Computer Generated Imagery: CGI and the Mathematical Interpretation of 3D Objects in a 2D Space

Griffin Ryan

University of Washington

MATH 308: Matrix Algebra

Autumn 2021

Abstract

Computer graphics are one of the cornerstones of human achievement. In the last century, digital graphics has humbly evolved from drawing pixels on a screen, to modeling the real-world with physics-based models. In terms of linear mathematics, computer graphics has seen some of the most advanced applications. This study explores the process of viewing a 3D space on a 2D plane. CGI can be expressed as a mathematical relationship using transformation matrices, rotation matrices, and luminance respectively.

Introduction

CGI can be simply summarized as a viewer interpreting 3D objects on a digital display. The display can be interpreted as a 2D plane in which points are plotted on. Every three-dimensional object shown on a screen can be interpreted as a collection of points on a two-dimensional plane.

Linear transformations are applied to these coordinate points to trick the brain into seeing depth instead of just a two-dimensional figure. A simple example of this is the torus. On a two-dimensional plane, this figure is shown as a circle or ellipse. When the torus is represented on a two-dimensional plane however, rotation matrices are applied to the set of points to give the illusion of depth in the screen. The following is an example of a rotation matrix being applied to a set of coordinates (x, y):

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$R\vec{v} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

According to the Encyclopedia of Mathematics, the webpage *Rotation* defines rotations of a Euclidean space around a given point form a group (Bazylev, 2002). In other words, the motion of a 3D object on a screen can be expressed with rotation matrices; this relationship tricks the brain into interpreting the depth of an object.

To consider an application of projecting 3D coordinates into a 2D space, we can consider the basis of the transformation as the viewer's perspective. In the field of 3D modeling, this origin is oft called the "camera".

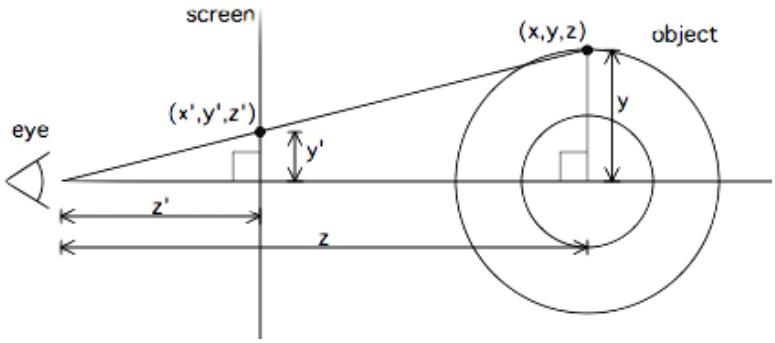

Figure 1: Perspective Rendering Model

In this model, the viewer is positioned z' units away from the viewing window, or plane. To project a single coordinate in R^3 space to R^2 space, the coordinate is scaled by the view distance z' (Treibergs, 2001). This translation is considered as the theory of 3D perspective rendering in computer graphics. To keep proportional equality in R^2 space:

$$\frac{y'}{z'} = \frac{y}{z}$$

$$y' = \frac{yz'}{z}$$

This relationship just defines the proportional equality. To project a singular coordinate in R^3 space to R^2 space, the coordinate must be scaled by the factor of the viewing distance, otherwise known as z'.

$$(x', y') = (\frac{z'x}{z}, \frac{z'y}{z})$$

A very common application of using this projection is in video games. Along with this coordinate projection, transformations can be applied to a set of coordinates to simulate movement. The torus is a very common shape in graphic design; it is just a circle rotated around the y-axis to give a 3D donut-like shape. One famous example in game production of the torus are the coins from Nintendo's *Super Mario* series.

Figure 2: Torus from the Mario series

The following examples will detail linear transformations in a 2D space, interpretation of 3D objects in a 2D space, and how to determine the illuminance of an object projected in 2D space.

Translation in 2D Space

Transformation matrices are used in 2D animation in order to move objects about our 2D plane, the screen. In game design, a very common transformation is to move the player character horizontally or vertically. This is called a *translation*.

To consider a specific translation in 2D animation, the original *Super Mario Bros.* can be used as an example.

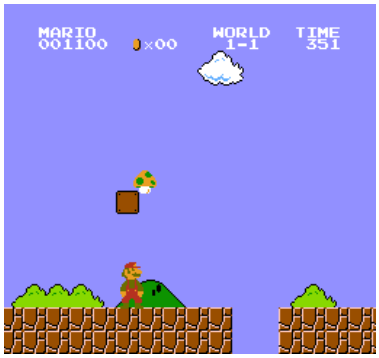

Figure 3: Super Mario Bros. translation example

To simplify the model, the player (Mario) can be represented as the single point (x, y) on a 2D plane. As the character moves, there are new target coordinates for the object to be positioned at. These target coordinates are represented as (x', y').

