2D Animation Using 3D Tools

By Michael J. Findler, Travis Samford

Abstract

Expressiveness is a major part of hand-drawn animation. The techniques used by traditional animators allow for artists’ intentions to be fully expressed, which is fundamental for stylized animation. The game industry largely focuses on photorealistic rendering techniques for modern game development, but is lacking in 3D tools for stylization. Creating better tools and techniques that allow studios to mimic traditional animation will allow for more expressive work. This paper will be covering some techniques that allow for an artists’ intentions to be accurately represented when creating game assets using Unity3D. The techniques covered include creating dynamic character outlines and shadows.

Introduction

The animated cartoons of the 1930s were highly stylized works that allowed for an artists’ intentions to be fully realized. The way these cartoons are drawn and how they are animated is a major factor in creating this stylized appearance. This style is very expressive but also labor intensive. If there were tools to get the same effect using 3D models, more studios may pursue this style for their projects.

The game industry has become obsessed with photorealism. Very few games try to replicate an expressive 2D style and focus mainly on accurately representing the real world. As a medium, games are creatively limitless, and it is a waste to not explore more imaginative, less realistic worlds and styles. Giving artists more control over animations and design decisions will allow them to make models with more character. The goal of this project is to replicate the 1930s style of animation and art. This will be accomplished by applying several non-photorealistic rendering techniques, and a custom workflow for creating assets in Autodesk Maya and importing them into Unity3D [1].

Cartoons generally have silhouetted edges around objects in the scene. This allows the viewer to easily distinguish the objects. The first step in matching this style will be to generate an outline around the 3D models. To accomplish this, a custom shader is created. This shader takes two passes. First, front face culling is used, and the faces of the object are expanded out in the direction of their surface normals. This first pass is then rendered in the color that the outline is going to be. In the second pass, back face culling is applied, and the object is rendered on top of the first pass at its original size. This creates a small overlap that becomes the outline of the object [2].

The characters are also followed around by a circular shadow under their feet. These shadows will change in size based on the distance from the ground the objects are. These shadows are either black, or change the color underneath them in a consistent way. To implement this feature, a Unity projector object can be placed above any objects that need to cast a shadow. The projector takes a special shader and projects a material onto the objects in front of it. Because of the nature of these shadows, they can be implemented using a cel-shader, where there is no diffusion: only shadowed or not. Once the projector is added, the character can be placed on a special layer and the projector can be configured to ignore that layer. These characters do not have any self-shading or specular highlighting themselves; It’s important to add them to this layer so that they are not affected by the projector [3].

Discussion

To create the outline effect, a two pass shader was created. The first pass is responsible for the outline of the object. In the first pass, the front face culling is turned on, causing any surface facing the camera to be removed. A 3D object consists of points, edges that connect points, and surfaces that connect edges. Each surface has a vector called a normal vector. The normal is a vector perpendicular to the surface, and signifies the front of the surface. To create the outline the first pass expands the surfaces along their normal vector increasing the size of the 3D object. The object is then rendered to the screen given a color for the outline. In the second pass, the back face culling is turned on, and the object is drawn at its normal size. The overlap from the first pass becomes the shadow around the object. To apply this shader in unity the user must create a material, select the outline shader, and then attach it to the object that needs an outline.

For an object shadow, a Unity Projector is used [3]. It is attached to the parent of the object and set directly above the object facing downwards. Once the projector is created it is set to ignore the layer that the object is on so the shadow is only cast on the ground. The cel-shading effect is accomplished by projecting a ramp texture with only values, shaded and unshaded, directly below the object [4].

Method

Subjects will be given four games to play. The games will consist of objects that have either a shadow, an outline, both, or neither. The players will have to complete the games in the allotted time to demonstrate which is the best highlighting technique. To complete these games, subjects will be given the objective to click on the objects of interest while avoiding clicking on anything else. If the subject clicks the correct objects, their score will increase. If they misclick, their score decreases. There is also a multiplier that increases by one for each correct click, and resets to one when the wrong object is clicked. The multiplier has a timeout of 5 seconds which will cause a reset to one.

