2004

Multi-timeline randomized story based web animation

Jonathan Fahnestock

Rochester Institute of Technology

A Thesis Submitted to the Faculty of the College of Imaging Arts and Sciences
In Candidacy for the Degree Master of Fine Arts

Multi-Timeline Randomized Story Based Web Animation:

By: Jonathan Fahnestock

Date: August 31, 2004

Jonathan Fahnestock 8/31/04

Jonathan Fahnestock Date

Thesis Committee Approval

Chief Adviser: Chris Jackson, Assistant Professor, Computer Graphics Design

Chris Jackson 8/31/04

Signature of Chief Advisor Date

Associate Adviser: Jim Ver Hague, Professor, Computer Graphics Design

Jim Ver Hague 8.31.04

Signature of Associate Advisor Date

Associate Adviser: Marla Schweppe, Associate Professor, Computer Graphics Design

Marla Schweppe 9/10/04

Signature of Associate Advisor Date

School of Design Approval

Chairperson: Patti Lachance, Associate Professor, School of Design

Patti Lachance 9-10-04

Signature of Chairperson Date

Multi-Timeline Randomized Story Based Web Animation:

By: Jonathan Fahnestock

Abstract:

Multi-Timeline Randomized Story Based Web Animation is the investigation of the use of randomness as a creative element and production tool for animations produced for the internet using Macromedia Flash™. It discusses the process of utilizing Actionscript to create animations more efficiently, production techniques for animating with Flash, writing for randomized animations, directing voice talent, optimization for web output, and advantages and limitations of the application. The application discussed, titled "Random Life," is an animation which incorporates randomness while remaining coherent by adhering to a traditional story line format.

Keywords:

Randomness, Random, Randomization, Animation, Random Life, Actionscript, Efficiency, Realism, Macromedia Flash™, Non-Linear Story Telling, Chance, Lip Sync, Tweening, Modular Design, Production, Directing.

Introduction:

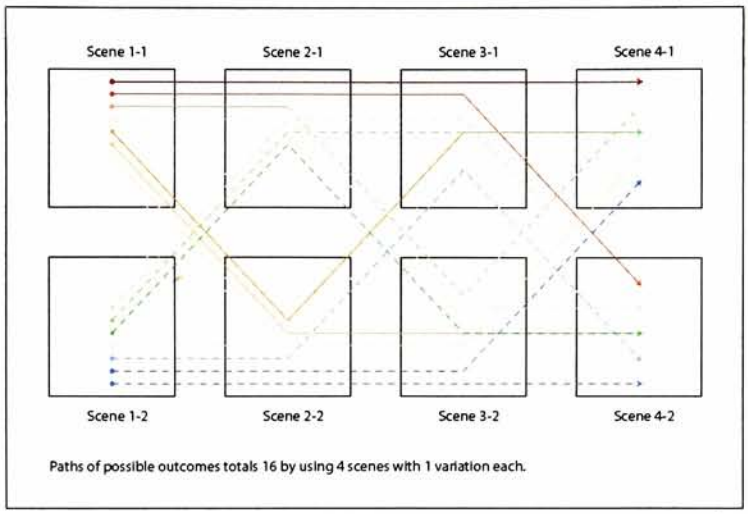
I've always been intrigued with the idea of randomness and how much of our lives' paths are often ruled by chance. Chances are, a random meeting or conversation had something to do with where you meet your wife, husband or best friend. Think about the random happenings that add so much to life, like seeing a shooting star or someone famous in a restaurant. Randomness was seemingly given to us to break up the monotony of our day to day making the world a more interesting and exciting place. With this in mind I began to wonder, if randomness has this power in real life possibly it has an equal power within the environment of animation created for the internet.

Many web cartoons exist today with varying levels of sophistication and quality, but the commonality is that they all have an equal playing field in terms of audience. Anyone with a web connection can view these animations foregoing the need for expensive TV or film production and distribution. According to a radio interview with Dwight Silverman, online news editor of the Houston Chronicle, broadcasted on NPR's Talk Of The Nation, internet animation is becoming very popular within mainstream pop culture (http://www.npr.org/features/feature.php?wfId=1753401).[1] Silverman explains that many web only animation sites such as HomestarRunner.com[2] or Rathergood.com[3] have cult followings which are generating a noteworthy buzz. Possibly I could combine the increasingly proven power of this new phenomenon with the element of randomness to create something unique and interesting adding to this budding web animation environment.

There are several ways I could incorporate randomness into web animation. One example, a randomized animation created in Flash titled "Testimony: A Story Machine" (http://www.myballoonhead.com/storymach.html)[4] by Simon Norton incorporates a more chaotic use of randomness. The Story Machine's structure is loose, creating a story that doesn't have a specific order or meaning. Because of its structure it is simplified with very little dialogue. Norton doesn't really care whether people understand what is happening, he leaves the user to construct their own ideas about the story.[5] This creates an interesting surrealistic effect, but I was more interested in something less simplified that was a closer representation of real life. Norton's example is intriguing and engages the user with its incoherence, but it could also confuse and disengage the user for the same reason. After reviewing this piece I began to think about finding a more controlled use of randomness. An animation that was varied enough to be interesting, but still remained coherent by following a more traditional story structure. By controlling the randomness it could possibly reach a larger audience by decreasing the amount of confusion. Thus began the production of the final project titled "Random Life."