The 2D translation matrix can now be represented. The vector (tx, ty) represents the scale at which the character moves horizontally, in the x-direction:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation in 2D Space

Often, these onscreen objects are manipulated and translated to convey 3D motion in the 2D space. For example, these are often rotated around the y-axis for a simple spinning animation. To examine a particular example of this rotation, the CGI ring animation from Marvel's 2021 film *Shang-Chi and the Legend of the Ten Rings*.

Figure 4: Shang-Chi and the Animation of the Ten Rings

This animation features a torus revolving around the x-axis, y-axis, and z-axis on the screen. To simplify the mathematical model, we can evaluate it as a torus spinning about the axes on a fixed point. The torus is modeled as a circle of radius R_1 centered at a point $(R_2, 0, 0)$. This circle is rotated around the y-axis from angle 0 to 2π. To model this transformation, the coordinates of the torus can be expressed as the following:

$$(x, y, z) = (R_2, 0,0) + (R_1\ cos\theta, R_1 sin\ \theta, 0)$$

With the coordinate system established for the torus, the rotation of the coordinates can be found by applying a rotation matrix. According to the webpage *Rotation Matrix* by

Wolfram Research, Inc., the rotation matrix that rotates a given vector v around an angle θ is given by:

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Considering this, the torus can be revolved about the y-axis and projected on the plane by the following transformation:

$$(R_2 + R_1\cos\theta\,, R_1\sin\theta\ \,, 0) * \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}$$

$$= ((R_2 + R_1\cos\theta)\cos\phi, \quad R_1\sin\theta\,, \quad -(R_2 + R_1\cos\theta)\ \sin\phi)$$

This transformation matrix gives the mathematical relationship for rotation about the y-axis with a torus. This rotation matrix applied to the coordinates of the torus is only one linear transformation. The torus should also spin about the x-axis and the z-axis by a factor of α and β respectively.

To model the rotation about all three axes, the same process is applied with more rotation matrices:

$$(R_2 + R_1\cos\theta\,, R_1\sin\theta\ \,, 0) * \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Thusly, the rotation process for the spinning torus found in *Sonic the Hedgehog* can be modeled as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (R_2 + R_1 \cos\theta)(\cos\beta\cos\phi + \sin\alpha\sin\beta\sin\phi) - R_1\cos\alpha\sin\beta\sin\theta \\ (R_2 + R_1 \cos\theta)(\cos\phi\sin\beta - \cos\beta\sin\alpha\sin\phi) + R_1\cos\alpha\cos\beta\sin\theta \\ \cos\alpha\sin\phi\,(R_2 + R_1\cos\theta) + R_1\sin\alpha\sin\theta \end{bmatrix}$$

This is calculated roughly every frame of the process. Therefore, the variables $\theta, \phi, \alpha,$ and β can be substituted for a value that determines the rate of spinning. For example, an angle of $\frac{\pi}{2}$ will spin twice as fast as $\frac{\pi}{4}$.

Determining Illumination of Object with Ray Tracing

To supplement the illusion of 3D space on a 2D plane, lighting is used to help convey depth to the viewer. In the 2006 movie *Cars*, Pixar Animation Studios used a process called ray tracing to provide luminance to the 3D models (Per, 2006). Only in recent years have modern computational components become powerful enough to utilize ray tracing in real-time.

Ray tracing is the process of calculating luminance of an object by modeling a light bouncing off the object (Christensen, 2006). Given a light source and object in a 3D space, ray tracing determines the object luminance by calculating the magnitude of the surface normal vector on the object.

Considering light is only cast in one direction, a sphere would have different luminance based on how close it is to the light source. The magnitude of the surface normal vectors varies at different points of the sphere. Therefore, the strength of luminance varies.

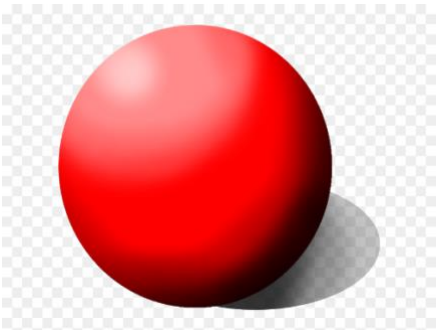

Figure 5: Sphere in 3D space showing luminance

Because Pixar movies tend to feature incredibly complex 3D models for characters, this illumination can be based off the spinning torus example above.

As the torus rotates, the strength of luminance is applied to different points based on a fixed light source. To simplify this model, the light source can be fixed at the point $(0, 1, -1)$ which is behind and above the viewer. Essentially the light is being cast onto the 2D plane from behind.

Before the luminance is determined, the surface normal N must be found. The surface normal is fixed at the origin of the torus. To have the lighting appear consistent with the rotating torus, the surface normal must have the same rotation process applied.

