Computer Graphics: Using Math and Science to Create Art

Alexander Neville

Brigham Young University

Table of Contents

Abstract……3

Introduction: What is Computer Graphics and How is it Related to Art?……4

Two-Dimensional Computer Graphics……4

Efficiency in 2D Graphics……5

Improving Traditional Animation……6

Creating Three-Dimensional Images……8

Accessibility for 2D Artists……8

Efficiency in 3D Graphics……9

Improving Aesthetic Quality in 3D

Three-Dimensional Animation

Aesthetics of 3D Animation

Efficient Procedural Generation

Conclusion

Appendix: Helpful Visuals

Abstract

This paper provides a basic overview of the field of computer graphics, as applied specifically to the realm of digital art. Computer generated digital art is all around us, from simple advertisements to full-length feature films and video games. While skilled artists provide the creative minds behind these beautiful works of art, computer graphics programmers are the driving force that makes computer-based art possible. They do the math and the physics behind the scenes, allowing the artists working in the digital realm to have as much creative capacity as possible.

This review will provide a basic overview of the most fundamental, essential aspects of computer graphics, and highlight the research that has thus far gone into each of these areas. Methods for 2D and 3D graphics will both be described, focusing primarily on the areas of aesthetic value, computational efficiency, and ease of use for artists.

Keywords: computer science, computer graphics, art, digital art, mathematics, efficiency, productivity, rendering, raytracing, painting, algorithms, computation, technology

Introduction: What is Computer Graphics and How is it Related to Art?

Technology is a dominant force in today's society. It is intricately entwined with everyday life, governing everything from lighting our homes at night to flying rockets into outer space. Art, on the other hand, has been around for millennia. Throughout the ages, it has been the primary way in which humans have expressed creativity and imagination. These two distinct fields are melded in the world of computer graphics.

Computer graphics is a sub-field of computer science. It is the science of digitally creating and manipulating visual content. At its core, it consists of the mathematical algorithms designed to draw two- and three-dimensional images onto a computer screen. This field fuses mathematics and art, and is used to create pictures and films with computers. Computer graphics has a wide variety of applications to visual art. In today's world, artists aren't limited to just a canvas and a set of paints. Instead, the realm of digital art allows artists and animators to create with the help of digital technology.

Because both mathematical algorithms and artistic principles are essential parts of this field, programmers in computer graphics often deal with problems relating to these things. Problems in computer graphics usually center around issues of computational efficiency (or how much of a computer's resources an algorithm uses), making software tools easier to use, and artistic quality. When graphics programmers find solutions to these problems, it is incredibly useful for digital artists. This is because it greatly increases the scope of what they can create.

Two-Dimensional Computer Graphics

2D computer graphics is a sub-discipline within computer graphics which focuses on creating and manipulating two-dimensional images on a computer screen. 2D images are, by defini-

tion, represented solely with the dimensions of length and width. There are many software programs available that can create and manipulate 2D images, including GIMP, Adobe Photoshop, and Krita. Artists can create cartoons, digital paintings, and pixel art for video games using such programs.

Efficiency in 2D Graphics

When painters and cartoonists are working digitally, they need software that will work quickly and efficiently in order to finish projects on time. For example, when an artist touches a digital pen to a drawing tablet, she expects it to make a mark on her computer screen without any lag. This will allow her to draw as though she were using a physical sketchbook or canvas. Another advantage to working in the digital realm is the ability to create perfect shapes and lines. Artists can greatly benefit from having the tools available which allow them to do this.

