Challenge: Making a raytracer (first step)

Abstract

The goal of this challenge is to design a simple raytracer that can be done step by step. In the first step, we will get acquainted with the mathematics required for vectors, points and three-dimensional volumes, and we will look at the method of projecting a wireframe.

To participate in this challenge, the use of any programming language is allowed, and only the final images obtained at the end of each step and the program source are required to participate in the challenge; I suggest that you share the code you wrote on Github or other suitable place with your friends so that they can use it too.

1 Basics

A free vector $\vec{v}$ is a 4-element vector of scalars (x_v, y_v, z_v, w_v), where the first three elements are the magnitude of vector $\vec{v}$ on each of the corresponding axes in the space and the last element (w_v) is always equal to 1.

We can calculate the length of a free vector with Pythagoras formula:

$$|\vec{v}| = (x_v^2 + y_v^2 + z_v^2)^{1/2}$$

A unit vector is a vector with length 1.

Inner product of two free vectors $\vec{w}$ and $\vec{v}$ is defined as

$$\vec{v} \cdot \vec{w} = x_v \cdot x_w + y_v \cdot y_w + z_v \cdot z_w = |\vec{v}||\vec{w}|\text{cos}\theta$$

A constant vector $\vec{P}$ is a vector with specific beginning and ending points. For example, if the vector $\vec{P}$ has specific beginning and ending points A and B we have:

$$\vec{P} = (x_B - x_A, y_B - y_A, z_B - z_A, 1)$$

A unit vector which is perpendicular to a plane in a space is called the *normal* of the plane and it is commonly shown as $\vec{n}$.

A transformation matrix M_t is a matrix which if multiplied with a vector, it moves the vector by $(x_t, y_t, z_t, 1)$:

$$M_t = \begin{bmatrix} 1 & 0 & 0 & x_t \\ 0 & 1 & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

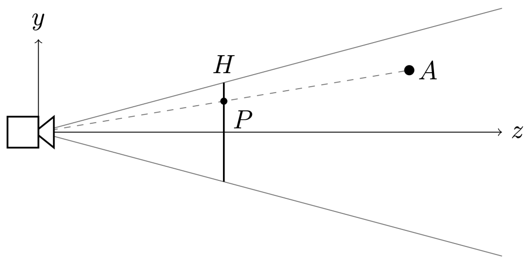

Figure 1: Projecting a point onto a virtual plane

A rotation matrix M_r is a matrix which if multiplied with a vector, it rotates the vector by θ around a specific axis. For each axis, we have:

$$M_r^x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_r^y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_r^z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2 Projection

Although our brain perceives the outside world in three dimensions, our eyes perceive the outside world in two dimensions. This is also true for the aperture of a camera lens. Our eye, or camera, does this by imaging three-dimensional objects on a virtual screen.

In the figure 1 we can see the process of projecting the point A on a virtual plane H. In this figure, we call the distance between the plane H and the camera *focal length* and it is dented as f. For the sake of simplicity, we assume the height of plane H to be equal to 1. With these assumptions, the position of the projection of point A on plane H is computed by:

$$y_P = f\frac{y_A}{z_A}$$

If we repeat these calculations in 3-dimension space on axis x (perpendicular to the plane), we have:

$$x_P = f\frac{x_A}{z_A}$$

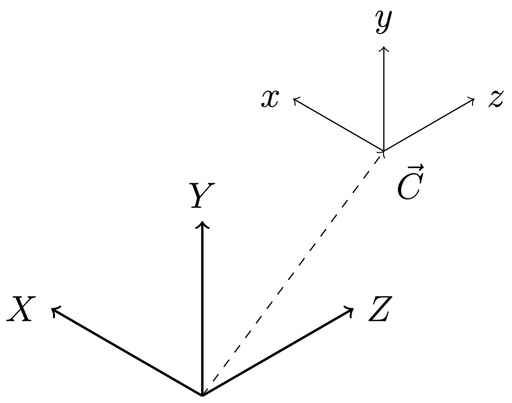

Figure 2: Global coordinate and local coordinate

In general, we have:

$$\begin{bmatrix} x_P \\ y_P \\ 1 \end{bmatrix} = \frac{1}{z_P} \times \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} = \begin{bmatrix} f\frac{x_A}{z_A} \\ f\frac{y_A}{z_A} \\ 1 \end{bmatrix}$$

In these calculations, we call the intermediate matrix the *projection matrix*. But this formula is only true when the camera is placed on the origin and aligned with the coordinate axes, but the camera we are considering must be located at any point in space and in any desired direction.