"Random Life" is a web animation created with Macromedia Flash which is built by randomly combining variations of its scenes as the story progresses. By using randomness the story and circumstances are different the next time a user watches, but it follows a traditional story line's order to remain consistent and coherent. To explain the structure of "Random Life," I like to use the example of a toy my sister had called Fashion Plates which consisted of several thin plastic plates that had interchangeable parts of outfits embossed on them. The plates were divided into sections: faces, shirts, dresses, pants, and shoes. The child would line up the plates in any combination they wished, then place a piece of paper over them. To create a picture of the new outfit they rubbed the paper with the lead of a pencil. The great thing about this toy was that the combinations were almost unlimited, by changing just one element the child would get a different outfit. As silly as it sounds, the structure of my concept is very similar to Fashion Plates. Instead of choosing from a group of shirts and pants my application selects from groups of varied scenes. The toy allowed the user to arrange the plates in any order, but unless you are Picasso, to make sense they had to be placed in the logical order. For the story to remain coherent as I wanted, the application would also have to select its scenes in a logical order.

Another great thing about Fashion Plates was its modular design. Any shirt selected would match with any pant selected which yielded the optimum amount of outcomes and gave room for additions. New modules could be created and sold separately to give the toy longer life and increase the amount of variations. "Random Life" was constructed with the idea of modulation in mind as well. Each group of scenes could be added to increasing the life and interest of the piece with minimal amount of work. The size of the project would increase factorially every time a new scene was added (see example 1).



example 1

Process:

Animating, especially with a one person team, is not an easy task. I really had to maximize all of my creative and programming skills to finish the project. This list was long but I felt that the project would be accomplished if it was organized and produced as efficiently as possible.

List of Tasks:

- Write two scripts that were entertaining and had scenes that can interchange between each other while remaining coherent.
- Design the characters and settings.
- Cast and record voice talent.
- Create a story board, animatic and prototype.
- Decide best practices for web output: file size, bandwidth, screen size.
- Devise methods for animating: lip syncing, body movement, motion and production.
- Develop Actionscript for randomized scene selection and playback control.
- Write and record sound track music.
- Add sound effects.
- Implementation and create final solution.

Writing:

The first task was to write the stories which not only had to be entertaining, but selected scenes within each story had to be interchangeable. The challenge was in keeping the scenes similar enough to be coherently interchangeable but different enough to be in-

teresting. The first step was to develop the core story line to act as the glue binding the scenes of the randomized animation. The core story line is as follows: the protagonist wakes up, gets a call from a friend to go somewhere, someone knocks on his door to visit, gets picked up from friend, they are presented with a choice, something happens from that choice. Then I began brainstorming on different paths the character could take while remaining true to the core.

Many times I was faced with the problem of something I wanted to happen during one scene would effect the rest of the following scenarios and would negate the effects of its alternating scenes. For instance, I wanted the friend that called the protagonist to be randomly chosen between three. Further investigation uncovered that if a different friend called, that same friend was going to have to be the one who picked up the protagonist in scene four. That would mean that from scene five on, there would have to be three different sections per friend and the application would have to know which friend was chosen. This would leave me, the only animator, to have to animate nine scene fives, nine scene sixes, etc. I soon discovered that for the application to be accomplishable I was going to have to make very calculated decisions about where the plots could go.

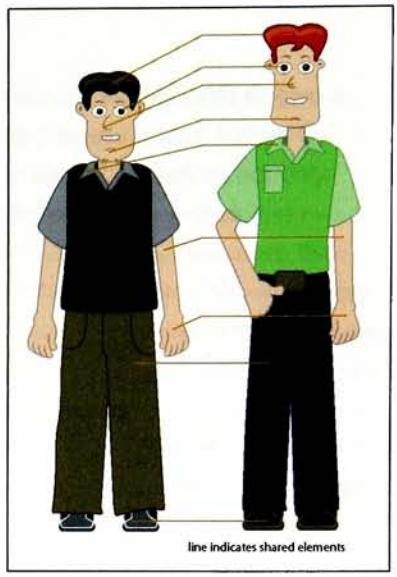
I soon decided that if I couldn't make the scene selections as random as I first conceived I would utilize randomization in a different way to make up for the loss of depth. By containing certain randomized elements within selected scenes I could achieve a similar effect but prevent those elements from affecting the other scenes. For example, within the animation the protagonist's answering machine picks up a phone call from a random selection of messages. The messages add insight to the protagonist's situation but do not effect the following outcomes. I also implemented this by having the characters say randomly selected lines which had similar meaning.

Writing the stories were challenging not only on a technical level, but on a creative level as well. I chose to loosely base the look and personalties of the characters on myself and people I know. I also chose those in my age group as the main target of the piece to ease the process of story development. By including those who are close to the same stage in life as myself I was able to create situations and themes that they could most likely identify with.

Character and setting development:

Developing characters for animation isn't an easy task to tackle. Looking at the scope of my project I decided that efficiency was the only thing that could save me from the doom of mediocrity. I wanted each character to be interesting, engaging and have individual characteristics, but at the same time other factors had to be taken into account. Animation is certainly time consuming and as they say, "Time is money." The intricateness and

amount of characters drawn increases the production time greatly, so a balance had to be found to complete the project. I utilized techniques described in chapter four of Sandro Corsaro's book, "The Macromedia Flash Animator." His advice was to recycle as many elements as possible to save drawing time and file size,[6] so I decided to have many of the characters share elements that took a lot of time to develop such as mouths, hands, arms and clothing (see example 2). By changing the color and orientation of the shared elements I was able to streamline the process while still retaining a character's individuality. This process was also very beneficial in keeping the file size of the project manageable for web output.



example 2

The movement and amount of expression of the characters was also a compromise. Once again I was going to have to juggle what I wanted with what I had time to accomplish. The compromise reached was to spend time on those elements deemed most important to make up for others that were deemed too time intensive or superfluous. The stories relied a great deal on dialogue, so accurate lip syncing, eye and brow movement and quality illustration were given priority over fluid appendage movement and complicated action shots. Interesting camera angles and editing were also used to give the piece more interest.

