Δangle*t)
```

Figure 5.21: Sine Wave Path location formula

A Sine Wave Path produces a path that follows the up and down pattern of a sine wave. Since we are using a sine wave for the path, the X value follows a vertical line, much in the same fashion as in the Line Path formula. The difference is the X value of the Sine Wave Path is incremented by a change in angle (Δangle) and not a vector value. The X value could follow any line vector, but due to time constrains only the vertical line traversal was implemented. As with the Elliptical Path, Δangle controls the speed of the path. Amplitude and frequency values control the height and length of each sine wave motion.

5.1.8 User Interface

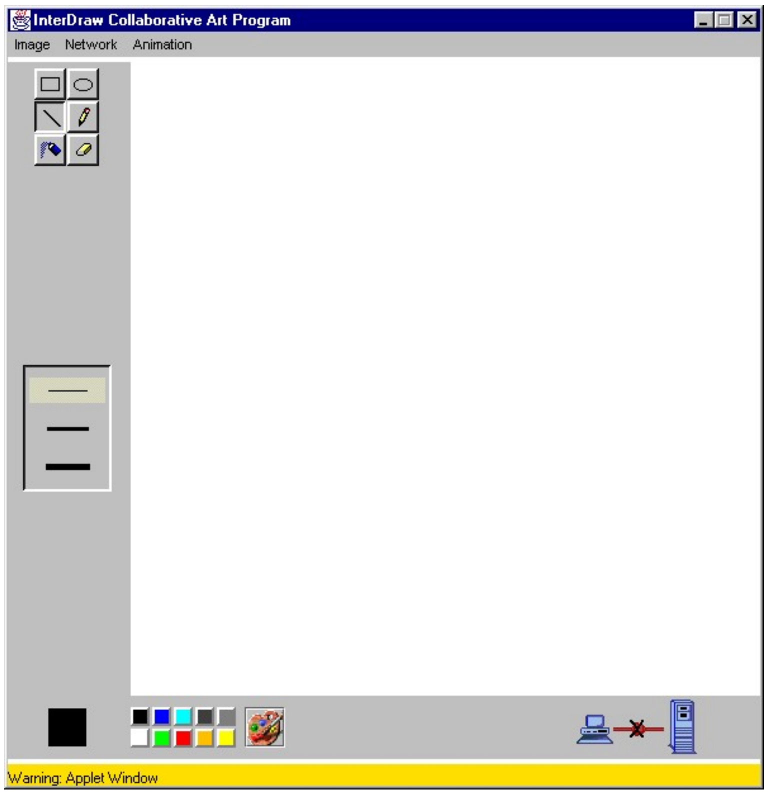


Figure 5.22: Main interface of InterDraw application

As stated in section 4.5.8 the design of the user interface for InterDraw was based on the user interface of Microsoft Paint. Following the basic format derived from the Microsoft Paint application (Figure 4.6) I was able to create the interface shown in Figure 5.22.

A few minor changes have been made to the interface to accommodate the networking and animation abilities of the application. One change was the addition of a server status image in the lower right hand side of the interface. This image provides feedback to the user on the current state of the client/server connection. The use of images rather than words is supported by the fact that people can recall images from their long term memory better than words.[May92] This ability help in making the user interface easier to use and understand.

Another change in the interface is the items appearing in the application menu on the top of the user interface. Similar to the "File" menu item of the Microsoft Paint application, the user interface of the InterDraw application contains an "Image" menu, which allows users to select from a number of images to work on. To handle the network capabilities of the InterDraw application a "Network" menu was also added to the application menu. The “Network” menu allows the user to update the current image and to test the server connection for errors. The final addition to the application menu was the “Animation” menu. This menu allows users to incorporate animated objects into the current image.

1. Select Image Interface

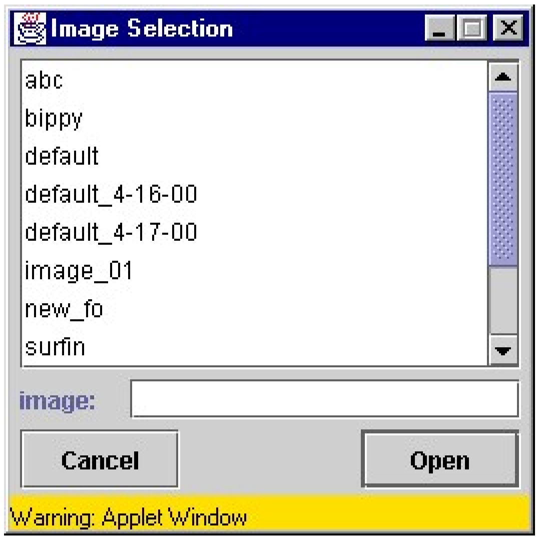


Figure 5.23: Image Selection user interface

The Image Selection interface was designed to be easy to understand and use. The overall look and feel of the interface is similar to the file selection dialog used in Microsoft Windows. Of course, this interface is much simpler as directories and file types do not exist.

A user is given two methods of selecting an image. The first method is to search through the list and using the mouse to select the image. The second method is to type in the name of the image into the text dialog provided. When a user selects an image from the image list, that image’s name is displayed in the text dialog. This feature allows the user to modify or change the name if they so desire.

2. Insert Animation Interface

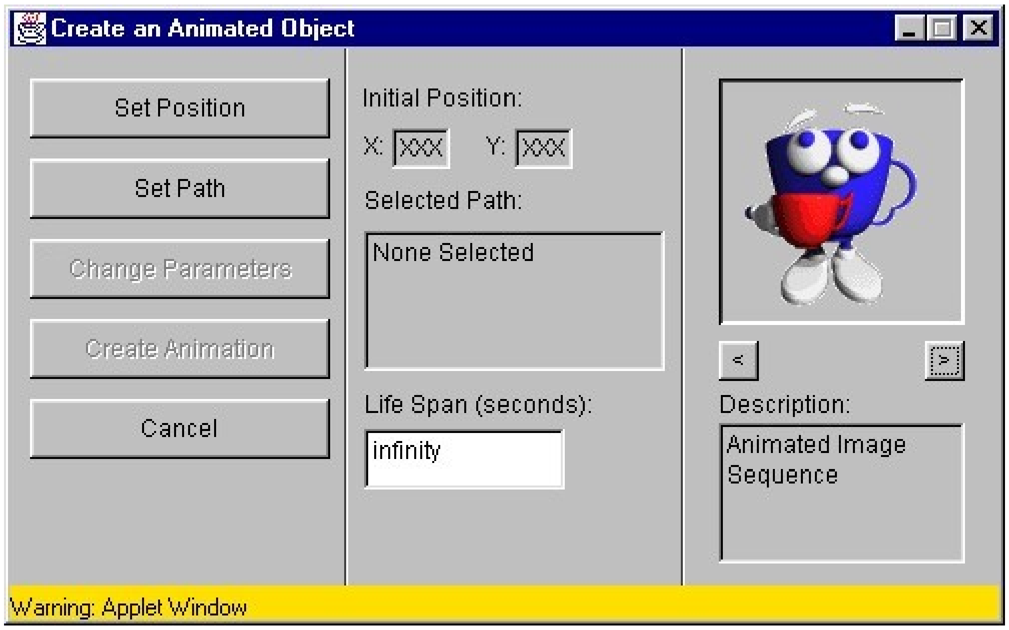


Figure 5.24: Insert Animation user interface

The user interface for the Insert Animation interface is broken up into three sections. The left section of the user interface contains the command buttons used in the creation of the animation object. The command buttons allow the user to set the different attributes of the animation object being created.

The middle section of the Insert Animation interface is used to display descriptions about the selected attributes of the animated object, such as starting location and path type. This section needed to be easily seen by the user and by grouping all of this information the middle of the interface the user does not have to search the interface for the information.

The final section (far right) is the animated object selection. This section contains information about the currently selected animation object. The current selected animation object is displayed in order to allow the user to have a visual representation of the object they are going to be adding to the image. Underneath this image are two buttons, which are used to change the selected animation object. The buttons are represented by less-than and greater-than symbols. These symbols were used for their likeness with left and right arrows. This similarity will allow the users to associate the symbols with going right and going left, which makes it easier for them to understand how to traverse the list of animated objects. Finally a description of the animation is provided to give the user a better understanding of the animation.

3. Animation Path Selection Interface

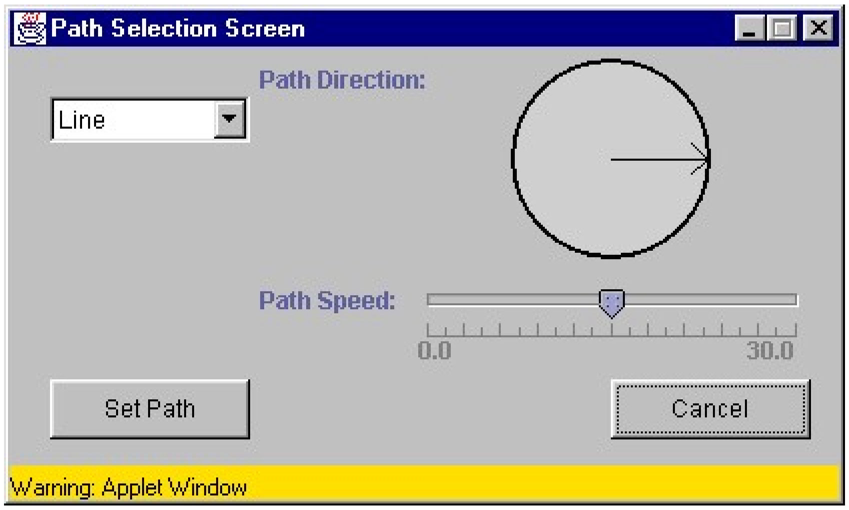


Figure 5.25: Path Select user interface

The Animation Path Selection provides an interface for the user to select an Animation Path for an Animated object. The interface makes use of graphical objects to set the parameters required for each type of Animation Path. The interface is broken up into two main sections. The bottom of the interface contains the command buttons used to either cancel or selected the Animation Path. The upper left area of the user interface provides the user with a simple drop down menu containing all available Animation Paths. The rest of the interface is used to provide an interface for the modification and selection of path parameters. Each parameter is represented by its name and a graphical interface such as a slider or drop down menu as illustrated in Figure 5.25.

In Figure 5.25 a Line Path has been selected. Two parameters can be set in a line path: the direction of the path and the speed that the path is followed. To select the path direction a rotating arrow interface is used. The user is able to rotate the arrow to point in the direction the path will follow. A slider interface is used to select the speed that the path is followed. This use of graphical interfaces to set the Path parameters simplifies the user interaction with the program and in turn making it easier to understand and learn.

5.2 Server Implementation

In an effort to limit the amount of developing needed to complete this project I searched for a server implementation that I could use and modify to create the server application for this project. After reviewing several different Java books in relation to developing server applications in Java, I came across a fully functional server model in *Java Examples in a Nutshell: A Tutorial Companion to Java in a Nutshell* by David Flanagan. After reviewing the documentation and the code for the server, I decided that this server would work for the basis of my server implementation.

There were a few reasons behind this decision. The first and most important reason being the code was free to use and modify according to the license presented in the code:

> "This example is provided WITHOUT ANY WARRANTY either expressed or implied. You may study, use, modify, and distribute it for non-commercial purposes."[Fla97]

Another reason for using Flanagan's code is the extensive documentation of the server in the book and within the source code. This documentation allowed me to easily see where to modify the code to incorporate my own server needs into his server model.
The final reason that I chose to use this server implementation was the number of features that were already implemented in Flanagan's server.

> "The Server class [sic] is a multi-threaded server that provides services defined by implementation of a nested [sic] Service interface. It can provide multiple services (defined by multiple Service objects) on multiple ports, and it has the ability to dynamically load and instantiate Service classes and add (and remove) new Services at runtime. It logs its actions to a specified stream and limits the number of concurrent connections to a specified maximum."[Fla97]

These are only some of the abilities that this server implementation contains. With all of these features incorporated into a robust server, I was able to spend more time on the more important aspect of this project. Instead of having to develop a complete and robust server, I now had to develop a Service object that could be added to the Server class provided by Flanagan's code. In effect, the InterDraw Service is a thread that services each client. The relationship between the InterDraw Service and the client applications is illustrated in Figure 5.26.

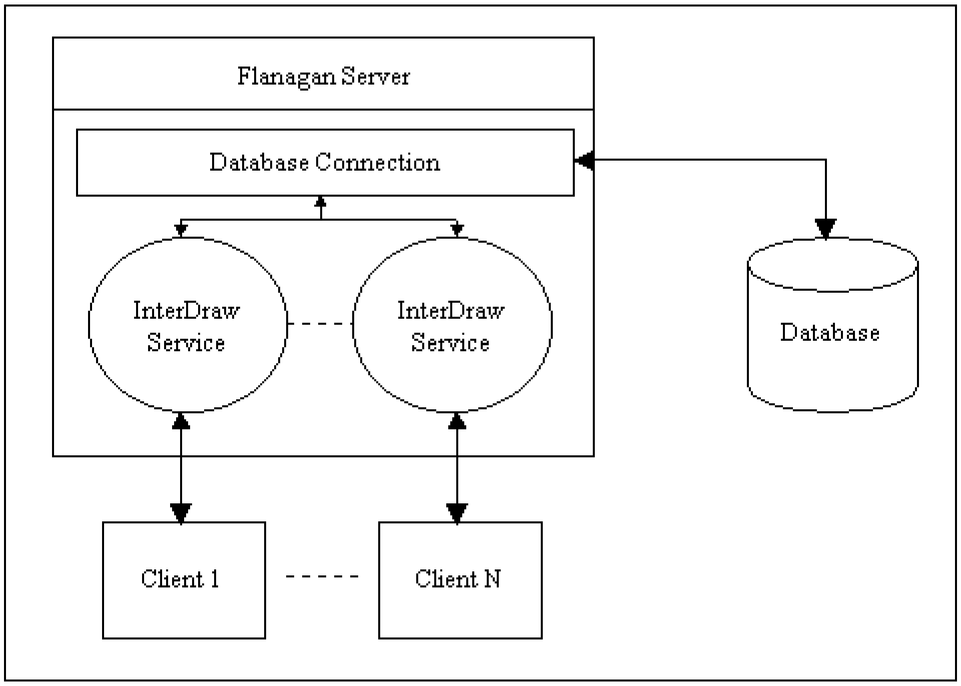


Figure 5.26: Server architecture using Flanagan Server

By using the server implementation developed by Flanagan, I was able to simplify the algorithm of the InterDraw Service into the following:

```
LOOP
        READ Data From Client
        EXIT SERVICE if no data
        PROCESS command
        IF ERROR
                SEND error message to client and EXIT SERVICE
        ELSE
                SEND data to client
REPEAT LOOP
```

This basic loop structure allows the client to perform multiple requests to the server without having to re-establish the connection.

5.2.1 Operations

There are 6 main operations or services that the InterDraw Service provides to each client. These service are the ability to return a unique client ID, return a time synchronization value to synchronize the client time systems with the server, retrieve all editable image names and to store, remove and retrieve image data

5.2.1.1 Return a Client ID

When a client requests a client ID, the server generates a 20-character non-null string that is returned to the client. This 20-character string is maintained by the server and updated to a new unique ID after every client request. On startup the server initializes each character in the client ID to the ASCII value of 1. As each new client requests a client ID the ASCII value of the rightmost character is incremented. When the ASCII value reaches the maximum allowed by a character in Java, the next character to the right is incremented and the rightmost character gets reset to 1. The other characters in the ID string follow the same pattern until all of the characters have reached the maximum ASCII value for a character in Java. Once this situation occurs, the client ID is reset to its original value of {1,1,1…1,1,1}. A simulation of this process is displayed in Figure 5.27.

