Virtual 3D Environments: Implementations of 3D Environments for Virtual Tours and Online Communication

Scott Bessey

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Virtual 3D Environments: Implementations of 3D Environments for Virtual Tours and Online Communication.

By Scott Bessey
May 1, 2013
Thesis Proposal for the Master of Fine Arts Degree

Rochester Institute of Technology
College of Imaging Arts and Sciences
School of Design
Computer Graphics Design

Title: Virtual 3D Environments: Implementations of 3D Environments For Virtual Tours and Online Communication

Submitted by: Scott Bessey

Date: May 1, 2013

Thesis Committee Approval

Chief Advisor
Associate Professor Chris Jackson, Computer Graphics Design

Signature of Chief Advisor

Associate Advisor
Professor Marla Schwepppe, Computer Graphics Design

Signature of Associate Advisor

Associate Advisor
Assistant Professor Shaun Foster, Computer Graphics Design

Signature of Associate Advisor

Thesis Candidate
Scott Bessey

Signature of Thesis Candidate
Reproduction Granted (Choose One):

I, __________________________, hereby grant/deny permission to Rochester Institute of Technology to reproduce my thesis documentation in whole or part. Any reproduction will not be for commercial use or profit.

Signature of Author

Date

Inclusion in the RIT Digital Media Library Electronic Thesis and Dissertation (ETD) Archive:

I, __________________________, additionally grant to Rochester Institute of Technology Digital Media Library the non-exclusive license to archive and provide electronic access to my thesis in whole or in part in all forms of media in perpetuity. I understand that my work, in addition to its bibliographic record and abstract, will be available to the worldwide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles and books) all or part of this thesis. I am aware that Rochester Institute of Technology does not require registration of copyright for ETDs. I hereby certify that, if appropriate, I have obtained and attached written permission statements from owners of each third party copyrighted matter to be included in my thesis. I certify that the version I submit is the same as that approved by my committee.

Signature of Author

Date
Abstract
Virtual tours can be found over the web in many different scenarios: touring a campus, displaying consumer products, documenting a trip. These tours provide information using images, movies and sound to provide users with navigation in such a way that they user feels they are experiencing the information in real time. These tours are usually driven by mouse clicks, whether it is clicking on an image gallery, movie gallery or panning around a panoramic image. While virtual tours are interactive in those examples, there is a lack of interaction between the users viewing the tour.

Online communication has become a substitution for the tradition interaction between people due to its ease of accessibility. These range from customizable avatars and custom messages to real time movement within an environment. This gives the user the ability to make a virtual persona; a way to experience a realistic interaction online. I will try to create a virtual tour that expands on previous virtual tour concepts and provide users with a virtual representation of an environment with different forms of interaction and communication.

My thesis is an exploration in designing a completely web based interactive tour with the purpose of teaching the user about the Computer Graphics Design program at the Rochester Institute of Technology while providing the user with a virtual re-creation of the program’s computer lab in hopes of bringing a more interactive approach to virtual tours. Divided into 3 sections, this thesis provides the user the option to choose their avatar, chat with other users and explore the program’s lab to learn about the program.

Project website http://www.seb8262.wordpress.com
# Table of Contents

## Existing Sites
- Baldwin-Wallace 7
- Gtalk (now Google Talk) 7
- Yahoo! Messenger for the Web 7
- IMVU Messenger 8
- Habbo 8
- FlashComs 8

## Book Reviews
- Flash 3D: Animation, Interactivity, and Games 9
- Avatars at Work and Play: Collaboration and Interaction in Shared Virtual Environments 9

## Process

### Design
- Graphics 12
- Isometric Drawings 13
- Colors 14
- Avatars 16
- Other artwork 15
- Login 19
- Start Screen 21
- Virtual Chat 22
- Choose Avatar 23
- Virtual Tour 24
- Info Boxes 25
- Code 26
- Structure of Library 26

### Development of Code
- Virtual Tour 27
- Layering and Grouping 28
- Collision Detection 29
- Depth Swapping 31
- Updating Avatar Movement and Depth 34

## Debugging the Application

## Technical Issues and Limitations
- Communication Between Users and Applications 37
- Collision Detection 37
- Depth Swapping 37
- Speed of the program 38
- Technical Resources 38

## User Feedback
- Login 39
- Start Screen 39
- Condensed Chat 39

## Usability Testing

## Conclusion

## Bibliography

## Thesis Proposal
Part A
Thesis Project Research
Existing Sites

Baldwin-Wallace
After researching virtual tours on the internet, several tours for multiple mediums were found. One in particular that stood out was the Baldwin-Wallace virtual campus tour. While all the other ones were QTVRs and image galleries, this one provided more of an interactive approach and an approach I saw for my thesis. In this tour, the user could navigate around the actual campus map and visit each building on campus. By clicking on the buildings, a window appeared showing a description of the building and other information useful for the user to know.

Disadvantages:
• Requires installation of QuickTime
• Clicking on the buildings for navigation
• Static learning (windows with text)
• No real time interaction
• No avatar movement or personalization
• Poor graphic quality

Advantages:
• Windows that can be minimized or expanded
• Map with zoom capability
• Flash based application

Link  http://www2.bw.edu/vr/

Gtalk (now Google Talk)
When I found this tool, it was limited to beta. Google Talk is now integrated into your Gmail account. It is a plug-in download that allows for the use of your address book as a friends list where you can chat over the web via text, voice chat or web cam. Google also offers Google Talk software for your windows desktop that acts as an instant messenger client with the same functionality.

Disadvantages:
• Chat room and instant messenger functionality only
• No virtual room or avatar movement.

Link  http://www.google.com/talk/

Yahoo! Messenger for the Web
This chat application is similar to Google Talk. Yahoo! Messenger for the Web may be used within your Yahoo! Mail account to chat with Facebook and Windows Live friends without requiring any other installation. This application was also offered as an external install (windows, mac and mobile), but was discontinued in November 1, 2011.

Disadvantages:
• Chat room and instant messenger functionality only
• No virtual room or avatar movement.

Link  http://messenger.yahoo.com/web
**Current Sites**

**IMVU Messenger**
This application is full 3D, external chat room environment. This allows for the user to customize and save an avatar, decorate your own room and interact in other user’s rooms. Chat bubbles appear above the avatar so the entire chat room to read and some interactions between avatars happen on a click basis. I found this messenger’s environment closely represents the environment I attempt to create in this thesis.

Disadvantages:
- Download and load times for the rooms
- To move around the room, you have to click on yellow circles to move to occupy that spot in the room.
- Only real time interaction is the built in system (e.g. if you click on a spot on an ice ring, your avatar will skate in a pre-determined rotation).
- External program to install.

