

Creating a Virtual Performance

Jeffrey Bolkhovsky, Dean Gianotti, James Tardanico

Advisor: Joseph Farbrook

Abstract

With the advent of shared virtual spaces, it has become possible to produce virtual live performances, similar to a play. When creating a virtual performance there is a process that one must go through step by step to achieve an end result. The first step in creating a performance is to choose a script and then find a suitable virtual space that gives a sufficient amount of resources to adequately perform it. Next, assets such as sets, props, and costumes must be constructed or acquired. Once all the assets are secured, to finalize a performance, the actors must be coordinated and properly equipped to interact. This document is intended to serve as a guide for all who endeavor to create such a performance.

Introduction

It is only recently that virtual space has evolved to be capable of things that until now we have only seen in the physical world. One of these big jumps forward was the ability to perform before a live audience in real time. Before now this had mainly been done using many prerecorded and scripted events that removed the live feel of a play performance. Interested in just how far this facet of virtual space could go we decided to test the waters by designing and executing a live virtual performance using as few prerecorded elements as possible.

This paper covers our adventure in virtual performing and is intended to help blaze a trail for you to follow. We hope that understanding what we did, problems we had, and our solutions will greatly ease the trouble you face. While things written here may not be current and do pertain mainly to only one medium, many of the trials we faced are universal to virtual space. We encourage you to use the things outlined in this paper as mere guidelines and for you to try new things so this form of media can evolve and grow into the tool we believe it can be.

Choosing a Script:

Deciding on a script for your performance is a decision you should begin to consider as early as possible. Once you have settled on a script you can begin to move ahead with bringing your performance to life. Remember you want as much time as possible to obtain assets, build sets, and rehearse. When it comes to selecting a script you essentially have three options available to you. You can write your own script, you can use someone else's script, or you can do a combination of both.

Using Someone Else's Script:

If you decide to use someone else's script then it is very important that you make sure you have the legal rights to use it. If your performance is non-profit or for a school project then you may have an easier time obtaining permission to use the script. There are many amateur authors who are more than happy to let you use their work for a non-profit performance. There are also many scripts in the public domain which you can take advantage of.

Writing Your Own Script:

Writing your own script provides its own unique benefits, but it also presents some challenge. Writing your own script gives you the option to tailor your performance to your audience and your virtual space (more on this later). Writing your own script lets you play to the strengths of your group however, without an experienced writer you run the risk of sacrificing quality.

Combination of Both:

So long as you obtain permission from the original author, you may find it useful to use a combination of their script and your own writing. Adapting an existing script can be advantageous to the success of your performance. Writing a good play is hard. Even if you have a lot of writing background it can be very difficult for someone who is not trained in the nuances of writing for the stage. By using a combination of your own writing and an existing script you can ensure that your play includes the scenes that you want, without compromising the overall quality of the performance. Adapting an existing script can also save a lot of time in the pre-production phase of the performance, affording you more time to devote to rehearsing and obtaining assets.

Type of Script:

No matter which method you use for procuring a script, it is important that you choose a script which lends itself to a virtual performance. Remember, creating and performing a virtual performance is very different from a live performance in the real world. For example in a live real-world performance facial expression and dialogue delivery are extremely important. Because actions are limited, dialogue is the most prominent and important way of conveying a story in a traditional real-world performance. In a virtual performance, actions are not limited by reality. Characters can fly, shoot lasers, cast spells, teleport, etc. At the same time, facial expressions and body language are limited by the software that you are using. In most cases broad, bold actions can convey the story much better in a virtual performance than body language and facial expressions. Make sure when you choose your script that you are able to perform every scene effectively within the virtual space you have chosen. A subtle emotional scene may be very powerful in a real-world performance, but it can easily lose its effect when the dialogue is delivered by a stiff emotionless avatar. Likewise, an exciting space battle is nearly impossible to perform on the stage, but in virtual space it can be extremely effective.

Choosing a Virtual Space:

Choosing which virtual space you will use to create your performance is one of the most important decisions you will make. The type of space you use will determine the shape and feel of the entire performance. There are many different virtual spaces to choose from, but some of them lend themselves particularly well to the success of a *live* performance. Keep in mind that different virtual spaces have different moods, restrictions, expectations, and assets. Before settling on what space you will use you must first ascertain what qualities you need your space to possess. Many of these requirements are determined by the script you have chosen. For example, if you intend to perform a drama, then you would most likely want a virtual space that allows avatars to easily convey a wide range of emotions. If your performance focuses on action and special effects than the ability for avatars to express emotion might be less important for

you than the ability to create special effects and handle media. Carefully research all of the options available to you and find out what each individual space has to offer. Once you have narrowed it down, decide on the space that best suits your script.

Here are some qualities you might want to keep in mind while researching potential candidates.

- Ability for avatars to clearly convey a variety of emotions.
- How does the virtual space handle outside media? Can you play video/audio tracks through the virtual space?
- Community. Does the space have a strong vibrant community that you can tap for help/resources as well as provide an audience for your performance? Will the community be interested in seeing the type of performance that you intend to create?
- Control. How much control will you have over the environment? Will you be able to restrict access to the performance to prevent heckling/accidental distractions? Will you be able to prevent the audience from interfering? Do you have access to private space where you can build sets and practice the performance?
- Familiarity. Are you familiar with the space and how to access information and obtain the assets you need? Are you technically proficient with the space (scripting, building set pieces and props etc.)?

Some examples of suitable virtual spaces include Second Life, World of Warcraft, City of Heroes, and other engines. These are not by any means the only options available to you. Almost any virtual space where you can maintain control over characters, sets, and props, and then deliver that performance to a target audience is an acceptable virtual space.

Ability to Convey Emotion:

Different types of performances require different levels of emotional commitment from the audience. It would be very hard to perform a tragedy if all of the actor's faces are constantly locked in an ambiguous semi-smile. Make sure that the space you choose has the ability to convey the level of emotion, be it from facial expressions or body language, which your performance requires in order to be successful. Many virtual spaces have built in gestures/animations. These animations vary greatly in quality and clarity. Keep in mind that the audience's distance from the virtual performers will be a factor in evaluating their emotional state. It may sometimes be beneficial to make use of body language over facial expressions

because it is much easier for the audience to read the performers actions. Some virtual spaces such as Second Life allow you to import custom animations, which can be created with various third-party software tools (see the “additional resources” section for more info).

Handling Media:

If you intend to utilize outside media (Music, Youtube videos, sound effects etc) in your performance then it is important that you choose a virtual space that will allow you to display this media to the audience. Keep in mind that your audience will not all be watching from the same place. They will come from a variety of places, with a variety of different Internet connections and a variety of different computer hardware. It is imperative that the audience is presented with a cohesive experience. You must ensure that any outside media is seen/heard by everyone at roughly the same time. Make sure your virtual space meets your media requirements in this regard.

