Modeling and Python Scripting in Maya

for the Animation Short *Style*

Department of Computer Engineering
California Polytechnic State University, San Luis Obispo
Gilenn Emille G. Collado

June 10, 2016

Abstract

The computer animation and graphics industry has dramatically improved over the years alongside the growth of technology. This paper discusses how that came about through a specific tool called Maya. It talks about its features and what it can create for you and the world. It also describes how scripting can simplify a job and make more of it. It highlights the effectiveness of the ability to manipulate code to create something amazing.

Introduction

Maya is a 3D computer graphics software that is often used to create 3D applications such as video games, visual effects, and of course, animated shorts and films. Although Maya is a complex tool, it is very powerful, useful, and versatile. Consequently, it has been used in many award-winning feature films including Disney's *Frozen*, Pixar's *Brave*, and Rupert Sanders' *Snow White and the Huntsman* [1].

As an avid fan of art, movies, animation, and technologies, this project idea was perfect. It integrated the four different subjects in a unique way. The intention was to learn how the animation process works, experiment in Maya, and create a animation short to showcase it all.

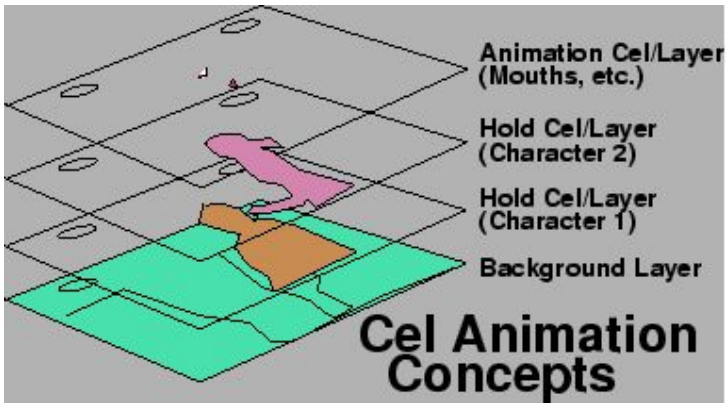
This project explores two ways of utilizing Maya and its features. The first is simply using the tools Maya already comes with and the other is scripting in either the Maya Embedded Language (MEL) or Python. For this project, the modeling and rigging of the character were both done using Maya's tools only while the animating and producing particle simulation was scripted in Python.

Users with background in computers, especially coding, will have an advantage. Scripting can get complicated if one does not understand what each line of code means. The ability to know what functions are and their purpose is most important because it will allow manipulation and therefore, produce a much better project. Since there is very little resources currently available not only for Maya itself, but also in scripting, this task would be difficult without the knowledge experience. With that said, however, it is still possible for anyone to do it. With some practice and time, just like taking coding classes in the different languages available to us, doing projects, and experimenting in class, this can become anyone's hobby or even, career.

Related Works

In this section, a few of the many evolutions of animation and technology is discussed. It briefly explains each of the process' or tool's attributes and their impact to the industry.

Figure 1: Cel Animation [3]



Traditional Animation

Before computers, each frame of an animation was drawn by hand. Eventually, this turns into a story reel, which is combined with a soundtrack and any necessary voice recordings. After editing and consulting with the director, the

animators will begin drawing a certain number of "key drawings" for each scene. The number is dependent on how the animator wants to show the point of that scene. Then, each drawing is transferred onto a sheet of plastic, also known as cel. Once that is finished, they take a picture each plastic sheet with a special animation camera and splice them into a Leica reel [2].

Notably, this process takes a lot of time and effort. Thus, it had a lot of limitations and was in definite need of improvement.

Skeletal Animation

One of the more commonly used animation processes today is skeletal animation. It is a technique that divides a character into two parts: the skin, also known as mesh, and bones, also known as the skeleton. Each bone is assigned a vertex, which is then assigned to a specific matrix of position, scale, and orientation. All of which are dependent on the movement an animator wants to create [4]. Then, each bone is matched to the skin. Thus, giving it more flexibility and freedom compared to traditional animation in regards to movement and storyline and impact the success of that movie, videogame, or special effect. In other words, this process basically mimics how a person's skeleton and skin work together.

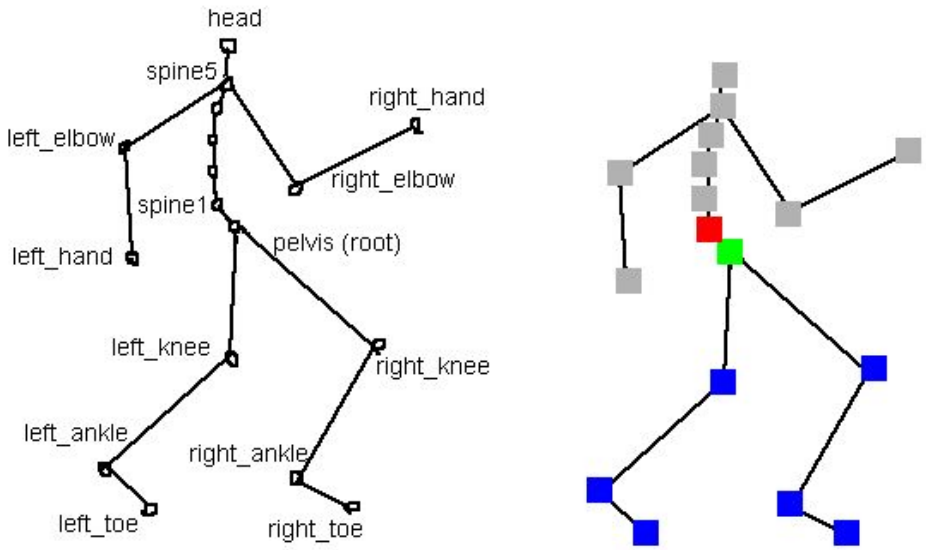


Figure 2: Example of Assigned Bones to a Character [4]

Open Graphics Library, or OpenGL, is a great tool for this type of animation. Once a model has been constructed, one can use OpenGL, a library created specifically for animation and graphics, to build its skeleton and skin by defining each bone and their matrices. Despite how good skeletal animation look on the screen, it is not exactly accurate as it does not account for

muscular and skin movement, which is not exactly bad a thing. It just depends on every animator. Other than that, it is another great accomplishment in this industry.