**Link**  http://www.imvu.com/

**Habbo**
Habbo is a completely web based environment with a Flash 2D chat system where 3D is in the form of isometric 2D art. Habbo has customizable avatars and rooms, mini games and online chat system built into their application. The user moves around by clicking on the ground tiles, sending the avatar walking to the spot in 3D space. This application has real time interactive features without hindering chat. You can easily add friends to your friend’s list and chat is in the form of private messages or public chat bubbles that the room can read. You can enter other friend’s rooms by “ringing” the doorbell.

Disadvantages:
- Although the most related application to my thesis, the application is still click heavy.
- There is no way to move within the 3D environment other than clicking on tiles.
- Chat bubbles can get cluttered and overtake the screen.

Advantages:
- Attempts realistic solutions for interactions (e.g. ringing the doorbell to be let into someone else’s room)
- Faster loading times due to the 2D isometric artwork
- Customization of the avatar and rooms
- Ability to complete tasks and gain

**Link**  Link: http://www.habbo.com/

**FlashComs**
This application is a Flash based chat room and instant messenger system. It has a dynamic whiteboard for users to write messages and draw pictures in real time. There is no avatar nor is there a 3D environment. Chat properties like font type, color, chat window skin, and other are customizable.

Disadvantages:
- No 3D environment
- No avatar
- Installation on a server is required.
- Seems to be for business teleconferencing.

**Link**  http://www.flashcoms.com/
**Book Reviews**

**Flash 3D: Animation, Interactivity, and Games**
This book provides the knowledge for creating 3D in Flash by means of 2D artwork. Topics are discussed such as the various types of 3D projects and how to achieve the imagery for a particular projection, given the appearance of 3D depth in a 2D Flash environment. This book discusses the creation of reusable templates to increase the performance of your project development.

Advantages
- Discusses how to create 2D isometric images
- Simulation of 2D camera using virtual reality concepts
- Discusses the trigonometry behind 3D concepts

![Figure 1](Image)
*Flash 3D book cover*

**Avatars at Work and Play: Collaboration and Interaction in Shared Virtual Environments**
This resource provides in depth articles about avatars. It solidifies my thesis and helps explain why and how the execution of my thesis will occur. These articles have a wide variety of range that will be useful in designing and planning out the interface. Shared Virtual Environments, or SVEs, are where all the interactions between users and their avatars happen. Avatars are exploited for different purposes by changing certain aspects about the avatar. This approach gives more personality and control over the avatar. If the user is of Asian descent, he could, in theory, create an Asian avatar (with the theory placed with my thesis). Focus on behavioral realism rather than representational realism becomes the issue, which will have major implications on the SVEs and my thesis.

Advantages:
- Teaches Conceptual approach to interaction
- Shared Virtual Environments (or SVEs)
- More personality and control over your avatar if these approaches are kept in mind.

![Figure 2](Image)
*Avatars at Work and Play book cover*
After researching thoroughly, my findings indicated the lack of an interactive, 3D environment in Flash as a virtual tour. There were chat room applications such as Habbo where a vector 3D environment exists with the intention of interaction and communication via avatars, but I found these lacked a teaching aspect. These applications are for entertainment and communication.

My thesis is to create an interactive, 3D environment for learning about a specific idea with the interaction of chat without hindering the learning experience. Virtual tours have been in existence for several years now, but they have not been as interactive as technology allows. Thus my thesis explores the virtual tour as a virtual space, where learning occurs in an interactive environment. The medium for these interactions are based around an avatar system, a communication system and a 3D environment for continued interaction and learning. The overall goal was to create this application for use by prospect students to find out more information about the program while providing a way for users to interact (both through interaction of the environment and communication of other users, who may current students, current staff or other prospect students).

Figure 3

Concept of theory for the application
How will the user be introduced to the tour? Should appropriate information be available outside of the interactive tour? What page will the user start on after log in? And how will these occur so none of these parts have a negative effect on the learning application but be driven by the specific parts?

To find these answers, I broke the application down into a chart, indicating the levels of the application and the different components (see figure 4). The chart is read from left to right, starting with the login. The user starts at the login, where they may create a user name and password. This information will be linked to their account, and to their avatar that they will customize. Once the user is registered, the main menu is loaded where the user can access their avatar, instantly chat or take the interactive tour to learn more about the program.
Design

Graphics
For the application to run with minimal loading times, the graphic choices are critical. The graphics needed to be small in size and easily updatable. At the time, 3D graphics were not chosen due to the complexity, both in concept and programming. 3D graphics can be more hardware intensive, could be seen as a distraction and would require a third party library such as Papervision3D. For these reasons, and by design choice, 3D graphics were not used.

The choice of 2D isometric vector graphics was introduced after reading “Flash3D: Animation, Interactive, and Games.” I needed a smooth, clean design to represent the overall feel of the application. By using vector graphics, my design was achieved with minimal load times. The file size of the application swf file is small and updating the graphics easy. Adobe Illustrator was used to create png files over jpegs for their small file size and transparency. The rest of the graphics were created using the tools in Flash.

After more researching about virtual environments and the different 3D perspectives, the decision was made to create isometric artwork based on Orthographic Projections. After viewing the website Habbo and testing the application IMVU Messenger, I decided to design the virtual environment from those two resources, taking the overall feel from Habbo and allowing clickable interactive objects as found in IMVU Messenger. I felt this type of drawing best fit the design I was attempting to achieve as it was easy to create and had depth.

There were two goals for the graphics of the application. First, the graphics were to be simple, clean and easily reusable. Reusing images allows for more customization as user avatars or an application skin. The other goal was to create a design that was graphically appealing; a design that may be taken off the computer screen and into print, both in application and theory.
Isometric Drawings

This type of drawing has all three axes drawn at a 30 degree angle, making all measurements along the axes exact. All of the isometric artwork created for this thesis was easily constructed using following the steps.

1. Using the rectangle tool in Flash, create a 60 by 60 pixel square. If the rectangle tool is not present, choose Window > Tools to open the tools window or click the R key.

2. Use the arrow selection tool to select the square. Select the Skew option and enter a vertical skew value of -30 degrees. This creates the right side of the object.

3. Created another square using the rectangle tool in Flash by repeating step 1. Select the square and enter a vertical skew of 30. This creates the left side of the object.

4. Created another square using the rectangle tool in Flash by repeating step 1. Select the square and enter a horizontal skew of 60 and a vertical skew of 30. This creates the top of the object. Depending on the direction of the top of the object, both skew values are either positive or negative numbers. In Flash, positive angles are measure clockwise.

To create the floor, the square (60, 30) created in the previous steps was duplicated and placed side by side until the desired shape was created. For other objects, such as the desks and computers, different pieces of the object were built using the outlines of the box as reference (see figure 6). With all the pieces were built and skewed properly, the pieces were placed together to form the object (see figure 6).

Figure 5

Isometric concept that was used to create artwork for the application

Figure 6

Artwork of a desk before the isometric transformation and the final product.
Design

Colors
The colors are directly influenced from the colors found in the Computer Graphics Design lab at the time (see figure 7). Light blue and grays were chosen from that palette with an orange-brown for the logo and menu. The other colors represent the colors used in the avatars.