Community:

A virtual space that has a large thriving community will almost always be more useful than one that does not. The virtual community is going to become your audience and so it is important that you consider their desires and expectations before choosing a space. Does the Eve online community want/expect to see a romance set in Victorian England? Probably not, but they might very well be interested in seeing a wacky comedy set in space. The community can also make your job a lot easier if you know where to look. Need a ray gun for scene three? Chances are that someone in the community already made a ray gun and would be happy to let you use it for your performance. Use the community to your advantage and your experience will be more enjoyable and your performance well received.

Control:

Control is a factor that is easily overlooked, but it is supremely important. Many virtual spaces *do not* offer significant amount of control over the environment. Having control over the environment gives you many more options and offers increased creative control. You need it to start raining on command? No problem, you can do that. Need to prevent the audience from wandering off backstage or from harassing your performers? No problem, you can prevent them from leaving their seats. Of course not all of these applications are necessary, but it should give you an idea of how important control really is. It would be unfortunate if a band of merry adventurers on a quest from the NPC in town wandered through your performance halfway through act one.

Familiarity:

Familiarity is probably the least important factor to consider, but nonetheless it should be considered. Most virtual spaces are designed with a shallow learning curve. It should be fairly easy for you to learn how to build, create, and script all the things that you need, but of course

having some familiarity would save you some time. The importance of time management will be discussed later, but needless to say, anything that saves you time and resources is beneficial to the success of the performance.

Remember, not all virtual spaces will meet all your criteria. Sometimes you may have to compromise in order to find a space that works for you. As long as you keep in mind what is *most* important to your performance you should be able to find a suitable space that meets most of your requirements.

The Next Step

Once you have your script and have decided which virtual space you plan to use its time to start putting together and building your performance. This paper will go into heavy detail about using Second Life as a virtual space and why it has the versatility to handle most types of performances and how to go about actually piecing together your performance. We chose Second Life because it offered the following advantages:

- Ability to create custom objects and scripts with functionality customized to our performance.
- A community with a marketplace that contains items that work for our performance in order to save time.
- A private area to create the stages, with full control over the area including access restrictions.
- Ability to place media onto an object and play it for an audience.
- Ability to have characters use custom gestures and to display a wide range of body language.

The next few sections will show you how to put your play together. The first place to start is stage design.

Final Presentation Format

Stage Design:

When looking at how to appropriately create stages for optimal audience view, many types of stage designs were tested and tried before one was finally chosen. The chosen design consisted of a large floor with a dome, shown below in figure 0.

Useful for creating a stage, are extremely large building blocks (referred to in Second Life as Mega Prims). Mega Prims allow you to make very large objects without having to stitch together smaller primitives. Mega Prims are not usually standard features for virtual space editors so you will have to get them from a third party such as Xstreet (more on this later).

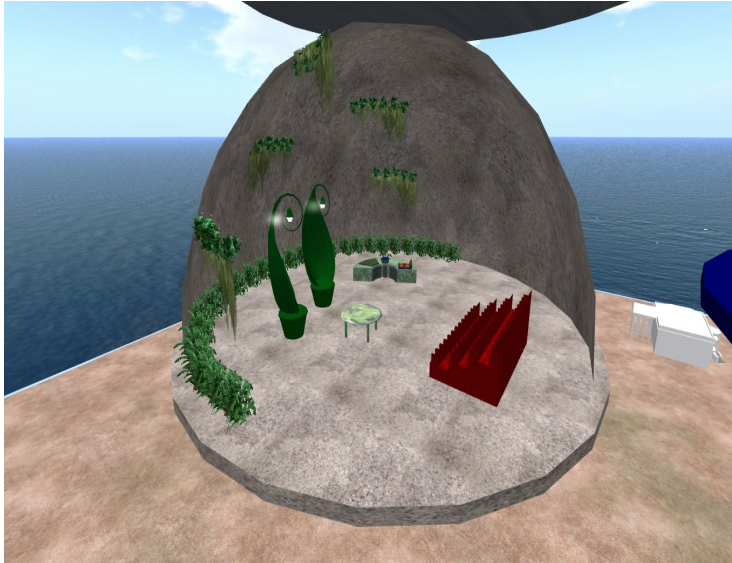


Figure 0

This design was chosen due to the fact that it does have a large open area that can be used to illustrate the interior of a building, or if the scene is outside, the dome can be removed for a view of the sky. While this was the base design for all of the scenes used, the dome structure was not constant throughout. For example one of our scenes used a cube-like enclosure to duplicate the interior of a currently existing building.

Scene Transition:

Each scene was done on a different stage (and these stages were all separate and far apart), so one of our initial problems was the need of a good method for scene transition. Scene transition needed to provide two things: The first was direction for the audience; when to move and where, (without any guidance the audience may not know when a scene is over, or where to go to find the next one). The second aspect was that of timing for the actors. Without any control, audience members might get to a scene before the actors do and they would be standing there confused in front of an empty set. The decided solution was to create seats that moved around, controlled by a stagehand. This turned out to be the optimal solution. A set of forty-eight seats was designed and given six sets of coordinates to move to at a specific speed, once given the command from a stagehand. This solution turned out to work extremely well and solved all of the problems for scene transition.

Controlling the Audience:

While not implemented, there were methods for dealing with an audience that were discussed. One of these methods was to create an invisible box around the seats that would not interfere with any audience member's camera angle, but that would hold them within a confined area should any member decide to stand up and walk around.

Acquiring Assets:

Once you have your script and the stage setup, it is time to acquire the stages, costumes, and anything else you need for a virtual performance. In second life there are two main ways of acquiring items. The first is creating them yourself. Second Life provides an interface to make items from predetermined shapes, which one has incredible control over. Sculpted items can be made in a 3D modeling program such as Maya. The second technique of acquiring items is to buy them. Within the community of Second Life, there are vendors that sell items for Linden dollars and there are also websites that do the same and deliver the items directly to your character in game. Finally once items are acquired, they may or may not need scripts in which to animate them or cause them to interact and make the performance more of a reality.

Building Blocks:

Why build your own items? Well first of all, the items you build are fully customized to be exactly what you need for a performance and secondly, they are completely free. When entering Second Life, one can find the build menu on top of the screen with the build (ctrl + B) option. When clicking that option, one opens the window shown below in figure 1.

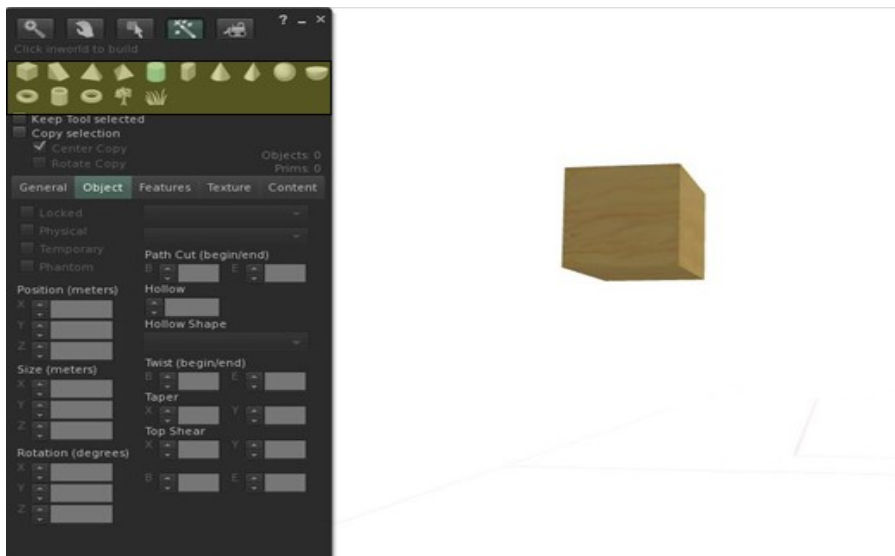


Figure 1

The highlighted shapes show the basic objects that one can use to start building any prop needed for a performance. To create a basic object or primitive within Second Life space, one only needs to click the desired shape and then click on an area to create the primitive in that space. Once the primitive appears, the build box is switched to the object menu (highlighted in figure 2).