The settings of the animations were heavily simplified and optimized for the sake of my strengthening nemeses, time and file size. The simplified environments also prevented them from taking too much focus from the characters. I utilized the same element sharing technique as I did with the character's but on a much greater scale. Most of the sets were created from what I call vector primitives which are movie clips consisting of one circle, one square, one triangle, and one oval. By reusing resized, rotated, skewed, combined and tinted instances of these shapes to make a car or road, I dramatically reduced the file size of the project and kept a consistent look throughout the piece. My experience in using this technique is that it is very useful, but can be easily taken too far. Every compromise you make for the sake of efficiency can potentially take more than what you gain from the overall quality of the final piece. For example, one of the scenes in my animation takes place while driving, and I wanted to have buildings passing by. Initially sticking to a strict policy of simplification, I just had resized squares in motion serving as the buildings. As efficient and easy as it was, it just didn't look as good as it should to communicate effectively. In this case as well as many others, quality design and illustration took priority over efficiency. Juggling process with creativity, I discovered, is one of the most important roles a leader of a creative undertaking has.

The Animatic:

When producing an animation, it is usually customary to draw out storyboards to give a rough example of shot compositions. An animatic takes this process a step further by animating the storyboards and adding sound. This gives the director a look at not only composition but the pacing of the piece as well. After the dialogue was written and recorded, scans of my hand drawn storyboards were imported into Flash. They were then animated simply in a slide show fashion along with the recorded dialogue. This was a very valuable step which I, at first, overlooked as a formality. Though time consuming, by having a rough I had a solid prediction of the final piece and was able to make adjustments before having to commit to any heavy animating. This probably saved me more time than it spent, because many of the editing problems were fixed during this stage giving me more time to devote to the final piece. The animatic also gave me material to create a prototype master shell including the Actionscript to control the random selection and play controls. By having a working model of the piece I was able to test the random selection concept and see if it truly worked the way I predicted before the main production began.

Animation methods:

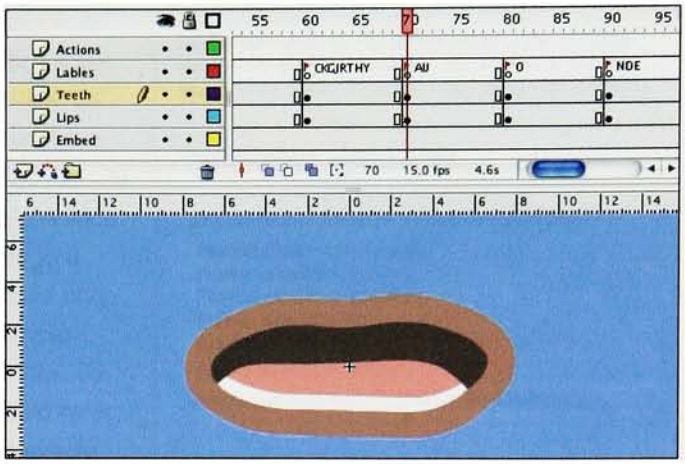
Animation takes a long time! This is where animating with Flash becomes beneficial. Along with the powerful animating tools, Flash also has a powerful scripting language. By utilizing these scripting capabilities I was able to achieve a great deal of traditionally "hard to do" animation with minimal effort, thereby streamlining my process and production time.

Having a set of stories that relied heavily on dialogue, my first concern was getting the characters to talk. At the time, the only method of lip syncing I knew of was to draw the character's different mouth states within separate movie clips. Every time you wanted the character's lips to change along the time line, you would have to make a new keyframe then swap the mouth with the mouth state you wish. This process seemed unimaginably tedious, time consuming and a poor solution for the project and its scope, so I decided to utilize Actionscript to save me time and from excessive tedium.

Taking from the more traditional method I first drew nine different mouth states which represent different letter sounds a character would make while talking, as well as some extras such as frowns and smiles. I then took each of these mouths and put them in their own frame of a movie clip along with a frame label describing the mouth state. For portability, the movie clip also contained the lip syncing Actionscript.

For this method to be efficient, easy and flexible I was going to have to be able to change a character's mouth state anywhere in the main movie without having to worry about long paths that could potentially change. Using Flash's global object, which removes the need to reference the path because it puts the path into memory, was the best method of

accomplishing this. I only had experience in making variables global, so initially I made a global variable to store which frame label I wanted the play head of the lip movie clip to go to and stop at. To constantly test the state of this variable, I called a custom function within an enterframe function which instructed the movie clip to go to and stop at a specific frame label defined by this variable. On the main time line I changed the variable every time I wanted the lips to change states (see code reference 1). The script also includes an array which is a list of the frame labels. Using an array was not necessary but I found that using frame labels and an array instead of referencing the frame numbers was a more organized way to build the movie clip (see example 3). This way I could have a direct association with the mouth state and its position.



example 3

This initial solution worked fairly well, but I could foresee potential problems. Enterframes and global variables can tax a processor during Flash's playback if many are used, and in this case I was going to be using them both many times. Eventually I came to the final, more optimized, solution by discovering that I could make custom functions global which eliminated the need for the global variable and enterframe (see code reference 2).