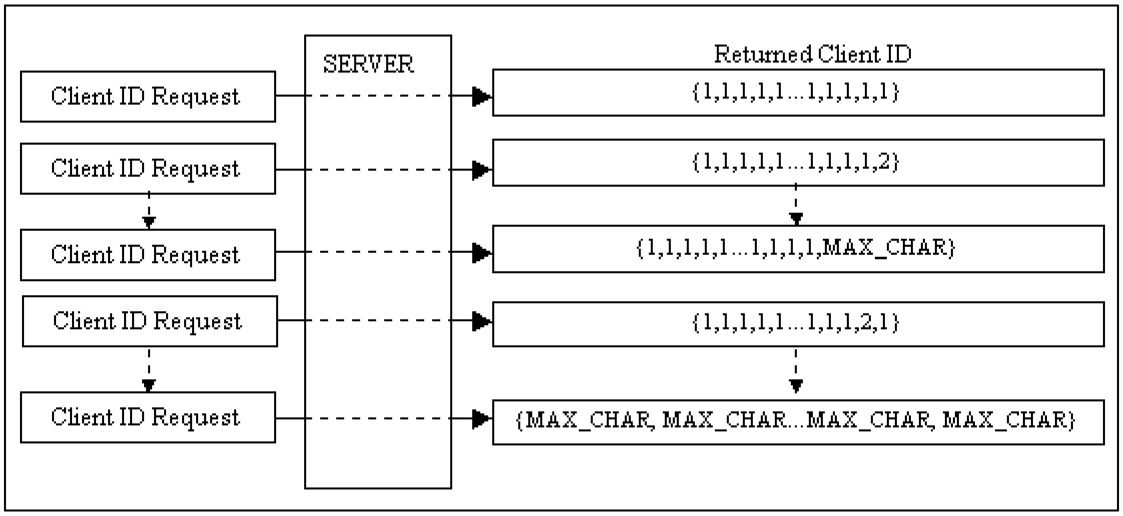


Figure 5.27: Simulation of Client ID generation by InterDraw Service

As can be seen in Figure 5.27 this algorithm has a maximum number of unique client IDs. With Java's current Unicode character scheme each character is 16-bits long. These 16-bits give each character $2^{16} - 2$ or 65534 different values after the removal of all null characters (ASCII value of 0). Using a 20 character string, the number of unique client IDs is $(2^{16} - 2)^{20}$ or 2.13468e96 different values. Using a generic 8-bit character this value gets lowered to 1.4625e48. In the worst-case scenario the client ID will get reset once in the next 4.638e37 years if we assume one new client ID is generated each millisecond.

5.2.1.2 Return a Time Synchronization Value

Time synchronization becomes a concern when animated objects are added to an image. The movements of the animations are related to the length of time since they were first created. This relationship makes it necessary to have a synchronized time between all of the clients in order to maintain the correct locations of the animated objects. This problem is resolved by using the server as a global synchronization clock. All client times are made in relation to the time maintained on the machine the server is currently running on. When a new client connects to the server, it tells the server what the local time is on the client in milliseconds. The server returns the difference in milliseconds between the current time on the server and the time received from the client. The client synchronizes itself with the server by adding its local time to the time difference returned from the server. This process is illustrated in Figure 5.28.

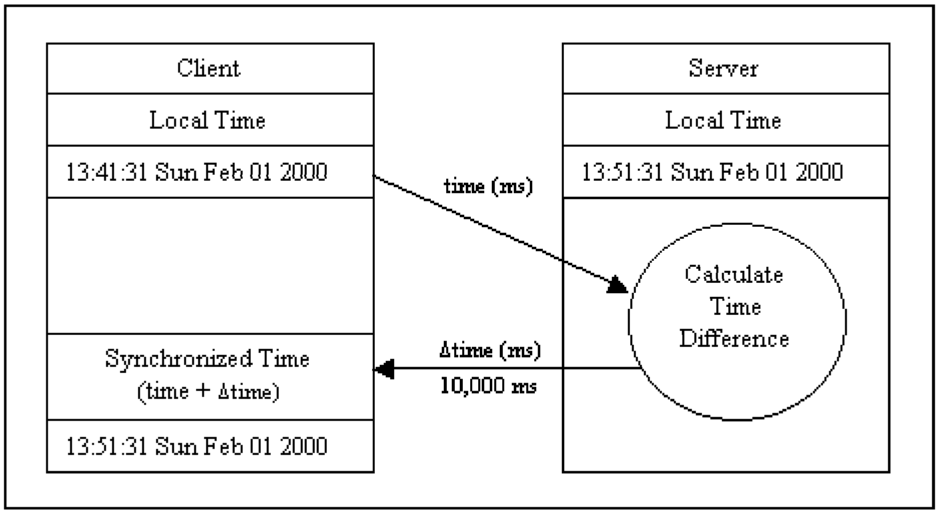


Figure 5.28: Flow chart of time synchronization process

5.2.1.3 Retrieve Editable Image Names

The retrieval of editable image names gives the client an up to date list of all images that are stored in the database. The retrieval of the image names is done through the use of a SQL (Simple Query Language) query. The query retrieves all of the image names in the database and sorts them in alphabetic order. The SQL query used is shown below.

```
SELECT image_nm FROM objects
                    GROUP BY image_nm
                    ORDER BY image_nm;
```

Where image_nm is the database field containing the name of the images and objects is the table within the database containing the image data.

5.2.1.4 Store Image Data

The storage of image data is done through the use of SQL insert queries. When the client makes a request to store image data to the database, the server receives a list of image data objects from the client. For each of the image data objects in the list the server uses an SQL insert query to insert the data into the database. For more information on this query see section 5.4.2: Server/Database Communication.

5.2.1.5 Retrieve Image Data

The server can respond to two forms of image retrieval. The first is a request for all image data pertaining to a certain image. For this request the server queries the database for all image data pertaining to the image that is specified by the client. For image data to be retrieved by the query, the data must not have 'expired' or have been removed from the image previously.

The second form of image retrieval is a request for all image data added to the image after a certain date by clients other than the one making the request. The server queries the database in a similar manner as above with a few differences. Instead of querying for all data, the server queries for only data that has been inserted into the database after the date specified by the client. For image data to be retrieved by the query, the data must not have 'expired' or have been removed from the image previously and it must have been inserted by a client other than the client requesting the data retrieval.

For more information on these queries see section 5.4.2: Server/Database Communication.

5.2.1.6 Removal of Image Data

Image data is not physically removed from the database. Instead the removal date field is set to the current server time. While this action does not remove the image data from the database, it effectively removes it from being accessed by the clients. The data becomes inaccessible by the server not returning any image data whose removal date has been set. The data is not removed from the database for two reasons. The first reason is to is to provide a method of updating other clients with the knowledge that the object data has been removed from the image. The second reason is purely for the maintenance of an image history. Keeping a history of all objects in the database will allow for the recreation of the image during various states in its life cycle. It could also provide valuable information into the effectiveness and evolution of collaborative art, which could be studied in future projects.

When a client makes a request for the removal of image data a list of object IDs is sent to the server. For each object ID in this list, the server makes an SQL update query to the database to set the specified object's removal date to the current server time. For more information on this query see section 5.4.2: Server/Database Communication.

5.2.2 Synchronization Issues

When using multi-process or multi-threaded application data synchronization becomes a considerable problem. Such issues as race conditions occur and must be dealt with. A race condition is a condition that occurs when two threads are accessing the same data. The actions of these threads may change what the final outcome of the action should be. When this happens, it is called a race condition. A popular example is withdrawing money from an ATM. The flowchart of this operation is shown in Figure 5.29.

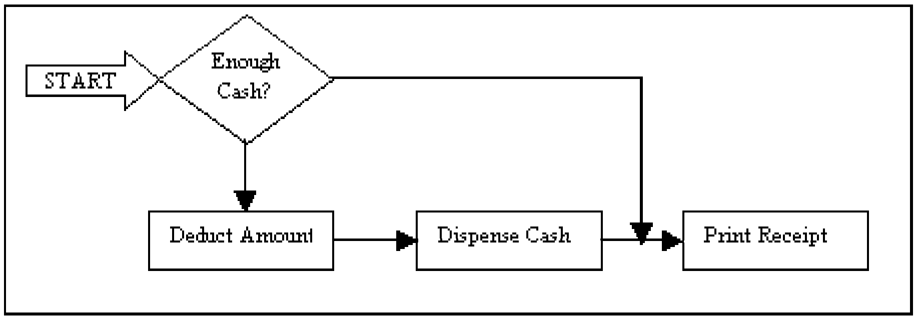


Figure 5.29: Algorithm flow chart for ATM withdrawal ([OW99], p 41)

The algorithm shown is a simple one. The ATM checks to see if the user has enough cash for the requested withdrawal amount. If there is enough available cash, then deduct the withdrawal amount and dispense the cash, otherwise go directly to the last step, which is to print out a receipt for the transaction. In a single threaded environment this algorithm works fine. If we add multiple threads or users in this case, a race condition is created. Take the following example:

> "One day, a husband and wife both decide to empty the same account, and purely by chance, they empty the account at the same time. We now have a race condition: if the two users withdraw from the bank at the same time, causing the [enough cash?] methods to be called at the same time, it is possible for the two ATMs to confirm that the account has enough cash and dispense it to both parties."[OW99]

From this example, it is easy to see how race conditions can be a problem with multi-threaded applications. The InterDraw Service(IDS) threads share two data sources. The first source is the client ID and the second is the database connection shown in Figure 5.26. In both cases problems would arise if the IDS threads modified either source at the same time as another IDS thread.

The solution to this problem was the use of BusyFlag objects. The BusyFlag can be looked at as a locking variable. When one thread needs to access a shared data source the thread must first acquire the BusyFlag by calling its getBusyFlag method. When the thread acquires the BusyFlag, it effectively locks all other threads from accessing the shared data source. The thread releases the BusyFlag by calling its freeBusyFlag method when it no longer needs the shared resource. The resource is once again free to be used by other threads. This pattern of acquiring and releasing the BusyFlag is illustrated in Figure 5.30.

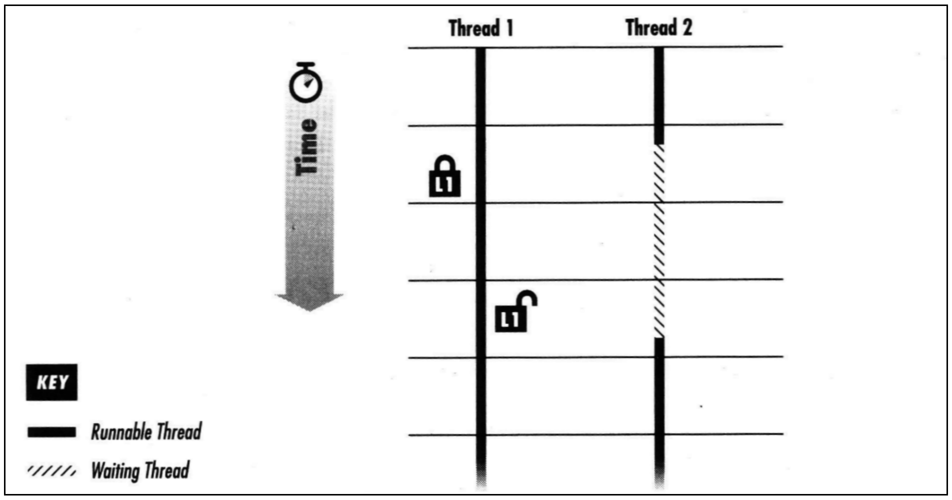


Figure 5.30: Acquiring and releasing a lock ([OW99], p 48) ***used without permission**

Since IDS threads share two data sources, two BusyFlag objects are used. One BusyFlag is used to facilitate the locking of the client ID data and the other to prevent access to the shared database connection.

5.3 Database

Table: OBJECTS			
Field Name		Field Type	Description
insert_dt	***key***	DATE	Contains the date that the image object was inserted into the database.
image_nm	***key***	VARCHAR(40)	Contains the name of the image for which the data is being stored.
object_id	***key***	VARCHAR(22)	Contains an object id that is unique to that client that is storing the object.
user_id		VARCHAR(20)	Contains the user ID of the client storing the image data.
removal_dt		DATE	Contains the date that this object was removed from the image.
death_dt		DATE	Contains the date that this object will expire and no longer be contained within the image.
data_str		LONG RAW	Contains the raw object data.

Table 5.1: "OBJECTS" table used to store image data in database

As shown in Table 5.1, OBJECTS uses three fields to create a unique key. These fields are 'insert_dt', 'image_nm' and 'object_id'. All three of these fields are required to create a unique key. When multiple objects are inserted into the table, the time required for the insert may take less than a millisecond, while the DATE field can only handle time in milliseconds. The end result is multiple inserts with the same insert_dt. Even adding the 'image_nm' field does not make the key truly unique. It is still possible to have multiple records with the same 'insert_dt' and 'image_nm'. To create a unique key, the object_id field was added. This field contains an id that is unique for each object within an individual client. Through the use of these three fields a unique key is created.

5.4 Communication

5.4.1 Client / Server

A client connects to the server using a TCP/IP socket. The client and server communicate with each other by sending frames over the socket connection. The frame structure consists of a starting frame delimiter '#@' the frame body and an ending frame delimiter '#!' (as seen in Figure 5.31).

#@
TRANSMITTED INFORMATION
#!

Figure 5.31: Client / Server frame representation

The use of frames allows the client and server to easily verify that all data has been received. Since we are using a TCP/IP connection for the transfer, we do not have to worry about maintaining data integrity, as the TCP/IP connection sustains a reliable, error free connection.

The body of the frame contains the following character delimiters.

Delimiter	How Used	Description
#C	<command string>#C	Represents the end of a command string
#U	<user ID string>#U	Represents the end of a user ID string
#D	<data string>#D	Represents the end of a data string
#I	<image name string>#I	Represents the end of an image name string

Delimiter	How Used	Description
#R	<data string>#R	Represents the end of a list of object ids to be removed from an image

Table 5.2: Delimiters used in body of transmitted frame

Within the text of each delimiter it is possible for the data string to contain a sequence of character that is equal to one of the frame delimiters. For example in the frame:

#@comm#Dand#

The correct parsing of the frame should be:

command comm#Da =

Instead the frame gets incorrectly parsed as:

command and = com data =

This incorrect parsing occurs from the parser using the '#D' in the string as a valid delimiter. To resolve this problem, I used character stuffing in the data strings of each delimiter. Whenever a '#' was found in one of the data strings an extra '#' was added to the string following the found occurrence of '#'. In the example of the frame:

#@comm#Dand#

With character stuffing the frame turns into:

#@comm##Dand#

When the frame gets parsed, the data strings are unstuffed, meaning the extra '#' are removed. If an odd number of '#' are found in a row, then the parse knows it has found a true delimiter. With this addition of character stuffing, the parsing of the frame will return with the correct result:

command comm#Da =

There is one other delimiter that has not been mentioned. When objects are being transferred between the client and server, they are being passed in the data portion of the communication frame. If more than one object is being sent at a time, there must be a way to separate the object data. This separation is done through the use of the data separation delimiter (#|). This delimiter can be placed within the data portion of the communication frame to separate the data of the different objects. An example of how to use this delimiter is illustrated in Figure 5.32.

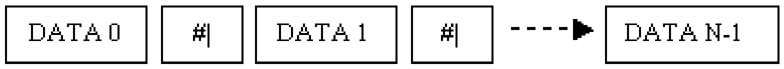


Figure 5.32: An example of how to use the data separator delimiter

The body of each frame must contain, at a minimum, a command followed by the #C delimiter. Depending on the command, more delimiters may be required. Table 5.3 shows a listing of these commands and their required or optional delimiters.

Command	Required Delimiters	Description
CMD_GET_UID	#C	Command sent by the client. It is a request for the server to create a unique client ID for the client.
CMD_GET_ALL	#C, #I	Command sent by the client. It is a request to retrieve all data for the specified image.
CMD_GET_TIME	#C, #D	Command sent by the client. It is a request to have the server calculate the synchronization time difference. The local client time is passed within the data delimiter.
CMD_GET_NEW	#C, #U, #I, #D [#R]	Command sent by the client. It is a request to receive all data that has been stored since a date specified in the data delimiter for the specified image. [optional: if the #R delimiter contains data, the server sets the removal date for the specified objects]
CMD_GET_IMAGES	#C	Command sent by the client. It is a request for the server to return a complete listing of all editable images.
CMD_ADD	#C, #U, #I, #D [#R]	Command sent by the client. It is a request to store the objects contained within the #D delimiter for the image specified. [optional: if the #R delimiter contains data, the server sets the removal date for the specified objects]

Command	Required Delimiters	Description
CMD_DATA	#C, #D [#R]	Command sent by the server. It is used when sending data, such as client ID to the client. [optional: if the #R delimiter contains data, the client removes the specified objects from the displayed image]
ERROR	#C [#D]	Command sent by the server. It is used to indicate an error on the server side. [optional: A description of the message may be included in the #D delimiter.]
OK	#C [#D]	Command sent by the server. It is used to indicate that an operation on the server has been successful. [optional: Any further information may be included in the #D delimiter]

Table 5.3: Communication commands between client and server

5.4.2 Server / Database

When the first client connects to the InterDraw Service of the server a database connection is established by creating an instance of the Oracle JDBC driver. The DriverManager JDBC class is then used to establish a connection to the requested database. To establish a connection using the DriverManager, a URL of the form “jdbc:subprotocol:subname”, a database username and a password must be provided by the connecting application.