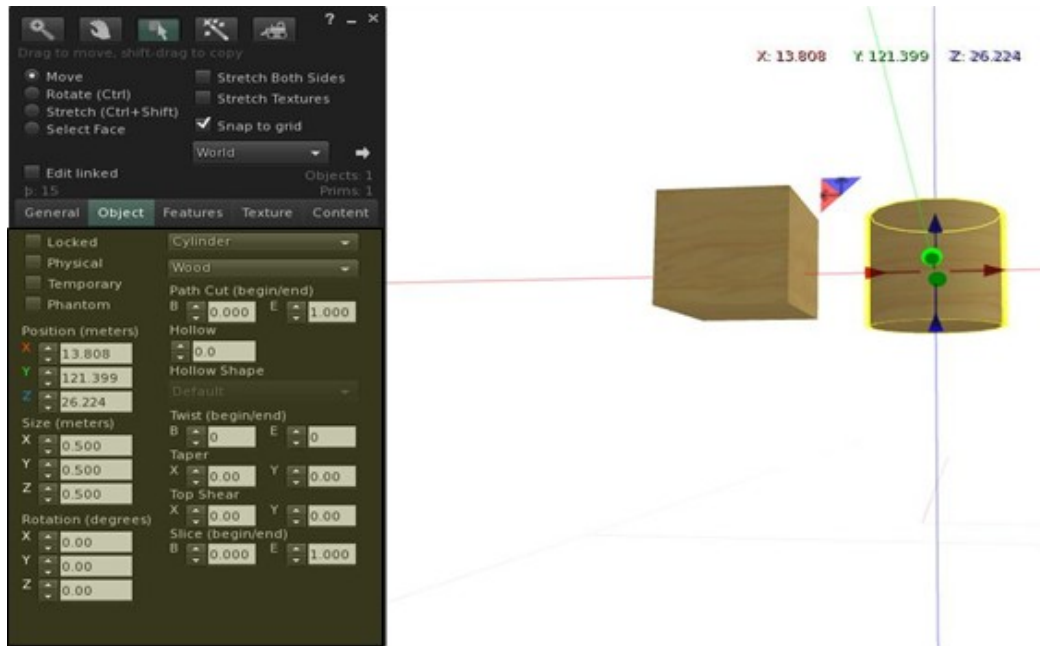


Figure 2

In the object menu, there are many options for editing the building blocks. Many of these options are described below in table 1.

Locked	Does not allow anyone to edit this object (including you).
Physical	Allows this object to be affected by gravity and other physical objects.
Temporary	Set this object to disappear within approximately one minute.
Phantom	Does NOT interact with other physical objects or avatars.
Position	Control the position of the object with X, Y and Z coordinates.
Size	Controls the objects size in the X, Y and Z directions.
Rotation	Controls the objects rotation in the X, Y, and Z directions.
Dropdown 1	Controls the basic shape of the object.
Dropdown 2	Controls the material base of the object.
Path Cut	Allows the user to obtain only a piece of the object.
Hollow Shape	Allows the user to control the shape of the hollow opening of the object.
Hollow	Hollows the object.
Twist	Allows the user to control how much the object is twisted from either

	end.
Taper	Allows the user to taper the object from one of two ends.
Top Shear	Allows the user to shear the object to one side from one of two ends.
Slice	Similar to Path cut it allows the user to obtain only a piece of the object.

Table 1

Different shapes have different options and table 1 only encompasses the options for a basic cube or cylinder. Many shapes have similar options. However, depending on the shape there are different ways to alter them.

Although Second Life has an incredible amount of customization, sometimes it is just lacking exactly what you need. Maya, a 3D modeling program, allows for users to create incredibly customized shapes known as sculpted primitives which can be uploaded and used as building elements or complete objects.

Putting the blocks together:

Simple shapes, even edited to be slightly more specific, are not enough to make the complicated items you may need for your performance. That is where linking comes in. The link (or unlink feature) is also found under the build menu, and allows one to link basic shapes into one item. Once all the shapes needed for an item are created, one can move them into position and link them into one item for use, see figure 3 below for an example.

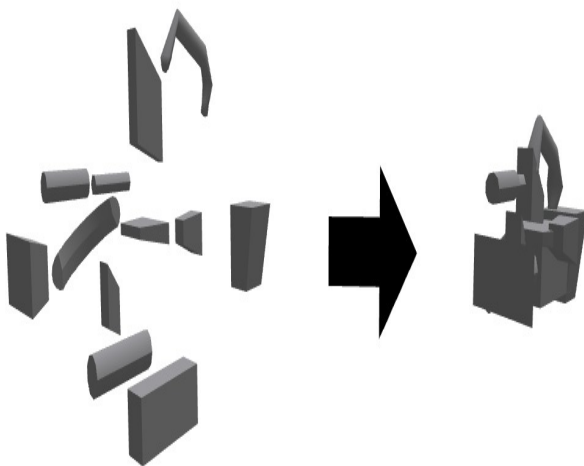


Figure 3

Making Things Pretty:

Once you have your objects, they probably need color, design, and texture on them to make them look more realistic. Once you have your object created in the build menu, there is a

texture tab that allows you to control the specific look of whatever you are building. You can texture objects, edit their colors, transparency, brightness and much more. The functionality of editing object in Second Life is shown in figure 4.