Code Reference 1: These actions below are place in the mouth movie clip and control which frame label the play head stops.

```
//_____ EnterFrame _____
this.onEnterFrame = function() {
  ChangeFrameLabel();
};
//_____ Function _____
function ChangeFrameLabel() {
  FrameLabel = new Array();
  FrameLabel[0] = "Smile";
  FrameLabel[1] = "Closed";
  FrameLabel[2] = "M";
  FrameLabel[3] = "FV";
  FrameLabel[4] = "WQOO";
gotoAndStop(FrameLabel[ _global.FRAME]);
}
```

This action below can be changed and placed anywhere in the movie. This would change the mouth state to the "WQOO" lip.

```
_global.FRAME = 4
```

After perfecting and implementing the lip syncing script, I soon tired of copying and pasting this code into every character's mouth movie clip every time I made changes to it or created a new character. To streamline my process I wanted to find a way to make the code more generic and movable. I found that by placing the lip syncing code in its own movie clip with a simple square on one of its layers (so I could see it) made the code easier to move around. To make sure that the square wasn't visible during playback I included a

piece of code to make its movie clip invisible. To make the code generic (reusable), I declared the custom function's name as the _parent._name, the instance name of the movie clip which the lip syncing code movie clip was embedded in (see code reference 2). When I wanted to add the code to a character's mouth movie clip, I would only have to drag the new action movie clip into that character's mouth movie clip. To control the mouth movie clip, I would refer to that clip's instance name. For example, if I wanted to add the code to the character Mike's mouth with the instance name MikeLips I would first place the action movie clip inside Mike's mouth movie clip. The custom function needed to control the lips would be _global.MikeLips(0). This method proved very useful in both saving time and easing editing the Actionscript. If I changed the code it would apply globally to all of the characters with no copy and paste. In the short run view, the time spent making the code generic could be seen as a potential waste. However, in the long run view, I was able to adapt this method for other animations such as arm movement, head position and eye position and was able to easily share that code between all of the characters. I spent hours of effort to save only a few minutes of time, but I saved those few minutes over and over again.

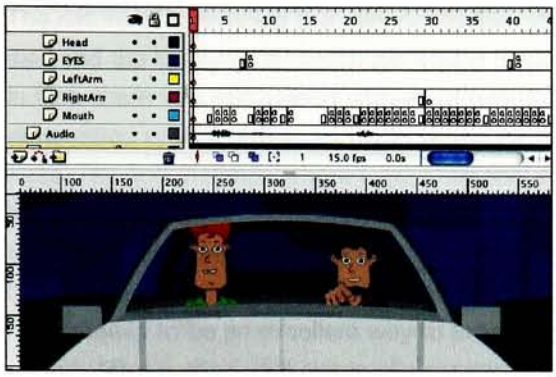
Code Reference 2: This action below is the modified version using a global function. The function's name is the instance name of the movie clip the action resides in.

```
//_____ Variable_____
FunctionName = _parent._name;
//_____ Function _____
_global[FunctionName] = function (theNumber) {
  FrameLabel = new Array();
  FrameLabel[0] = "Smile";
  FrameLabel[1] = "Closed";
  FrameLabel[2] = "M";
  FrameLabel[3] = "FV";
  FrameLabel[4] = "WQOO";
_parent.gotoAndStop(FrameLabel[theNumber]);
}
this._visible = false;
```

This action below can be placed anywhere in the movie. This would control the mouth with the instance name MikesLips and change the mouth state to the "WQOO".

```
_global.MikesLips (4)
```

After the script was written, I was able to change the mouth state anywhere in the movie, no matter where the character's mouth was located. Next it was time to develop a process for production. I found the most convenient method consisted of first placing the audio of the dialogue on a layer with the audio mode set to stream. That way I could hear it within Flash's authoring environment. It was also helpful to separate the dialogue into smaller chunks. I found it easier and more organized to deal with smaller bits of the audio and it gave me a good idea of the timing of the scenes for editing. Then I created a blank layer for the lip syncing actions directly under the audio layer. I would then listen to the audio carefully and when I wanted the mouth state to change I would add a keyframe on the layer. Within this keyframe I would call the global function which passed the mouth state to the embedded Actionscript located within the character's mouth changing it to the desired mouth state. As the play head moved past the keyframes, the custom function would be called corresponding to the audio (see example 4).

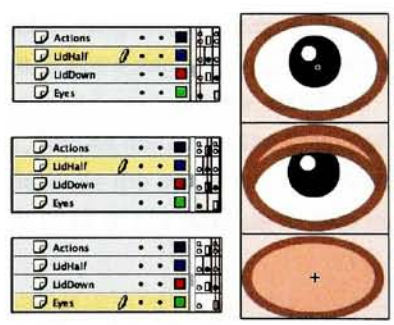


example 4

Listening to the audio and manually placing the code each time I wanted the lips to change proved to be the most tedious and time consuming aspect of the whole process, but it was much easier than the more traditional method for many reasons. If the lip syncing was done the old fashioned way, every time I wanted to change the character's positioning, scale or dialogue I would most likely have to completely redo the lips manually. This would definitely leave much of the work set in stone whether I liked it or not. By letting code control the lips, once the lip sequence was set, I was given the benefit of changing my mind in terms of the scale and position of the characters. I was able to easily make on-the-fly editing decisions easing the process and giving more time for other aspects of the production. I also found it much easier to change a line of code than to constantly have to swap graphics.

This method proved very efficient and versatile so I implemented similar methods for controlling arm, eye and head positions. The disadvantages were few but are as follows. Since you are using code, you cannot view the animation within Flash's authoring environment, you have to test movie. This makes it a little more difficult to time scripted animations with sound effects and tweened animations. Being somewhat processor dependent, it also could cause syncing problems if the processor was slowed because of other scripts or tweened animations. Finally, since the custom function is called on keyframes it is locked to the frame rate the movie was set at. If you were to change the frame rate you would have to change the position of the scripted keyframes.

