IB Mathematics AA Higher Level

Exploring and implementing 3D computer graphics using linear algebra

L. Sawadsky

Internal Assessment: Mathematics AA HL

Session: May 2026

Word Count:

Contents

1 Introduction

Representing our three-dimensional world in the two dimensions our computer screens allow us has proven not to be a simple task, but thankfully one that centuries of research in linear algebra happen to assist greatly in. Simulating a three-dimensional environment in a computer system is done largely using research in physics: laws and constants that govern the behaviour of rigid bodies, fluid, and light are replicated in programs to create a world that functions, more or less, like ours. However, there is a problem designers run into that has no parallel in our world. We understand the laws by which physical things change, but *representing* these changes is a challenge unique to computerized versions of the world. This investigation will explore two aspects of this non-physical issue: the projection of a set of given vertices in 3D space onto a 2D screen, and some common manipulations of those vertices: translations, stretches, and rotations. It will primarily focus on the mathematics behind them, but in parallel I will develop an implementation of these principles in C++ using the 2D graphics library SDM.

2 Projection of arbitrary points

The first challenge this investigation will tackle is the visualization of a given set of points in $\mathbb{R}^3$ on a 2D display. The approach taken by modern 3D graphics engines is to turn 3D objects into 'meshes': collections of vertices connected by lines and faces. To draw these meshes on the display, the engine first draws the vertices using the following strategy: treat the observer as a point and then create a finite plane representing the display which moves around the observer, normal to the line between its center and the observer with the distance of that line constant, in a sphere. This movement represents turning the 'camera'; alternatively, one might imagine the display as a sort of window into the space represented. A point $\mathbf{P}$ in that space is represented on the display by placing a point at the intersection of the line from the observer to $\mathbf{P}$ with the display plane.

For now, since we are dealing with two different instances of $\mathbb{R}^3$, let $\mathbb{R}^3_{comp}$ be

'computer space' and $\mathbb{R}^3_{real}$ be 'real space', with the same scaling. Let $\Pi_{display}$ be the display plane and **O** be the position of the observer, with coordinates (O_x, O_y, O_z) in $\mathbb{R}^3_{comp}$ and $(0,0,0)$ in $\mathbb{R}^3_{real}$. Let d be the distance between **O** and the center of $\Pi_{display}$, θ_x the horizontal angle between the lines connecting **O** to the edges of $\Pi_{display}$, and θ_y the vertical angle. Let point $\mathbf{P}(P_x, P_y, P_z)$ be an arbitrary point in $\mathbb{R}^3_{comp}$ with $(P_x - O_x, P_y - O_y, P_z - O_z)$ being its coordinates in $\mathbb{R}^3_{real}$, i.e. using **O** as the origin. The vector $\overrightarrow{\mathbf{OP}}$ is therefore: $\begin{bmatrix} P_x - O_x & P_y - O_y & P_z - I \end{bmatrix}$.

2.1 Derivation of equations

The equations for $\mathbb{R}^3 \Rightarrow \mathbb{R}^2$ are deceptively simple. Given point $P(X, Y, Z)$, let its projection on the display plane be $D(x, y)$. The user is at the origin, so the line from the user to point P is $\begin{bmatrix} X & Y & Z \end{bmatrix}$ and every point on that line will be of the form $(\lambda X, \lambda Y, \lambda Z)$. Let d be the constant distance from the user to the screen plane. Then, for all points on the display plane $\Pi_{display}$,

$$d = \lambda Z$$

$$\therefore \lambda = \frac{d}{Z} \tag{2.1}$$

The projection of any point P onto the display plane is thus

$$(\frac{dX}{Z}, \frac{dY}{Z}, d)$$

which, in $\mathbb{R}^2$ on $\Pi_{display}$, is simply

$$D(\frac{dX}{Z}, \frac{dY}{Z}) \tag{2.2}$$

One might notice that dividing each coordinate by Z also matches up with our intuition—objects seem smaller the farther away they are. After calculating this position, a simple check can be used to decide whether the coordinates are within the screen or not.

2.2 Application

3 Transformation

A 3D graphics engine that only projected points and provided no methods for manipulating them would not be so much an 'engine' as a simple window into a given space. The types of transformations that will be covered by this investigation are translations, stretches, and rotations. Now that we have defined the method of projection, all calculations may be done for $\mathbb{R}^3$ with no regard for the position of the viewer or the display plane. To project these points, the origin O will simply be shifted to the viewer's position using

$$P(P_x, P_y, P_z) \Rightarrow P'(P_x - O_x, P_y - O_y, P_z - O_z)$$

and the points will be projected using the described equations.

3.1 Homogenous coordinates

3.2 Translations

The simplest of these transformations, translations along lines are achieved by adding to or subtracting from any of the x, y, and z values of a point. If a point is to be translated along a 3D vector $\mathbf{V}$, the translation is simply

$$P(P_x, P_y, P_z), \begin{bmatrix} \mathbf{V}_x & \mathbf{V}_y & \mathbf{V}_z \end{bmatrix} \Rightarrow P'(P_x + \mathbf{V}_x, P_y + \mathbf{V}_y, P_z + \mathbf{V}_z). \tag{3.1}$$

3.3 Stretches

While translations can be performed on individual vertices, stretches are only unique from them when applied on multiple vertices, generally forming an object.