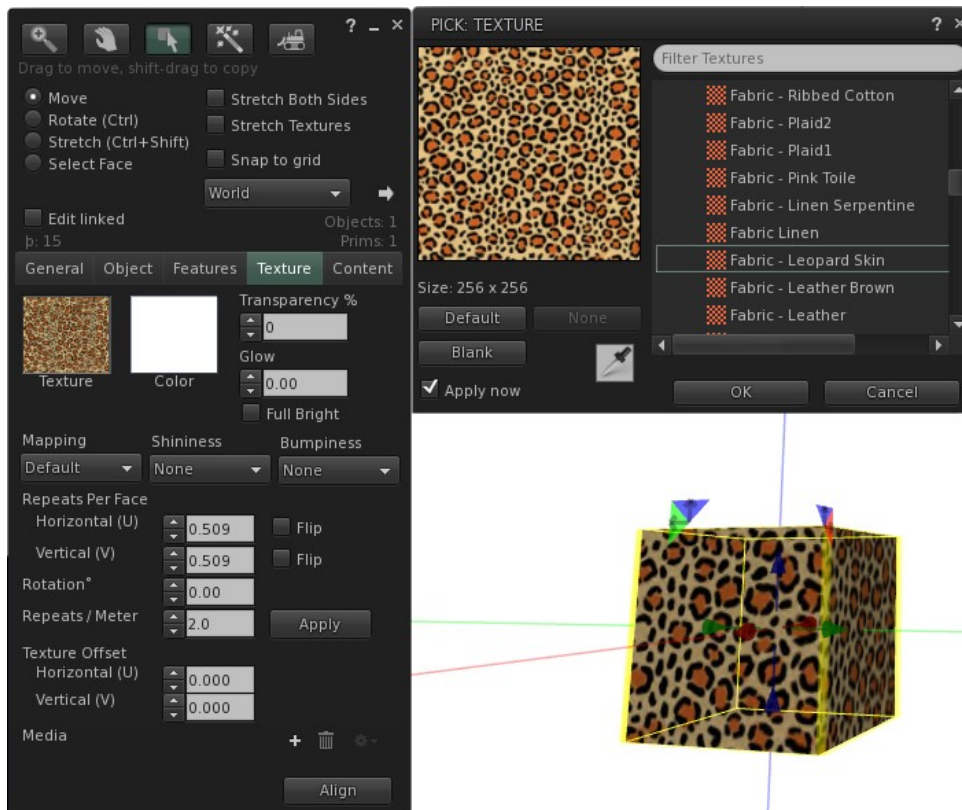


Figure 4

You can edit an object's appearance either by using the menu within the build interface or by dragging a texture from your inventory onto the face of an object, to put that texture on a specific face (this functionality is also available within the build interface but is much easier if you choose to drag and drop). Eventually all your objects should be built and look exactly how you want them.

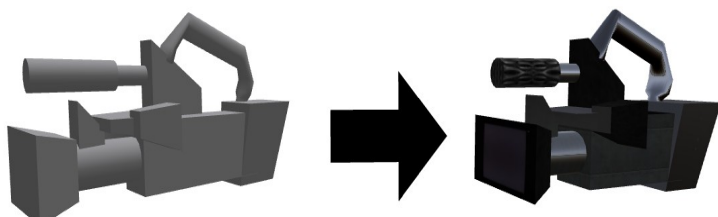


Figure 5

Once your objects are already in Second Life space, it is possible to right click them and select edit to enter the build interface window. You can also right click and select remove->Delete to get rid of any unwanted objects.

Cooperation:

When creating a virtual performance, you probably will not have a one-man show. So you must make sure that the other designs and actors in the show have the ability to modify the objects you create and vice versa.

There are three types of permissions that exist in Second Life. They are copy, modify, and transfer. Copy permissions allow for a person to make as many copies of an object as desired. Modify permissions allow for someone to change an object in any way. Finally, transfer permissions allow for one person to give the object to another person. Allowing any number of those permissions gives the creator of an object the ability to control how far his or her object can go. There is one other type of permission that is only relevant in a group: Sharing, which grants you and everyone in your group full access to an object.

There are two ways to give permissions to objects. The first is in the general options for the object (shown on the left) and this allows you to change permission rights on the object you are currently editing. The second method (shown on the right) is by right clicking the object in your inventory and this allows you to change permissions in the item profile.

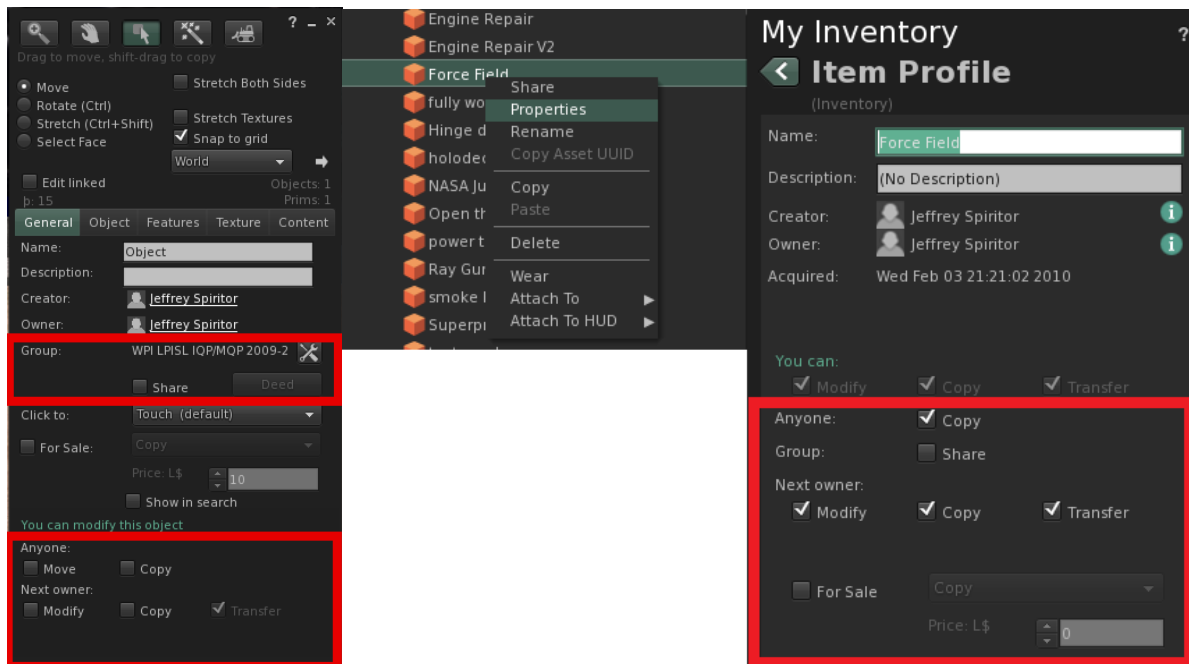


Figure 6

If You Don't Have the Time:

Of course sometimes, you do not have the time or resources available to sit down and build everything you need. Luckily there are other options for acquiring objects. In our performance there were three main resources used to acquire items we did not build ourselves.

The first is the second life library, found in the inventory. This library contains hundreds of objects, textures, scripts (discussed later), and more, and it is completely free. So before you go looking anywhere else, check your library to see if you already have something you need.

The second resource is XStreet (www.xstreetsl.com), which is a site linked with Second Life where people are able to post items for sale. Again, if you are lucky, you might be able to get some things for free or at extremely low cost. Once you have found the items you need, you can purchase them from XStreet and they are delivered directly to your Second Life account.

The third possibility for acquiring items is finding an in-game vendor, where you are able to buy the items you need right on the spot.