Avatars
For the avatars, I had a friend created the graphics. First he drew started with the head, sketching several heads based on current students (see figure 8). Once approved, they were brought into the Flash timeline and colorized (see figure 9).
The next step was to create the rest of the heads facing the different directions. Front left was already created (see figure 9). The other directions would give the appearance of the head in 3 dimensions on the screen: front right, back left and back right (see figure 10). After the head, the body and legs of the avatar were sketched and colorized (see figure 11).

The avatar’s design was the defining factor on how the rest of the tour’s graphics were going to look. I created the rest of the graphics for the project, trying to keep the design similar to the avatars (see figure 12).
Design

Other artwork

Figure 13

Image of computer

Figure 14

Final artwork for computer
Design

Figure 15

Image of a computer and desk

Figure 16

Final artwork of a computer and desk
Design

Figure 17

*Image of the lab with computers*

Figure 18

*Final artwork of the lab with computers*
Design

Login
The user may log in to the application with their username and password (see figure 20). If they have not registered, a register button can be found on this page that will direct them to the register screen (see figure 21). The user will be required to log in to take the tour with their selected avatar or use the chat system.

If the user is attempting to acquiring information about this application, this screen has links to all the resources for the thesis as well as the thesis blog, FlashChat blog and a special thanks screen thanking all the individuals and groups that made this thesis possible is also found in the links section (see figure 20).

Figure 19
Early concept of the login screen

Figure 20
Final concept of the login screen
Design

Figure 21

*Final concept of the register screen*
Design

Start Screen
From the start screen, the user has the option to choose an avatar, take the tour or chat with other users (see figure 23). It seemed unnecessary to make the user log into the tour if they are using the application to just chat with other users, so an option to chat outside of the tour was added to the application.

Figure 22
Early concept of the start screen

Figure 23
Final concept of the start screen
Design

Virtual Chat
From this screen, the student is given an option to go to the Virtual Chat. Typical to the layout of most instant messenger system, this was designed to be easily recognizable to the typical user and limit any unnecessary directions (see figure 24). Users have the ability to send private messages by selecting a user from the Users list, typing a message in the bottom box and clicking Send. Clicking “Deselect” will deselect the user from the list and messages sent will be viewed by the entire lobby (see figure 25).

Figure 24
Users chatting in the virtual chat lobby

Figure 25
User sending a private message in the virtual chat lobby
Choose Avatar

Before a user may enter the tour, they must select an avatar (see figure 26). Each avatar is sorted by gender, providing several options. Once an avatar is selected, the user must click on “Save Changes” to confirm and save the selection (see figure 27). The user clicks on “Close” to return to the screen.

Choose avatar screen without the avatars

Choose avatar screen with the avatars
Design

Virtual Tour
If the user wants to take the tour, he/she may choose so by selected the Tour, which too had communication via a condensed version of the Virtual Chat. The main purpose of the tour is to provide the user with information about the program while providing an interactive environment to chat (see figure 30). Depending on what is currently on the screen, the bar at the top of the application provides navigation for the tour’s sections (see figure 28).

Figure 28
Image of the tour menu bar found at the top of the screen

Figure 29
Image 1 of the tour

Figure 30
Image 2 of the tour
Design

Info Boxes
The info boxes are the yellow boxes with the letter i across them (see figure 31). These are scattered throughout the tour and teach the user about the program. You are introduced to the first box as you first enter the tour, providing direction for use of the tour (see figure 32). The student clicks on the info boxes to display an interactive window where you may choose to either go in order of the pages, or skip through the pages by clicking the numbers at the bottom of the screen (see figure 33).

Figure 31
Image of info boxes

Image of an info box in the tour

Figure 32

Image of the interactive window that loads when an infobox is opened

Figure 33
Design

Code
This application relies heavily on code to manage interactions, tracking variable and states, chatting and customization. For this reason, the decision was made to have external classes. Actionscript 3 was the primary code behind this application with PHP code to handle server side code for saving user login information and avatars. Actionscript 3 is far more superior to its predecessor, containing new features as well as speed increases. Along with Actionscript 3, a Flash Media Server was needed to handle the server calls between chatting users. This required server script to be written using a native form of Actionscript 1.

Structure of Library
Library structure is very important not only for organization but for ease of coding. Instead of having a single class files with thousands of lines of code, I broke functionality out into several classes, naming these classes based on the code functionality and application section (see figure 34).

Figure 34
Image of the package explorer in Flash Builder
Development of Code

Virtual Tour
For my thesis to function as a virtual interactive tour, a library was built. Due to the scope of this thesis and what core components needed developed, several key classes needed to be written to manage the functionality. These classes include .as, .asc and .php files. To better understand how the movement of the avatar in the lab was going to function, I purchased an isometric role playing game tutorial from Cartoonsmart which helped in the development of code as well as some of the artwork. The tutorial was very helpful and explained in detail the logic behind the setting up of files as well as coding. I recommend this tutorial to anyone who may be interested in gaming in an isometric fashion.

This section required the most development. This takes advantage of all three code types, as there are many steps to achieve the dynamic, real time interaction. Not only does this section have a condensed form of the chat room system, but each user’s avatars occupy in the 3D lab environment, updating movement in real time as the users takes the virtual tour of the program. When one user moves their avatar on the screen, each of the other users screen is updated with the new position.

Figure 35
Image of the Virtual Tour
Layering and Grouping

The layering of the objects in the lab is critical for many reasons. For this, I used the structure from CartoonSmart Flash Tutorials in High Definition Video tutorial. Aside from the reason of the layout of graphics for the 3D environment representation, the layering and grouping of objects are how depths and collisions are managed. The floor (or background) layer is just that; the floor and any other objects that do are not directly involved in either the collisions or depths. The depth of this layer always stays 0 (always in the back). The only transformation the floor layer receives is movement when the avatar walks around the lab.

The foreground objects are the desks, doorways, avatars and other objects in the lab that have depth manipulations. The middle layer is for one purpose: collision detection. Whether it’s the user walking into a wall or walking between two desks, this layer is used to find which action occurred.

Figure 36

Concept of how the objects are layered and grouped within the tour
Development of Code

Collision Detection
Collision Detection refers to two or more objects hitting (or overlapping) each other. This happens when either part of the whole of each object occupy the same space on the page. Flash has collision detection in the form of two methods: hitTestObject() and hitTestPoint().

The hitTestObject() method is used to evaluate if the bounding box of the object is intersecting a bounding box of another object, while hitTestPoint() evaluate if a display object intersects a specific point. However, due to the nature of the isometric artwork, hitTestObject() did not work due to the detection of the bounding box. For the artwork, it would to be test shapes (non-bounding box) collisions. The hitTestPoint() detection is more true to what is required for the appropriate collision detection, having the option to test the shape of a display object. Unfortunately, there is no real way to recreate the collision detection desired on a point by point basis.