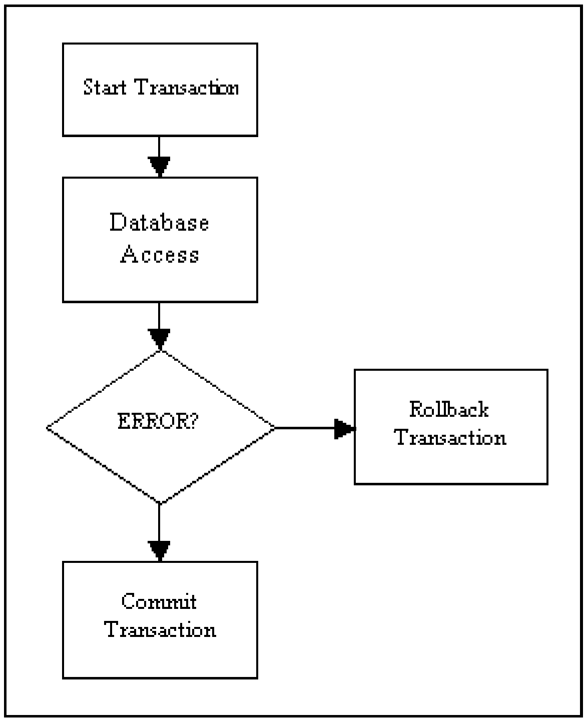


Figure 5.33: Flow chart of a database transaction

To provide better error support, the server makes use of database transactions. Transactions provide error handling abilities when adding to or updating information within the database. A transaction is started before any update to the database. If at anytime during the database access an error occurs, then the transaction is cancelled and all modifications to the database are removed. If an error does not occur, then the transaction is committed and all modifications to the database are made final.

The server uses several SQL queries to interact with the database. The queries used and their results are listed below. Field names are represented in italics, variables are represented with left and right arrows (e.g. <variable>) and tables are represented in bold letters.

1. Image Name List Retrieval Query

SQL Text:

SELECT *image_nm* FROM **objects**
GROUP BY *image_nm*
ORDER BY *image_nm*;

Result:

The query returns a record set containing an alphabetical listing of all image names stored within the database.

2. Image Data Storage Query

SQL Text:

```
INSERT INTO objects
        (insert_dt, user_id, object_id, image_nm, death_dt, data_str)
        VALUES (<current_dt>,<user_id>, <object_id>, <image_nm>,
<death_dt>, <data_str>);
```

Result:

The query creates a new record in the database with the specified current date, user ID, object ID, image name, date of object death and data string.

3. Image Data Retrieval Query

Two queries were used in the retrieval of image data. The first of these queries is used to retrieve all image data for a specified image, while the second is used to retrieve all information for a specified image that was not inserted by the specified user.

SQL Text:

```
SELECT * FROM objects
        WHERE (image_nm = <image_nm>) AND
                (removal_dt is NULL) AND
                (death_dt > <death_dt>)
        ORDER BY insert_dt;
```

Result:

The query results in a record set containing all image data objects for the specified image that have not been removed from the image or expired.

SQL Text:

```
SELECT * FROM objects
        WHERE (NOT (user_id = <user_id>)) AND
                (image_nm = <image_nm>) AND
                (insert_dt > <insert_dt>) AND
                (removal_dt is NULL) AND
                (death_dt > <death_dt>)
        ORDER BY insert_dt;
```

Result:

The query results in a record set containing all image data objects for the specified image that have been inserted after the specified date, have not been removed from the image, have not expired and were not inserted by the specified user.

4. Image Data Removal Query

SQL Text:

```
UPDATE objects
        SET removal_dt = <current_dt>
        WHERE (user_id = <user_id>) AND
                (image_nm = <image_nm>) AND
                (object_id = <object_id>);
```

Result:

The query sets the removal_dt field of the stored data object specified by the user ID, image name and object ID, to the current date. This action effectively removes the image date from being returned by any of the other database queries.

5.5 Image Representation

The image data is stored in two ways. On the client machine the image data objects are stored in StoredObject classes. These classes maintain all information about each image data object. During network transfer and database storage the StoredObject classes convert the image data objects into Unicode character strings that are easily transmitted and stored in the database.

5.5.1 StoredObject representation

The StoredObject representations of image objects are used by the client application to maintain the current state of the displayed image and to keep track of objects that are uploaded or downloaded from the server. The StoredObject class maintains all of the attributes of image objects that are required to restore an image object visually to the screen. Table 5.4 displays the parameters that are required to maintain these attributes.

Object Parameter	Description
Object ID	A string ID that is unique for objects within the client. It is created by combining the client ID with a 2-character string that is incremented as the client ID in section 5.2.1.1.
Object Type	An integer value that represents the type of the object represented by the StoredObject class.
Creation Date	A long integer representing when the object was created in order to maintain the correct drawing order of the objects.
Expiration Date	A long integer representing when the object will no longer exists within the image.
Object Color	A Color class that represents the base color for the object.

Object Parameter	Description
Object Bounds	A Rectangle object that represents the area encompassed by the object.
Extra Data	A character string containing more object specific information. (same format as described in section 5.5.2.2).

Table 5.4: Parameters maintained by StoredObject class

The client stores the StoredObject classes within two Linked List structures. One list is used for locally created objects and the other for imported objects from the server. Two lists were used to minimize the amount of management required in deciding if an object was a local object or an imported object. With two Linked List structures, this problem is removed. Determining if an object is a local object or an imported object becomes the simple task of checking to see if the object is contained within the local or imported object Linked List.

A Linked List structure was used instead of a Hash Table or a Java Vector, because of the ease that an object can be inserted in between two other objects in the structure. When an object is downloaded from the server, it may have an earlier creation time than other objects downloaded previously. To maintain a correct display order, the object must be placed into the correct location in the Linked List structure, which is ordered from the oldest object to the youngest. This action may require the insertion of the object between two others in the Linked List. This operation is easily attained with the Linked List structure as opposed to the other possible data structures.

5.5.2 Character String Representation

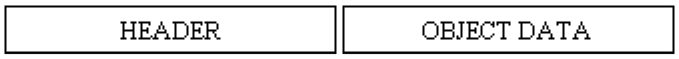


Figure 5.34: Character string representation

The character string representation of an image object is used for transferring object data between the client and server and for the storing of the object data in the database. As shown in Figure 5.34 the character string representation of an image object is broken into two parts; the header and object specific data. While the header contains general information about an image object, the object specific data contains information that relates to each individual object.

5.5.2.1 Header

The header is a 42-character string containing general information about the image object. It is similar to the attributes stored in the StoredObject representation.

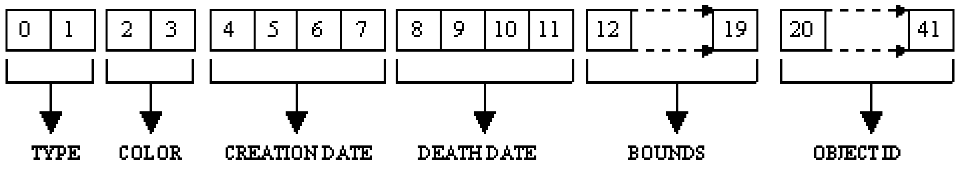


Figure 5.35: Header format for character string representation

Figure 5.35 illustrates a breakdown of the header section of the character string representation. The first two characters of the string contain the character representation of the object type. Characters 2 and 3 contain the RGB color representation of the object's base color. Characters 4 through 7 contain the character representation of a long integer of the date/time when the object was created. Similarly characters 8 through 11 contain the character representation of a long integer of the date/time when the object will be removed from the image. The next 8 characters (12-19) represent the object bounds. The first two of these characters contain the upper left x value of the object bounds and the third and fourth contain the upper left y value of the object bounds. The final four characters in this block represent the length and width of the area bounded by the image object. The final 22 characters of the header form the unique object ID for the image object.

I have been simply stating that the characters in the header are representing both integers and long values. How can this be? In Java and other languages an integer contains twice as many bits as a character, and a long integer has twice as many as a normal integer. Let's say our system is using 8-bit characters, 16-bit integers and 32-bit long integers. To convert the 16-bit integers into a character representation we must place the first 8 bits of the integer into one character and the second 8 bits into a second character. Instead of having an integer value of -1 we now have a character string {ASCII (256), ASCII (256)}. The same method applies to converting a long integer into a character representation. A long integer value of -1 will be converted to the character string {ASCII (256), ASCII (256), ASCII (256), ASCII (256)}. This process is clearly illustrated in Figure 5.36.

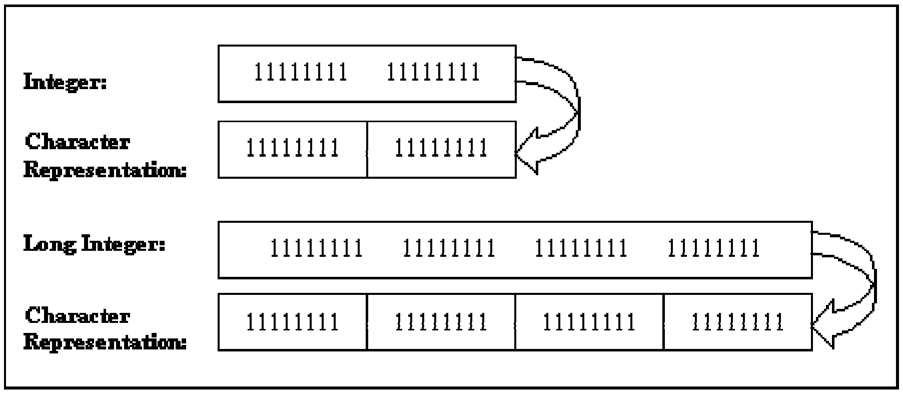


Figure 5.36: Integer and long integer conversion to character representation

5.5.2.2 Object Data

The object data portion of the character string representation contains information that is unique to each individual image object. There are three basic types of image objects; animated sprite objects, rubberbanding objects, and free drawing objects. Each type has its own basic format for storing unique object data. The following sections describe in detail these formats.

1. Free Drawing Objects

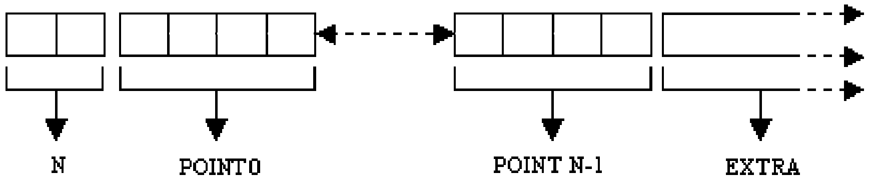


Figure 5.37: Free Drawing object storage format

Free drawing objects keep track of all mouse locations to create a precise path that can be used for such drawing tools as free hand drawing and airbrush painting. To store this precise path all recorded locations must be maintained in the object representation. The first two characters of the drawing objects data are the character representations of the total number of points in the recorded path (see N in Figure 5.37). The following N*4 characters contain the actual xy locations incorporated within the drawn path. These points are stored in blocks of four characters, where the first two characters are the x value and the second two the y value of the given point. Any characters following the location data are specific to classes extending the Drawing object class.

2. Rubberbanding Objects

Rubberband objects are objects that can be visually seen changing shape as the user adds the object to the current image. Examples of rubberband objects include lines, rectangles and ellipses. Rubberband objects are drawn based on an invisible boundary "box" that is created by the user. This process is illustrated in Figure 5.38.

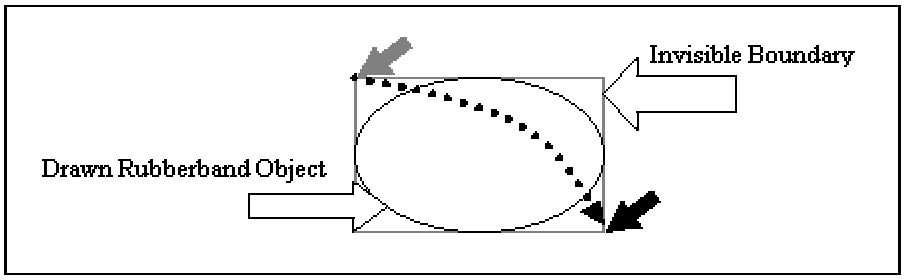


Figure 5.38: Example showing how a Rubberband object is created

Since a rubberband object is drawn largely based on the boundary box discussed above, the object data must contain the location and dimensions of this box. The storage of the boundary box is done in two parts. The first part is storing the top left xy location of the box in the first four characters of the object data and the second is storing the bottom right xy location in the following four characters as can be seen in Figure 5.39. The four characters for each consist of the first two characters representing the x value and the second two characters representing the y value of the location. As with drawing objects any characters following the boundary box specification data are specific to classes extending the Rubberband object class.

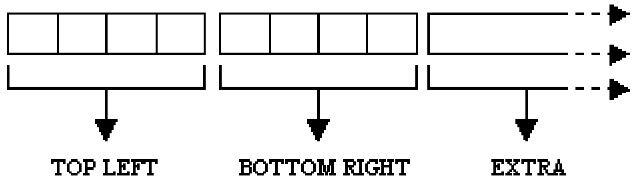


Figure 5.39: Rubberband object storage format

3. Animated Sprite Objects

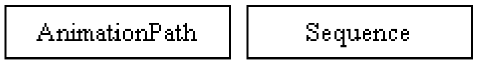


Figure 5.40: Animated sprite object storage format

Animated sprite objects can consist of anything from rotating 3-Dimensional models, to a modulating series of images. Unlike the Rubberband or the Drawing classes, the Animated sprites do not store locations or bounding boxes within the object data. Instead, the sprites' data contains two serialized objects, as shown in Figure 5.40. A serialized object is an object that can be written to an output stream, which in this case is a character string. The first serialized object that gets written into the object's data is an AnimationPath. The AnimationPath class maintains the current displayed location of the animated object. The second serialized object is a Sequence class. The sequence class is in effect the animated sprite's image generator. It is up to the Sequence to return the next image in the animated sprite's animation.

5.6 Evaluation

The evaluation of the InterDraw project was done in two steps. The first step was user testing, where users ran the application on different computers, Internet connections, operating systems and browsers. This step was used to see how the InterDraw application operated in both high and low stress conditions. The second step was a user survey. The user survey was used to retrieve specific information from the user on the usefulness and durability of the application.

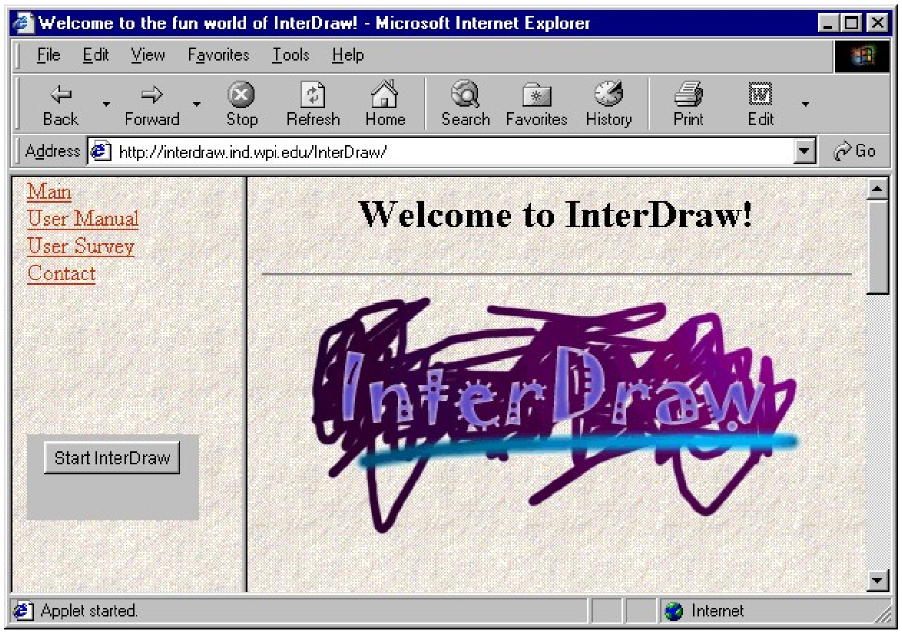

Welcome to the fun world of InterDraw! - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Print Edit
Address http://interdraw.ind.wpi.edu/InterDraw/
Go
Main
User Manual
User Survey
Contact
Start InterDraw
Welcome to InterDraw!
InterDraw
Applet started.