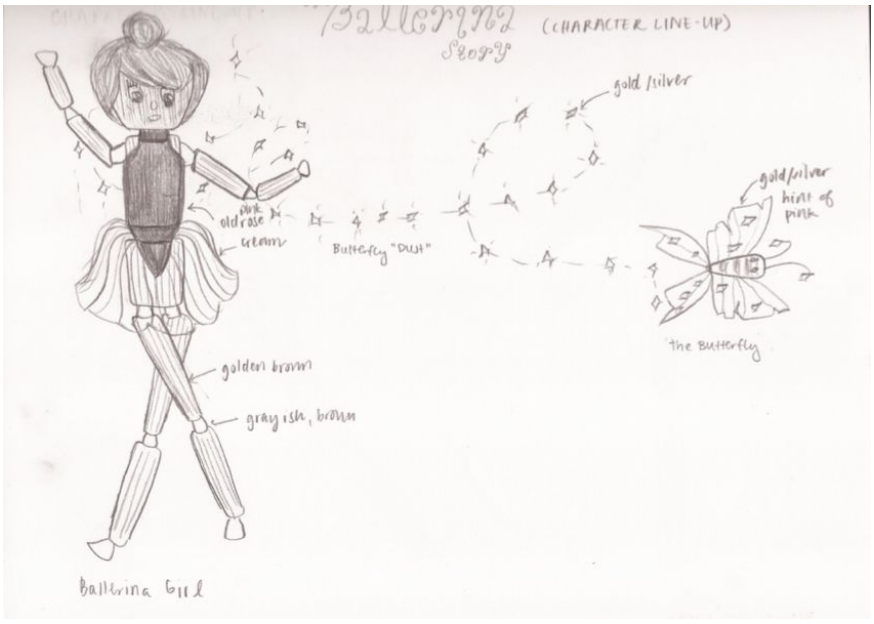
Blender

Due to the advancements in technology, specifically in modern GPUs, there are many open-source 3D computer graphics software that allow users to model and animate characters, like in Maya. One of them is called Blender. It is very similar to Maya and can be viewed as either better or not as good. It is different for everyone. The main differences is the video editing feature, the cost, because it is free unlike Maya, and the complexity, because Maya is definitely a lot more complicated and meant for bigger, more advanced productions [5].

On the contrary, Blender's features include exactly what Maya offers such as sculpting and rigging tools, UV and camera manipulation, library extensions, and particle simulations. More in depth information and downloads can be found on their website [6].

Design, Development, and Production Stages

This section discusses the different stages that were involved in creating this specific animation short. These stages were highlighted because of their significance and were modeled after the class, Digital 3D Modeling and Design, taken during the creation and production of this project. There could definitely be more or less stages than what has been done here; they are unique to every application.



Stage 1: Storyboarding

The first step to any type of design project is to make a plan. In the computer animation industry, this is called storyboarding. This is an important part of the process because not only does it allows you to outline and visualize your story shot by shot, but also helps you

strategize how to approach the subsequent stages [7].

Figure 3: Original Character Lineup

The original storyline for this animation short included a butterfly and a ballerina in her jewelry box. However, with limited time and experience, only the ballerina character was fully developed and used.

Another change made during storyboarding was the total length of the animation short. In the beginning, there were more shots that were supposed to be included. Hopefully though with more time and experience, they could eventually be added. Nonetheless, storyboarding made editing and cutting out scenes easier because each shot was pictured.

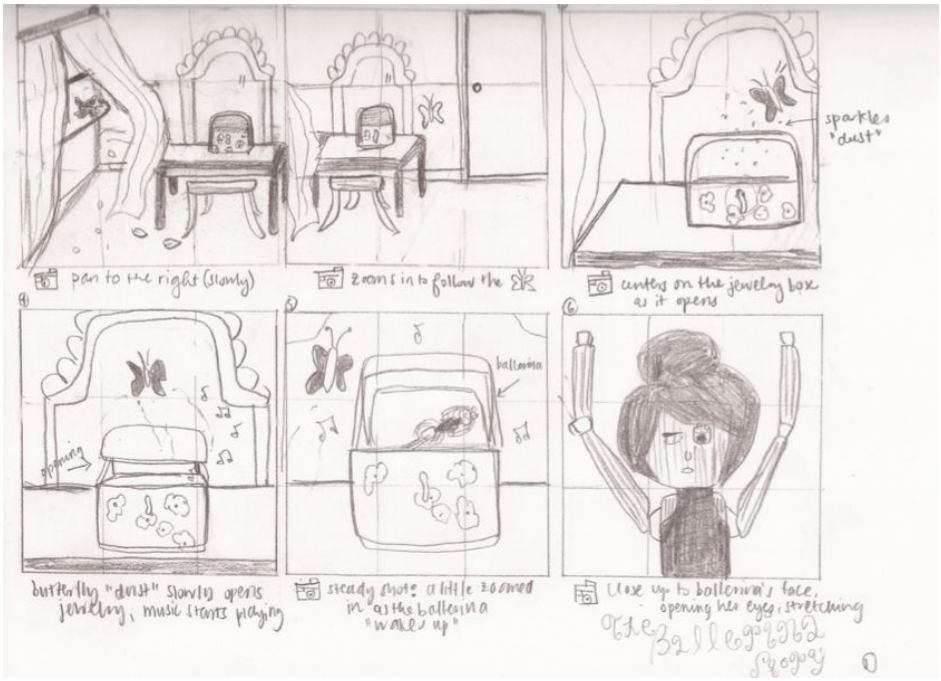


Figure 4: Beginning of Original Storyboard

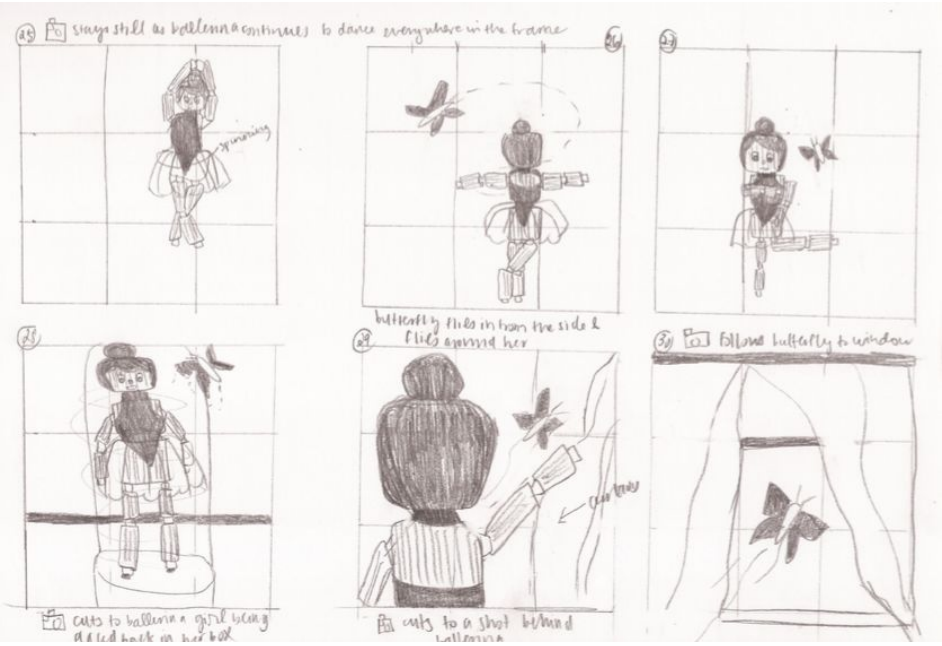


Figure 5: End of Original Storyboard

Stage 2: Modeling

Once a storyboard has been put together, the next stage is modeling. The first step in this stage involves uploading the character's sketch onto Maya as reference for the skeleton of the model. For this project, the front and side views of the character were used to build the ballerina. That way, the full 3D effect could take place.