For this purpose, we defined two distinct coordinates. The *global coordinates*, which is the reference by which we do all of our calculations. We show the global coordinates with capital letters X, Y, and Z, and for brevity, we call the origin of the global coordinate, the origin of coordinate. Note that based on this definition, axis Y is the axis which is visually facing up (Figure 2).

On the other hand, we have the local coordinates whose origin is camera's position. We consider the direction of the z axis of these coordinates on the side of the camera's view and the direction of the y axis perpendicular to it and facing the top of the camera, and define the x axis relative to the two axes z and y.

With these definitions, we need to convert the vectors of each point from global coordinates to vectors in local coordinates. Obtaining this conversion is long and beyond the scope of this challenge, so here we only summarize the final conversions.

Assume that the vector $\vec{P}$ has $(X_P, Y_P, Z_P, 1)$ coordinates in global coordinates and we have a local coordinate with origin $\vec{C}$, where $\vec{C}$'s position in the global coordinates is $(C_X, C_Y, C_Z, 1)$. Then, we can obtain the local vector $\vec{P}_c$ with

$$\vec{P}_c = M_r^r \times M_r^y \times M_r^p \times M_t \times \vec{P}$$

In this formula, M_t is a transformation matrix:

$$M_t = \begin{bmatrix} 1 & 0 & 0 & -C_X \\ 0 & 1 & 0 & -C_Y \\ 0 & 0 & 1 & -C_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and M_rs are the rotation matrices:

$$M_r^x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_p & -\sin\theta_p & 0 \\ 0 & \sin\theta_p & \cos\theta_p & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_r^y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_r^z = \begin{bmatrix} \cos\theta_r & -\sin\theta_r & 0 & 0 \\ \sin\theta_r & \cos\theta_r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

.

Here, θ_p, θ_y, and θ_r are relative to the camera's natural position - when the local coordinates of the camera are aligned with the global coordinates - and they are measured relative to the axes x, y, and z, respectively. In technical terms, these angles are called *pitch*, *yaw*, and *roll*, respectively.

3 Challenge

The first step of the challenge is to write a program in any preferred language and technology to create a wireframe image of a 3D model from the perspective of a camera using the formulas we have discussed so far. This program should read the model and camera information from the text file whose format we explain in the next section. To participate in this challenge, you must put the code you wrote where it is accessible to everyone, and share the image of the program output for all cameras.

4 Format of the text file

The text file contains 3 parts: vertices, faces, and cameras. The vertices part starts with the command `VERTICES` and includes a vector in each line, representing the location of a vertex. For example:

```
VERTICES
1.0 0.0 0.0 1.0
1.5 2.1 0.0 1.0
1.0 0.0 3.0 1.0
...
```

Note that each vector contains 4 floating points, where the first three are vector ith global coordinates X_i, Y_i, and Z_i and the last number is always 1. Vertices are indexed from 1.

The second part begins with the command `FACES` and each line represents a triangular face. For example:

```
FACES
1 2 3 1.0 0.0 0.0 1.0
1 4 5 0.0 1.0 0.0 1.0
2 5 6 0.0 0.0 1.0 1.0
...
```

Each face is defined by 3 decimal and 4 floating numbers. The decimal numbers are the index of the triangle vertices, and the 4 floating points determine the normal vector.

The last part, or the cameras section, begins with the command `CAMERAS` and contains a camera in each line. For example:

```
CAMERAS
5.0 6.0 1.0 1.0 30.0 45.0 10.0 0.25 FAR
1.0 0.5 3.0 1.0 20.0 30.0 15.0 0.50 NEAR
...
```

The definition of a camera includes 8 floating numbers, where the first 4 numbers are the position of camera relative to the global coordinates; the next 3 numbers are pitch, yaw, and roll; and the last number is focal length of the camera. At the end of each line, there is a string representing a unique name for the camera.