Internet

Figure 5.41: Evaluation website

A user testing website was developed to provide a focal point for the evaluation. The website provided the testers with background knowledge of InterDraw. The web page included an online user manual, which provided useful information about the program and its features. The site was set up to provide easy access to both the user testing and the user survey portions of the evaluation. As shown in Figure 5.41 the user survey and InterDraw application are both located on the menu frame (left side) of the user testing website. By providing the user testing and survey in the menu frame the testers were able to access them, regardless of the page being viewed in the content frame (right side).

1. User Testing

The server application for the user testing was set up on a Dell Inspiron 3500 with a Pentium II 400 MHz Intel processor. An Oracle database owned by WPI was used for the user testing storage. For each client that connected with the server, a log was generated that contained that IP address and date of the client connection. This log was generated in order to provide the load and connection rate data of the server during testing.

2. User Survey

The user survey consisted of 14 questions and a comment section. The first four questions were used to collect demographical information about the user, such as their name and connection speed. The following seven questions were answered based on a yes/no format or a scale from 1 to N (N being 5 or 10). These questions were aimed at retrieving information about the users opinions of the program. The questions ranged from how easy the program was to learn to whether or not they enjoyed their experiences with the program.

The next three questions allowed the users to request additional features and to mention features they felt should be removed from the application. The questions also contained a section for the users to submit any bugs that they found in their testing of the program. The last piece added to the survey was a comment section that gave the users the ability to mention anything extra they felt was relevant to the project.

When a user submitted the survey, the contents of the survey fields were sent to a CGI script that inserted the information directly into a database. The information in the database could then be used to easily sort and analyze the data.

For a look at a text version of the web-based survey, see Appendix A.

6 Results

In this section, we will be taking a look at the results of the evaluations and information retrieved during the testing of this project. It will include looking at how users responded to the InterDraw program and a look at the types of images that were produced from the collaborations of the different users. All charts are located in Appendix D.

6.1 User Testing Results

User testing occured over a period of two and a half weeks. During this time period over 55 different computers successfully interacted with the InterDraw program. This amount does not take into account different users using the same computer to interact with the program at different times, which means the total number of users testing the program could be much higher than this number. While this number is, at best, a good estimate of the total number of users, it provides valuable feedback on the use of the InterDraw program.

One interesting aspect of the computers that were used in user testing was their location. Figure D. 3 illustrates that almost half of the 55+ computers used in user testing were from outside of the WPI domain. This number included computers from Colorado, Florida, Massachusetts, Michigan, Vermont and others. These numbers illustrate the diversity of the users that tested the InterDraw application.

Figure D. 1 illustrates the number of users testing the InterDraw program per day during the user testing period. This graph shows that the maximum user load for any given day was around 22 users, with an average per day of between 3-5 users. These numbers do not include days where there was not any user activity.

Figure D. 2 represents the number of users testing the InterDraw program per hour during the user testing period. As with the previous graph, this graph does not show hours where there was not any user activity. This graph can be used to determine the number of users simultaneously connected to the InterDraw application. The maximum load on the server for any given hour was 7 users, with the average load staying between 1 and 3 users.

The graph in Figure D. 4 displays the speed of the connection that was used during the user testing. The majority of the data on connection speed was from users with a T1 or greater Internet connection. The other connections ranged from 28.8 bps modem connections to DSL connections. An important feature of the InterDraw program is its accessibility to the users. This accessibility requires the program to work well under all connection speeds, from 28.8 bps modem to a T1 line. These results show that the users tested InterDraw on a wide variety of connection speeds.

These numbers illustrate the capabilities of the InterDraw server. Although it would have been good to see a higher number tested, it has been shown that the server can handle at least 7 simultaneous connections. After checking server logs, it was shown that the only problems with these connections was an increase in the time to upload and download data from the server. This issue was caused by the synchronization methods in the server, which only allow one client to have access to the database at any time. Any clients that try to connect to the database while another client is accessing it have to wait until the database is free, thus adding a delay to the upload or download process.

6.2 User Survey Results

The results from the user survey were not as impressive as from the user testing information. Out of the 55+ users that successfully testing the program only 27 of them submitted a survey. Out of the 27 surveys only 3 were from non-WPI students. While this set of results is not as diverse as in the actual user testing of the program, it does provide useful feedback on the users' opinions of the InterDraw program.

Figure D. 5 illustrates a graph representing the user responses on how understandable the buttons and icons were in the program. This question was asked in order to determine whether or not the user interface was effective. The results were very positive. As illustrated in the graph, the majority of users felt that the buttons and icons were understandable.

Another question on the survey asked the users how easily they learned how to use the InterDraw program. The results of this question are shown in Figure D. 6. The results of this question were not as one sided as with the previous question. While the majority of users did find the program easy to learn, there was a small minority that had some difficulty in learning how to use the program. These results show that work still needs to be done to make the InterDraw program easier to use for everyone.

In relation to the previous question, the users were also asked how easy was the InterDraw program to use. As with the last question, the majority found the program easy to use, while a few others found it difficult or confusing to use. These results add to the conclusion that more work is needed to make InterDraw easier to use and learn.

In order to get accurate data on how InterDraw performed on various Internet connections, the users were asked to rate the loading speed of the program. Figure D. 9 illustrates the result of this rating displayed against the type of connection used by the user. The results were not as useful as I had hoped. There does not appear to be much of a pattern in the ratings given by the users. Users with a fast T1 or greater connection gave results varying from 8 responses for a fast loading time and 7 responses for a slow loading time. Even users with slower 28.8 bps connections gave results ranging from 3 to 8 on a scale to 10. These results leave me to conclude that the loading time of the InterDraw program is more related to the congestion of the Internet and the type of computer the user is using, than with the application itself. A definite conclusion cannot be determined using the data available from this survey.

One of the last questions asked of the users was whether or not they felt InterDraw was a useful art tool. The results shown in Figure D. 10 illustrate that almost half felt that InterDraw was a useful art tool, while the other half did not. I think that these results really depend on the user. If the user is of an artistic type, he or she may be more inclined to use InterDraw as an art tool, but if the user is not very interested in art, the results may be the opposite. Another possible reason behind this distribution is the lack of more sophisticated drawing tools. Only a few basic tools were incorporated into the testing application. As a result, the users may have felt the InterDraw application was not as usefully than other drawing applications with more advanced tools and features.

6.3 Future Work

Future avenues or work to enhance the InterDraw project include:

- Adding more extensive tools and animations.
- Adding user name creation and recognition.
- Adding user-to-user interactions, such as chat rooms.
- Incorporating a proxy server on the server side to provide client routing to multiple servers and multiple data stores.
- Adding greater user interaction with the objects within the image, such as modifying object attributes after the object has been created.
- Enhancing interface

All of these items will help to enhance the users ability to work with others to create unique, collaborative pieces of art and to make the collaboration as enjoyable as possible.

6.4 Conclusions

This thesis describes an interactive, collaborative art program for the Internet. This program, named InterDraw, incorporated many features that are common in today's graphical applications, while adding the extra ability to collaborate with other InterDraw programs to create truly unique images. The goal of this project was to enhance the ability of artists to communicate with each other in order to facilitate their collaboration in developing graphical images. Overall, I feel that InterDraw successfully meets this goal and provides a step towards collaboration in other areas of the arts as well.

Appendix A

User Survey

User Information

First Name ____________________

Last Name ____________________

Email ____________________

Connection Type:

28.8 bps	64K ISDN	DSL
33.6 bps	128K ISDN	T1+
56K bps	Cable Modem	

Approximately how long did InterDraw take to load? (*1 being fast, 10 being slow*)
1 2 3 4 5 6 7 8 9 10

Was the program easy to learn how to use? (*1 being easy, 5 being hard*)
1 2 3 4 5

Was the program easy to use? (*1 being easy, 5 being hard*)
1 2 3 4 5

Were the buttons understandable? (*1 being very understandable, 5 confusing*)
1 2 3 4 5

How would you rate this program as a useful art tool? (*1 being very useful, 5 not useful at all*)
1 2 3 4 5

How entertaining was the experience? (*1 being enlightenment, 5 being an 8 hour lecture*)
1 2 3 4 5

Would you use this program again?
Yes No

Anything you would like seen added to the program?

Anything you would like seen removed from the program?

Any Bugs to report?

Appendix B

Programmer's Manual

B.1 Programmer Notes

InterDraw is written completely in Java using the Java 2 JDK. The InterDraw files make up a library called "idp" and includes an oracle JDBC library and the gjt library developed by David M. Geary. The following information is intended to help anyone who may want to modify the InterDraw code.

B.2 Files

B.2.1 /

- InterDraw.java – Starting class of InterDraw. The InterDraw program is activated through this class.

- Server.java – Contains the InterDraw server application class. Includes all Service objects used to process client and database connections.

B.2.2 idp/

- AnimationPanel.java – Contains the main Animation Insertion interface. Manages the creation of the Advanced Sprite object parameters and transfer to main application.

- AnimationPathPanel.java – Contains the AnimationPath selection interface. All path and parameter selection are managed within this class.

- BDirection.java – Generates a compass looking user interface for selecting a direction.

- BSlider.java – A simpler version of the JSlider Java Swing class, with the added ability of allowing the user more control in its display parameters.

- CloasableFrame.java – Provides a Java Frame that closes on a WindowClose event.

- ImageSelection.java – A user interface for selecting a string from a list of strings. Used in InterDraw to allow the user to select the current image from a list of image names.

- InterDrawPanel.java – Extends the AdvancedDrawPanel. Implements the Advanced Object and Network Manager layer of the InterDraw application. This layer controls all network communication along with the creation and management of all advanced objects such as animations.

- MainMenuFrame.java – Contains the main Java Frame for the InterDraw application. The application is initialized from this frame. The color selection Frame, which provides the user interface for selecting the current drawing color, is also contained within this file.

- ServerStatusPanel.java – Listens for ServerStatusEvents and displays a visual representation of the Server Status.

- Stopwatch.java – A modified version of the Stopwatch class by David Geary. It simulates a physical stopwatch by keeping track of the passed time since a certain event.

- StopwatchClient.java – An interface used by classes to receive "ticks" from the Stopwatch, which mark the passage of time. Modified from StopwatchClient class by David Geary.

- StoredObject.java – Contains a stored representation of a drawing object. It is able to build and parse the string representations of all drawing objects.

- ToolButton.java – A lightweight image button, which stays down when pressed. Only one ToolButton may be pressed down at any time. All others will be "unpressed" when another is selected.

- ToolOptionPanel.java – Provides the user interface for selecting different options for a drawing tool.

- ToolPanel.java – Provides the user interface for selecting from a group of drawing tools.

- UpdateThread.java – A thread that runs in the background of the InterDraw program and generates Update events every 5 minutes to automatically update the current image.

B.2.3 idp/AdvRubberband/

- Rubberband.java – An abstract class for drawing tools/objects with "rubberbanding" capabilities. It provides all mouse handling and storage capabilities for these implementing classes.

- RubberbandEllipse.java – Extends the Rubberband class. Generates elliptical shaped rubberbanding drawing objects.

- RubberbandLine.java – Extends the Rubberband class. Generates rubberbanding line shaped drawing objects.

- RubberbandRectangle.java – Extends the Rubberband class. Generates rectangular shaped rubberbanding objects.

B.2.4 idp/anim/

- AdvancedSprite.java – An animated object that may be placed within a SpriteAnimator Container class. It uses an AnimatedPath to calculate its location within the Container and a Sequence class for maintaining its current image.

- AnimationParameter.java – Holds parameter values and objects that are used to recreate an animated object.

- AnimationPath.java – An abstract class for maintaining a time based path within a certain boundary area.

- CollisionArena.java – An interface for defining an area where collisions of AdvancedSprite objects may take place. Modified from CollisionArena class by David Geary.

- EdgeCollisionDetector.java – Detects collisions between a boundary and an AdvancedSprite object. Modified from EdgeCollisionDetector class by David Geary.

- SpriteCollisionDetector.java – Detects collisions between two or more AdvancedSprite objects. Modifed from SpriteCollisionDetector class by David Geary.

- CirclePath.java – Extends the AnimationPath Class. Generates a time varying path in the shape of an ellipse.

- LinePath.java – Extends the AnimationPath Class. Generates a time varying path in the shape of a line or vector.

- SinePath.java – Extends the AnimationPath Class. Generates a time varying path in the shape of a sine wave.

B.2.5 idp/anim/sequence/

- Sequence.java – An abstract class for providing a sequence of images.

- CubeSequence.java – Extends Sequence class. Provides a dynamic sequence of images, creating a 3-Dimensional rotating wire-frame cube animation.

- ModelSequence.java – Extends Sequence class. Provides a dynamic sequence of images, simulating a 3-Dimensional rotating wire-frame model.

- ImageSequence.java – Extends Sequence class. Provides a static sequence of images that may be repeated as needed.

B.2.6 idp/database/

- Database.java – Contains all database connection information.

B.2.7 idp/draw/

- AllPurposeDrawPanel.java – Implements the Drawing Tool Manager layer of the InterDraw client application. It controls the selection and management of drawing tools.

- AdvancedDrawPanel.java – Extends AllPurposeDrawPanel. Implements the Basic Object and Display Manager layer of the InterDraw client application, which controls the display of all basic drawing objects.

- Drawing.java – An abstract class for drawing tools/objects that require intricate "free hand" like capabilities. It provides all mouse handling and storage capabilities for these implementing classes.

- DrawAirbrush.java – Extends Drawing class. Generates airbrush like drawings.

- DrawPencil.java – Extends Drawing class. Generates free hand line drawings.

B.2.8 idp/event/

- AnimationEvent.java – An Event object that is generated when an Animated object needs to be created or updated.

- AnimationPathEvent.java – An Event object that is generated when an AnimationPath has been created and needs to be updated in another object.

- ServerStatusEvent.java – An Event object that is generated whenever the connection status of the InterDraw server changes.

B.2.9 idp/interfaces/

- AnimationEventListener.java – An interface for objects receiving AnimationEvents.

- AnimationPathEventListener.java – An interface for object receiving AnimatinPathEvents.

- Drawable.java – An interface a drawing tool must implement. It provides the ability to control the Graphic object being drawn too, the bounds of the object, the color and whether or not the drawing object/tool is currently active.

- Fillable.java – An interface that allows a drawing object to have outline and filling options.

- ServerStatusListener.java – An interface that allows an object to receive ServerStatusEvents.

- Sizeable.java – An interface that allows a drawing object to have sizing capabilities.

- SpriteAnimator.java – An interface that allows an object to contain AdvancedSprite objects.

- Storable.java – An interface that all drawing objects must implement in order to store and restore the object.

B.2.10 idp/model

- Cube3D.java – Extends j class that provides the ability to easily generate and manipulate a 3-Dimensional model.

B.2.11 idp/net

- BusyFlag.java – A synchronization flag that provides exclusive access to data and methods.

- NetTransferObject.java – Sends and receives InterDraw object data from and to the server and client.

B.2.12 idp/utils

- Convert.java – Contains useful conversion utilities, such as converting an integer into two character variables.

B.2.13 oracle/

- Contains all classes required by the Java JDBC API to access an oracle database.

B.2.14 gjt/

- Contains a useful graphical library by David Geary.

Appendix C

User Manual

C.1 Layout Help

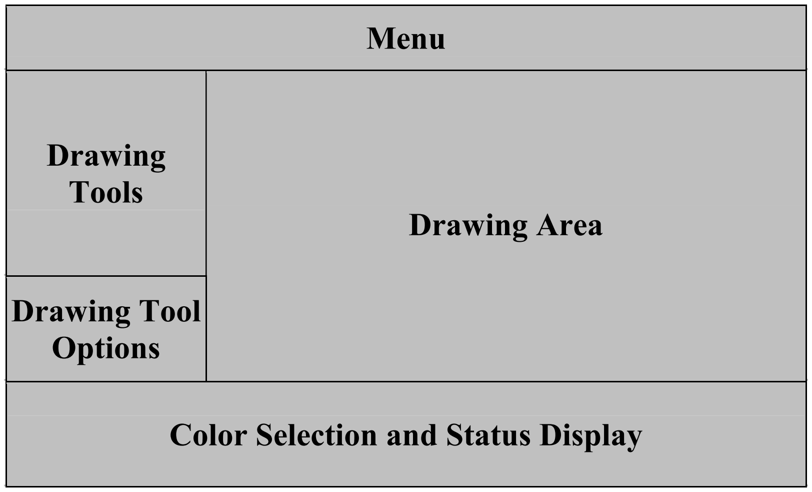


Figure C. 1: Layout of the InterDraw interface

C.2 Drawing Tool Icon Help

Icon	Description
╲	This button allows you to add lines to the image.