Constants

The order of the levels and their difficulty do not change from test group to test group. The targets for each level are a constant, as is the control system, and the graphics.

Variables

The order in which the users face these levels will be changed. Some will see levels without the shaders first, and others will start with them. There may also be mixed levels where some things are shaded, and others are not. The line thickness of the object can grow and shrink. The amount of points or the multiplier may increase depending on how quickly the highlighted object was seen and interacted with. Which object is highlighted depends on the level order.

Measuring

The goal is to see if user reaction time was heightened by the outline, shadow, or both. The user’s experience will be measured using a QUIS (Questionnaire For User Interaction Satisfaction) survey, and the user’s interactions with the program will be tracked using Unity analytics. How long was the item on the screen before it was clicked? Once an item is clicked, count the number of other items on the screen. How many items were on the screen when the clicked item appeared? How close was the clicked item to the last clicked item? If there is a timeout feature, how many of the relevant objects were missed? Was the mouse closest to the object that just appeared on screen when it was clicked?

Apparatus

The test subjects will be broken into groups of 8 and will be playing with different combinations of effects on 3 different lab computers. The lab computers are Dell precision towers 7910 with Dell U24156 monitors. The program was built using Unity 2017.1. Users interact with the program using a standard mouse and keyboard. The players will be given a total of 5 QUIS forms to fill out. They will do a per level QUIS and a QUIS for their final thoughts.

Results

Reviewing the data, the length of time a subject played had a much bigger effect on their performance than the effects. The combination of effects had little effect on the results. For each level a subject played, the number of correct targets selected increased. Level one and level four had a large increase showing a statistical difference in the subject’s ability by the end of the game.

The gender of the subjects had a large effect on how correct targets were selected as well. The male subjects selected more than the female subjects.

The gender also effected the precision of the player. The female subjects clicked less but had less miss clicks than their male counterparts. This implies that while male subjects clicked more correct targets female players took their time and tried to make their clicks count.

The time that the targets were alive were significantly longer for female subject. This implies that males have a higher reaction time.

Conclusions

Using a combination of a vertex shader and a Unity projector, two non-photorealistic effects were achieved. A two-pass vertex shader was used to create an outline on objects. The projector was used to create a shadow underneath an object.

To test the visibility of these effects, a game was created with four levels. Each level had a different combination of the effects. Subjects were then asked to play the game and answer questionnaires about their experience.

Statistically there was not a significant difference in visibility between effect types.

Future work

In future work, we would like to explore dynamically generated objects. For example, creating smoke puffs or speed lines when a character jumps or runs.

Creating tools to handle the exaggerated animation of these objects is needed; following the 12 principles of animation tools can be used to achieve the effects of animation [5].

Integrating these effects into a workflow that includes a 3D modeling program would reduce the workload and further streamline working with non-photorealistic techniques.

# References

|  |  |
| --- | --- |
| [1] | D. C. a. J. L. Mitchell, "Non-Photorealistic Rendering with Pixel and Vertex Shaders," in *In Direct3D ShaderX*, Wordware Publishing, Inc, 2002, pp. 319-333. |
| [2] | "Unite Europe 2016 - A Crash Course to Writing Custom Unity Shaders!," Unity, 28 6 2016. [Online]. Available: https://www.youtube.com/watch?v=3penhrrKCYg&t=1943s. [Accessed 19 6 2017]. |
| [3] | Unity, "Projector," Unity, 10 6 2017. [Online]. Available: https://docs.unity3d.com/2017.1/Documentation/Manual/class-Projector.html. [Accessed 19 6 2017]. |
| [4] | R. R. Luque, "The Cel Shading Technique," 2012. |
| [5] | F. T. Ollie Johnston, The Illusion of Life: Disney Animation, Disney Editions, 1995. |
| [6] | S. M. D. a. T. Wien, "Abstract Cartoon Style Rendering". |
| [7] | C. M. M. H. M. B. Adam lake, "Stylized rendering techniques for scalable real-time 3d animation," 2000, pp. 13-20. |
| [8] | P. B. a. J. T. a. L. Markosian, "X-toon: An extended toon shader," in *In Proceedings of NPAR 2006*, 2006, pp. 127--132. |