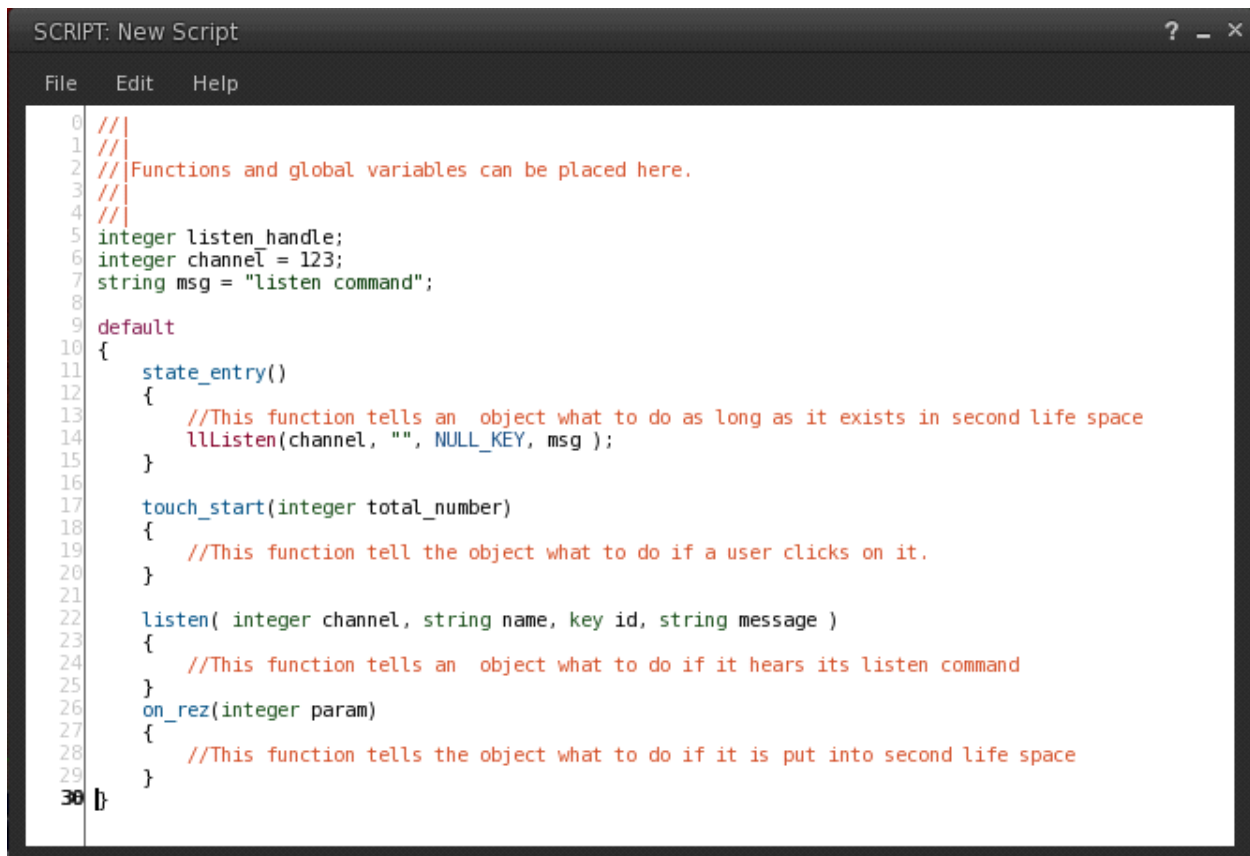
Neither of the second two methods is superior to the other, although XStreet provides a much greater searching capability. Both will provide items with similar costs, though you are much more likely to find items you need for free on XStreet, since you can search specifically by price. Both methods have nearly instant delivery time and there are some items that are unique to either XStreet or in-game vendors. So if you do not have the resources to get everything you need, there are a number of options.

Not So Boring:

Once you have your objects, they may not do anything. This is where scripts come into play. Under the contents tab of any object, you will find what looks like an empty folder. There you can input scripts, objects, textures, or whatever you need your object to hold. While putting objects and textures into the contents folder will not accomplish much, putting a script in there will allow your object to be more than a hunk of primitives.

What are scripts? Basically they are pieces of code that tell your object to do something. This can be anything from commands to move around, animate textures, or even act as a working gun. Scripts in second life are written in a language very similar to JAVA or C, known as Linden Scripting Language, or LSL.

A script in Second Life can have the basic outline similar to that shown in figure 7.



```
0  ///  
1  ///  
2  ///  
3  ///  
4  ///  
5  integer listen_handle;  
6  integer channel = 123;  
7  string msg = "listen command";  
8  
9  default  
10 {  
11     state_entry()  
12     {  
13         //This function tells an object what to do as long as it exists in second life space  
14         llListen(channel, "", NULL_KEY, msg );  
15     }  
16  
17     touch_start(integer total_number)  
18     {  
19         //This function tell the object what to do if a user clicks on it.  
20     }  
21  
22     listen( integer channel, string name, key id, string message )  
23     {  
24         //This function tells an object what to do if it hears its listen command  
25     }  
26     on_rez(integer param)  
27     {  
28         //This function tells the object what to do if it is put into second life space  
29     }  
30 }
```

Figure 7

Like JAVA or C, LSL can have different variables, functions and methods in order to control objects. The language has full capabilities of basic programming. It has functions such as statements, for loops, while loops, or whatever else you may need. There are also many Second Life related functions already build in to make creating a script for your object easier. But before we get into the specifics, lets go over some of the basic uses that the objects might have in the play.

Passive Items:

Some items will be passive, or rather they may always be animated or perform fixed functions (such as rotating a texture). For these items, all commands will be put into the state_entry() area to ensure that whatever the objects needs to do, it will always be doing that action.

Active Items:

While there are many ways for an item to do something on activation, there are three ways specifically that we used in this performance and that will be discussed in this guide. The first is activating by clicking.

Within the touch_start() section, any commands put in will only be executed when the object with the current script is clicked by a user within Second Life. This is ideal for things such as door or elevators that make logical sense to activate when touched.

The second way to manually activate an item is to have it listen to a channel. For this to work, the item must be listening to a channel when it is active and this can be accomplished by calling the llListen command in state entry. This command takes 4 commands. The first is an integer and represents the channel in which the object listens. The second is a string representing the name of an object or avatar that the object should be listening for. This section can be left simply as "" to ensure that it listens to everything. Third is a UUID for an avatar to specifically listen only to one person. This can be set to NULL_KEY so that it will not discriminate who the object listens to. Finally, the fourth parameter is a string that the object hears in the specific channel and will activate the listen method. The listen() method contains anything that the object should do when the listen script is activated for that object.

Finally, there is the on_rez() function which contains any task the object should perform when it is placed into Second Life space from one's inventory.