□	This button allows you to add rectangle shaped objects to the image.
⬭	This button allows you to add elliptical objects to the image.
✏	This button allows you to add free hand lines to the image.
🖌	This button allows you to add free hand lines to the image using an airbrush effect.
⌫	This button allows you to remove drawn objects from the image. ***Note: you can only remove objects that YOU create and not those created by others***
🎨	This button allows to you to select a different color than the basic ones displayed.

Table C. 1: Description of the drawing tool icons

C.3 Menu Help

Menu	Option	Description
Image	*Select*	Allows you to either select another image to work on or create a new image.
Network	*Update*	Will send all new information to the server and update your image with any new objects that others have added to the image. This option is disabled if the connection to the server is lost. **Note: this also occurs automatically every 5 minutes as well.**
	Check	Checks to see if a connection can be made to the server.
Animation	*Insert*	Allows you to create an animated object to add to the current image.

Table C. 2: Application menu option descriptions

C.4 Status Image Help

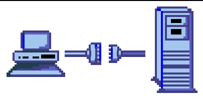
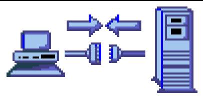
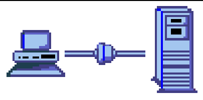
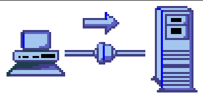
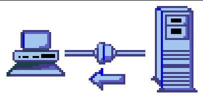
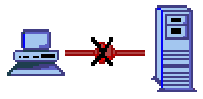
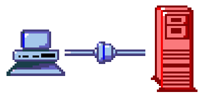
Status Image	Description
	A connection to the server has not been established.
	Attempting to connect to the server.
	Connection has been made to the server
	Sending image data to server
	Receiving image data from server
	The connection to the server was lost
	An error has occurred on the server.

Table C. 3: Status image descriptions

C.5 Animation Help

C.5.1 Main Interface

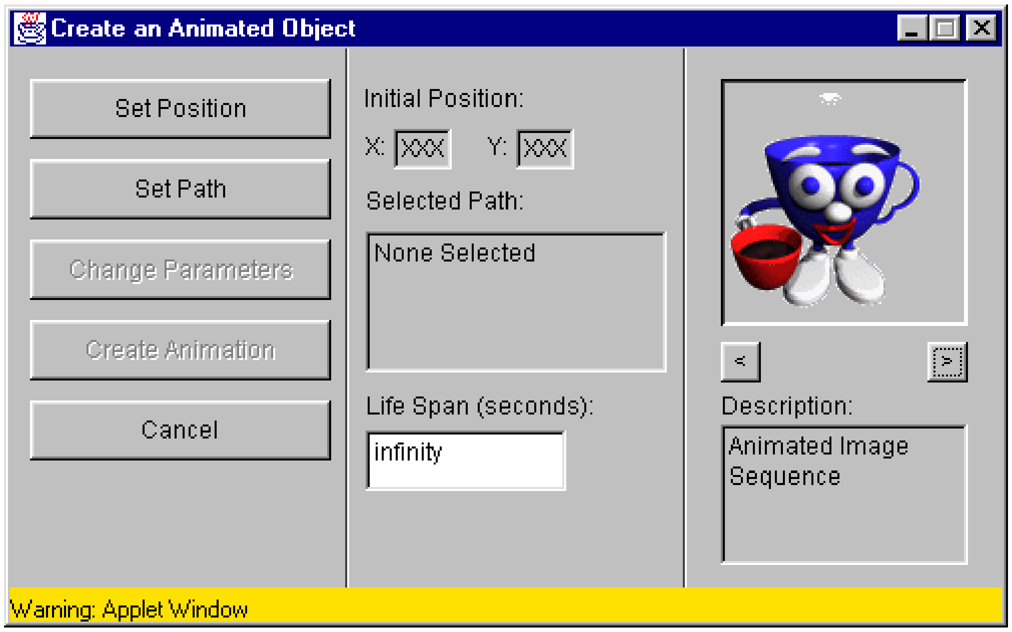


Figure C. 2: Screen shot of Animated object creation screen

The main interface is split into three sections. The left section contains the option buttons, which allow the user to set the different options for an animation. The center section contains descriptions of the selected attributes for the animation to be created. The right section contains an active view of the animation being created and a description of the animation.

C.5.2 Option Buttons

Button Name	Description
Set Position	Allows you to set the starting location of the animated object. When this button is selected, you are brought back to the main application where you use the mouse button to select a location on the drawing area.
Set Path	Allows you to select a path for the animated object to follow. Brings up a path selection dialog.
Change Parameters	Not Implemented. Will allow user to modify animation specific options.
Create Animation	Creates the animated objects and adds it to the current image.

Button Name	Description
Cancel	Closes animation creation dialog and returns you to the drawing screen.

Table C. 4: Option button descriptions

C.5.3 Information Section

Name	Description
Initial Position	x and y screen position of the starting location for the animated object.
Selected Path	A text description of the selected path the animated object will follow.
Life Span	An editable field that allows the user to specify how long the animated object will last within the current image. The time is specified in seconds.

Table C. 5: Descriptions of the information being displayed in the information section of the Animated object creation screen

C.5.4 Animated Object Selector

Item	Description
[<] and [>] Buttons	Move through the list of animated objects that are available for insertion into the current image. *Currently only 2 sample animations have been implemented and added to this list.*
Description	A text description of the displayed animated object.