The point of origin is this case is $(\cos\theta\,, \sin\theta\,, 0)$. Applying the same three rotation matrices found, the model is as follows:

$$\left(N_x\ ,\ N_y,\ N_z\right) = \begin{bmatrix}\cos\theta \\ \sin\theta \\ 0\end{bmatrix} * \begin{bmatrix}\cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi\end{bmatrix}\begin{bmatrix}1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha\end{bmatrix}\begin{bmatrix}\cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1\end{bmatrix}$$

This is just the expression for the surface normal at the point (x, y, z). This expression must be multiplied by the point of light $(0,1,-1)$.

$$L = \left(N_x\ ,\ N_y,\ N_z\right) * (0,1,-1)$$

$$L = \begin{bmatrix}\cos\theta \\ \sin\theta \\ 0\end{bmatrix} * \begin{bmatrix}\cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi\end{bmatrix}\begin{bmatrix}1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha\end{bmatrix}\begin{bmatrix}\cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1\end{bmatrix} * (0,1,-1)$$

$$L = \cos\phi\cos\theta\sin\beta - \cos\alpha\cos\theta\sin\phi - \sin\alpha\sin\theta + \cos\beta\,(\cos\alpha\sin\theta - \cos\theta\sin\alpha\sin\phi)$$

Therefore, the luminance L for the rotating torus model is calculated as such. The two expressions for luminance L and the points of the torus (x, y, z) together both use the same rotation matrix transformations. However, we have two different expressions:

$$\begin{bmatrix}x \\ y \\ z\end{bmatrix} = \begin{bmatrix}(R_2 + R_1\cos\theta)(\cos\beta\cos\phi + \sin\alpha\sin\beta\sin\phi) - R_1\cos\alpha\sin\beta\sin\theta \\ (R_2 + R_1\cos\theta)(\cos\phi\sin\beta - \cos\beta\sin\alpha\sin\phi) + R_1\cos\alpha\cos\beta\sin\theta \\ \cos\alpha\sin\phi\,(R_2 + R_1\cos\theta) + R_1\sin\alpha\sin\theta\end{bmatrix}$$

and

$$L = \cos\phi\cos\theta\sin\beta - \cos\alpha\cos\theta\sin\phi - \sin\alpha\sin\theta + \cos\beta\,(\cos\alpha\sin\theta - \cos\theta\sin\alpha\sin\phi)$$

Conclusion

Considering computer architecture, these complex equations are possible in fractions of a second due to the advancement of the GPU. A traditional CPU handles calculations singularly, whereas the GPU runs the processes in parallel. This allows for the computations happen at the same time. All considered, it still takes massive processing power to create a 3D scene.

To put the torus model into perspective, this is a basic singular 3D model. Compared to a Pixar animation, there are thousands upon thousands of different objects that are transformed and illuminated with each frame. At Pixar Animation Studios, it takes an average 24 hours to render a single frame (Hery, 2013). As such, a 60 frames/second movie that is 100 minutes long would take 144,000 hours to render on a single machine. Equivalently, this figure

is almost 16.5 years of render time. With the advancement of CGI comes the advancement of computational processing.

The scale of the torus model above is only a small fraction of what it takes to create the modern CGI we are used to seeing. This process of rotation and illumination is the essential to rendering 3D space on a 2D plane. Modern computer-generated imagery follows the same process, but on a much more massive scale.

\

References

Bazylev, V.T. “Rotation.” In *Encyclopedia of Mathematics vol. 1.* Springer-Verlag New York Inc., 2002.

Christensen H. Per, Julian Fong, David M. Laur, Dana Batali. *Ray Tracing for the Movie ‘Cars’*. Pixar Animation Studios, September 2006. https://graphics.pixar.com/library/RayTracingCars/paper.pdf.

Christophe Hery, Ryusuke Villemin. *Physically Based Lighting at Pixar*. Pixar Animation Studios, July 2013. https://graphics.pixar.com/library/PhysicallyBasedLighting/paper.pdf

Christensen, Per H. *Point-Based Global Illumination for Movie Production*. Pixar Animation Studios, July 2010. https://graphics.pixar.com/library/PointBasedGlobalIlluminationForMovieProduction/paper.pdf.

Eisenberg, David J. “Translation: Moving the Grid.” in *2D Transformations*. Processing Foundation. https://processing.org/tutorials/transform2d.

Leon, Steven J. “Linear Transformations.” in *Linear Algebra with Applications (9th Edition)*. Pearson, August 9 2014.

Treibergs, Andrejs. *The Geometry of Perspective Drawing on the Computer.* University of Utah, July 24 2001. https://www.math.utah.edu/~treiberg/Perspect/Perspect.htm#refs.

Weisstein, Eric W. “Rotation Matrix.” in *MathWorld – A Wolfram Web Resource*. https://mathworld.wolfram.com/RotationMatrix.html.