Consequently, it is evident that drawing lines efficiently is a paramount area of research in 2D graphics. According to Charles Oliver Coleman (2003), a scientific illustrator at the Institute of Systematic Zoology in Berlin, Germany, it is possible to draw perfect, precise lines quickly by using *vector graphics*. Vector graphics is a type of 2D graphics which uses geometric primitives (lines, points, and shapes) in order to create art. Drawing precise lines with vector graphics is done using the concepts of *path* and *fidelity*. Conceptually, a *path* is a special type of line, known as a Bezier curve, that is made up of straight and curved segments. Each segment is equipped with special handles that allow great control over the shape of the line. Mathematically, these lines use *linear interpolation* to create a curve: First, two lines are created, between which the curve will be drawn. These lines can be adjusted by the artist, using the aforementioned handles. Three points on one line are selected, which connect to corresponding positions on the other

line. Points between these form a curve. As the artist adjusts the handles, these points are either narrowed or widened, allowing complete control over the appearance of the curve. *Fidelity* is important because it controls how many of these points are created. In most 2D drawing applications, the artist can adjust this attribute. Coleman asserted that vector graphics is the optimal method for drawing perfect lines in 2D, because it allows for mathematically precise lines.

Murrell (2011) opined that instead of vector graphics, *raster graphics* are a more optimal way to draw two-dimensional lines to a computer screen. Instead of using geometric primitives, raster graphics uses a grid of x and y coordinates in order to control what an image looks like. Each coordinate in the grid is associated with a color, and is mapped to a pixel on the screen. In order to draw a line onto the screen, two coordinates are recorded: the beginning and the end of the line. A mathematical formula is then used to calculate the points in between these two coordinates, and all those points are filled in with the selected color. Murrell mentioned that raster graphics are more easily converted into different file formats, such as PDF. Also, while it is not as mathematically precise as the lines drawn by vector graphics, a raster line is able to hold a greater spectrum of color variety. Additionally, since raster graphics generally tends to be easier to code and maintain, more software applications support it.

Ultimately, though, no consensus has quite been reached on which of the two methods is better or more efficient. Both have their unique advantages and disadvantages. Consequently, software programs exist for both vector and raster graphics, and lines can be drawn effectively with either method.

Improving Traditional Animation

While 2D animation has taken a backseat to the 3D works of studios such as Pixar and DreamWorks in recent years, it is still alive and well today. American and Japanese studios still often use 2D animation in television shows and films. Instead of being drawn on paper, though, nowadays 2D animation is usually done in software programs, such as Adobe Animate and Toon Boom.

An important area in digital 2D animation is that of *inbetweening*. 2D animated works consist of myriad still images, called *frames*, which are shown very quickly one after the other. There are 24 frames required for each second of film (Whited, Noris, Simmons, Sumner, Gross, & Rossignac, 2010), and the final product is quite like that of a lengthy flip book. The frames that highlight the beginning or end of a visual transition are known as *key frames*, and inbetweening is used for the images that occur in between these frames, which give it the illusion of motion. Back when animation was only done on paper, inbetweening was an expensive process, requiring the talents of multiple artists with a high level of technical ability. In digital animation, though, inbetweening is done with the help of special software tools. Nobody has yet figured out how to digitally automate the process of inbetweening, but semi-automatic methods have been proposed.

Whited, et al. (2010) proposed a new software program, called BetweenIT, designed to improve semi-automatic inbetweening. These researchers also stressed the importance of artists in the inbetweening process, arguing that if the process is entirely automated then the unique creative touch of the animators would be lost. Thus, BetweenIT only handles *tight inbetweening*, which is used when the two key frames are very similar in shape. At any point, artists are allowed to modify it. Essentially, the program uses an algorithm for *composite interpolation*,

which is similar to the linear interpolation method described earlier. However, instead of using Bezier curves to create its lines, it uses a *logarithmic spiral*. This is a type of curve that is found often in nature, has less mathematical precision errors than Bezier curves, and is considered aesthetically pleasing. de Juan and Bodenheimer (2006), researchers from Vanderbilt University, suggested a more intricate semi-automatic approach. While the algorithms they use are very similar, de Juan and Bodenheimer's research also suggests adding a library of re-usable animations that have already been in-betweened. Data from these animations is used in subsequent projects. This saves time for future computations.