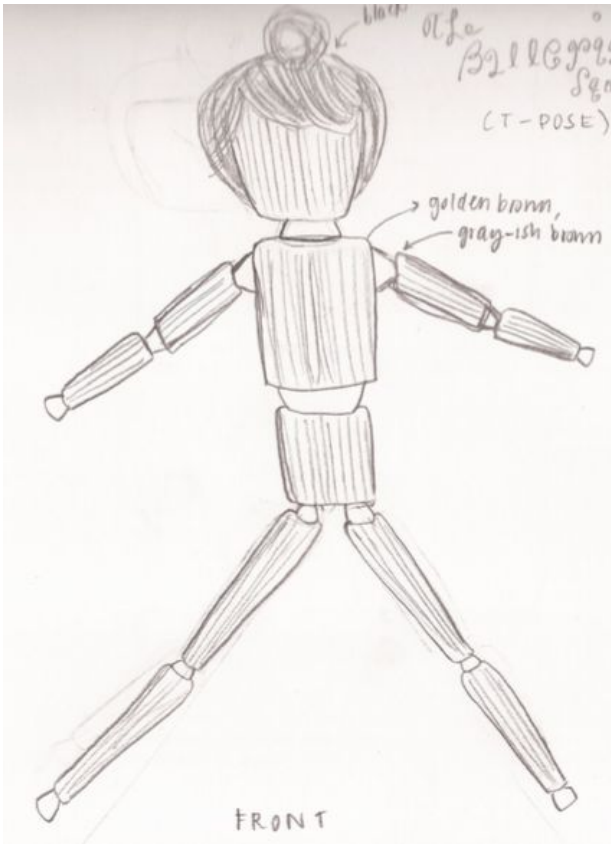


Figure 6: Front View in T-Pose

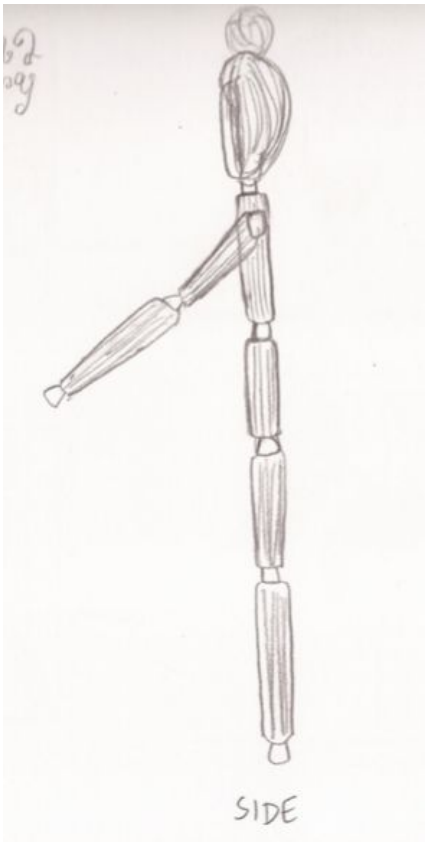


Figure 7: Side View in T-Pose

Then, in Maya, both the polygonal and non-uniform rational b-splines (NURBS) modeling features were manipulated to create the model's body. Polygonal modeling uses primitive shapes like cubes, spheres, and cones to give a character the proper generic, geometric look. They can be modified by scaling and rotating. NURBS modeling, on the other hand, is the complete opposite because it allows customization using the polygons for more natural and smooth shapes. It incorporates sculpting tools such as beveling and pinching [8].

After a few design reviews and learning more of what Maya can do, some changes were made to the ballerina model. Both polygonal and NURBS modeling were used in the process.

Figure 8: Front View of Final Model

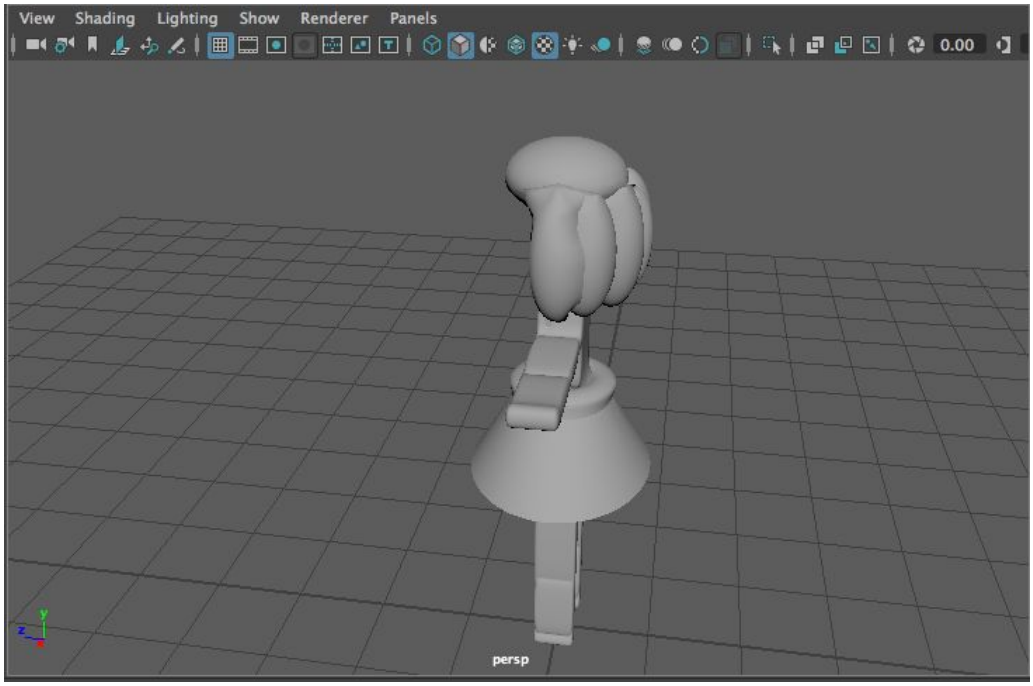


Figure 9: Side/Back View of Final Model

Stage 3: Lighting, Texturing and Applying Materials and Colors

When the modeling stage is done, the next is choosing the lights, colors, materials, and textures to use on the character. Maya offers a great a selection of these. It includes patterns like wood,

checkers, and water as well as various types of lights i.e. spotlights, volume lights, and area lights.