After researching online, I found the Corey O’Neil’s Collision Detection Kit, which offers more efficient and precise collision detection based upon pixel to pixel overlapping. Along with that site, I found the Lab: Autopsy of Skinner’s Collision Detection in AS3 website to be very helpful in developing with the kit, breaking down some of the functionality and adjust them for personal interaction. A color uint tell the SDK which pixels to ignore when collision detection occurs by using the excludeColor() method. This is how transparent (or non-colored) pixels are not detected from the artwork. This detection is resistant to any transformations (scaling, resizing, rotation, etc) and is relatively fast. Below is an example of how the SDK was used to detect whether an avatar struck a wall or desk.

```
_collisionList = new CollisionList(myAvatar.testAvatarHotSpot, this.floor.mc.wall_mc, this.floor.mc.hotSpots);
_collisionList.excludeColor(color);
```

**Figure 37**
Code that sets up the collision object’s list which can be found in the Tour.as

1. CollisionList objects are used when you have more than two objects to test for collisions.
2. When a user enters the tour, the collision list is loaded with the avatar’s hot spot, the lab walls and the floor hot spots (see figure 37).

```
var userCollisions:Array = _collisionList.checkCollisions();
```

**Figure 38**
Code that creates the array of overlapping objects

3. The checkCollisions() method returns an array with the objects that are overlapping (see figure 38). This occurs each time the KeyboardEvent.KEY_DOWN event is dispatched which moves the avatar around the lab.
Development of Code

4. If the collision array’s length is greater than 1, then the user’s avatar does not move and we set its position to the previous position (see figure 39). This does not move the avatar or the floor. Otherwise, the floor is moved to its correct position depending on the direction the avatar is facing.

```javascript
if(userCollisions.length) {
    myAvatar.inTransit = false;
    avatarX = avatarXOriginal;
    avatarY = avatarYOriginal;
} else {
    myAvatar.inTransit = true;
    myAvatar.pointX = avatarX;
    myAvatar.pointY = avatarY;
    scrollFloor();
}
myAvatar.avatar.moveAvatar();
```

Figure 39

Code that sets the position of the avatar and scrolls the floor (if applicable), which can be found in Tour.as.

Figure 40

Image of the avatar colliding with the blue tiles under the printer
Development of Code

Depth Swapping
All static objects are linked to other objects by groups. These groups dictate how the code decides which groups need to be placed in front or behind other groups and avatars. This is the first of two steps to correctly swap the depths to achieve the appropriate movement in the 3D environment. The second step for the depth swapping system is hot spots along the floor. These hot spots are linked to specific groups by means of the naming convention given to the hot spots.

1. On the creation of the hot spot on the floor, a CollisionGroup object is created using the user avatar hot spot and the newly created hot spot located on the floor. The object tells us when there is a collision between these two objects. To achieve the correct collision detection, we exclude a specific type of pixel. The CollisionGroup object reads a uint variable as a transparent pixel (e.g. var color:uint = 0x00000000). Any color may be passed for exclusion so long as it is a uint (see figure 41).

```plaintext
// should be transparent (Alpha = 0x00, empty and transparent)
public var color:uint = 0x00000000;
swapDetection = new CollisionGroup(userAvatar.testAvatarHotSpot, this);
swapDetection.excludeColor(color);
```

**Figure 41**

*Code that sets up the collision group with transparency which can be found in SwapSpots.as*

2. An event listener for the ENTER_FRAME event is added to continuously test for the collision.
3. The sendSwapName method calls the CollisionGroups.checkColisions() method. If any collision occurs, the points are returned to us in an array. The array is then checked for content. If the array is empty, no collisions between the avatar and the hot spot occurred. If the array is not empty, code is executed to swap the appropriate groups to achieve the correct depth (see figure 42).

```plaintext
this.addEventListener(Event.ENTER_FRAME, sendSwapName);
public function sendSwapName(e:Event) {
    var swapCollisions:Array = swapDetection.checkCollisions();
    if(swapCollisions.length) {
        if(is_hitTest == false) {
            is_hitTest = true;
            // sends the name of the swap spot walked on to the class to be deciphered
            floorClass.identifySwap(e.target.name, userAvatar.name);
            userAvatar.run_setDepth(e.target.name, userAvatar.name);
        }
        else {
            is_hitTest = false;
        }
    }
}
```

**Figure 42**

*Code to execute the swapping of objects within the tool which can be found in SwapSpots.as*
Development of Code

4. The Flash Media Server is told to save the new avatar depth, which is sent to the other connected user’s application.

When a hot spot is hit by a user’s avatar, the code below is executed to evaluate which depths should be swapped (see figure 43).

```actionscript
public function identifySwap(sName:String, user){
    var containerDepth:Number;
    var backGroupDepth:Number;
    var frontGroupDepth:Number;

    //splits all three pieces of information into an area
    parameters = sName.split("_", 3);

    //front group needs to be first
    fGroup = this.getChildByName(parameters[1]);
    bGroup = this.getChildByName(parameters[2]);
    _user = this.getChildByName(user);

    containerDepth = this.getChildIndex(_user); //avatar
    backGroupDepth= this.getChildIndex(bGroup); //back group
    frontGroupDepth = this.getChildIndex(fGroup); //front group

    // NEEDS TO BE:
    /////////////// Higher Num - Lower Num
    /////////////// Front < Container < Back

    if (containerDepth < backGroupDepth){
        this.removeChild(bGroup, this.getChildIndex(_user));
        containerDepth = this.removeChild(_user);
        backGroupDepth = this.removeChild(bGroup);
    }

    if(containerDepth > frontGroupDepth){
        this.removeChild(fGroup, this.removeChild(_user));
        containerDepth = this.removeChild(_user);
        frontGroupDepth = this.removeChild(fGroup);
    }

    if(frontGroupDepth < backGroupDepth){
        this.removeChild(fGroup, backGroupDepth);
        frontGroupDepth = this.removeChild(fGroup);
        backGroupDepth = this.removeChild(bGroup);
    }

    _user.setSendDepth(containerDepth, fGroup.name, frontGroupDepth);
}
```

Figure 43

Code that isolates tiles in front and behind the avatar to check, which can be found in SwapSpots.as
Development of Code

1. The identifySwap() method swaps the objects in the lab with the avatar to achieve the correct depth order.
2. Using the name of the hot spot that the collision detected, the front and back groups are found to be swapped with the avatar in a specific order. An example of a hotspot name would be front_group9, group8. This allows for the correct depth order in any situation when dealing with the local user’s avatar (see figure 44).
   a. If the container (or avatar) depth is less than the back group depth, then place the container in front of the back group.
   b. If the container (or avatar) depth is greater than the front group depth, then place the container behind the front group.
   c. If the front group depth is less than the back ground depth, then swap the two depths.
3. Finally, the depths are updated on the server and sent to the other users via the Flash Media Server.

```java
FrontGroup > Avatar > BackGroup

if (avatarDepth < backGroupDepth)
    // swap avatar and backGroup

if (avatarDepth > frontGroupDepth)
    // swap avatar and frontGroup

if (frontGroupDepth < backGroupDepth)
    // swap frontGroup and backGroup
```

*Figure 44*

Concept of depth swapping used within the application
Development of Code

Updating Avatar Movement and Depth

As a user moves within the lab, other users will need to see these changes. This is where the Flash media server plays its role in the tour. The Flash Media Server is where data is store to be accessed by all users logged into the tour. When one user updates a position of their avatar, for example, it sends a call to all the other users, informing the clients that there was a change made to the server data (see figure 45). Depending on what call was made, the appropriate response is executed. In this case, the user’s avatar state is to be updated.