The other movements of the characters such as head, eye and arm were animated in a very similar fashion. Like the mouths, each part was given its own movie clip with its different states placed within it. They are also controlled by a global function on the main time line. These parts were assembled in a single master movie clip, so the character made up of these parts could easily be moved, copied and scaled. By making the functions global, the separate parts could be nested anywhere within the master clip, making the editing and rearrangement of a character's attributes easily achieved. This feature also allowed the parts to be independent from the master. For example, in one shot I only needed the top half of the characters because they are sitting in a car, so I created a new movie clip made up of only the parts that would be seen.



example 5

The nature of lip syncing is a very controlled, straight-forward process. The words are said and the lips have to match the sound being heard. The code eased the process, but in fact, there was still a good amount of tedium. For the sake of my brain and time, I was determined to limit tedium as much as possible so I began exploring another tool, randomness. The first instance where I used this tool was making a character's eyes blink. Humans blink randomly all of the time and to make animated characters seem more real they have to as well. One way to do this is to make a movie clip of a tweened animation of the eyes blinking on a loop, but loops can become monotonous and are often easy to spot. I found randomness to be an excellent way to break this monotony and that it could be easily coded. After setting up a movie clip containing one blink sequence (see example 5), you simply have a random number generated and a conditional statement testing for a specific number within an enterframe function. Once the chosen number is generated the movie clip is told to play (see code reference 3). Other randomized elements were set up similarly such as head and body movement, rainfall, car movement, a fight sequence and random dialogue selection. I found that randomness is an extremely valuable tool for animating. A tremendous amount of animation time was saved and a more believable and unique result was achieved.

Code Reference 3: This action below is set to make the eye movie play when a random number between 0 and 25 is 2.

```
//_____ Randomly Blinks
this.onEnterFrame = function() {
  Blink = Math.round(Math.random()*25);
  if (Blink == 2) {
  JLEye.play();
  JREye.play();
  JRBrow.play();
  JLBrow.play();
  }
  };
```

Audio:

Recording voice talent is a lot more than pressing record on the recording software and say talk. It was very important that the voice talent was directed and knew exactly what was needed for their particular character. During this process I found that a great deal of editing of the script was done. It turned out that it was quite essential to the success of the project to allow the voice talent to have flexibility and feel free to add to the script. All of the talent gave input and was allowed to improvise lines which was fun and added a great deal of depth to the story. The process consisted of first rehearsing and fine tuning the lines. We then took several takes of the scene. The benefit of not doing a live performance is having the ability to edit, and I can safely say that I never took a perfect take. All of the finalized scenes were a combination of the separate takes. This let me adjust the timing of the performance and use the best aspects of each take. Because the audio was recorded in a sound proof booth, once edited, I added a slight reverb to all of the dialogue to give the sound a more realistic quality.

The music selection turned out to be a harder process than I first thought. The music used in the animation ended up being short simple musical interludes between transitioning

scenes much like a TV sitcom uses music. I first conceived using music in a more cinematic manner with music playing during the scenes enforcing the mood and emotion. I found that the sitcom format was more appropriate for the project and it allowed me to easily add the element of randomness. I was able to randomly select an interlude during the transitions. In my tests I discovered if I were to randomly select cinematic music during a scene, the random selection wouldn't always be appropriate for the scene or the timing wouldn't quite match. I think it could be achieved, but on a more complex level than I was prepared to handle. The music would have to be grouped into moods much like the scenes and the selection would be limited to the appropriate group. In this case there would have to be a lot of music created. In the final solution I had seven interludes to choose from which gives quite a bit of variety. In the other case, to be effective there would have to be many selections per group and the work would increase factorially.

In terms of effectiveness, the random selection of music became much more than an interesting aspect to add to the project. The random interludes seem to operate on an almost subconscious level. The music is subtle and short and most people do not notice that it is randomized. However, because of the randomization, the users cannot pick out a pattern of the music selection. Therefore the music doesn't seem repetitious or monotonous when they rewatch. This adds to the experience in an almost mysterious way because the user isn't bothered by monotony, but doesn't know they aren't. The positive effect is also achieved with less work than permanently selecting the music, because less music has to be written. Let's say there are ten scene transitions. Each one needs to have a different interlude. If they are randomly selected, there isn't a pattern which breaks monotony and adds interest, so conceivably five interludes could do the work of ten. Another positive aspect of the method is that new music can easily be added to the bank because of its modular nature. If the music was set permanently it would limit the amount of music to the amount of scene transitions. The drawback to using music in this method is the loss of control. You no longer get to choose exactly which music will be played for a specific effect.

Project Structure:

For a web animation, the project's scope was quite large and needed to be well is structured. It was essential for the project to be as modular as possible so I could easily add new scenes to the final piece without having to edit or alter the scenes already created. Having the final piece in separate chunks would also be a more efficient way for users to load the animation in terms of download speed. Each scene was exported as a separate *swf* file which, when selected, it is loaded into a master *swf* shell. This kept each scene independent from one another so edits or bugs within one scene wouldn't effect the others. In a team scenario this method would be quite effective because separate teams could work on separate scenes which could be compiled with minimal effort.

Within each scene's *fla* file, I found it best to separate each camera shot into separate Flash scenes. This made it easier to edit than having the duration of the scene on one big time line because I could edit at any point without effecting the rest of the camera shots. I named each flash scene numerically with even numbers so if I needed to rearrange or add a scene I could do so without having to do a large amount of renaming (see example 6). As mentioned previously, I separated the dialogue of the scene into small chunks before animating which made it easy to incorporate into my editing method and gave me a good idea of the scene's pacing.

example 6

With all of the scenes of the project separated, I needed to create the shell to act as the master control. I soon found out, to keep the code accessible to all of the loaded scenes, this is where the bulk of the Actionscript would have to be written and things like player controls should be kept. Because the randomization only happened at precise points and not every scene had a variation to choose from, it was quite challenging to devise a method for randomly loading the separate scenes. The challenge came with picking a random number when there are different amounts of scenes to be selected. If there are two scene 2s you pick a number between one and two, but what happens when there are three scene 5s or only one scene 6? It took some brain racking. I needed a common variable that stored a random number, but since a random number is selected between a set of numbers I needed those numbers to be different depending on the amount of scenes to be selected. If only there was a way to organize a separate list of each set of scenes, then I could select a random number based on the length of those lists. Ah ha! Arrays do just that (see code reference 4).