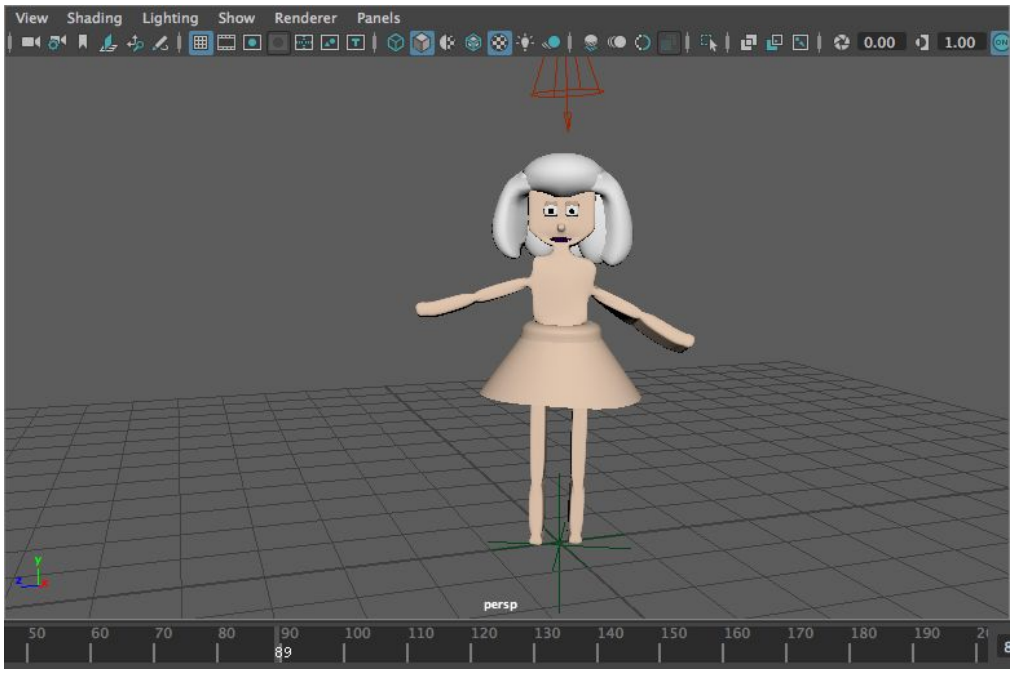


Figure 8: Front View of Final Model with Base Colors and Lights

Figure 9: Side View of Final Model with Base Colors and Lights

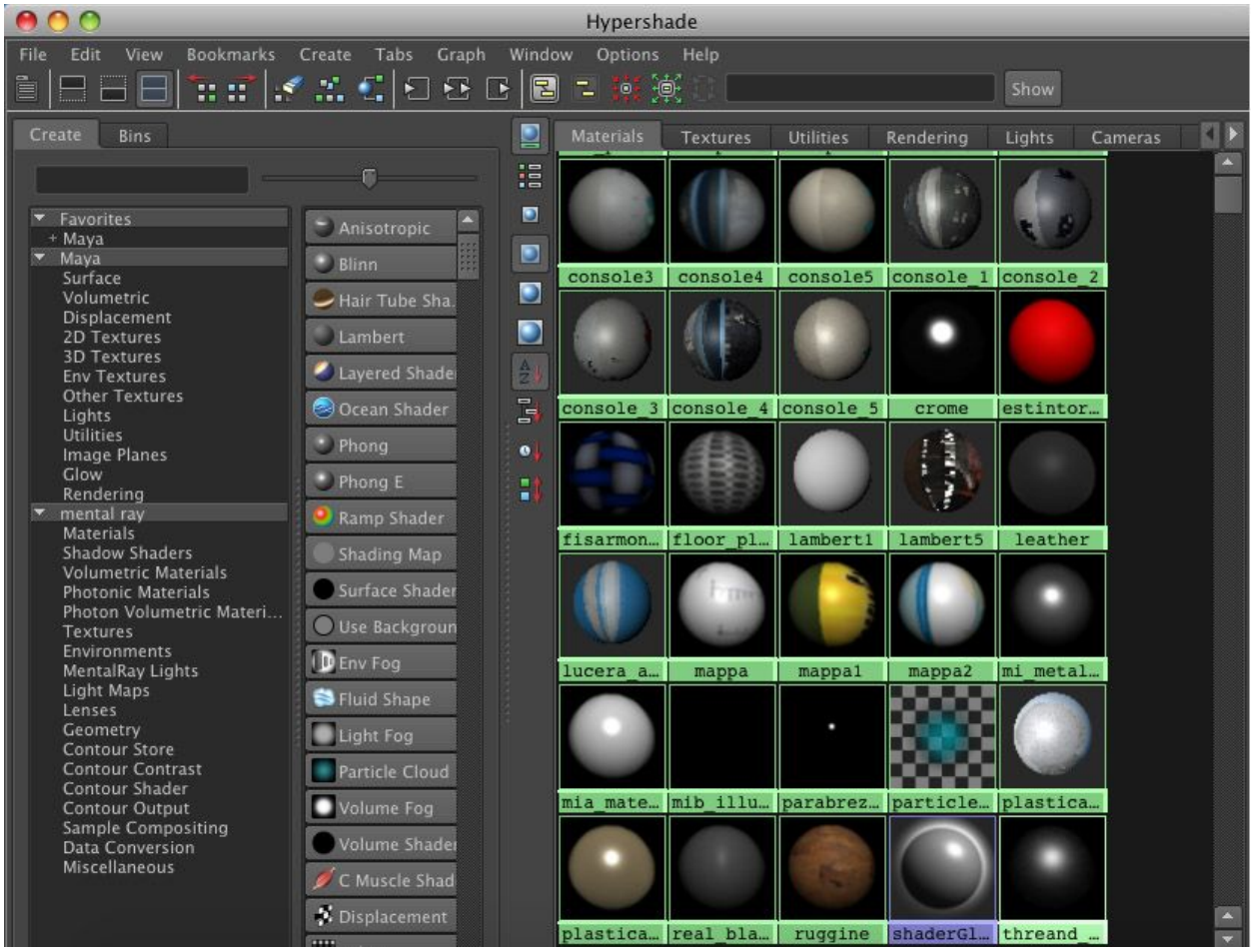


Figure 10: Different Textures and Materials in Maya [9]

Additionally, it provides several textures such as Phong, Lambert, and Blinn that enhances not only the character but also the colors and materials that is added with them. This part of the process is probably the most simple but can take a lot of time due to the many options.

For this project, many different combinations of materials and textures were applied until one was found that fit the vision for the story the most. Experience and time should also take into account since each of these materials and textures can be manipulated to so much greater.