The Functionality:

Now that the basis for scripting objects has been set, we can move on to what you can actually do with the scripts. Below in table 2, there is a list of commands that were frequently used in our performance for many of our objects.

llSetText(string texture, integer face);	Sets the texture of an object
llSetTextAnim(integer m, integer f, integer x, integer y, float s, float l, float r);	Sets texture animations
llGetRot();	Returns a rotation vector
llSetRot(rotation rot);	Sets a Rotation
llSetColor(vector color, integer face);	Sets a Color
llPlaySound(string sound, float volume);	Plays a sound
llLoopSound(string sound, float volume);	Loops a sound
llStopSound();	Stops Sound
llGetPos();	Returns a position vector
llSetPos(vector pos);	Sets a new position for the object
llRegionSay(integer channel, string msg);	Broadcasts a message in a channel
llSleep(float sec);	Puts the script to sleep for sec seconds

Table 2

Above are commands that allow for the changing of object appearances and the general movement of objects. Of course there are many more commands available that can be found by

searching the Second Life Linden Scripting Language Portal found at http://wiki.secondlife.com/wiki/LSL_Portal.

It is also possible to purchase scripts on XStreet or from Vendors. Objects acquired from these sources very often come with their own scripts. For some help or ideas for your performance, many of the scripts we wrote or used can be found in Appendix A.

Equipping Your Objects:

Now that you know how to create your own objects and get them to perform the actions you need to use them for your play, it is time to talk about equipping some of those objects for either your costumes or your UI. When right clicking on any object, it is possible to wear it or attach it to any part of your avatars' body. Once attached, you move the object to the optimal position with respect to your avatar.

You might notice that not only do you have an "attach to" option, but you also have an "attach to HUD" option as well. This option will allow you to attach an object to your user interface so that only you can see it. Why would this be useful? Well as a group, we created a control board that attached to our user interfaces with buttons that used the `llRegionSay()` command in the `touch_start()` function to allow for a single person to control stage parts and performance queues, that used the `listen()` function to operate, from anywhere on or off the stage. A stagehand was then able to control the objects on queue.

Slight of Hand, Getting your items to play their parts:

Unlike in real life, a virtual space poses many complications for simple tasks relating to items and objects that your actors must use. In virtual space much is reversed. It is easier to move a house from one place to the next, but passing a ball from one person to the next is anything but easy. There is no easy solution to these problems, though with work, they can be overcome.

One issue that you may encounter is that at some point in your performance, there will need to be an exchange of props from one character to another. This simple interaction brings about four major hurdles that must be overcome. In order, these hurdles are lag, render time, human reactions, and time. Really, lag is a non-issue since there is very little that can be done to alleviate this, let alone control when and how bad it will be. Because lag is relatively common, you can get away with the slight delays that it can cause though you need to be aware that it exists.

Render time is a big thing that you tend to forget about after having done the performance a few times. Your audience generally won't render what they can't see, so anything new will not only have to be rendered from scratch, but it will also be put in a cache. Both of these things take time. Small amounts of time, provided that the object is relatively simple and doesn't have too many complicated scripts running, but time nonetheless. During this period, your audience will see one of two things: Nothing, or the object rendering one polygon at a time. Again, the

only real thing you can do to alleviate this is keep your objects simple. Another fix which can work (though it can cause hairline fractures in the immersion) is having a copy of the object somewhere within the set so the audience already has the object cached and therefore can render it almost instantly.

Human reaction is really the easiest problem to fix, though it will require your actors to be very comfortable navigating the interface in order to attach and detach items at the right times. They need to practice getting their timing down otherwise you will have two copies of the item at once, or none. This is due to the only real way to “pass” an object visibly is to have one avatar derez the object as the other rezes it, giving the illusion that the object passed from one actor to another.

While passing objects between actors is the hardest trick to pull off, interacting with objects can also be difficult to pull off seamlessly. While Second Life has an animation to accompany a user clicking on an object, it is little more than the avatar pointing to an object. This means that different gestures and animations will need to be used to simulate typing, opening doors, turning knobs, and so on and so forth. While again, buying different gestures and animations is effective and can yield high quality results, it can be either expensive or difficult to find the specific animations you are looking for. A program that helps create animations specifically for Second Life is Qavimator, which is free and can be downloaded from <http://www.qavimator.org/>. It is worth mentioning that while creating the animations is free, uploading them does cost 10 Linden dollars (about 3-4 cents).

One object that we spent a fair amount of time working on was the actual seats that the audience sat in. Because we had to work around an awkward stage placement, we needed to create a way for our audience to move from scene to scene in a way we could control. Moving seats was the answer we found, turning the play into something like an amusement park ride. Using props spread in between the different sets, we were able to alleviate the breaks between the action with mini transitions indicating a change in location or time. This allowed us to better immerse our audience into the world of the performance, since we were able to show them where they were as opposed to telling them.

Media:

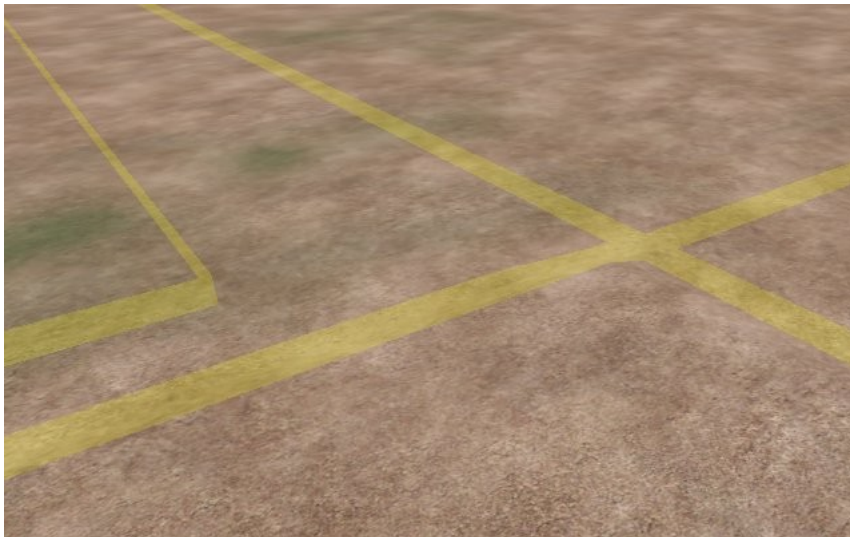
Second Life offers many ways to convey parts of the story to your audience other than just performing a scene. One of these other methods that we made use of was the ability for you to map video and audio onto objects in order to create movies. This system is not without its flaws and complications that you should be aware of before trying to include this kind of viewing experience within your own performance, but should you overcome these hurdles, it can add a different experience from what your audience will be expecting.

With the advent of Second Life’s newest, (at time of this writing), user interface dubbed “Viewer 2” there now exists two methods for creating fully animated media textures. The

“traditional” method makes use of the feature that allows a parcel of land to be assigned a media source to project onto a specified texture. This was the primary method we used, mainly because Viewer 2 was finished within the last third of our project and we wanted to make sure that our performance would be viewable by anyone, not just those using Viewer 2. If viewer 2 does not meet your needs there are a number of third party viewers available to you including the popular Emerald viewer.

Regardless of which media format you decide to use, you should be aware of two things that can create problems for both you and your audience. First and most important is that your audience won't load anything they can't see. While this is not just a problem that exclusively arises with media, it heavily impacts the display and timing of the media. With the traditional format of using parcel media, the playing media is separate for every audience member, thus causing the exact time when the media starts and stops playing to vary drastically from user to user.