1. When a student opens up the tour, a shared object is created with a specific name that is linked to the user by means of the NetConnectionClient Object that was created on log in.
2. Event listeners are set up to listen for specific calls from the server.
3. When the user moves their avatar, specific data (direction, position, depth) is saved in the shared object on the Flash Media Server. The avatar object stores all of its states and changes for updating the Flash Media Server.
4. The setProperty() method sends the information to the server for updating. As it is updated, an event is sent to the rest of the users connected to update that avatars position on their screen.
5. The event listener is set up at the time the avatar is created in the tour to listen for the server call to update avatar movements (see figure 46).

```
public function AvatarMC(userName:String, MainMC:NetConnection, DocClass, isUserAvatar:Boolean, avatarNumHolder:Array, direct) {
  ...
}
```

**Figure 45**

*Code that creates the avatar and all the appropriate server objects which can be found in AvatarMC.as*

```
  userXY_so.addEventListener(SyncEvent.SYNC, doUpdate);
```

**Figure 46**

*Code that creates the event listener for updating server objects which can be found in AvatarMC.as*
Development of Code

6. The doUpdate() method is executed, testing the name of the event from the setProperty() call above. The functionality ranges from updating x and y positions to setting depths of groups and its respected avatars (see figure 47).

```javascript
private function doUpdate (se:SyncEvent):void {
    for (var client: cl < se.changeList.length; cl++) {
        if (se.changeList[cl].code == "change") {
            switch (se.changeList[cl].name) {
                case "xpos":
                    this._x = userXY_so.data.xpos;
                    break;
                case "ypos":
                    this._y = userXY_so.data.ypos;
                    break;
                case "direct":
                    this._setDirection = userXY_so.data.direct; cr;
                    break;
                case "inTransit":
                    this._inTransit = userXY_so.data.inTransit;
                    break;
                case "unlockUser":
                    unlockUser(userXY_so.data.unlockUser);
                    break;
                case "sendAvatarDepth":
                    if (userXY_so.data.sendAvatarDepthUser != _docClass._user06j.userName) {
                        var userInputObjects = _docClass._user06j.sendAvatarDepthUser;
                        var frontGroup = userXY_so.data.sendAvatarDepths.frontGroup;
                        var frontGroupDepth = userXY_so.data.sendAvatarDepths.frontGroupDepth;
                        var theGroup = _docClass._user06j.sendAvatarDepthUser(frontGroup);
                        _docClass._user06j.sendAvatarDepth(userInputObjects, frontGroupDepth;
                        _docClass._user06j.sendAvatarDepth(userInputObjects, frontGroupDepth;
                    }
                    break;
                default:
            }
        }
    }
}
```

Figure 47

Code that updates server side objects with the appropriate data which can be found in AvatarMC.as
Development of Code

Debugging the Application
There are several ways to debug in Flash: toggle points, the debug menu. Flash builder’s code hinting and the build the library without having to publish the swf file is a really nice tool. For this project, the best debugging tool was tracing. But due to the server side of this project, I wanted to be able to see what data was being passed between users and the Flash Media server as well as between users. Server side functionality does not work in publish preview. Each time I wanted to debug involving anything about the server, I would have to publish the swf file and open it externally. But tracing only occurs in Flash.

To better server by debugging needs, I used the DropDownMenu class built for the friends list as a debug window that would act as the output window for all my traces. The window was versatile, may be expanded and minimized, and traced any string value I needed for debugging.

```
dropMenu = new DropDownMenu("Debug", 34.4.8, 38.2, "DOWN", 172.35, 242.0, "TextArea", _MC);
dropMenu.tabEnabled = false;
addChild(dropMenu);
```

Figure 48

*Code that creates the debug window*

This creates the debug window that is added to the application (see figure 48).

```
dropMenu.textBox.htmlText += "userRequest Called";
```

Figure 49

*Code that writes text into the debug window*

This line displays the text within the debug window (see figure 49). This is how the server code is debugged, writing the sent and caught data to the window.
Technical Issues and Limitations

Communication Between Users and Applications
The first technical issue I had to overcome was how to connect users together over the application to allow for updating avatar positions and sending chat messages in real time. The Flash Media Server was the perfect solution; however it is a more expensive than other servers. Coding for the Flash Media Server as well as PHP was a new frontier for me as I had never written in those languages proper to my thesis.

Soon after setting up the server and client code for updating avatar positions, I noticed delay between the time it took the user to send data and when the avatar position was being updated. The solution was removing unnecessary array populations, taking great care in when sorting of these arrays and nested for statements. This optimized my code to the best extent, but there was still a slight delay, which was anticipated.

Collision Detection
Hit tests are a common occurrence in flash, whether it is when a mouse rolls over a button or an object is dragged and dropped over another in a flash game. Collision detection was vital for the application, but the common solutions for collisions did not meet the criteria for this application. It was limited in what the hit test tested and when the hit test dispatched the event telling flash it had collided.

As stated earlier, the two built in functions in flash are hitTestObject() and hitTestPoint(), but these functions did not achieve the proper collision detection. The application required a Collision detection that was fast, efficient, and capable of detecting when objects collided (not their invisible bounding boxes). This is why I chose Corey O'Neil’s Collision Detection Kit. It offered a way for me to ignore specific colors (thus specific pixel types) from detecting a collision, thus allowing the application to correctly detect the collision of two irregular shapes.

Once I figured out how the kit worked, it was a matter of updating the user’s avatar appropriately. At first, the user avatar would get stuck on the colliding object. This occurred because the users avatar’s hot spot would overlap the wall’s hotspot, executing code that disabled avatar movement. The collision test was always true where once the wall threshold was crossed, it was always crossed. This was correct functionality if the avatar continued to move in the wall’s directions, but this did not allow the avatar to move in the other 3 directions. The solution to the issue was to have the application test for the collision before the collision occurred, so the application to stop the movement of the avatar before it actually collided. After the first attempt at the solution resulted in the avatar bouncing off the wall, I found the correct solution by setting the x and y values of the avatar to the original values before the move.