Figure 11: Close Up Shot of the Final Ballerina

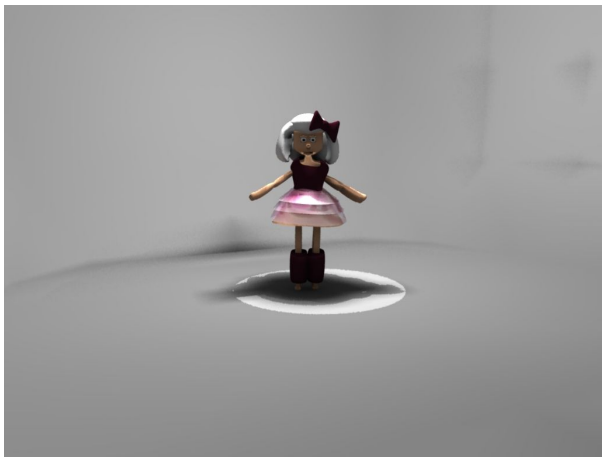


Figure 12: Far Shot of the Final Ballerina

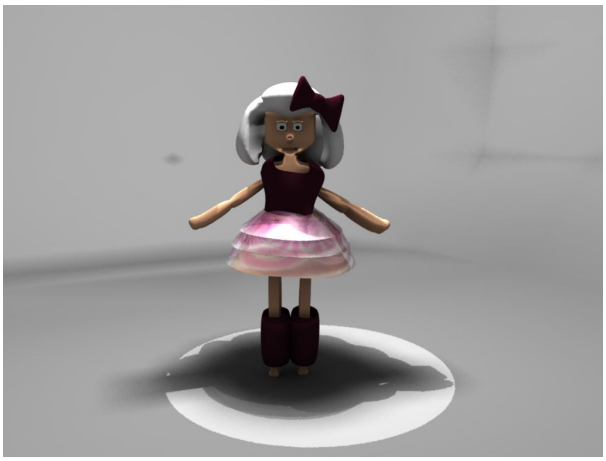


Figure 13: Full Body Shot of the Ballerina

Stage 4: Rigging

For this stage, it takes some planning to figure out which parts of the model's body to move. The story and reality of a real body must taken into account. Otherwise, it will be ineffective and look unflattering. Rigging in Maya is a lot like skeletal animation, except without the code. Maya has a very easy to use, flexible tool called HumanIK, which builds the model's skeleton and assigns the individual weights and scales for the bones. These values can be manipulated, depending on the animator and what he or she is trying to.

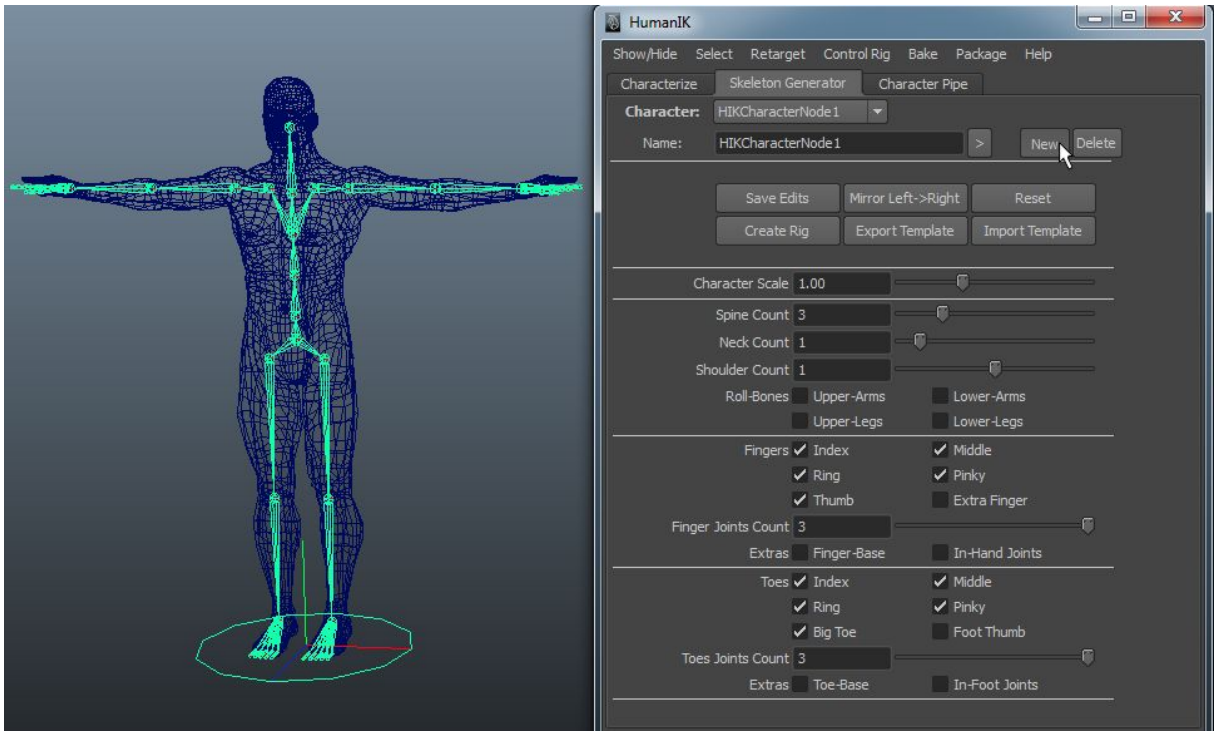


Figure 14: Example of the HumanIK Tool in Maya

Furthermore, this tool lets animators rig every part of the body, including the joints in fingers and toes. It is not a complicated part of Maya, but can take some time and practice to master and come up with excellent, natural-looking rigging.

Stage 5: Animating

Unlike the previous stages, this part of the process involves the scripting tool Maya offers. Not only can you customize existing features of Maya but also make up your own. The animating stage for this specific project requires a lot more versatility and include some particle simulation. Even though there is a tool called nParticles in Maya. There was only so much one could do with what was learned from the available resources that demonstrate its use. Thus, there would a limit to just the basic simulations. So, it made sense to write scripts in Maya's Script Editor in Python to allow tailor-made features, given the experience and materials for self-teaching available. All of which are listed in detail in the Appendices section.

There were a total of three scripts used from a Youtube channel called "Autodesk Scripting and SD Learning" and a fourth one that was derived from two of the three. Each were done separately to allow experimentation of what this specific script could do, other than what was said in the tutorial. So, it was definitely good to have a coding background to understand the different functions used, the syntax, and what the process is actually doing.

The first script concerned the random instances of an object aspect of the animation. In the figure below, there are pink particles floating around the ballerina.