Table C. 6: Descriptions of the objects contained within the Animated object selection section of the Animated object creation screen

C.5.5 Animated Path Dialog

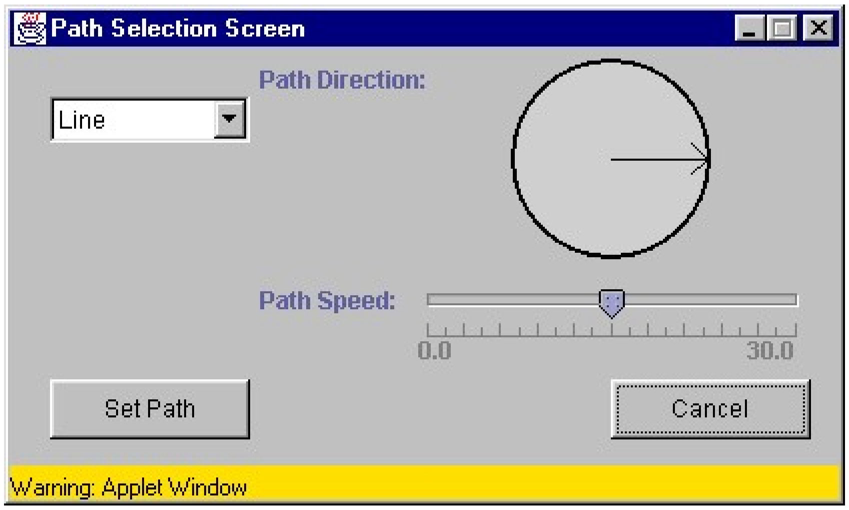


Figure C. 3: Screen shot of path selection screen

The following table contains a listing of all available path types and their options.

Path Type	*Option*	Description
Line Path	*Path Direction*	Allows the user to specify the direction of the path.
	Path Speed	If we consider the path as a vector, then the path speed is the length of the vector. For example, if the direction is to the right, that would give us the normalized vector of (0,1). If we wanted a speed of 20, then the vector length would equal 20. Therefore the line vector is (0, 20).
Ellipse Path	*X Radius*	Allows the user to specify the width of the elliptical path.
	Y Radius	Allows the user to specify the height of the elliptical path.
	Delta Angle	The Delta angle is the change in angle that is incremented to the current angle to create the next location on the elliptical path. For example, if the delta angle was 10, then the path would follow an elliptical pattern at 0, 10, 20, 30, 40...350, 360 degrees.
Sine Path	*Amplitude*	Allows the user to specify the amplitude of the sine wave used in the sine the path.
	Frequency	Allows the user to specify the frequency of the sine wave used in the sine path.

Path Type	*Option*	Description
	Delta Angle	The Delta angle is the change in angle that is incremented to the current angle to create the next location on the elliptical path. For example, if the delta angle was 10, then the path would follow a sine pattern at 0, 10, 20, 30, 40...350, 360 degrees.(*can be converted to radians as well*)

Figure C. 4: Descriptions of the settable options available for each path type

Appendix D

User Testing Data

D.1 Quantitative Information

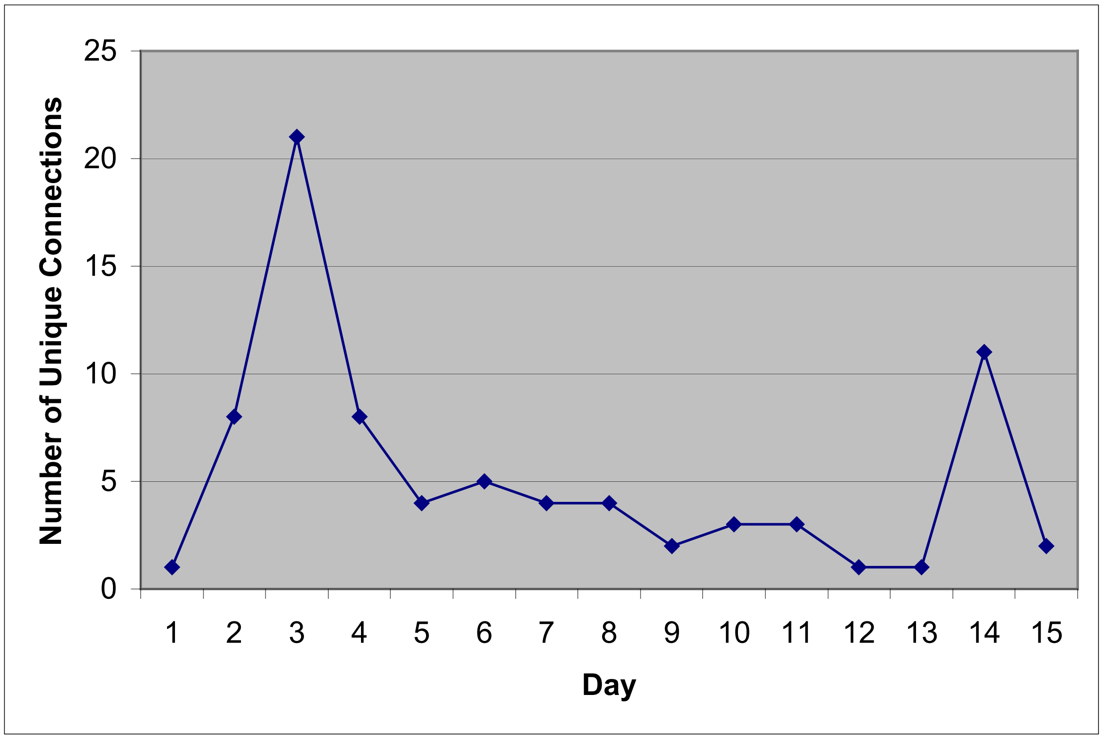


Figure D. 1: The number of unique user connects per day during user testing

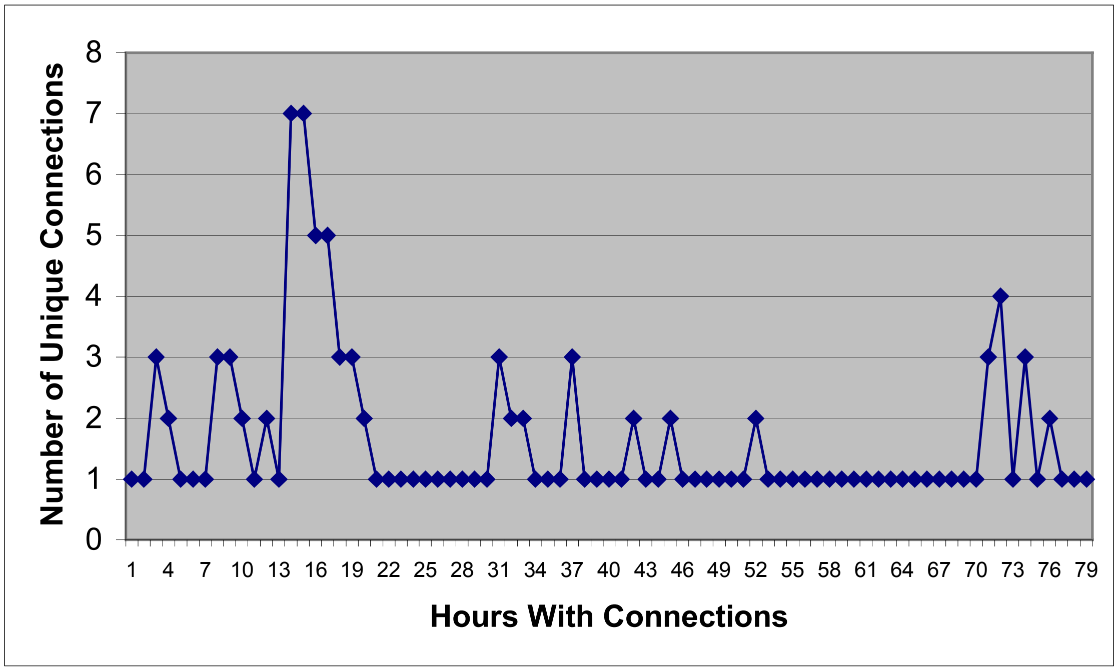


Figure D. 2: The number of unique connections per hour during user testing

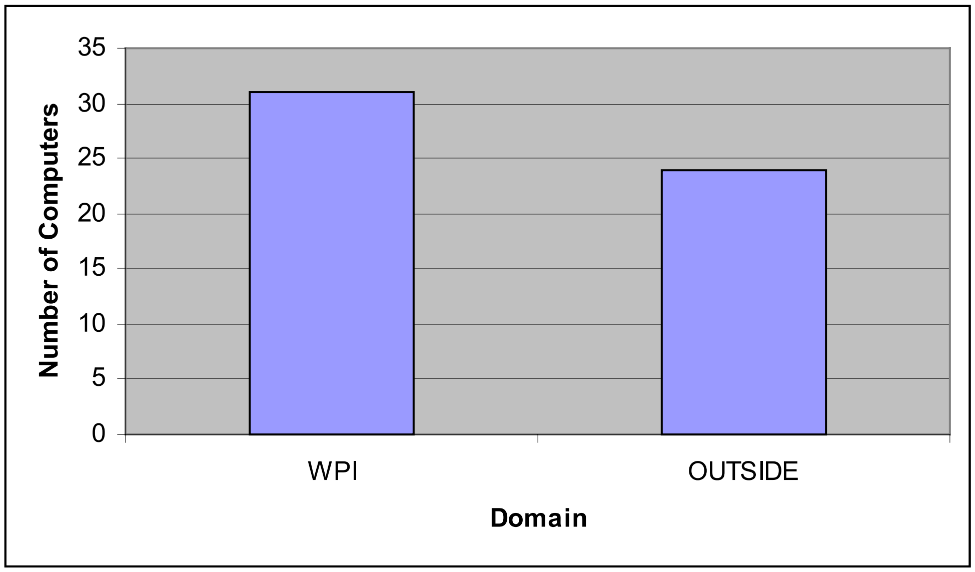


Figure D. 3: The number of computers within the WPI domain used for in user testing Vs the number of computers outside of the WPI domain

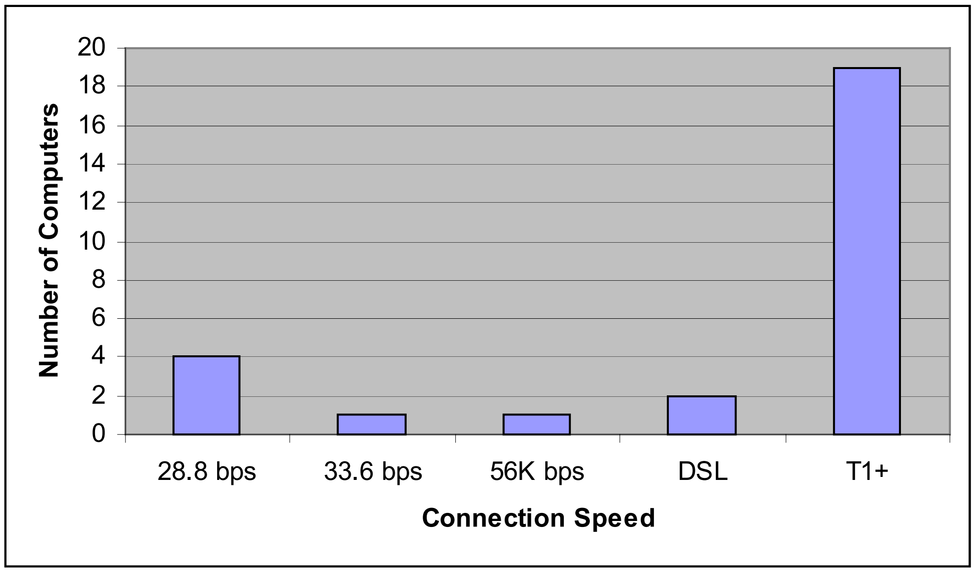


Figure D. 4: The connection speeds of the computers used in user testing

D.2 Qualitative Information

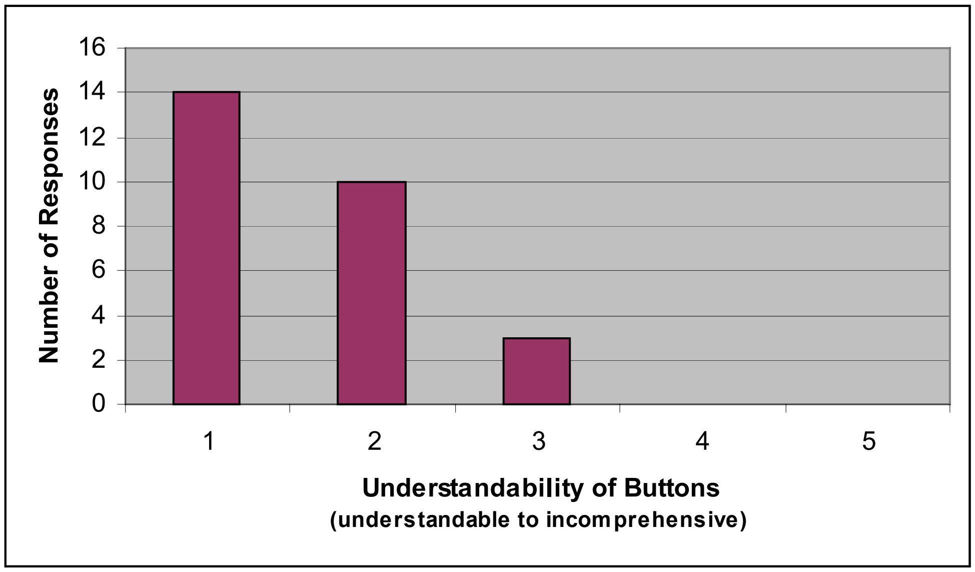


Figure D. 5: Results of survey as relates to the understandability of the buttons used in the InterDraw application

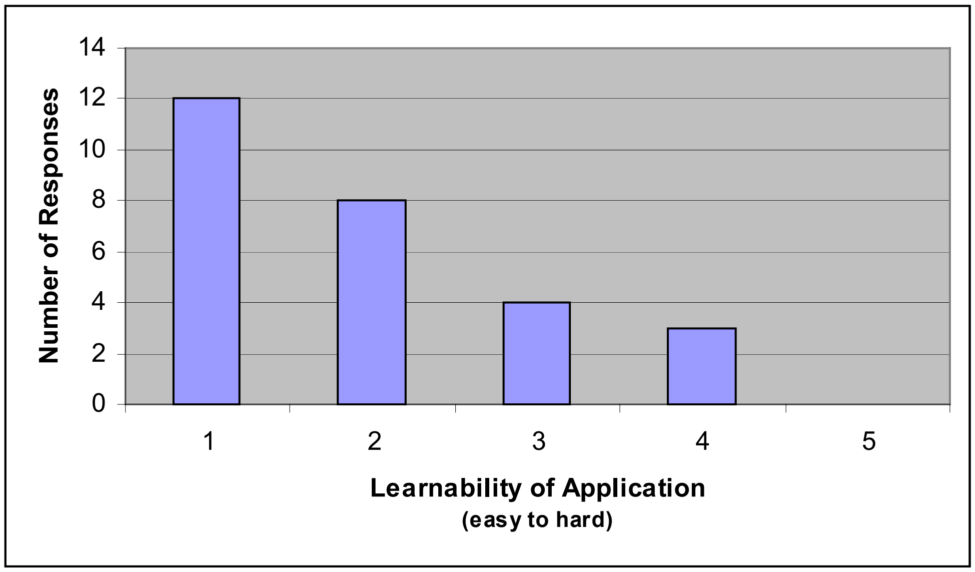


Figure D. 6: Results of survey as relates to how easy the InterDraw program was to learn

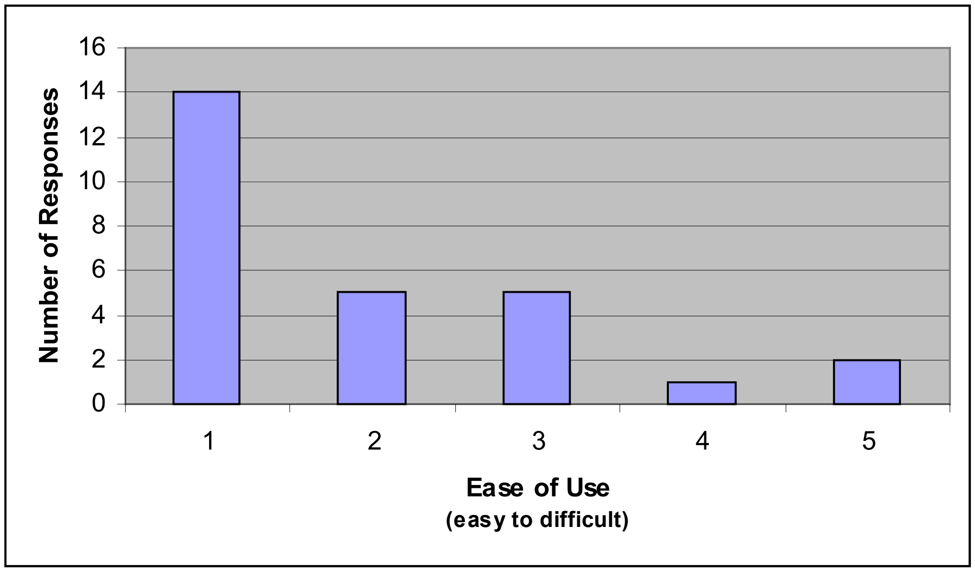


Figure D. 7: Results of survey as relates to how easy the InterDraw application was to use

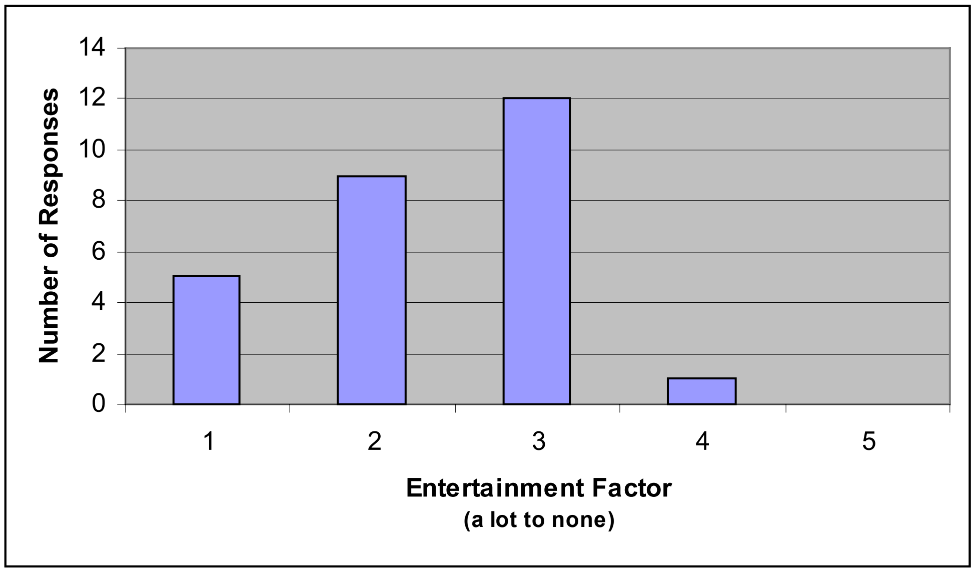


Figure D. 8: Results of survey as relates to the entertainment level of the users from using the InterDraw application

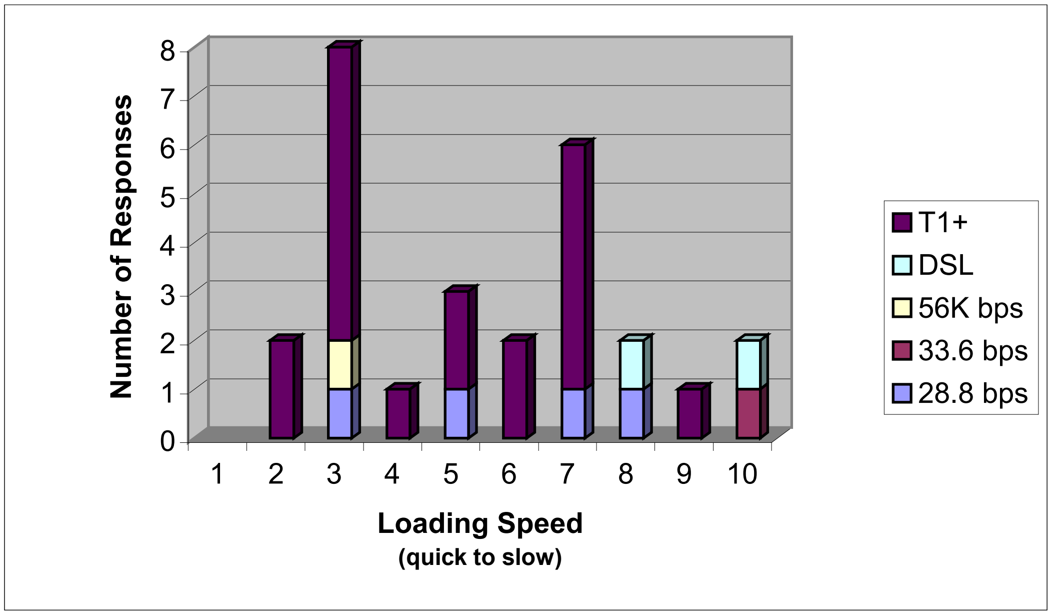


Figure D. 9: Results of survey as relates to the loading time of the InterDraw application and the speed of the user's connection

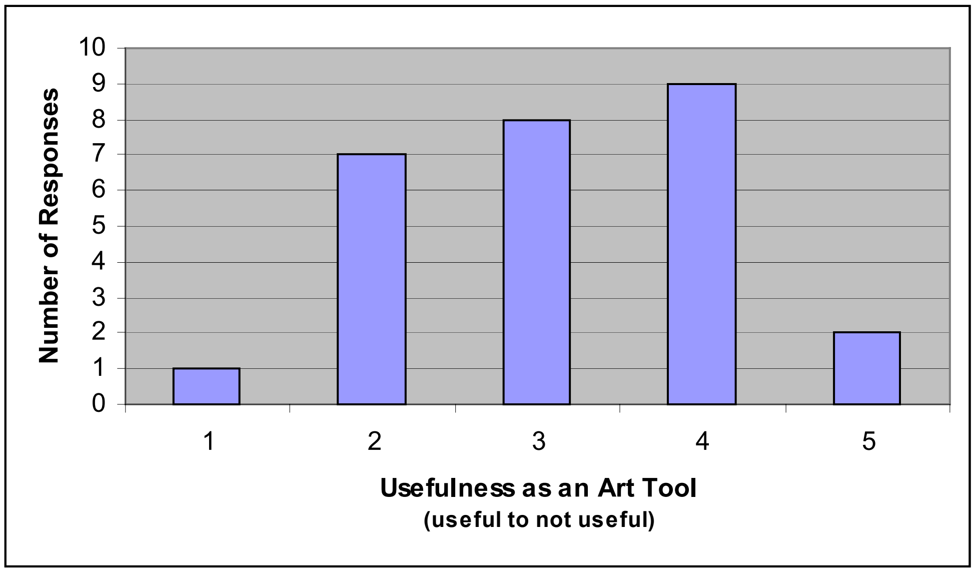


Figure D. 10: Results of survey as relates to the usefulness of the InterDraw applications as a useful art tool

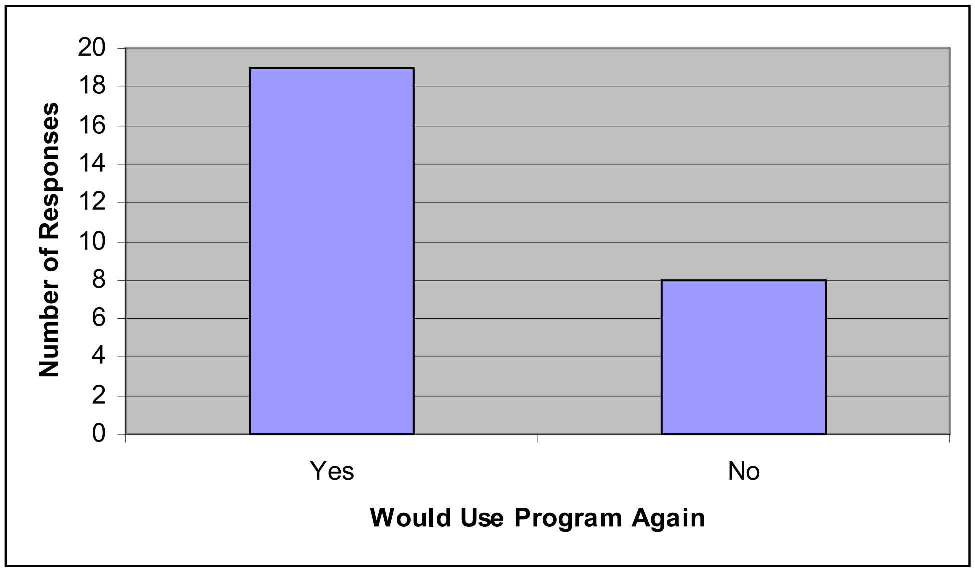


Figure D.11: Results of survey illustrating the number of users who would use the InterDraw application again Vs those who would not

Appendix E

Sample Collaborative Images Created With InterDraw

E.1 Image 1

Image 1 is a good example of how one artist's vision can be completely turned around by the vision of another artist. The image starts out as a serene ocean view. As the image progresses, the addition of birds, clouds and a boat help to add to this theme. The final artist changes the feel of the scene by adding a hunter taking aim at the birds from a camouflaged boat.