Depth Swapping
Depth swapping in Actionscript 3 can be tricky because depth has to be a valid number between 0 and 1 minus the number of children in that particular display list. In the early stages of this thesis, the avatar swap depth with the desk associated with the hidden hot spots found throughout the tour. This would work on the first pass, but as the avatar continued to walk through the tour, the desks would not swap properly. This was noticeable with any desks behind the avatar, sometimes cutting off the head of the avatar.

To solve this, the desk behind the avatar would need updating. I decided to create a set of hot spots placed in between desks in the tour. The hot spot’s name tells the application what desk is in front and what desk is behind. When the avatar walks between two desks, the avatar is placed behind the front desk and in front of the back desk.

After testing further, I noticed that desk’s depths were still not correct throughout the tour. This occurred because as the depth of one desk was updated, desks with higher depths had their depths incremented by one, and that new depth was usually incorrect. To fix this, all desks needed to be stored and updated in an array after each depth swap, and have their depth manually set to the array position after each depth swap. After the two desk’s depths and the avatar’s depth are set, the array of tables would be sorted through and their depth’s updated to match the new order. This made sure that all table depths would be correct to the avatar.
Technical Issues and Limitations

Speed of the program
Arrays can be resource heavy when constantly sorted and positions manipulated. This can be seen as a limitation, but it depends on how arrays are implemented and maintained. An alternative to arrays are vectors. Vectors, approximately 60% faster than arrays, can greatly increase the performance of your code. Vectors are similar to arrays in that they are unsorted lists. The only limitation to vectors is that they can only hold a single variable type, which has to be declared upon instantiation. This is vectors surpass arrays in speed. One solution to get around the single variable restriction is by using Vector<Object>, where objects are able to store multiple types of data. Unfortunately, vectors were not used in this application due to the collision detection kit only supporting arrays. Achieving this would either take an update of the collision detection kit or manually changing the collision kit code to save in vectors instead of arrays.

Technical Resources
Resources can be hard to come by at times for many reasons. A new code library has not matured yet, leaving people to figure out their own solutions to their project-specific questions. Or perhaps the tutorial or solution for the problem someone else has fixed does not make sense to you, maybe by their code or the lack of an explanation for a solution.

During my thesis, there was not much on resources when it came to Corey O’Neil’s Collision Detection Kit, Flash Media Server, and Vectors. I found some books, tutorials and forums that were very useful, but it seemed they only explored a small portion of theories and concept that I needed for my thesis. Tutorials and other code examples would get me on the right track, but it was up to me to continue and mold the concept of the code to my needs.
User Feedback

This thesis went through many phases, many of which were technical phases behind the scenes. User feedback occurred through all stages either by colleagues, friends or thesis committee members. User feedback was mostly related to the design of the application: the navigation, login screen and start screen.

Login
I had several responses about the original design of the login screen (see figure 20). Some users disliked the empty (or negative) space around the login area while other users through the screen lacked some substance. It functioned as intended but could be used for something more. I decided to add a links section that would provide a way to access outside information related to my thesis. This would be my blogs for my thesis as well as a special thanks section as tribute to all the individuals that shaped this application (see figure 21). The registration page was a replica of the login screen in both the early design and final design, thus it had the same design changes (see figure 21).

Start Screen
The start screen loads once the user is logged into the application. This is the main navigation for the application and provides buttons for each section with a small description of what the section offers. The early design, being a place holder for the final concept, focused on emphasizing the sections as 3 large buttons, where the tall button had enough space was provided to show the user’s selected avatar as part of the button (see figure 23). As a design choice and user feedback, the avatar was removed as it was deemed not necessary. This opened up the possibility of shrinking the buttons and having more space for secondary information (see figure 24).

Condensed Chat
The original concept for the condensed version of the virtual chat resembled an instant messenger client (see figure 59). I wanted the design to be recognizable to most users for a minimal learning curve but at the same time enhance the user experience while adding a new interactive interface, both in design and functionality. On the left side of the message box, you would find current messages, which the user would be able to swap between the messages. But it leaves to question how the user will start new messages.

After receiving feedback from other users, there were several issues with the design. Other than the issue of starting a new message, a comprehensive user list of every user connected to the chat was missing. I decided to incorporate a chatroom design as it is still recognizable and has all necessary functionality (see figure 60). This design, influenced by the virtual chat section, fixes the missing comprehensive list, allowing all messages between users to be seen in one screen.
In an attempt to better improve this project in the future, I created an evaluation form for users to fill out after viewing my thesis. Here are a couple of forms that users filled out.

<table>
<thead>
<tr>
<th>Thesis Title</th>
<th>Virtual 3D Environments: Implementations of 3D Environments For Virtual Tours and Online Communication.</th>
<th>Scott Bexley MFA, Computer Graphics Design</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Evaluator Background</th>
<th>Age</th>
<th>Occupation</th>
<th>Gender</th>
</tr>
</thead>
</table>

**Directions**

Please best answer the following questions below.

Your responses are anonymous and will not be shared with anyone.

If you do not understand a question, please leave it blank.

Feel free to write comments below each question.

### Project Evaluation

1. The project is easy to navigate.

2. The directions for the project are easy to follow.

3. I like the colors scheme and artwork chosen for the project.

4. The info boxes found within the lab are effective in displaying the information about the program.

5. I feel I am exploring an actual lab while taking the tour.

6. Walking around the lab is simple, easy to use and enjoyable.

7. Chatting is simple, easy to use and enjoyable.

8. Chatting does not distract me from the tour.

9. The intent of the project is clear.

**Figure 50**

*Blank evaluation form page 1*
The tour information is in a logical order (e.g., the order in which the info boxes are found throughout the lab upon exploration make sense with the information they provide).

This project answers my questions or directs me to the answers I have about this program effectively.

I would choose a school or program only having taken a virtual tour.

I would use a virtual tour application such as this project in selecting a school or program.

What did you like / dislike about this project?

____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________

Any other comments? Anything you would like to see added to this project?

____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________
____________________________________________________________________________________

Figure 51

Blank evaluation form page 2
### Usability Testing

<table>
<thead>
<tr>
<th>Thesis Title</th>
<th>Scott Branyan</th>
<th>M.A. Computer Graphics Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 Student 1</td>
<td>Male</td>
<td>M.A. Computer Graphics Design</td>
</tr>
</tbody>
</table>

#### Project Evaluation

- The project is easy to navigate: Agree, Agree, Agree, Agree, Agree.
- The directions for the project are easy to follow: Agree, Agree, Agree, Agree, Agree.
- I like the colors scheme and visual choices for the project: Agree, Agree, Agree, Agree, Agree.
- The information found within the tab can effectively display the information about the program: Agree, Agree, Agree, Agree, Agree.
- I felt I was exploring an actual lab while taking the test: Agree, Agree, Agree, Agree, Agree.
- Walking around the lab is simple, easy to control, and enjoyable: Agree, Agree, Agree, Agree, Agree.
- Cheating is simple, easy to use and enjoyable: Agree, Agree, Agree, Agree, Agree.
- Cheating does not distract me from the test: Agree, Agree, Agree, Agree, Agree.
- The intent of the project is clear: Agree, Agree, Agree, Agree, Agree.