Further research should be conducted in order to determine whether or not inbetweening should be fully automated, and if so, how precisely to do that. Much more in-depth mathematical research and exploration will be necessary in order to achieve this.

Creating Three-Dimensional Images

3D computer graphics is another sub-field of computer graphics, dealing with the creation, manipulation, and motion of 3D images on a computer screen. Unlike two-dimensional images, images created in three dimensions have the attributes of length, width, and depth. Many modern-day films and video games have been created with the help of 3D graphics. Furthermore, many programs exist to create images in 3D, including Blender, Autodesk Maya, Autodesk Mudbox, and Houdini.

Accessibility for 2D Artists

3D graphics requires a higher level of technical skill than traditional or 2D art. Most artists are not trained in mathematics or computation, and consequently can struggle when beginning to learn 3D. This generally happens when an artist starts working in the film or video

game industry. Mahmudi, Harish, Le Callennec, and Boulic (2016) proposed a method for posing and animating 3D characters using 2D techniques. Artists can make line strokes with a stylus, which is translated to a corresponding portion of the 3D skeleton. They can then animate the characters using those lines. This interface is much more intuitive for traditional artists who are accustomed to working in 2D, thus making it easier for them to fill important roles at film and game studios.

The algorithm that these researchers used consists of four steps: First, *stroke-chain matching* is used to match an artist's digital brushstroke to a corresponding limb or appendage of a character. Then, *intermediate representation* makes modifications to the initial match, in order to adjust mathematical error. It might scale, rotate, or translate the match. Third, *feature extraction* selects the points that are most likely to represent the character's skeleton. Finally, *joint alignment* computes where the joints of the character are, in relation to the skeleton. The artist can manually override any of these steps, because they are imperfect and do not always produce what the artist wants. However, overall they are much more effective in helping artists who are used to drawing to translate their skills into 3D artwork and animation.

Efficiency in 3D Graphics

3D computer graphics is still a relatively new field, and consequently the issue of *efficiency* is a prevalent one. Computers, especially average computers made for everyday use, can handle complicated graphics, but do not usually do so very efficiently. Using an average laptop with a good graphics card, it can still take hours to render a high quality image created with a 3D modeling program such as Autodesk Maya. Thus far, many strides have been made in terms of efficiency in computer graphics, although there is still a long way to go.

Raytracing is a method of realistically creating light and color in a 3D image. A virtual ray of light is bounced off of different objects in a scene, and light or shadows are computed based on the perspective of the viewer. This method is known for creating beautifully rendered images, such as those used by Pixar in their animated films. However, even the powerful computers Pixar uses can take weeks to finish rendering a scene, and this is in part because raytracing is very computationally expensive.

Barringer and Akenine-Möller (2013) proposed a method for faster raytracing. Their method involves using a *binary tree* structure to internally represent data. A binary tree is a special type of list, used to stored data in a computer. Visually, a binary tree looks like an upside-down tree in nature. Data is stored in *nodes*, which look like leaves on the tree. Each node can have up to two *children*, which branch off from it. Data used to compute color is stored in each of these nodes. Using this binary tree proves to be much faster than normal methods of data representation, because it requires less memory and therefore less overhead computation. Efficient raytracing leads to shorter render times, which is advantageous for artists.

Zou, Wang, & Wan (2010) did not refute or argue Barringer and Akenine-Möller's proposal. However, they did offer a different approach to improving efficiency in 3D graphics. Their logic involved attacking the problem of rendering directly, increasing its speed through new methods. Most 3D scenes use *textures*. Textures are 2D painted files wrapped around 3D objects, used to control their color and shadows. In most 3D software programs, data representing textures is loaded in all at once. In the method described by Zou et al., though, texture data is only loaded when it is absolutely necessary — i.e., when the artist sees it. Scenery that is behind other objects, for example, is not loaded into data unless the artists moves around the scene and it di-

rectly intersects with their line of vision. The rest of the data is compressed, in order to save both time and memory. Both of these proposed methods are computationally efficient, and are used in much of today's digital art software.