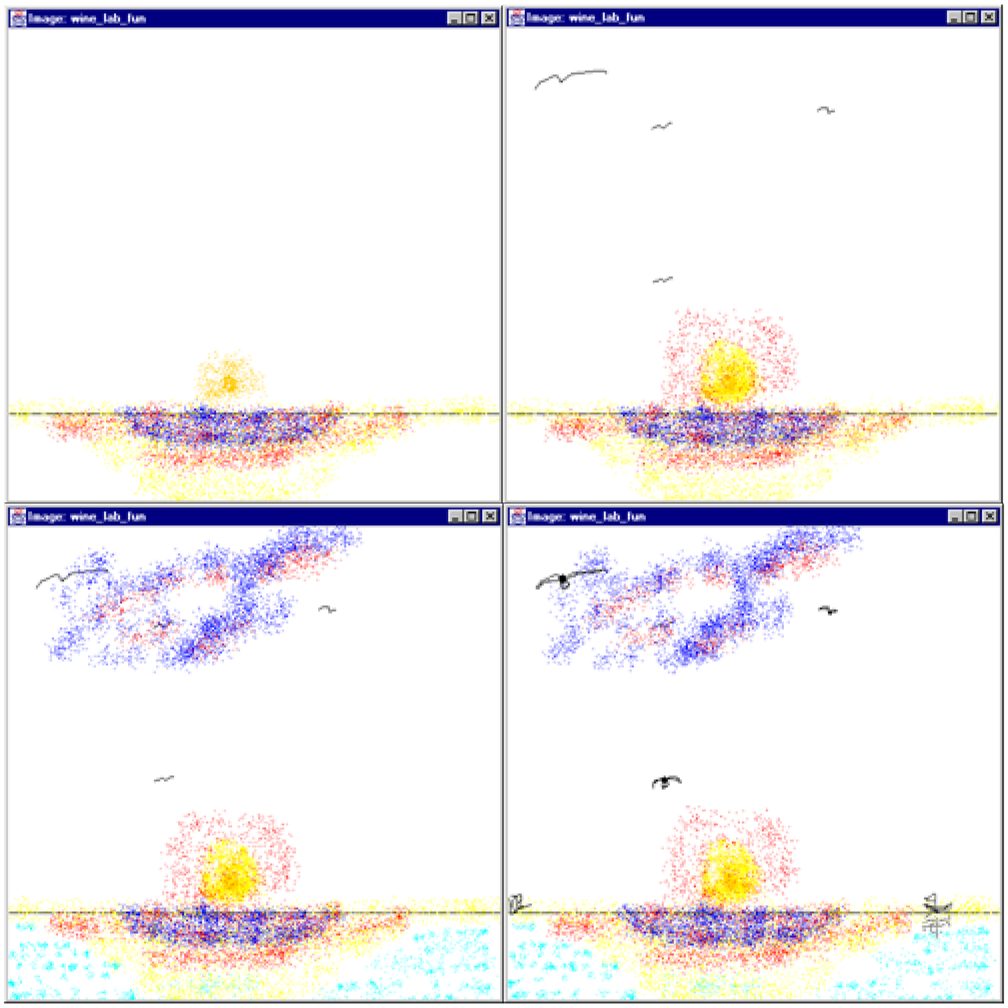


Figure E. 1: Progression of image 1

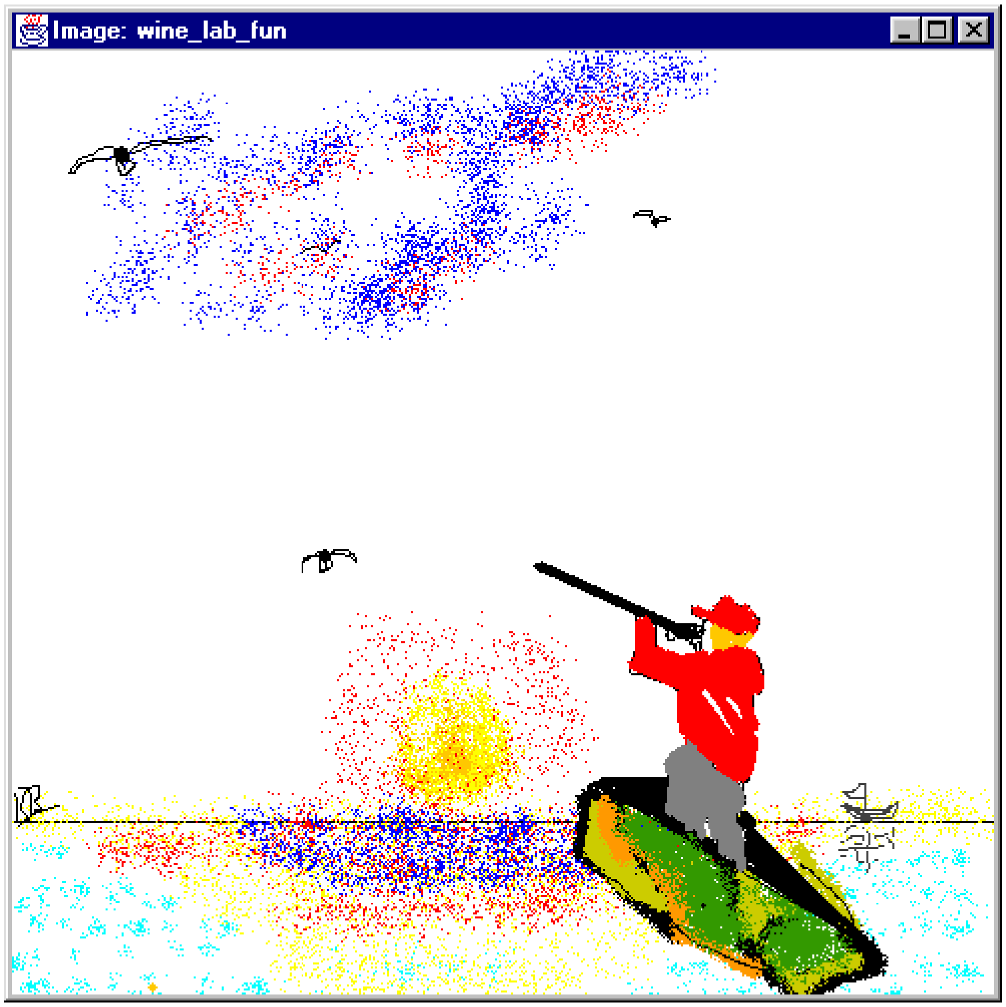


Figure E. 2: Completed image 1

E.2 Image 2

Image 2 did not go through as many iterations as image 1, but even with the few iterations available it is easy to see the different views of the artists. The original artist once again started out with a calm sea scene of a surfer riding a wave. The next artists changed the mood of this image by providing the addition of a giant shark attacking the surfer. Following this trend, the next artist drew a missile flying toward the shark.

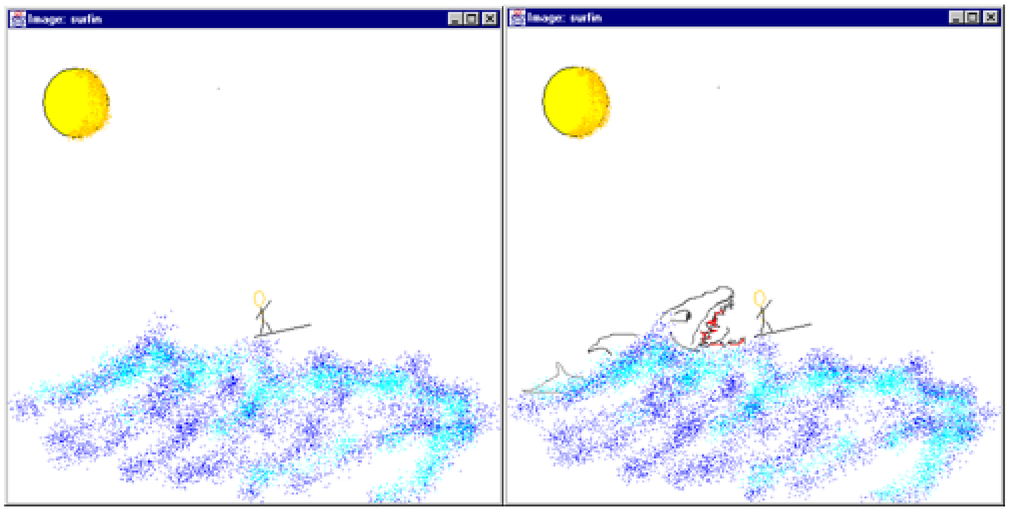


Figure E. 3: Progression of image 2

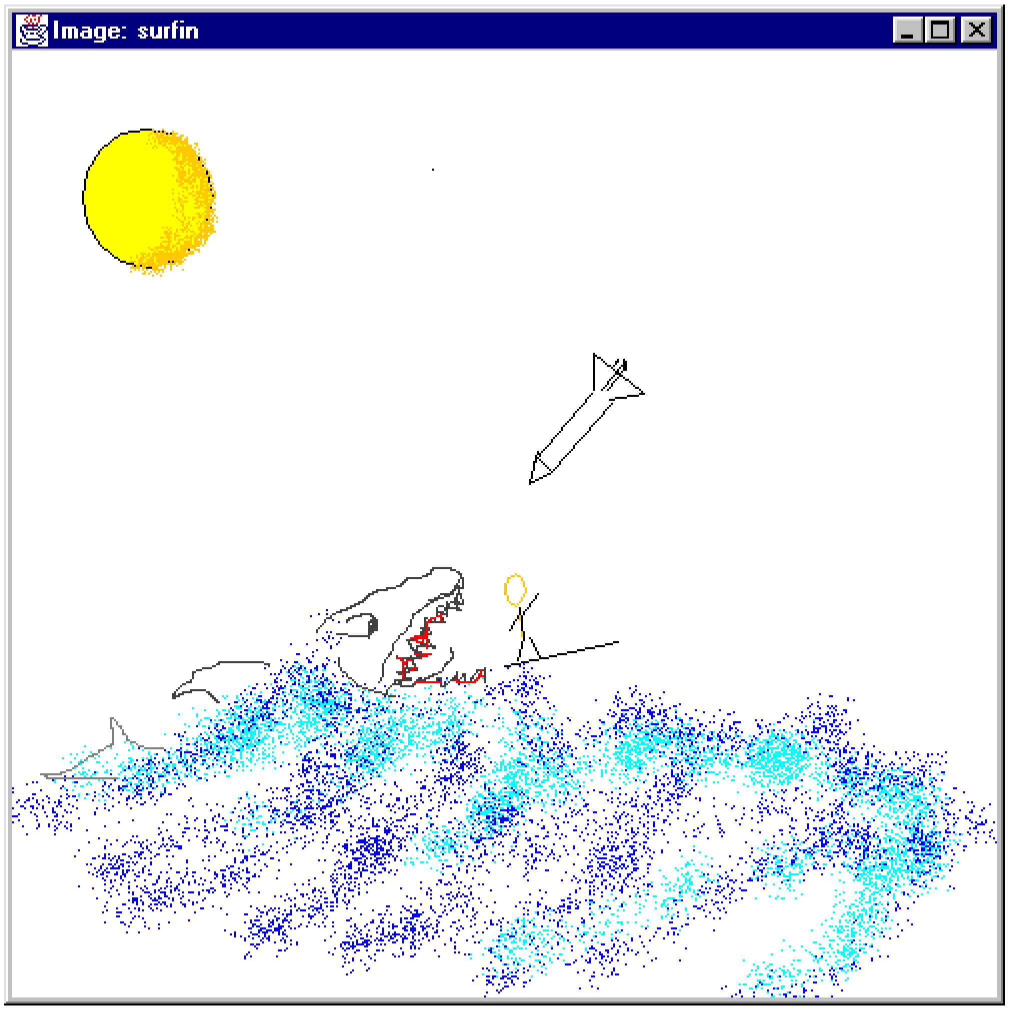


Figure E. 4: Completed Image 2

E.3 Image 3

Unlike with the previous two images, the overall theme of image 3 did not change over the life of the image. Instead, this image represents what happens when two artists modify an image simultaneously. If you notice, in the third progression in Figure E. 5 both artists have added buildings to the scene. This overlap can create some unexpected results. In the final image, the artists worked together to generate a compromised and completed image.

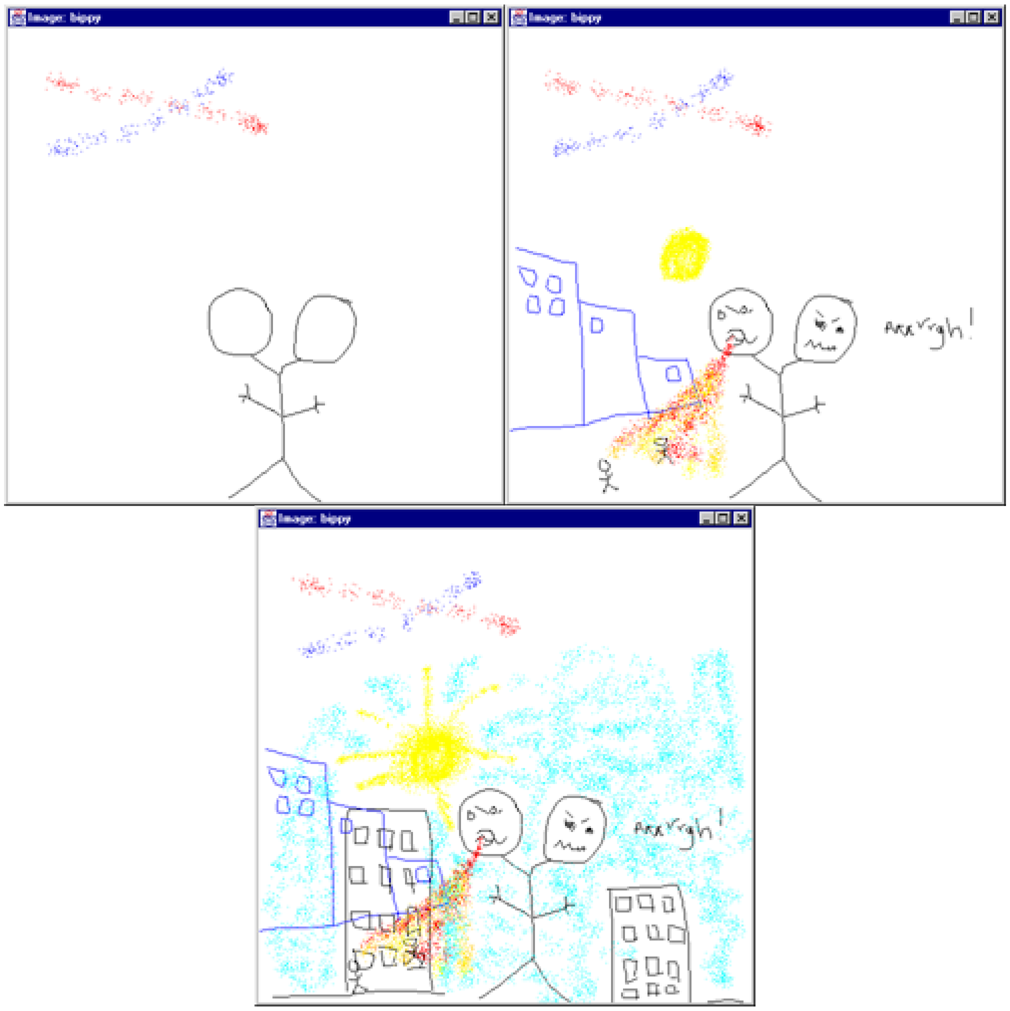


Figure E. 5: Progression of image 3

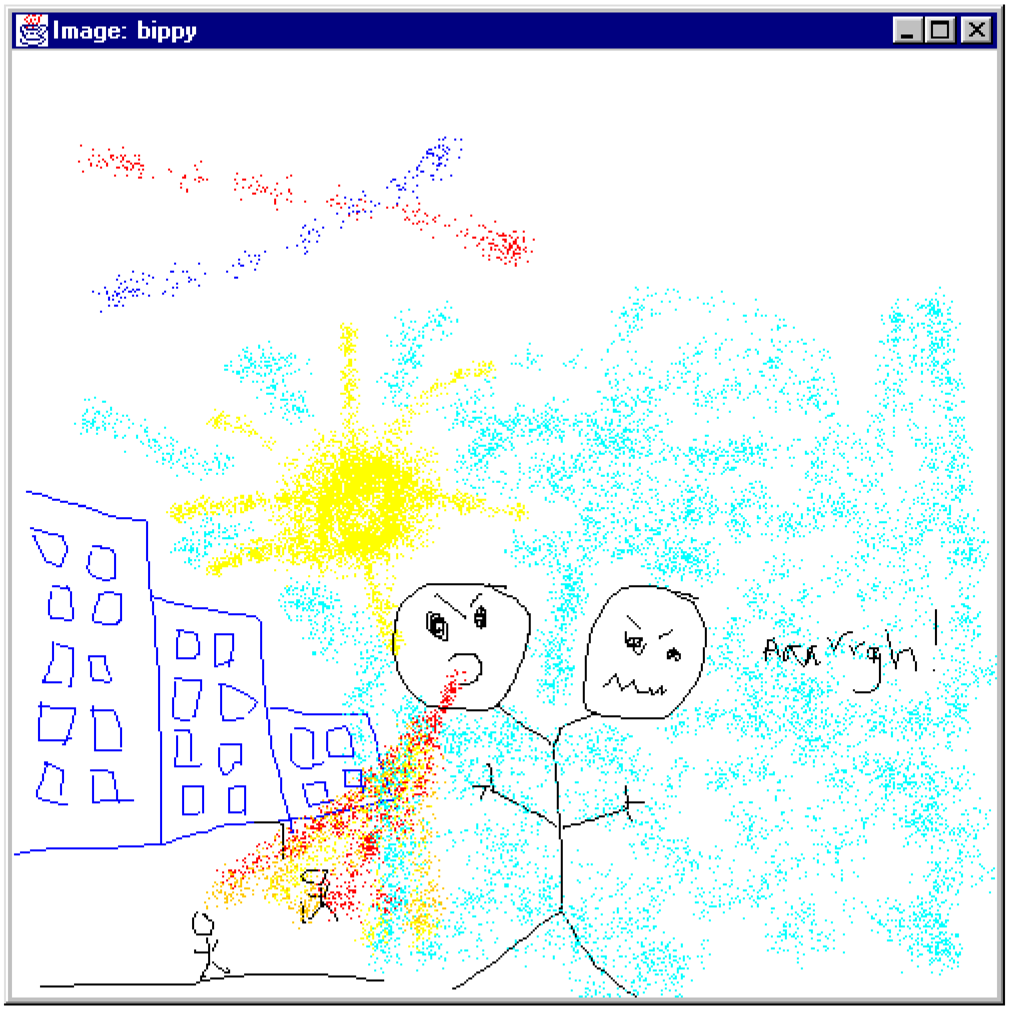


Figure E. 6: Completed image 3

E.4 Image 4

Image 4 is a wonderful example of an image without direction. It is apparent from the start that the original artist did not have a goal in mind when he or she started the image. This trend is carried over into the work of the following artists, where images and objects were added seemingly at random. The resulting image is a truly unique image that would not have been possible without the collaboration of a number of artists.