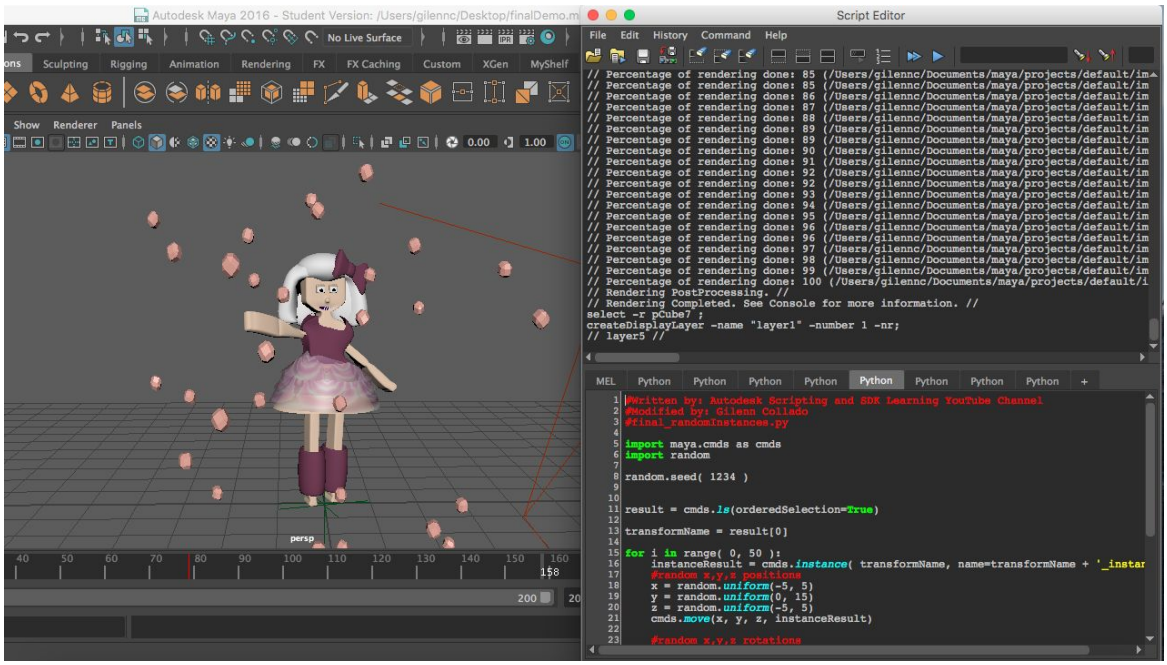


Figure 15: Ballerina with Her Particles Alongside the Script Editor

This script, listed in Appendix A, mimicked exactly what the nParticles could do and generated a modified version of cubes in different sizes at different locations. Rather than using Maya's tools and doing each task i.e. create the main particle look, generate various sizes, copying them, and adjusting each particle's position separately, the script can do it all at once. This is important because this allows a user to expand their design and give more freedom with what they can come up with. It also saves time and resources. I can easily generate thousands and thousands of particles in an instant.

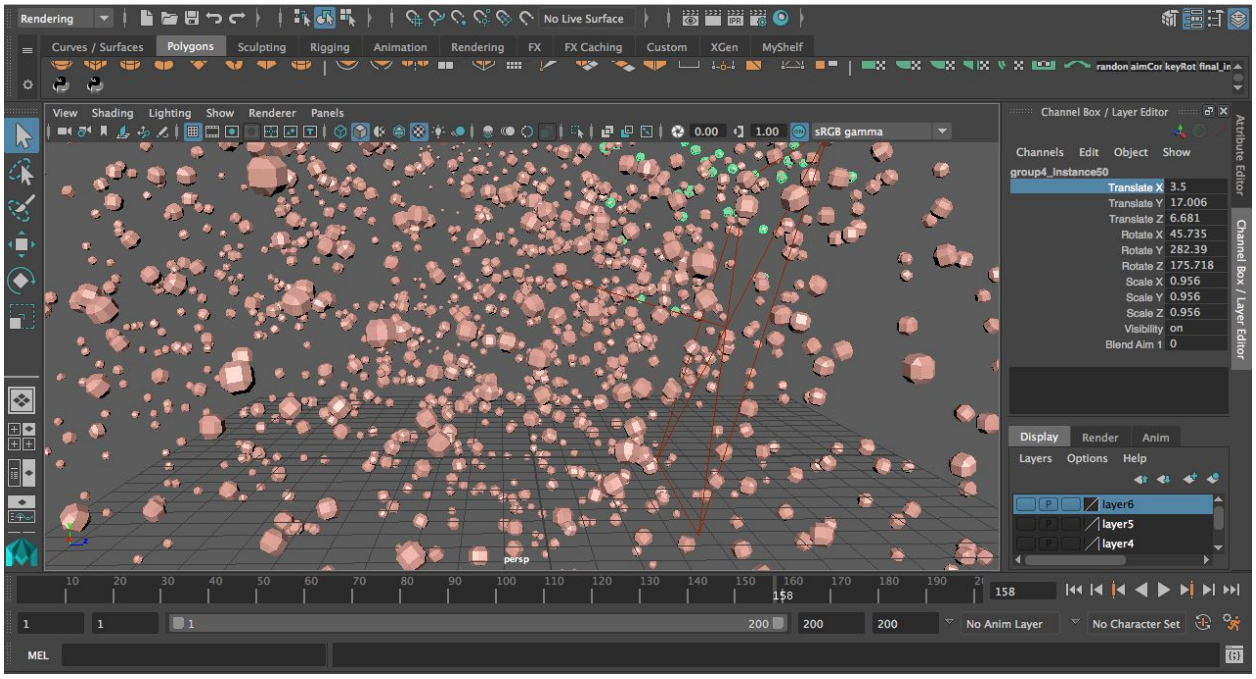


Figure 16: Example of Instancing Thousands and Thousands of Particles

The second script is responsible for the shielding effect of the particles. This is similar to how electrons and the nucleus attract and repel each other, where the electrons are the particles and the nucleus is the ballerina. The particles will move alongside the ballerina and then move away when she gets a certain distance close to that particular particle. With my experience and knowledge of Maya, as of now, this can only be done through script, which can be found in Appendix B. No features can currently do with this does.

The third script detailed in Appendix C controls the animation part of the sequence. The particles rotate around the ballerina as she dances across the floor. Like the first script this can be done through a tool, using the Animation features of Maya, which is what was used for the ballerina's dance movements. Like the first script, however, this has to be done in multiple and separate steps. Consequently, this takes more time and more effort. Therefore, the third script is especially helpful with time-constrained and limited resources type of projects.

The fourth and last script is a combination of the third and last scripts. Although they could have been kept separate, it was discovered that they worked better being together. It also eliminates another step from the process. The first script, instancing, could definitely have been added to the final script. However, it was kept separate since it dealt with a different idea while the other two were practically the same.

Even though all the scripts are written in Python, the functions were still built into Maya. For example, a function built in Maya is polyCube. It is generated to create a cube object in the middle of the screen with the default option to relocate its position and have other options to scale and rotate it. Another example is the pair cutKeyFrame and setKeyFrame. Normally, in Maya, this takes clicking on a key on the timeline to start, assigning the scheduled animation, and clicking on another key to indicate the end time. So, with the scripts, there is less clicking and moving around involved. Therefore, more intuitive and promising to create a success animation.

This is significant because it can showcase a user's ability to manipulate code to do less work with the same amount of product, if not more. This simply show how the advances in modern technology has impacted the world of computer animation and graphics positively. Without the script editor, this short animation would not have been possible in the time frame that is given.