Parcel Media, The Traditional Approach:



An example of parcel divisions

Parcel Media, as it will be referred to, is named after the ability for an area of land to be sectioned off into separate areas referred to as parcels. The reasons for parceling is varied; from restricting access to certain areas of an island, to determining what actions can and cannot be made within that area. For our purposes, we mainly made use of the media playing features of the parcels, but the other features can be just as helpful when it comes to your own performance.

To start, it is only possible for the owner of the island to subdivide land into parcels, so it is important for either someone within your group to be the owner or for the owner to be easily reachable so they can subdivide land for you when it is needed. The owner is also needed to set the media assigned to the parcel as well as what the media texture for that parcel happens to be. Because of this, we strongly recommend that it be someone within the group itself.

It is important to note that for the most part, control over the media projected by the parcel is extremely limited. There is no way to exert any control over what others are seeing and hearing once the media has loaded, beyond controlling when it loads. Once the media starts loading, it will finish loading, and once that happens it will play once, and precisely once, in its entirety regardless of if there still exists a surface for which it to play it on. This implies the following things:

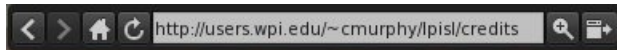
- A parcel media file will play exactly once every time you enter the parcel
- There is no way to force a replay of a media file that has already completed aside from reentering the parcel
- A media file will continue playing even if the surface it was being projected on is removed. This is most notable with media that contains audio as it will continue to play until it finishes.
- If the media texture is displayed **anywhere**, regardless of whether it is visible or not, at the time of entering a parcel, the media will load and play

This last point caused us a fair amount of trouble as we had screens placed in multiple stages across the island. We were having an issue with media playing at the wrong time. To remedy this, we created a pair of scripts (MED-001, MED-002) that in addition to hiding the screens when not in use, would also change the texture to something other than the media texture when the screen was not visible. While this proved effective, it left very little room for error on the part of the stagehand.

It is also important to understand that what media plays where depends entirely on where the viewer's avatar is not where the screen is located. The audio from parcel media is parcel wide and as such, will play regardless of how far away from the source you get so long as you stay within the same parcel. The following example illustrates the prior two issues: an avatar standing in parcel A will hear A's media even if the screen is located in parcel B.

Website Media, The Viewer 2 Approach:

The new viewer 2 approach to media simplifies many of the problems that the old media system had while adding in some new ones of its own. While we can't speak extensively about this method because we were constrained to viewer 1, (needing to accommodate the SL population that hadn't yet migrated to viewer 2), we would strongly recommend exploring it as a replacement for the old parcel media as opposed to regarding it an equal alternative.

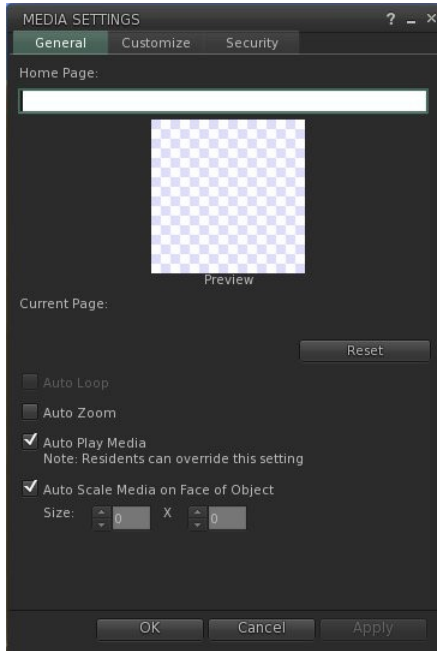


The Media bar that appears over media textures in Viewer 2

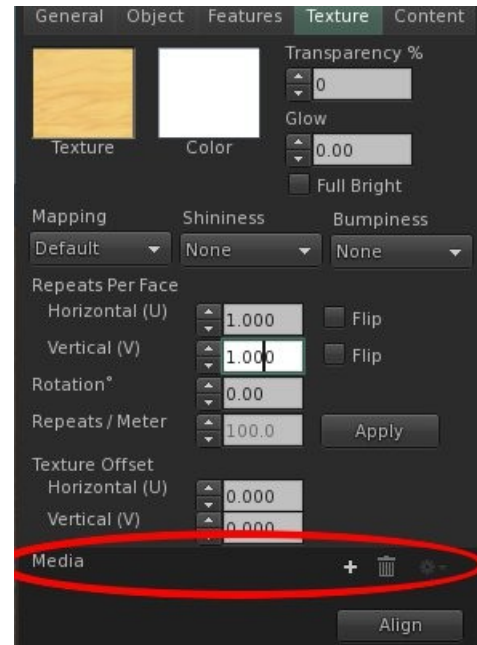
The viewer 2 approach makes placing media onto an object very easy and intuitive, due in part to the interface changes that made the viewer itself work and feel very similar to a web browser. With viewer 2, you can place any media that is linked to a URL (note: not a SLURL, but a normal URL) onto an object's face. The web content that is on that face will function exactly like the page itself would including allowing you to move along hyperlinks and fill in text fields. The only restriction is that pop-ups do not display, so any login page that uses a pop-up cannot be used. Downloads also won't work. The viewer itself is compatible with flash and anything that can play in QuickTime, but beyond that, we are unsure of the exact limits. This means that more obscure movie file formats might have trouble playing even if they are working properly in a common web browser.

One of the main advantages we found when exploring this form of media usage was that whatever is being displayed, it is the same for everyone viewing it. While this doesn't eliminate loading time differences (which can cause some audience members to miss the first part of a media clip), it does make it easier to control what your audience is seeing and when they are seeing it.

When using this media solution, it is very important that you set your permissions correctly for the media itself. At default settings, a media texture will work like a web browser except everyone can use it at the same time. It is important that you keep your audience from changing the media from the intended file to some video on Youtube. These settings are all controlled from the media settings screen that pops up when you click the gear icon on the texture tab for the object itself.



The media settings screen



The texture tab of an object

One thing worth mentioning is that because the media texture will display more or less like how the page would actually display in a web browser, chances are your media will have a progress bar displaying at some point. It is important to determine if this causes too big a break in the performance's atmosphere. An additional media aspect to consider with regards to the feel of the performance is consistency in choice of styles in your media. For example, doing a performance in Second Life with the avatars, but having your media be of live people, can cause cracks in the continuity of the performance that may or may not be desired, depending on exactly what you want from your media.

Acting:

As one might imagine, acting within a virtual environment is very different than doing so in the physical world. The skills needed are drastically different and many of the things that are necessary for an actor to do on a physical stage production are eliminated or greatly simplified with virtual acting. There also are elements that become much more important, such as multitasking and timing.

Picking actors:

Picking actors for a virtual performance is similar to standard acting however because the only part of the actual actor that is carried through the medium is their voice, it becomes much more important that you actors sound how you want them to. Because of this, actors that can do multiple voices that are dramatically different and unique are invaluable, especially if your cast is small and you have certain actors changing characters from scene to scene.