Once I devised a method for randomly selecting scenes, I set up a test with the animatics. The prototype worked, but after watching the test several times, I realized that with only a

Code Reference 4: The action below consists of variables used to randomly choose which scene will be loaded. Every time the ChangeScene function is called the sceneNumber adds one. The randomNumber variable chooses a random number between 0 and the length of the current array. For example after Scene 1 is loaded and played it will call the ChangeScene function. The scene number will become 2, the array name will then be Scene2Array. It will then pick a random number based on the length of Scene2Array. Finally the MovieLoaded will be a random position in Scene2Array.

```
function ChangeScene() {
sceneNumber++;
var TheArrayName = "Scene" + sceneNumber + "Array";
var randomNumber = Math.floor(Math.random() * this[TheArrayName].length);
var MovieLoaded = this[TheArrayName][randomNumber];

reference: Scene2Array = newArray("Plot1_Scne2","Plot2_Scne2","Plot3_Scne2");
...
```

few variations of the scenes and the way the code was written, the chances of the user seeing the same scenes in the same setting was great. It was very possible that they could watch the animation again and only see one or two variations, thus lessening the project's effectiveness. If I had a large amount of scenes to select from, this would be less of an issue, but to have the maximum amount of variation I had to alter the script a bit.

By working with arrays as the storage containers for the scenes, I was able to remove items from those arrays once they were selected. This way, if the animation was replayed the scene that was previously viewed wouldn't be in the list and therefore couldn't be selected again. Of course, eventually the array would become empty so I also had to set up a reset function to rebuild the array once it was empty (see code reference 5). This posed yet another problem because initially I was creating all of the arrays with one custom

Code Reference 5: The action below is the resets the arrays when they are empty. It also removes the selected scene from the array so it cannot be selected until it is reset.

```
...
if (this[TheArrayName].length == 0 && resetting) {
  resetting = false;
  resetArrays(TheArrayName);
  sceneNumber -= 1;
  ChangeScene();
} else {
  loadBox_mc.unloadMovie();
  loadBox_mc.loadMovie(MovieLoaded);
  preload(loadBox_mc);
}
// this removes the scene selected from the array so it cannot be selected again
this[TheArrayName].splice(randomNumber, 1);
};
```

function, so once one array was empty they all reset decreasing the amount of variety. To fix this problem, I had to modify the function to only reset the array that was empty.

The naming convention of the files was also important because this was their reference for being loaded in the proper order. Each scene was named with a plot number and a scene number. The different variations of a plot's scene shared the same scene number but had a different plot number. For instance, the project has two scene fours to choose from, one from plot one named Plot1_Scene4 and one from plot two named Plot2_Scene4.

The project also includes sub-randomized elements that are selected inside a scene rather than in between the selection of the scenes. For instance, in scene one, the protagonist has a dream which is selected between two choices. The selections, which are linked movie clips located in scene one's library are chosen during its playback. As explained earlier, when developing the code for the scene selection I solved the problem of users

re-seeing randomized elements by using an array to store the references to the scenes and removing the selection made to prevent re-seeing that scene. I did not want the same problem occurring during the selection of sub-randomized elements, but the previous solution would not work the same way because the scenes are being loaded, viewed and then unloaded. If the arrays were placed within the scene they would be reset every time the scene was loaded, which would not solve my problem. As a fix, since the shell is the only file always loaded when viewing the animation, I put the sub-scene array on the shell. In the case of the dream selection, it required me to call a custom function located on the shell from scene one at the point I wanted the selected dream to be displayed. I then had to set up a return to send back the selection from that array (see code reference 6 & 7). Scene one would then place that selection on the stage. By keeping this information in the shell I was able to track selections, even ones that occurred in the middle of loaded scenes.

Code Reference 6: The actions below are located on the shell and include an array of the possible selections that could be made, a random selection based on the length of the array, a reset once the array is empty, and a return to communicate the scene that is requesting the information.

```
//_____ dream selection _____
resetDreamArray();
  function resetDreamArray() {
  delete dreamArray;
  dreamArray = new Array("Dream1_mc", "Dream2_mc");
}
function randomDreamSelect() {
  var randomNumber = Math.floor(Math.random() * dreamArray.length);
  var clipLoaded = dreamArray[randomNumber];
  dreamArray.splice(randomNumber, 1);
  if (dreamArray.length == 0) {
        resetDreamArray();
  }
  return clipLoaded;
}
```

Code Reference 7: The actions below are located on the scene which asks the shell for information. Since the scene is loaded into the shell, _root refers to the shell. The code on the shell returns the information requested. The selections are movie clips linked within the scene's library.

```
clipLoaded = _root.randomDreamSelect();
this.attachMovie(clipLoaded, "Dream_mc", 100);
```

As stated before, while writing the stories I came across many problems in terms of keeping the story cohesive. In one case the two characters, Jon and Mike, talk about another character, Cliff. Cliff's scene is randomly selected between a choice of two, so it was possible that Jon had just spoken to Cliff before Jon spoke about him to Mike. It was also possible Jon spoke to a different character. In the later scenario it would be appropriate to mention that he had just spoken to Cliff, but not in the other. To solve this problem I recorded two different lines that were similar but one mentioned that he had just spoken to Cliff. I then needed to create a conditional statement to choose the appropriate line to be played depending on whether Jon had or had not seen Cliff previously. To accomplish this, another array was created on the shell called scenesSelected. This stored all of the scenes selected during the animation in the order they were played. When the Jon's line came up a test looked at the array on the shell to see if Cliff's scene was the last scene selected. If it was or was not, it would play the appropriate line. I decided to play the line during a shot of the back of the character's heads to eliminate the need for lip syncing.