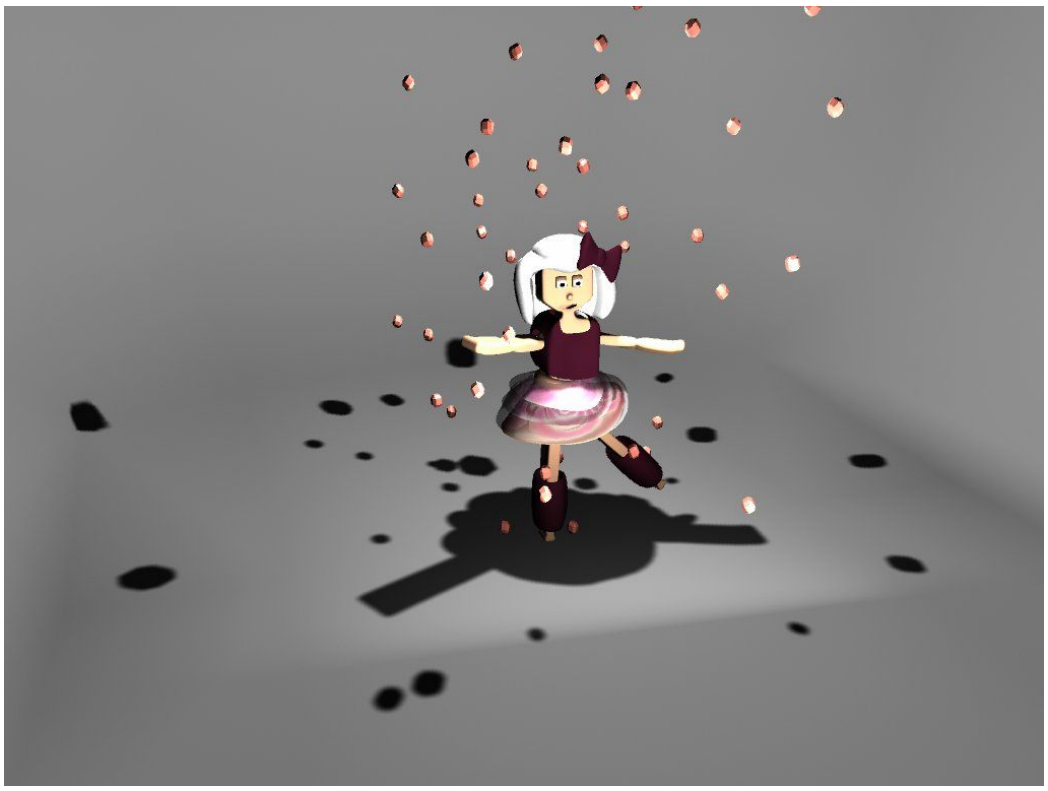


Figure 17: Ballerina Model with Particles - Angle 1

Figure 18: Ballerina Model with Particles - Closeup

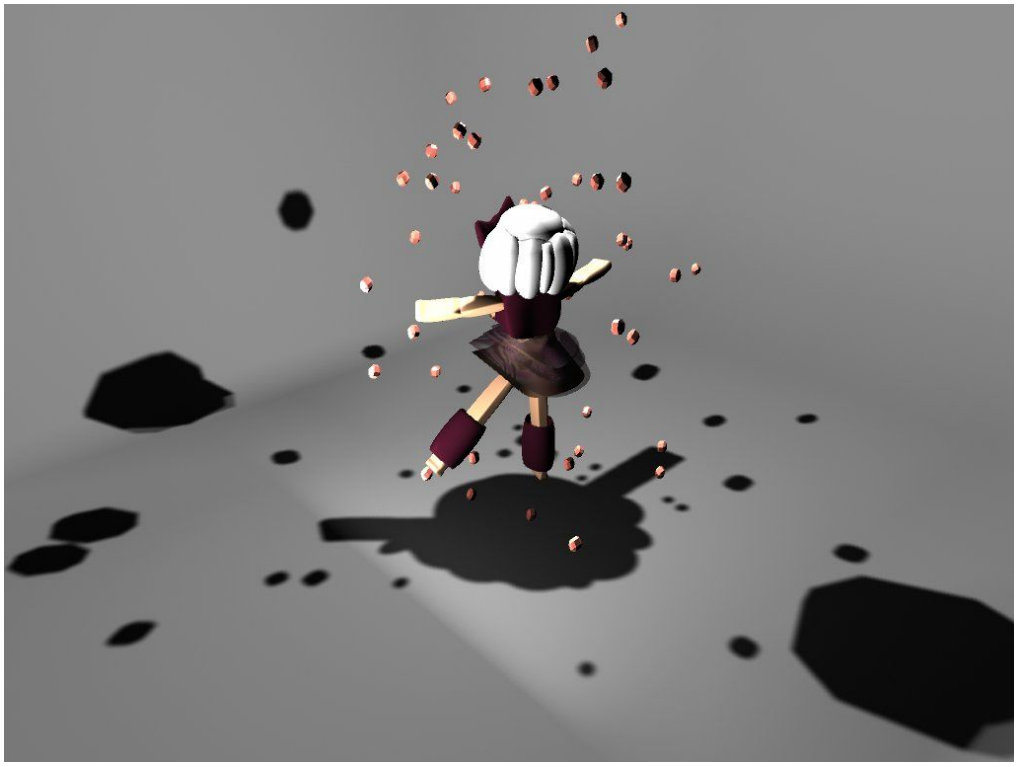


Figure 19: Ballerina Model with Particles - Back shot

Figure 20: Ballerina Model with Particles - Angle 2

Stage 6: Rendering

This stage probably took the most time because it is influenced by the number of frames per second and number of scenes the animator decides to use, which are both affected by how smooth he or she wants the flow of the story to be.

For this project, 240 frames per second was selected so that there would be more options for each scene. It definitely took a long time because there were different angles and poses for the ballerina model to capture.

The steps for certain part of the project involves Maya's tool, Batch Render. Basically, it lets the animator choose the appropriate settings i.e. number of frames, camera angle, type of output file etc. Then, all at once, it produces each frame, one by one, that it has been instructed to do based on the settings--whether that be 240 frames for one second or 240,000 for ten seconds.

Stage 7: Editing

The final stage of this entire process is editing. It involves using a video editor software like Adobe Premiere and Final Cut Pro. For this project, however, the chosen software is iMovie,

mainly because of previous experience and its simplicity. Unfortunately, there was not a lot of video editing opportunities to showcase but it does the job of producing a quality animation short. It took a bit of time to select the best frames in the hopes to avoid making the video choppy or too long. Besides that, a background music, title, and credits were added to the final product.

Future Works

Because of limited time and experience, the animation short only had one character and is very brief. In the near future, I would like to take some time to improve my current character, develop more characters, improve the particle simulations, add more to the storyline, and extend it altogether. In addition, I would like to create an animation series built on this project alongside other content that I can come up with as part of my post grad career in the computer animation and graphics industry. Lastly, I would also want to take what I have learned these past two quarters, regarding computer animation and graphics, and apply it to designing and developing tools to make it doable for inexperienced aspiring animators to make their very own pieces of art.

Results and Conclusion

This project definitely took a lot of time and effort to come together. I could only wish to have had more time, discipline, and experience to develop a more solid animation short. However, even after all the delays and plenty of self-teaching, I am fairly satisfied with what I was able to put together. Taking supplementary classes in both computer animation and Maya definitely helped. I could not have imagined where I would be if I had not. It was actually really interesting and inspiring to see the code behind Maya through the Introduction to Computer Graphics class I took concurrently. I would like to do more work in these two fields in the near future.

As for the results of this project, I was able to fully develop a character and produce a simple animation with a particle simulation that demonstrates the two ways Maya can be manipulated to create applications: its accompanying tools and scripting features. Both aspects were hard to do, especially in a short period of time. There were just very limited resources available with the Maya 2016 version. Yet, they both are also very rewarding because now, I know the different stages it takes to make my own animation, what the process is, and everything else along the way. I could definitely go a long way with this knowledge alone.