One idea that we were unable to test due to lack of physical bodies, (but could ideally yield a better overall production) would be to use two physical bodies for each virtual character. By using one person to move the avatar and another to provide the voice, you alleviate some of the stress on an actor when they are required to say a line while also manipulating their virtual puppet, or counterpart.

Virtual role assignment:

While not usually different from the assignment of roles in a traditional performance, virtual roles allow some flexibility. Because instantaneous character changes are possible, it is very feasible to have one actor play multiple roles provided that at no time those characters are needed to appear at the same moment. This is where the individual talents of each actor need to be scrutinized. Primarily actors should only be given roles that they can appropriately voice. For example only one of the five people in our group could do a female voice so it was required that he play the female roles in the performance.

Virtual Acting:

In the same sense that hitting keys on a keyboard is different from walking across a room, virtual acting is different from normal acting. As has already been mentioned, virtual acting is made or broken based entirely on the writing of the dialogue and how the actors are able to deliver this dialogue. This is especially important in a medium like Second Life, where your actors are severely limited in their control over their avatar's facial features. Emotions must be delivered entirely through how dialogue is delivered and what is being said.

Beyond the limitations of the system used, your actors must be intimately familiar with the system they are using. Much like how lines and blocking must become second nature for an actor in a physical performance, keystrokes must also be memorized and practiced for a virtual performance. This familiarity will allow the actors to ease the stress of doing multiple things at one time and helps a great deal in allowing the performance to run smoothly.

Conclusion

Over the nearly year-long process of planning, execution and aftermath of our project, we learned a lot. This use of virtual space is still in its infancy and as such, there is vast room for improvement and learning. In the time we spent we only scratched the surface. If there had been more of us with backgrounds more suited to writing and acting we could have produced something far beyond what we finished with. If live virtual performances are to evolve and grow into a widely accepted and used process, others from outside the interactive media field will need to get interested. We hope that we have gotten the ball rolling and that we can encourage others to keep it moving in the right direction.

The included recording with this paper is of a full run through of our performance, but it is an early run before we had all the effects and costumes finalized. Many of the issues that we discuss above are clearly visible at this point.

Appendix A

Movement script used to move audience seats from one stage to another:

//scene 2 is the position to move the object, rotate is the final rotation of the object, activate is the word in the channel 43284 needed to activate the script and the speed is how fast it goes.

```
vector scene2 =<32,116,56>;
```

```
vector rotate = <90,90,0>;
```

```
string activate = "scene1";
```

```
float SPEED = 1.0;
```

```
//please limit the speed to a maximum of 5.7, otherwise the object can jump more than 10 meters and stop early
```

```
vector pos;
```

```
integer listen_handle;
```

```
default
```

```
{
```

```
state_entry()
```

```
{
```

```
listen_handle = IListen(43284, "", NULL_KEY, activate);
```

```
}
```

```
listen( integer channel, string name, key id, string message )
```

```
{
```

```
pos = IGetPos();
```

```
ISetRot(IIEuler2Rot(rotate * DEG_TO_RAD));
```

```
while(pos != scene2)
```

```
{
```

```
if( IIFabs(pos.x - scene2.x) > SPEED)
```

```
{
```

```
if( pos.x > scene2.x )
```

```
{
```

```
pos.x = pos.x - SPEED;
```

```
}
```

```
else
```

```
{
```

```
pos.x = pos.x + SPEED;
```

```
}
```

```
}
```

```
if( IIFabs(pos.y - scene2.y) > SPEED)
```

```
{
```

```
if( pos.y > scene2.y )
```

```
{
```

```

        pos.y = pos.y - SPEED;
    }
    else
    {
        pos.y = pos.y + SPEED;
    }
}

if( lIfabs(pos.z - scene2.z) > SPEED)
{
    if( pos.z > scene2.z )
    {
        pos.z = pos.z - SPEED;
    }
    else
    {
        pos.z = pos.z + SPEED;
    }
}

llSetPos( pos );

if( lIfabs(pos.x - scene2.x) <= SPEED && lIfabs(pos.y - scene2.y) <= SPEED && lIfabs(pos.z
- scene2.z) <= SPEED)
{
    pos = scene2;
    llSetPos(pos);
}
}
}
}

```

A Button Script used for the buttons on the attached User Interface Control Board:

```

default
{
    touch_start(integer total_number)
    {
        llRegionSay(43284, "smoke");
    }
}

```

Script to change windows to a different texture and make the move very quickly:

```

string texture1 = "stars bright";
string texture2= "stars-texture-3";
float time = 5.0;
integer side = ALL_SIDES;
integer listen_handle;

swith(string texture)
{
    IISetTexture(texture,ALL_SIDES);

}
default
{
state_entry()
{
    listen_handle = IISetListen(43284, "", NULL_KEY, "warp");
    IISetTextureAnim(ANIM_ON | SMOOTH | LOOP, ALL_SIDES,1,1,1.0, 1,0.25);
}
listen( integer channel, string name, key id, string message )
{
    swith(texture1);
    IISetTextureAnim(ANIM_ON | SMOOTH | LOOP, ALL_SIDES,1,1,1.0, 1,1.25);
    IISleep(time);
    swith(texture2);
    IISetTextureAnim(ANIM_ON | SMOOTH | LOOP, ALL_SIDES,1,1,1.0, 1,0.25);

}
}

```

Scripts for hiding/revealing the video windows

MED-001 (Revealing script)

Note: string texture1 is the texture we used for our media texture while string texture 2 is anything but the media texture

```

float cloakSpeed = .1;
string texture1 = "7bef4ce1-3a28-d9f2-bc83-1bde90d48957";
string texture2= "Walnut";
integer listen_handle;
swith(string texture)
{

```

```

    IISetTexture(texture,ALL_SIDES);
}
default
{
    state_entry()
    {
        listen_handle = IIListen(43284, "", NULL_KEY, "screen3");
    }

listen( integer channel, string name, key id, string message )
{
    IISetPos(<31.029,137.584,62.459>);
    swith(texture1);
    integer x;
    float xf;
    for (x=1; x<11; x++)
    {
        xf = x * .1;
        IISleep(cloakSpeed);
        IISetAlpha(xf,ALL_SIDES);
    }
    state default;
}
}

```

MED-002 (Hiding Script)

```

float cloakSpeed = .1;
string texture1 = "7bef4ce1-3a28-d9f2-bc83-1bde90d48957";
string texture2= "Walnut";
integer listen_handle;
swith(string texture)
{
    IISetTexture(texture,ALL_SIDES);
}

```

```
default
{
  state_entry()
  {
    listen_handle = IListen(43284, "", NULL_KEY, "screen3off");
  }

  listen( integer channel, string name, key id, string message )
  {
    ISetPos(<31.029,137.584,72.459>);
    swith(texture2);
    integer x;
    float xf;
    for (x=9; x>=0; x--)
    {
      xf = x * .1;
      ISleep(cloakSpeed);
      ISetAlpha(xf,ALL_SIDES);
    }
  }
}
```