Improving raytracing and rendering speeds are but two of the many ways that 3D graphics can be made more efficient. Other approaches include optimization of graphics cards, faster bump mapping, and making improvements to shading algorithms. There is still a lot of research to be done in this area, and as computers improve each year, the possibilities of what can be accomplished with them increases significantly.

Improving Aesthetic Quality in 3D

In addition to the problem of efficiency, it is also important that 3D computer graphics algorithms allow artists to produce beautiful works of art. One key to improving aesthetic quality in 3D graphics is balancing computational efficiency with artistic sensibility. Most of this paper has focused on methods for efficiency and ease of use, which are both very important for computer graphics. However, in the omnipresent quest for quicker runtime and less memory usage, artistic quality must not be overlooked.

Gulbrandsen and Framestore (2014) describe an algorithm for improving the appearance of metallic surfaces in 3D. This method creates beautiful materials, and is also relatively easy for artists to use. Most algorithms that control the appearance of metallic surfaces use what is called a *Fresnel equation.* This is defined as a mathematical formula which calculates the reflection of light on different surfaces. Gulbrandsen and Framestore's algorithm does use a Fresnel equation; however, it simplifies it in order to make it easier for artists to use. It also produces a very high quality metal. In sum, their formula uses the concepts of *edge tint* — the color of the metal under

different types of light — and *reflectivity.* Artists are able to adjust these values through sliders, and they are much easier to understand than the complicated *index of refraction* that most metal algorithms utilize. This algorithm is implemented in a software package called 3Delight Metal. Because it is easier for skilled artists to understand and manipulate, it tends to produce very realistic looking metals.

Another interesting procedure for beautifying 3D graphics was proposed by Toth (2013). Toth's approach tackles the problem of wrapping a two-dimensional texture around a 3D object. When a 3D object's shape isn't uniform, or a texture is too small, *texture seams* can occur. These are unintended inconsistencies in a texture, usually in the form of lines or mismatched colors. Toth's solution to this problem is to discard certain *filter taps*, which are devices used to make parts of a texture that are closer to the viewer appear to be more vivid and detailed. In Toth's algorithm, the filter taps that are not immediately visible to the viewer are discarded. As a result of this, texture seams are less likely to occur, because the computer has less of a need to compensate for perceived deficiencies.

Villemin and Hery (2013), both artists at Pixar Animation Studios, also wanted to improve the artistic quality of 3D computer graphics. They described a method for making more beautiful flames when building and animating fire scenes. Until beginning their research, most 3D software packages created flames that looked realistic when surrounded by certain surfaces (wood and brick, for example) but not others (such as glass and water). Villemin and Hery's algorithm created flames using *light sampling* and *BDRF sampling*, which compute the likely appearance of a flame based on two parameters: the surrounding lights and the nearby objects. The end result is flames that look realistic even if viewed around materials such as glass. So, in a film

or video game, if a character is looking outside a window at a forest fire, the fire still looks real and convincing, instead of being blurred by the glass material.

Certainly, the discussed methods are only a few of the myriad ways computer graphics researchers have worked to improve 3D computer graphics. Additionally, this small snapshot indicates that while great strides have been made in these areas, further research is still necessary in order to allow 3D artwork to reach its full potential.

Three-Dimensional Animation

In the previous section, this paper discussed algorithms for improving 3D computer-generated images. Another equally important facet of computer graphics is animating those images across a screen. While the previously discussed algorithms can certainly be applied to animation, they are just as easily applied to still images. Other algorithms, though, are better understood when analyzed in the context of 3D animation.