This project is a testament to how people inexperienced in art, movies, animations, and/or technologies can assemble their very own video games, visual effects, or films at home. This shows how advanced our industry has gotten and the many opportunities for anyone to grow as an artist and animator. This is important to note to hopefully encourage those who are simply curious or want and are looking to try something brand new and different.

Acknowledgements

Zoë Wood
Senior Project Advisor
Introduction to Computer Graphics Professor
California Polytechnic University, San Luis Obispo

Enrica Lovaglio Costello
Digital 3D Modeling and Design Professor
California Polytechnic University, San Luis Obispo

Music by http://www.bensound.com/: "Memories"

Modeling and Python Scripting in Autodesk Maya 2016

Modified scripts from the Autodesk Scripting and SD Learning YouTube Channel:
https://www.youtube.com/user/ScriptingSDKHowTos/about

References

[1] Animation Mentor. "5 Reasons Why 3D Animators Should Known Autodesk Maya."*Animation Mentor*. Next Education LLC, 11 Sept. 2014. Web. 1 June 2016.

[2] "Traditional Animation Process." Web log post. *Divyen Blog*. Blogspot, 15 Jan. 2015. Web. 1 June 2016.

[3] "Cel Animation Model." *Cel Animation Model*. Automanga, n.d. Web. 05 June 2016.

[4] "Skeletal Animation." *OpenGL*. OpenGL, 03 Mar. 2014. Web. 05 June 2016.

[5] "Where Blender Functionality Is Better Than Maya's." *Plural Sight*. Digital Tutors, n.d. Web. 05 June 2016.

[6] www.blender.org

[7] GoAnimate. "What Is A Storyboard And Why Do You Need One?" *What Is A Storyboard And Why Do You Need One?* GoAnimate, 03 Dec. 2015. Web. 01 June 2016.

[8] "10 Basic Concepts To Understand Maya & The 3D Environment | Maya 101 | Learn Maya 3d | Polygonal Modeling vs NURBS Modeling." *M5 Design Studio*. M5 Design Studio LLC, 30 Aug. 2011. Web. 08 June 2016.

[9] *LuGher 3D*. LuGher 3D, n.d. Web. 05 June 2016.

Appendices

Parts of the follow code were obtained from the YouTube channel called: Autodesk Scripting and SD Learning Channel. They were modified to accommodate this project's design and objectives. The final animation short produced used three scripts that were eventually cut and manipulated into two.

Appendix A

One for generating random instances of an object, similar to what the nParticles tool in Maya, can do.

```
#Written by: Autodesk Scripting and SDK Learning YouTube Channel
#Modified by: Gilenn Collado
#final_randomInstances.py

import maya.cmds as cmds
import random

random.seed( 1234 )

result = cmds.ls(orderedSelection=True)

transformName = result[0]

for i in range( 0, 50 ):
    instanceResult = cmds.instance( transformName, name=transformName + '_instance#' )
    #random x,y,z positions
    x = random.uniform(-5, 5)
    y = random.uniform(0, 15)
    z = random.uniform(-5, 5)
    cmds.move(x, y, z, instanceResult)

    #random x,y,z rotations
    xR = random.uniform(0, 360)
    yR = random.uniform(0, 360)
    zR = random.uniform(0, 360)

    cmds.rotate(xR, yR, zR, instanceResult)

    #random x,y,z scales
    scaling = random.uniform(0.3, 1.5)
    cmds.scale(scaling, scaling, scaling)

instanceName = instanceResult[0]

bevelResult = cmds.polyBevel(at=180, oaf=1, ws=1, name=instanceName + '_bevel#')
```

Appendix B

This second modified script allows the newly generated random instances of an object to yield at another object. Thus, giving a shielding effect.

```
#Written by: Autodesk Scripting and SDK Learning YouTube Channel
#Modified by: Gilenn Collado
#final_aimAtConstraint.py

import maya.cmds as cmds

selectionList = cmds.ls(orderedSelection = True)
if len (selectionList) >= 2:
    targetName = selectionList[0]
    selectionList.remove(targetName)
    for objectName in selectionList:
        cmds.aimConstraint(targetName, objectName, aimVector=[0,1,0])

else:
    print 'Please select two or more objects.'
```

Appendix C

This third modified script creates the animation rotation between two objects. In this project, I selected the ballerina to be the main object and the instances of another object to do the rotation.

```
#Written by: Autodesk Scripting and SDK Learning YouTube Channel
#Modified by: Gilenn Collado
#final_keyRotation.py

import maya.cmds as cmds

def keyFullRotation(pObjectName, pStartTime, pEndTime, pTangentAttribute):
    cmds.cutKey(pObjectName, time=(pStartTime, pEndTime), attribute=pTangentAttribute)
    cmds.setKeyframe(pObjectName, time=pStartTime, attribute=pTangentAttribute, value=0)
    cmds.setKeyframe(pObjectName, time=pEndTime, attribute=pTangentAttribute, value=360)
    cmds.selectKey(pObjectName, time=(pStartTime, pEndTime), attribute=pTangentAttribute)
    cmds.keyTangent(inTangentType='linear', outTangentType='linear')

selectionList = cmds.ls(selection = True, type = 'transform')

if len (selectionList) >= 1:

    startTime = cmds.playbackOptions(query=True, minTime=True)
    endTime = cmds.playbackOptions(query=True, maxTime=True)

    for objectName in selectionList:
```

```
        keyFullRotation(objectName, startTime, endTime, 'rotateY')

else:
    print 'Please select one or more objects.'
```

Appendix D

This final script is a combination of the shielding effect and animation rotation.

```
#Written by: Autodesk Scripting and SDK Learning YouTube Channel
#Modified by: Gilenn Collado
#final_script.py

import maya.cmds as cmds

def keyFullRotation(pTargetName, pObjectName, pStartTime, pEndTime, pTangentAttribute):
    cmds.cutKey(pObjectName, time=(pStartTime, pEndTime), attribute=pTangentAttribute)
    cmds.setKeyframe(pObjectName, time=pStartTime, attribute=pTangentAttribute, value=0)
    cmds.setKeyframe(pObjectName, time=pEndTime, attribute=pTangentAttribute, value=360)
    cmds.selectKey(pObjectName, time=(pStartTime, pEndTime), attribute=pTangentAttribute)
    cmds.keyTangent(inTangentType='linear', outTangentType='linear')
    cmds.aimConstraint(pTargetName, pObjectName, aimVector=[0,1,0])

selectionList = cmds.ls(selection = True, type = 'transform')

if len (selectionList) >= 1:

    startTime = cmds.playbackOptions(query=True, minTime=True)
    endTime = cmds.playbackOptions(query=True, maxTime=True)
    targetName = selectionList[0]
    selectionList.remove(targetName)

    for objectName in selectionList:

        keyFullRotation(targetName, objectName, startTime, endTime, 'rotateY')

else:
    print 'Please select one or more objects.'
```