#### Project Evaluation

- The project is easy to navigate: Agree, Agree, Agree, Agree, Agree.
- The directions for the project are easy to follow: Agree, Agree, Agree, Agree, Agree.
- I like the colors scheme and visual choices for the project: Agree, Agree, Agree, Agree, Agree.
- The information found within the tab can effectively display the information about the program: Agree, Agree, Agree, Agree, Agree.
- I felt I was exploring an actual lab while taking the test: Agree, Agree, Agree, Agree, Agree.
- Walking around the lab is simple, easy to control, and enjoyable: Agree, Agree, Agree, Agree, Agree.
- Cheating is simple, easy to use and enjoyable: Agree, Agree, Agree, Agree, Agree.
- Cheating does not distract me from the test: Agree, Agree, Agree, Agree, Agree.
- The intent of the project is clear: Agree, Agree, Agree, Agree, Agree.

---

**Figure 52**

**Page 1 of student 1 evaluation form**

**Figure 53**

**Page 2 of student 1 evaluation form**

---

**Figure 54**

**Page 1 of evaluation for student 2**

**Figure 55**

**Page 2 of evaluation for student 2**
Usability Testing

Figure 56
Page 1 of evaluation for student 3

Figure 57
Page 2 of evaluation for student 3
Conclusion

Can Flash build a 3D environment with acceptable feedback and interaction implemented to an online chat room application while not hindering online chat? The answer is yes. I have developed a virtual tour of the Computer Graphics Design program that contains the following: a Flash 3D environment, a chat system, and an interactive learning environment that with the chat system, allowing interaction between users and their environment. This visual representation of the Computer Graphics Design program provides an interactive, instructional and online approach where potential students gain an understanding of where they will be spending their time and money if they decided to join our program.

This application strays away from the QuickTime Virtual Reality and panoramic tours, and provides a new approach to virtual tours and how these tours present information. Fast, interactive and completely web based, this application is through a browser without the need to install an executable file. Developed in Actionscript 3, Flash, and php, the application is compatible across all browsers which allows for the same user experience.

How can this application be improved? One improvement is to change how the lab and its floor are created. Currently, the floor and all objects are placed manually within the Flash movie clip and the entire movie clip is moved to give the appearance of movement in the room. While researching, I found an interesting solution to this improvement in the development of tile generators. A tile generator dynamically creates environments using an external file to tell the application what types of tiles are placed at which location in space. The typical tile generator uses a system of 1s and 0s and stored in an xml or text file. The Os usually represent a tile that can be walked on and the 1s represent a tile that cannot be walked across. This system can be added upon with more numbers (or ever letters) to represent different artwork (e.g. the number 3 or letter T could tell the generator to place a tile populated by a tree). This approach improves performance, creating a more efficient than the current solution and may perfect the swap technique that currently is not perfect. This keeps the published swf file smaller and updating tiles more efficient. Changing a value in an external xml or text file will automatically update the generator when loaded.

This application was developed for Flash for the ease of accessibility over the web. This application could be published for mobile and tablet devices. Motion sensors and touch technology add to the possibility of new navigate and how the user receives their information. This would involve a re-design of the existing application and its navigation. This application would also make an interesting Facebook application, providing the user with integration to their Facebook accounts and ease of accessibility.

From the usability feedback forms, there were several comments about increasing the customization of the avatars. One approach could be allowing the user to change attributes like hair, shirt or skin color. This would add a more dynamic and personal feel to the application. Another comment I received was involved the moving of the avatar to appropriate tiles. The user stated that he was trying to click on a tile to move the avatar, and was asking why this was not implemented. This would leave all non-chatting actions to mouse clicks, which is not a bad idea. The current solution to avatar movement inadvertently assumes the user has some video game knowledge and the use of the W, S, A, and D keys to move the avatar in the respective directions.
Bibliography

Books


Internet Resources


Title: Virtual 3D Environments: Implementations of 3D Environments For Virtual Tours and Online Communication

By Scott Edward Bessey

Problem Statement

My thesis is an exploration in developing 3D environments in Flash for the purpose of promoting The Computer Graphics Design program of the Rochester Institute of Technology. Completely accessible through the computer by means Internet and Flash Player, this virtual tour will be built through Flash using vector, 2D elements while managing the elements in 3D space. This gives the ability for movement throughout the 3D environment, chatting with the staff in the school of design, avatar creation, and accessibility of exploring the rooms related to the CGD program.

After doing research, I have not found much on 3D environments in Flash, only “Flash 3D: Animation, Interactivity, and Games,” by Jim Ver Hague and Chris Jackson. Other research on the implementations of online chatting was found to be more pertinent. Yahoo and Google have already created Flash online chat systems. Both are still in beta as of today. But that still leaves the ability for a 3D chat room. Can Flash build a 3D environment with acceptable feedback and interaction implemented to an online chat room application while not hindering online chat?

Background

With the Internet set in place, the World Wide Web was introduced in the early 1990s. After that, fast streaming “web pages” and the Internet took off. Now, data of many kinds may be found from movies to news. But something was created at the beginning of this. Something created by Time Warner called AOL, one of the first really popular Internet service providers (ISP). What made it stand out from the other starter ISPs was a concept know as “instant messaging” and with this concept also came “chat rooms.” Chat rooms are places where you can instant message many people at once for everyone in that particular chat room to see. They are these little box windows on the computer monitor where you interact from people all across the globe.

Today, instant messaging is still popular as when it was introduced, but the interaction has stayed the same. As for chat rooms, they have died down with AOL, and stepped aside for Roadrunner and new high speed Internet. Being 5 times faster, Roadrunner offers a simpler, new way of connecting where you are instantly connected on broadband, bringing new possibilities of the Internet. Downloading programs, watching movies and streaming brought on a whole new market with the ability to bring more dynamic and conceptual content to users. With these bigger, faster and stronger programs, the interaction between individuals over the Internet may be enhanced for a more realistic, simulative environment, creating a new and more powerful approach to portraying information.

After searching the Internet for virtual tour, quite a number of virtual tours for multiple mediums were found. One in particular that stood out was the Baldwin-Wallace virtual campus tour. This one provided more of an interactive approach and something I saw my thesis project could become. In this tour, the user could navigate around the actual campus map and visit each building on campus. By clicking on the buildings, a window appeared showing a description of the building and other information useful for the user to know. It is not what this thesis will become, but it is a step in the right direction. The other examples of virtual tours I found during my research were QTVR, or QuickTime Virtual Reality, tours that were controlled through QuickTime. QTVR allows for the creation and viewing of panoramas, using images taken at multiple viewing angles. These seamed images allow you to pan and zoom their way around the space they are providing. There is software out there to help you create these simple, virtual QTVR galleries.
My thesis surpasses these applications by creating a new process behind depicting real environments by means of vector, 3D representations. Why use vector? Using vector imaging will produce fast, 3D rendering. The user will navigate and interact with the environment and other avatars in order to gain knowledge about the program. With the addition of non-playable characters (NPCs), the user can have access to the information all the time. The professors and department heads do not need to be logged in for you to get the knowledge needed. They avatars will always be there to answer all of your questions from set responses. Eventually, a form of online chat will be created where the users may chat amongst themselves and other faculty logged in. The process of getting to the chatting capabilities will change however, taking a bit longer to reach the chatting screen. This project has potential for many upgrades, from online conferences and streaming video presentations on the web. All the functionality of my thesis above makes Flash the best great medium for multi-platform capability.