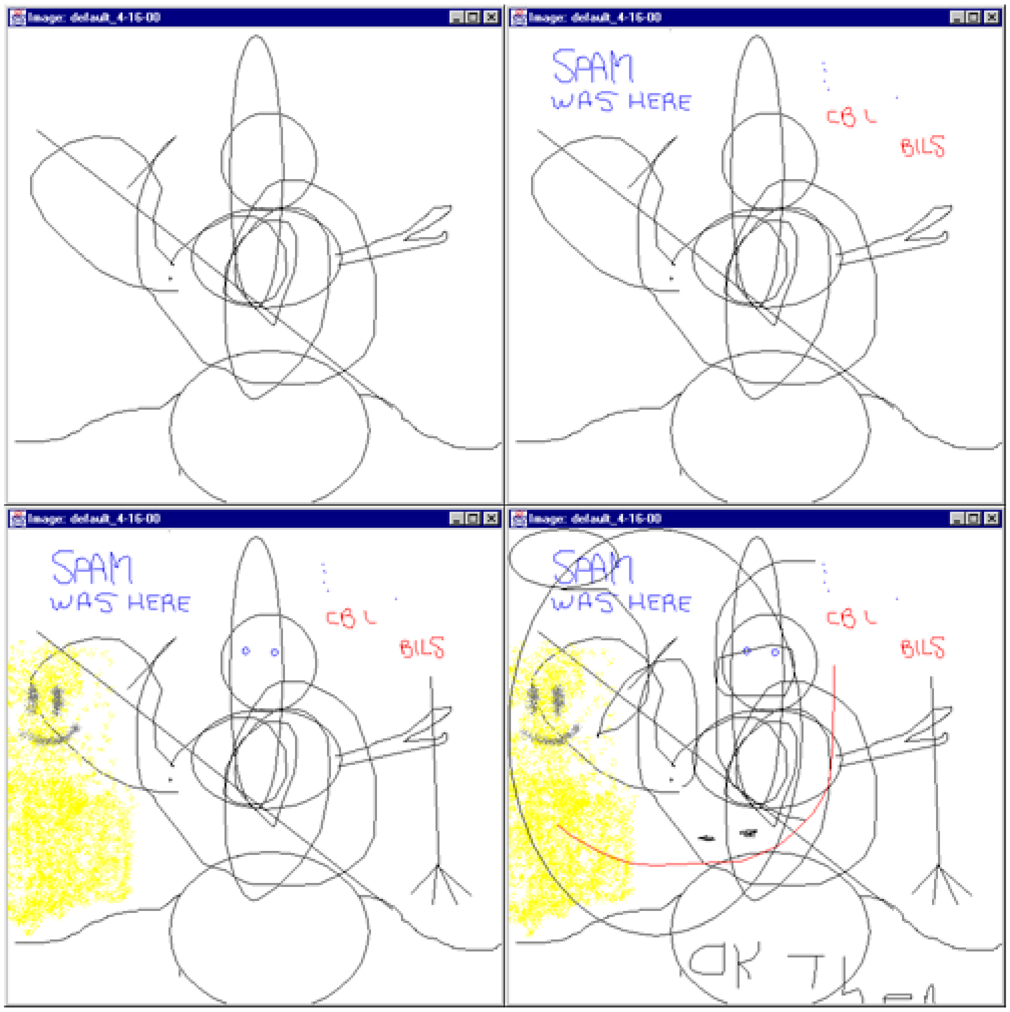


Figure E. 7: Partial progression of image 4

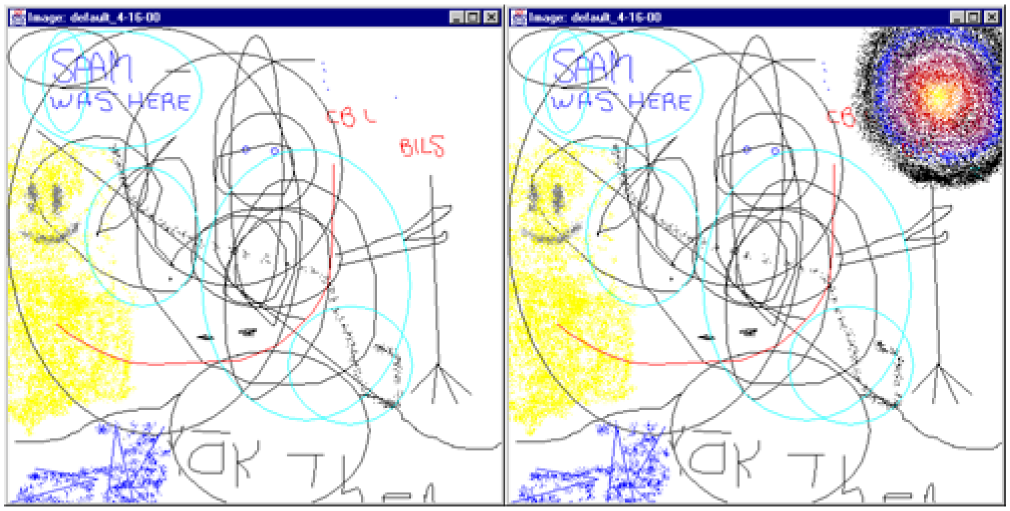


Figure E. 8: Final progression of image 4

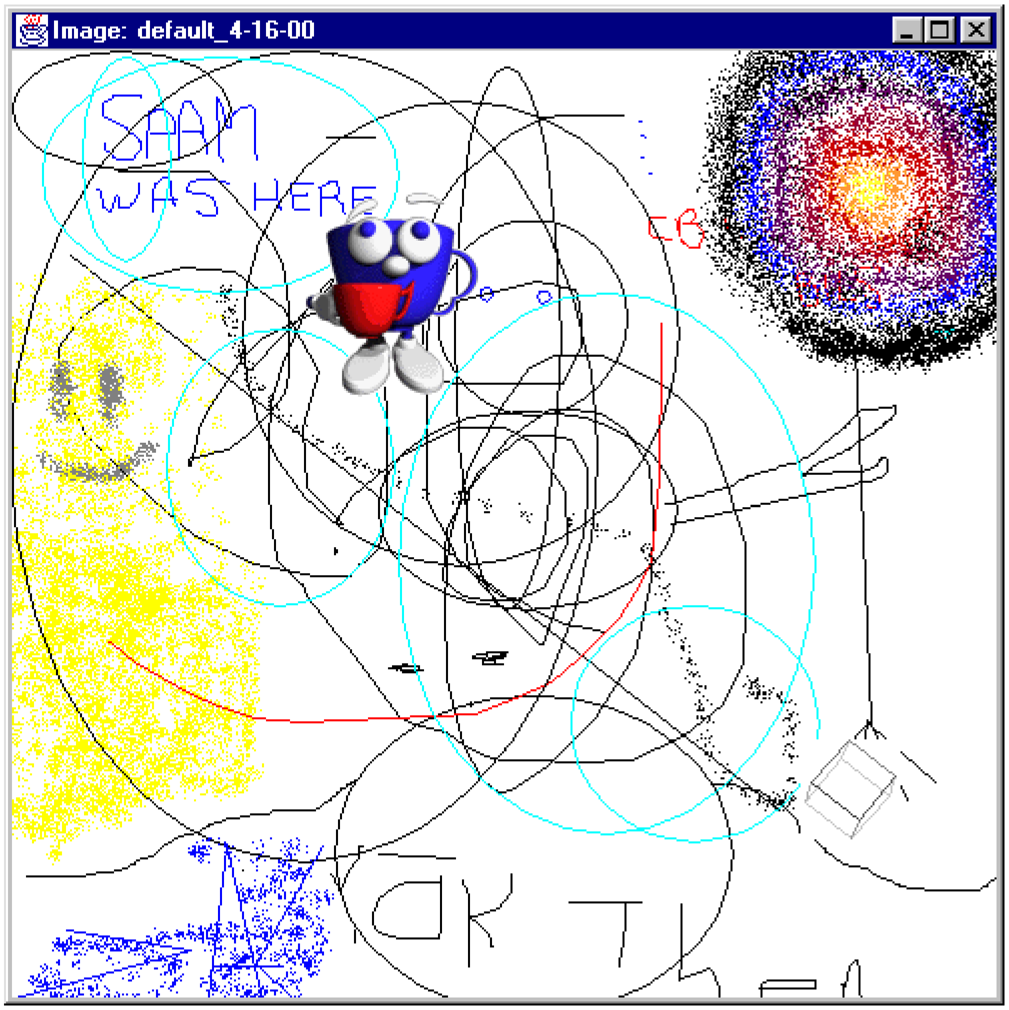


Figure E. 9: Completed image 4

Appendix F

Software Availability

F.1 Where to get InterDraw

InterDraw is a free, open source application and may be obtained at **http://davis.wpi.edu/~InterDraw/**.

F.2 How to compile and run

To compile the InterDraw application, the Java 2 JDK version 1.2.1 or higher is required. This may be downloaded from Sun Microsystems at **http://java.sun.com/.** You can then use the **javac** compiler in the JDK to compile the **.java** files included with the InterDraw application.

The client application can be run from an Internet browser using the .html file included with the source code, or using the **appletviewer** program provided by the JDK. The server application is run using the **java** program in the JDK with the following syntax:

```
java Server Server$InterDrawService port_num
```

Where **port_num** is the port number the InterDraw client applications will user to connect to the server.

Bibliography

[Ben96]	Bentley, Chris, Animating Multidimensional Scaling to Visualize Large N-Dimensional Data Sets, M.S. Thesis, WPI, 1996.
[Cho97]	Choi, Yearn Hong, “Crafting a definition of art.”, World and I, (July 1997): pp. 112 – 113.
[CT98]	Christiansen, Tom and Torkington, Nathan, Perl Cookbook, O’Reilly & Associates, Inc., Sebastopol, CA, 1998.
[DDN00]	Deitel, Deitel and Nieto, Internet & World Wide Web: How to Program, Prentice Hall, NJ, 2000.
[Die00]	Dietz, Steve, Curation (on) the Web, Online, Internet, April 2000, Available: http://www.archimuse.com/mw98/papers/dietz/dietz_curatingtheweb.html
[Eur00]	European Paintings.com, Italian Paintings Galery, Online, Eruopean Paintings.com, Internet, April 2000, Available: http://www.europeanpaintings.com/italian
[FD97]	Flanagan, David, JAVA in a Nutshell: A Desktop Quick Reference, O’Reilly & Associates, Inc., Sebastopol, CA, 1997.
[Fla97]	Flanagan, David, Java Examples in a Nutshell: A Tutorial Companion to Java in a Nutshell, O’Reilly & Associates, Inc., Sebastopol, CA, 1997.
[GC99]	Cornell, G. and Horstmann, C.S., Core JAVA 2: Volume I – Fundamentals, Sun Microsystems Press, Palo Alto, CA, 1999.
[Gea97]	Geary, David M., graphic JAVA 1.1: Mastering the AWT, Sun Microsystems Press, Mountain View, CA, 1997.
[Gol99]	Gollifer, Sue, Collaborative Art Projects, Online. University of Brighton, Internet. March 1999. Available: http://www-ctiad.adh.bton.ac.uk/ctiad/sue/collabe.html
[Gom66]	Gombrich, E. H., The Story of Art, The Phaidun Press, Greenwhich, Conn., 1966.
[Gre61]	Greenberg, Clement, Art and Culture, Beacon Press Boston, Toronto, 1961.

[Gro99] Growing Through It Org., Growing Through It, Online, Growing Through It Org., Internet, March 1999, Available: http://www.growingthroughit.org/hp/index.htm

[Har97] Harold, E.R., JAVA Networkd Programming, O'Reilly & Associates, Inc., Sebastopol, CA, 1997.

[Hil90] Hill, F.S., Computer Graphics, Prentice Hall, Englewood Cliffs, NJ, 1990.

[Kre93] Kreiner, L.E., "Toward a Definition of Art", Art Education, (May 1993): pp. 7 – 11.

[Lei52] Leicht, Hermann, History of the World's Art, Spring Books, Czechoslovakia, 1952.

[Mas98] Mason, A.L., WEBART: Collaborative Development of Artwork on the Web, A Major Qualifying Project, WPI, 1998

[Mer94] Merriam-Webster, Inc, Merriam-Webster's Collegiate Dictionary, 10, Merriam-Webster, Inc., USA, 1994.

[May92] Mayhew, D.J., Principles and Guidelines in Software User Interface Design, Prentice Hall, NJ, 1992.

[Mit99] Mitchel, Bonnie, International Internet ChainArt Project (6 March 1994), Online, Syracuse University, March 1999, Available: http://ziris.syr.edu/chainartdocs/chainart.html

[MPL65] Murray, Peter and Linda, Dictionary of Art and Artists, Frederick A. Praeger, NY, 1965.

[NS98] Naughton, P. and Schildt, H., The Complete Reference: JAVA 1.1, Osborne McGraw-Hill, Berkeley, CA, 1998.

[Ora00] Oracle Corporation, Oracle Software Powers the Internet, Online, Oracle Corporation, Internet, Available: http://www.oracle.com

[OW99] Oaks, S. and Wong, H., JAVA Threads, O'Reilly & Associates, Inc., Sebastopol, CA, 1999.

[Pal99] Palmatier, Gary, Ideas to Images (3 March 1999), Online, Sonic, Internet, March 1999, Available: http://www.sonic.net/~ideas/index.html

[PBS00] PBS Online, Life on the Internet, Online, PBS Online, Internet, Available: http://www.pbs.org/internet/timeline

[Ray09] Raymond, G.L., Genesis of Artform, G. P. Putnam's Sons, NY, 1909.

[Ree97] Reese, George, Database Programming with JDBC and JAVA, O'Reilly & Associates, Inc., Sebastopol, CA, 1997.

[Ros99] Rosenfeld, Klause, HypArt. (9 Apr 1999), Online, Internet Informationssysteme GmbH. Internet, March 1999, Available: http://www.work.de/cgi-bin/HypArt.sh

[Sau00] Sausalito Art Festival, The Sausalito Art Festival – Computer Art, Online, Virtual Success, Internet, April 2000, Available: http://www.sausalitoartfest.org/computer.html

[Sch99] Schach, S.R., Classical and Object-Oriented Software Engineering: With UML and Java, WCB/McGraw-Hill, U.S.A. 1999.

[Spa99] Spalter, Anne, The Computer in the Visual Arts, Addison-Wesley, USA , 1999.

[Str00] Strauss, Howard, The future of the Web, Intelligent Devises and Education, Online, Educause, Internet, April 2000, Available: http://www.educase.edu/ir/library/html/cnc9809/cnc9809.html

[SM00] Sun Microsystems, Java Discussion Forum, Online, Sun Microsystems, Internet, Available: http://froum.java.sun.com

[Sun00] Sun Microsystems, The Source for Java™ Technology, Online, Sun Microsystems, Internet, Available: http://java.sun.com

[Tan96] Tanenbaum, Andrew S., Computer Networks, Prentice Hall, NJ, 1996.

[Tex99] TexShare, Libraries, Universities and Colleges (11 Aug. 1995), Online, TexShare, March 1999, Available: http://www.texshare.utexas.edu/Search/libs.html

[Til84] Tilghman B.R., But is it Art?, Basil Blackwell Publisher Limited, England, 1984.

[Tol62] Tolstóy, Leo, What is Art? And Essays on Art, Oxford University Press, London, 1962.

[Var00] Varian, Varian's Dreamview Collaborative Art Project, Online, Internet, April 2000, Available: http://www.varian.net/dreamview/dcollaborative/index.html

[WCS96] Wall, Christiansen and Schwartz, Programming Perl, O'Reilly & Associates, Inc., Sebastopol, CA, 1996.

[Wei97] Weintraub, Annette, "Art on the Web, the Web as Art." Communications of the ACM, 40 (Oct 1997): 97-102.

[Wor99] World Wide Arts Resources, World Wide Arts Resources, Online, Internet, March 1999, Available: http://www.wwar.com

[WW96] Watt, A. and Watt, M., Advanced Animation and Rendering Techniques: Theory and Practice, ACM Press, Great Britain, 1996.

[Yah00] Yahoo! Inc., Yahoo! Arts:Artists:Collaborative Projects, Online, Yahoo! Inc., Internet, April 2000, Available: http://dir.yahoo.com/arts/artists/collaborative_projects/