The process that was developed is the main reason for the project's successful completion. The careful organization, preparation and use of Actionscript allowed many aspects of the project to be done easily and fast. I would say that having the desire to not only do something, but finding the best and easiest way to do that something is a must for someone undertaking an animation of the scale of this project. It is also very important to see the art in not only the final aesthetics of the project but also in the whole process of its completion.

Conclusions:

Testing:

The application was tested by fifteen people: eight males, seven females from the ages twenty to twenty-seven in different settings. The test involved the subjects watching the animation once and answering a series of questions involving the over all quality of the piece. They were not told about the random selection feature of the application. They then were asked to watch the animation again and answer a second set of questions involving their response to the random selection of the scenes. All of the subjects had seen web animation before and in terms of the story remaining coherent all those tested could follow the story lines of both variations. The first section's results were quite positive with a very high percentage of subjects finding the animation enjoyable and agreeing that they would recommend the animation to their friends.

The results of the second half of the test, testing the random selection, had noticeable positive increases. 73.3% strongly agreed and 26.7% agreed that the addition of the new scenes made the animation more enjoyable. When asked if they would watch the animation again 86.7% agreed or strongly agreed on the first half of the survey. On the second half, the percentage increased to 93.3% when asked if they would watch the animation again to find more scenes. Interestingly, 33.3% of subjects strongly agreed that they would recommend this animation to their friends on the first half. This percentage increased to 40% after watching the second time with the random scene selection. The neutral response of this same question strangely, increased 6.7% and the agree response decreased 13.4% on the second half showing some inconsistencies. However, none tested gave a negative response to this question.

The highest neutral response was 46.7% when asked if they wished they had more control over the outcomes of the animation. This was a question often asked during the development of the piece. I decided that having the computer make the selections seemed more mysterious yielding a more positive experience. It seems that the subjects of this test were on the fence and didn't seem to have a strong opinion one way or the other.

About 80% were surprised to see the different scenes. By not telling the users that the scenes are randomly selected, I was trying to create a kind of Easter egg effect. Most

didn't expect to see a different animation, were surprised and would watch it again to see if any more selections existed, this could potentially add much to the over all effect of the piece. The random selection seems to trigger the users curiously, possibly keeping their attention for much longer than a traditional web animation. In terms of applications developed for the internet, this result is noteworthy because the internet is typically an environment were retaining attention is often difficult. I can also see a scenario where the users connect with their friends to find new animations. This potential buzz is also noteworthy because today the primary means of advertising for web cartoons is word of mouth.[7]

Test Results

Please watch the animation one time and answer the following questions using the scale provided:

	strongly agree	agree	neutral	disagree	strongly disagree
I have watched animation on the internet before:	53.3%	26.7%	20%	0%	0%
I found this animation enjoyable:	33.3%	66.7%	0%	0%	0%
I could relate to the subject matter:	26.7%	26.7%	46.7%	0%	0%
I would recommend this animation to my friends:	33.3%	66.7%	0%	0%	0%
I would watch this animation again:	46.7%	40%	20%	0%	0%
I would watch new episodes of this animation:	46.7%	40%	13.3%	0%	0%

Please watch the animation again and answer the following questions using the scale provided:

	strongly agree	agree	neutral	disagree	strongly disagree
I noticed there were different scenes the second time I watched the animation.	73.3%	26.7%	0%	0%	0%
I was surprised to see the different scenes:	46.7%	33.3%	13.3%	6.7%	0%
The addition of the different scenes made the animation more enjoyable:	73.3%	26.7%	0%	0%	0%
I could follow the story lines of both variations of the animation:	73.3%	26.7%	0%	0%	0%
I wish I had more control over the outcomes of the animation:	20%	13.3%	46.7%	13.3%	6.7%
I would watch this animation again to find more scenes:	40%	53.3%	6.7%	0%	0%
I would recommend this animation to my friends:	40%	53.3%	6.7%	0%	0%

example 7

Limitations:

The primary goal of the project was to use randomness but in a very controlled way to insure the story line and themes remain coherent to the audience. If it were a more chaotic approach it would be very easy to infinitely add scenes and scenarios, because you wouldn't have to be concerned with the cohesion between scenes. Because of the control, the variation between the scenes could be quite great but not infinite. This is where I can see possible limitations. As discussed earlier all of the scenarios have to remain true to the main underling story. Even if there are ten variations of every scene, though having much more longevity and depth than a traditional animation, I can foresee users eventually seeing a pattern and wanting the characters to vary passed the core story. I believe it is certainly possible to break from this limitation, but with significantly more work and planing. I found by just varying slightly from the core story increases the amount of work and thought greatly and can become quite a chaotic chore to organize.

The random selection is controlled so the viewer cannot see a random element twice in a row when replaying the animation. This is effective, but it will not work if the user hits the refresh button on their browser, or comes back to the site at a different time because the selections are reset every time the project is loaded. This is a limitation because a user could potentially go back to the site and see essentially the same scenes negating the positive effects of the random scene selection. It could possibly be corrected by using a cookie or some other more sophisticated programming technique to keep track of the scene selection outside of Flash's domain. It also become less of an issue when more scenes are available for selection because the chances of variation are increased.

Another limitation may ironically come from one of the projects strengths. By using Actionscript to streamline the work flow, the process requires not only creative talent, but technical knowledge as well. Though not extremely complex the scripting cannot be used by someone only interested in the animation capabilities of Flash. The complexity of some of the code may also cause the application to run slowly on older machines limiting the audience to those with newer machines. The use of code also limits it's distribution. While a traditional animation created with Flash can easily be transferred to film and video, my animation can only be viewed with a computer.

The scripted animation techniques are quite effective and efficient, but they do not always produce a smooth and fluid looking animation. The arm, head and eye positions are fairly jerky which I find appropriate for my piece, but those wishing to have a more tweened look to their animations may find the techniques more complex to modify and utilize. This piece could also take the idea of randomization much further. The modular design of the project would allow the depth and position of camera shots, the clothing of the characters and the setting to all be randomized.