Aesthetics of 3D Animation

Making 3D animation aesthetically pleasing is of utmost importance to artists, who are aiming to produce professional caliber work. Thus, much research has been done in this area. Bai, Yu, and Liu (2016) proposed a method for improving cloth animations. Prior to their work, cloth animation was generally considered a natural consequence of character animation. Animators would concentrate their efforts on the math and physics behind their characters, and simple algorithms would cause clothing to move whenever a character did. However, this allowed for less control over the clothing. Bai et al. created a new algorithmic paradigm which focused specifically on what the character was trying to do with the clothing. This algorithm computed the physics behind the character's joints, as well as their motion directly applied to the cloth.

This allowed for cloth that looked much more natural, because it could be adjusted to different situations.

In a similar vein, Bradbury, Subr, Koniaris, Mitchell, and Weyrich (2015) created another algorithm used for creating more realistic 3D animation. In 3D video games, players explore immense virtual worlds. For artists trying to meet release dates, it is not efficacious to model and sculpt every single tiny detail of these worlds. It is simply too time-consuming. Yet at the same time, beautifully detailed graphics can help market and sell a game. Bradbury et al. focused their efforts on developing an algorithm to quickly create a virtual forest. Animators would model a small number of different trees, and their algorithm would take over after that. The algorithm simulates a forest in the real world by using the same logical bases as ecologists do. Plants are generated throughout the terrain based on environmental factors: how much rainfall and sunlight the area receives, for example. Not only does this save the artists time, it also allows gamers to have a much more realistic gameplay experience. In fact, this algorithm is very similar to the topic covered by the next section: procedural generation.

Efficient Procedural Generation

As mentioned in the previous section, it is often not feasible to spend hour after hour painstakingly modeling each little aspect of an enormous video game world. Thus, artists require the assistance of computers in order to meet the rigorous deadlines of the video game industry. *Procedural generation* is the term for algorithms that computers use to create repetitive content — such as rocks, trees, and mountains — automatically. Parberry (2014) described an effective means of generating infinite terrain using real-world elevation data. In essence, Parberry's algorithm reads in real-world information using geospatial data. This information is translated into

computer code, which is in turn transformed into 3D mountains, valleys, lakes, and the like. As a result of using real-world data, there is a very natural, logical flow from one landscape to another. This algorithm has proven to be very helpful for video game artists.

Conclusion

Ultimately, computer graphics is a multifaceted field with many areas that have not yet been explored. While many problems in computer graphics have been solved, many are still in need of optimization and improvement. And there are many problems that have not yet been explored. Problems such as aesthetic quality, ease of use of software applications, and computational efficiency truly form the heart of this industry. When graphics programmers find effective solutions to these problems, they benefit the artists by greatly increasing the scope of what they can create.

References

Akenine-Möller, T., & Barringer, R. (2013). Dynamic Stackless Binary Tree Traversal. *Journal of Computer Graphics Techniques, 2*(1), pp. 38-49. Retrieved from http://jcgt.org.

Annum, G.Y. (2014). Digital painting evolution: A multimedia technological platform for expressivity in fine art painting. *Journal of Fine and Studio Art, 4*(1), pp. 1-8. Retrieved from http://www.academicjournals.org/journal/JFSA.

Bai, Y., Liu, C. K., & Yu, W. (2016). Dexterous Manipulation of Cloth. *Eurographics, 35*(2), pp. 2-11. Retrieved from http://www.cc.gatech.edu.

Bodenheimer, B., & de Juan, C. N. (2006). Re-using Traditional Animation: Methods for Semi-Automatic Segmentation and Inbetweening, presented at Eurographics: ACM Siggraph Symposium on Computer Animation, Vienna Austria, 2006. Nashville, TN : Vanderbilt University.

Boulic, R., Harish, P., Le Callennec, B., & Mahmudi, M. (2016). Artist-oriented 3D character posing from 2D strokes. *Computers & Graphics, 57*(1), pp. 81-91.

Retrieved from http://www.elsevier.com/locate/cag.