**Scope**

The creation of 3D elements in 2D space is not a new concept, nor is virtual tours. The implementation of a 3D environment in Flash remains to be researched while Instant Messaging and online chat are not. As for virtual tours, I have seen QuickTime VTR tours out on the Internet, but only for hotel advertising and not for institutions. I know from personal experience that it is difficult to visit prospect schools when there are so many choices spread out across the globe. My thesis will bring new means of visiting the CGD program, allowing users to interact with not only the environment, but also with non-playable characters (NPCs), or computer driven avatars, who will answer any questions about the program. Eventually, other users will be able to log in and chat amongst other users. This experience will be more than just sitting at the computer and conversing with a school program director through email or over the phone. The design of this 3D tour addresses two design problems: ordinary, static interaction and relating information to people who are not well accessible. The designs of the 3D environment and the code behind the application have relationships with computer graphics.

**Literature Survey**

“Flash 3D” by Jim Ver Hague and Chris Jackson provides the knowledge required for 3D in Flash. Some of the topics discussed are the various types of 3D projections and how to achieve the imagery. Creating illusions of depth, tweening, interactive movements and the trigonometry functions are required for the creation of 3D environments. The reading also talks about how to simulate a 3D camera while using virtual reality concepts. After reading the beginning of this book, Orthographic Projection provides the best solution, and the drawing type set for this projection type is Paraline drawings. These drawings provide a number of different types, ranging from Isometric, to different versions of Dimetric and Oblique. What is important out of those types is the one I choose, which would be Dimetric, due to these types being a little more realistic because of foreshortening, or the illusion of direction causing an extension in space. This book will be the driving force towards the creation of my thesis and the simulation of a 3D environment.

“Avatars at Work and Play: Collaboration and Interaction in Shared Virtual Environments” provide in depth articles about avatars solidified my thesis and helps explain why and how the execution of my thesis will occur. These articles have a wide variety of range that will be useful in designing and planning out the interface. Shared Virtual Environments, or SVEs, are where all the interactions between users and their avatars happen. Avatars are exploited for different purposes by changing certain aspects about the avatar. This approach gives more personality and control over the avatar. If the user is of Asian decent, he could, in theory, create an Asian avatar (with the theory placed with my thesis). Focus on behavioral realism rather than representational realism becomes the issue, which will have major implications on the SVEs and my thesis.

These collections of articles become good resources on how people or groups of people interact together through computers, networks, and by avatars by using these. This will help in figuring out what would be the best way to overcome the limitations of nonphysical interaction through networks and by means of interaction through avatars in a SVE, which does not seem to be implemented in the older chat room and instant messaging systems.
Methodology

The implementation of this virtual 3D environment involves interaction at its most basic form. The environment is a nonphysical entity. The goal of this project is for users to be in the same nonphysical space and to communicate. An instant message is the same theory; only it allows the user to interact one on one to many people. Virtual tours are a little different when it boils down to interaction. These are more static images in space, not leaving too much to the imagination. QTVRs are real nice and a new approach, but still too static. To take the notion of group interaction and applying it to a nonphysical approach to a virtual tour is the problem my thesis is going to solve.

To achieve this, a Flash 3D environment built into a virtual tour and chat system will be created. With this system, we are tackling my thesis problem by pushing ahead with new means of displaying information, Internet interaction through chat, and 2D/3D elements through the web adding more realistic approaches to interaction. The other part of my thesis involves tackling the 3D aspect of flash to create the environment and coding movement of avatars on top of compiling the content needed to achieve the second purpose of my thesis.

The following will be used/created to implement my thesis:

• Flash application for the web
• Actionscript 3.0
• Controlled Test of this virtual environment
• A survey with questions related to the experience of my thesis.

Some questions may be:

o Relevance of this application?
    o Would they like/use a virtual touring application for visitation of a school that they would not be capable of physically visit?
    o Would it enhance your communication over the Internet?
    o What comes to mind when you hear this project?
    o Does it answer all the questions you may have?
    o Can any questions be added to make the content more reliable?

Limitations

The limitations I would encounter would be hardware. To create 3D effects in Flash involves processes and the more processes a computer has to calculate, the slower it happens and the more hard is it on the computer itself, eventually crashing. If the environment and the processes become too processor heavy, it will slow the computer down significantly, dampening my success rate on the program, but not the concept.

Storage and hosting for this virtual tour and/or chatting environment is a limitation needing to be addressed. If the user enters the virtual environment, where do I store the virtual environment? Would I be running processes on the users computer or on a server? It does not make sense to store the information needed to run the room on individual computers. The room should be localized somewhere that has Internet access 24/7 so that everyone who has the curiosity may enter the environment. Also, I do not know how to go about storing the information needed to run the virtual tour or the content for the whole entire program. This will require further research.

Implications of the Research

As stated above, I have found out that Google and Yahoo both have instant message services available through flash while AOL has AIM express, which seems to be loaded from the website, but I do not believe its flash based. As for the limitations, hardware would be the one significant limitation. Another would be the familiarity with the new Actionscript 3.0. I have only recently starting learning Actionscript 3, coding in Actionscript 2.0 before. The differences between the two languages are drastic and will take a good portion of the thesis to handle.

Peer Review

The plan for this project is not only to create something never seen before, but also to create a concept that may be marketable out of college. I would not only submit to Siggraph this coming summer, but also submit my thesis to Google, AOL, and Yahoo, to see what they have to say. My thesis uses concepts that would be beneficial to their cause (chat room, online communication through Flash). It only seems fair to see what they have to say about it. Also, other institutions may have good feedback and thoughts on if it portrays they right information and has the right feel to be affective. Future projects with other schools are possible.
**Target Audience**
- Male / Female
- 20-45, or whoever is interested or has knowledge in this technology
- Ability to open up the website and follow the directions
- Prospect students looking to attend the Computer Graphics Design program

**2 Typical Users**
- 21-year-old female high school student from the US who is interested in attended RIT’s Design program
- 33-year-old international student from India who does not have the time or funds to check out RIT

**Software and Hardware**
- Mac or PC equivalent to run normal Instant Messengers/Web Site activities
- Web browser
- Server Space
- Macromedia Flash Player with ActiveX object capabilities
- Actionscript 3.0 and PHP

**Timeline**
See Timeline file