Final Words:

Randomness can be an effective creative tool which adds interest, the element of surprise and mystery to animation produced for the internet. It seems to tap into the audiences' curious nature, keeping their attention and encouraging repeat viewing and word of mouth advertising. Because of this feature, it can serve as an alternative for what Tom Winkler, web animator and founder of doodie.com, suggests makes good content on the web; changing the content of a web site daily.[8] By adding more scenes to "Random Life" you easily change the content but in a more in-depth way because the project grows factorially. Randomness also is an excellent production tool easing the process of animation and adding realism. I found that a great deal of time was saved, and more positive outcomes were achieved by letting randomness control the characters rather than my strict direction.

The organization, planning and efficiency of the production can be directly linked to the successful completion of the project. I found that when undertaking a project such as "Random Life," The ability to find the easiest most efficient way to accomplish something while retaining the creative integrity of the project is the most important trait a director or animator could have. It is absolutely essential to be curious and intrigued with not only the solution, but with the problem and the process as well. To complete most of the tasks by one's self, it takes artistic, analytic, technical and organizational creativity and suites those who have many good skills rather than a few great skills.

"Random Life" also serves as a successful use of controlled randomness by creating interest and sparking curiosity, but remaining coherent and understandable to the audience. By making randomness less chaotic this aspect creates a comfortable and enjoyable environment for users. The testing of the project yielded positive results among its target audience and is proof positive that the use of Actionscript to streamline an simply production can be a tremendous advantage. Though successful, "Random Life" only utilizes a small portion of the potential of using randomness creatively. Hopefully, it will serve as a positive example from which others can learn from and expand on.

Endnotes:

1. Conan, Neal; Internet Animation Goes Mainstream guest Dwight Silverman; Talk Of The Nation; http://www.npr.org/features/feature.php?wfld=1753401; NPR; March 9, 2004.
2. Chapman, Matt and Mike; HomeStarRunner.com; http://www.homestarrunner.com.
3. Veitch, Joel; RatherGood.com; http://www.rathergood.com.
4. Norton, Simon; Testimony: A Story Machine http://www.myballoonhead.com/story-mach.html; 2002.
5. DiNucci, Darcy; Macromedia Flash Interface Design (Twelve Effective Interfaces and Why They Work); Macromedia Press; 2002; p56.
6. Corsaro, Sandro; The Macromedia Flash Animator; New Riders Publishing; 2002.
7. Conan, Neal; Internet Animation Goes Mainstream guest Dwight Silverman; Talk Of The Nation; http://www.npr.org/features/feature.php?wfld=1753401; NPR; March 9, 2004.
8. Corsaro, Sandro; The Macromedia Flash Animator; New Riders Publishing; 2002; p71.

Review of Literature:

Books:

Corsaro, Sandro; The Macromedia Flash Animator; New Riders Publishing; 2002.

This book emphasizes Macromedia Flash animating techniques. It includes information on optimization, efficiency, tricks of the trade and interviews with leaders within the industry.

DiNucci, Darcy; Macromedia Flash Interface Design (Twelve Effective Interfaces and Why They Work); Macromedia Press; 2002.

Examines 12 interfaces including an in-depth look at Testimony: A Story Machine which is a randomized web animation by Simon Norton. Discuses the effectiveness of and process creating the site.

Franklin, Derek / Makear Jobe; Macromedia Flash MX Actionscripting Advanced; Peachpit Press/Macromedia Press; 2002.

Step by step guide to learning advanced Actionscripting. It includes information on generating random numbers, controlling sound and creating and utilizing arrays. Very good source for learning Flash's programming language.

Makar, Jobe; Macromedia Flash MX Game Design Demystified (Official Guide to Creating Games With Flash); Peachpit Press/Macromedia Press; 2003.

Great source for creative Actionscripting techniques. It is a book geared towards gaming, but can be utilized for many different types of applications.

Radio Brodcasts:

Conan, Neal; Internet Animation Goes Mainstream guest Dwight Silverman; Talk Of The Nation; http://www.npr.org/features/feature.php?wfld=1753401; NPR; March 9, 2004.

This is an interview with Dwight Silverman, online news editor, Houston Chronicle discussing internet animation. It gives incite into this budding new genera explaining some of the successes and why they are successful.

Web sites:

Bauer, Andrej; The Gallery of Random Art; http://gs2.sp.cs.cmu.edu/art/random.

Includes art work generated randomly by a computer.

Brunson, Richard; eDork Random Poetry; http://www.edork.com/Poetry/.

Poetry that is randomly generated by users imputing words.

Chapman, Matt and Mike; HomeStarRunner.com; http://www.HomestarRunner.com.

Homestarrunner.com is a very popular animation website. This site is an excellent example of great character work and how web animation should be done.

Chevrel, Sebastien; Sketches; http://www.sebchevrel.com/sketches/.

An experimental web site that includes a Flash application that randomly sketches lines in various styles. Excellent links to other experimental Flash sites.

Dix, Alan; In Praise of Randomness; http://www.hiraeth.com/alan/topics/random.

An in-depth study of randomness. Includes mathematical explanations, uses of randomness and philosophical discussion.

Norton, Simon; Testimony: A Story Machine http://www.myballoonhead.com/storymach.html; 2002.

This award winning website is a randomly generated animation created with Macromedia Flash. Its navigation is loose and exploratory and the themes and story line are very open.

Veitch, Joel; RatherGood.com; http://www.Rathergood.com.

Rathergood.com is a popular web animation site that has an array of odd and funny animations. Some of them have been featured on television.


Multi-Timeline Randomized
Story Based Web Animation
by Jon Fahnestock
Rochester Institute of Technology