Bradbury, G.A., Koniaris, C., Mitchell, K., Subr, K., & Weyrich, T. (2015). Guided Ecological Simulation for Artistic Editing of Plant Distributions in Natural Scenes. *Journal of Computer Graphics Techniques, 4*(4), pp. 28-53. Retrieved from http://jcgt.org.

Coleman, C.O. (2003). “Digital inking”: how to make perfect line drawings on computers. *Organisms Diversity and Evolution, 3*(1), pp. 303-304. Retrieved from

http://www.urbanfischer.de/journals/ode.

Gulbrandsen, O. (2014). Artist Friendly Metallic Fresnel. *Journal of Computer Graphics Techniques, 3*(4), pp. 64-72. Retrieved from http://jcgt.org.

Hery, C., & Villemin, R. (2013). Practical Illumination from Flames. *Journal of Computer Graphics Techniques, 2*(2), pp. 142-155. Retrieved from http://jcgt.org.

Parberry, I. (2014). Designer Worlds: Procedural Generation of Infinite Terrain from Real-World Elevation Data. *Journal of Computer Graphics Techniques, 3*(1), pp. 74-85. Retrieved from http://jcgt.org.

Toth, R. (2013). Avoiding Texture Seams by Discarding Filter Taps. *Journal of Computer Graphics Techniques, 2*(2), pp. 91-104. Retrieved from http://jcgt.org.

Wan, W., Wang, X., & Zou, X. (2009). A Fast Method for View-Based 3D Scene Texture Rendering. *IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2009), 2*(1), pp. 361-364. Retrieved from http://ieeexplore.ieee.org.

ALSO: https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Murrell.pdf

https://www.siggraph.org/education/materials/HyperGraph/raytrace/rtaccel.htm

These will be properly cited in my final paper

Appendix A: Helpful Visuals

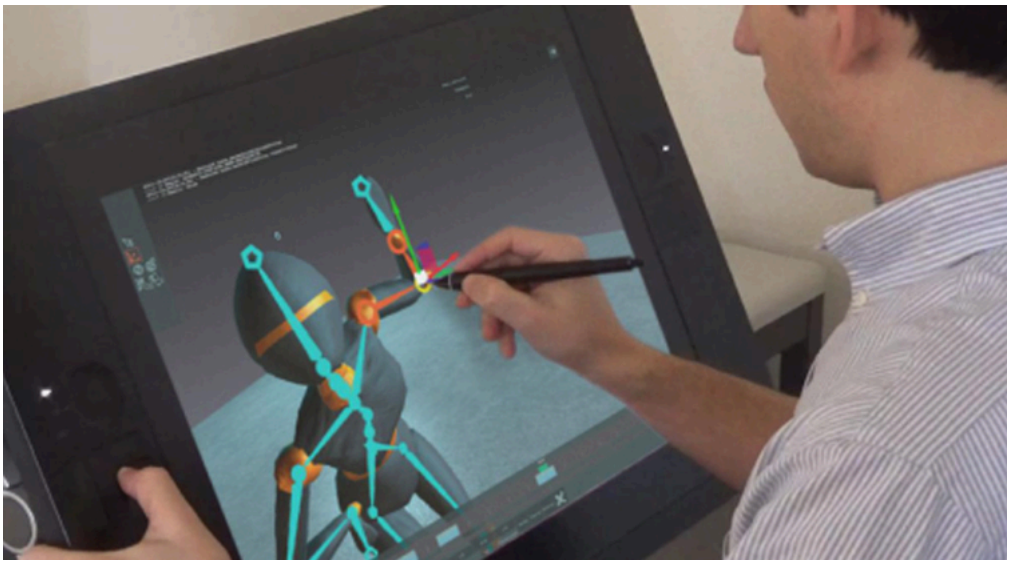

Figure 1: An artist uses the software program created by Mahmudi, Harish, Le Callennec, and Boulic.