Linear Algebra Analysis of How Desmos 3D Graphing Tool Projects 3D Graphs onto 2D Screens

Aryo Wisanggeni – 13523100[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523100@mahasiswa.itb.ac.id, [2]arrrryow@gmail.com

***Abstract*— Graphing calculators are essential tools for visualizing mathematical concepts, especially in areas like trigonometry, calculus, and matrices. While traditional calculators focus on 2D graphs (with two variables, x and y), 3D graphing calculators address functions with more than two variables. Since most modern devices use 2D screens, 3D graphs are displayed using perspective projection, a method pioneered by Filippo Brunelleschi in the 1400s. One of these graphing calculators, Desmos effectively uses perspective to render 3D objects on 2D displays in a comprehensible manner. This study explores the mathematical principles, particularly linear algebra, that enable such projections in Desmos. These concepts include the three main transformations that are used: view transformation, projection transformation, and camera transformation, all using matrices for implementations. In addition to that, these ideas underpin not only graphing calculators but also broader applications in computer graphics and visualization.**

***Keywords*—Linear Algebra, Desmos, Graphing Calculator, 3D Object Projection to Screens.**

I. Introduction

Graphing calculators are tools to perform complex calculations, plot graphs, and analyze data. These calculators are especially helpful for those that are learning math topics like involving trigonometry, calculus, matrices, and many others. This is because a lot of mathematical topics are function based and can be visualized. These visualizations help students grasp mathematical concepts intuitively.

The most basic type of graphing calculator is a graph for two dimensional functions. These are functions that only have two variables, usually x and y, where x is the free variable, whose value is decided by the input, and y is the dependent variable, whose value depends on x. The graph that visually represents these two variable functions will be laid out in a 2D layout, with the horizontal axis representing the x value, while the vertical axis represents the y value. It is no mystery that these graphs will also fit on most screens, as most screens of this age are also 2D.

Though as expected, the 2D dimension is far too constricting, there are many functions that have more than just two variables. Looking at the practical implementation, the only other higher order dimension that can be naturally perceived by humans are 3D objects. This means that 3D graphing calculators are possible as long as they are placed in the three-dimensional world. Now the other challenge presented with implementing these 3D graphing calculators would be how to make it comfortable for developers and learners to use. The obvious choice is to represent it in a 2D perspective as the devices that we have now phones, laptops, actual calculators, and many others use a 2D display to show users the data presented.

Though this presents another problem: how would we make 3D objects appear on 2D screens? Thankfully this is not difficult problem to solve. Since early 1400s, Filippo Brunelleschi created what is known to be the first painting to use linear perspective, The Baptistery in Florence. This painting is proof that since long ago humans are able to represent 3D objects in 2D mediums, with the help of perspective.

This implementation has been successfully done by several modern software. An example of this, for 3D mathematical graphing calculator, is Desmos. Desmos is an online graphing calculator that provides both 2D and 3D graphs for its users. Like other 3D modelling software, it is deploying the same perspective strategy to successfully visualize those 3D object to 2D screens in a way that is understandable.

This study delves into the mathematical principles that make projecting 3D objects onto 2D screens possible, focusing on how linear algebra enables this transformation. By exploring concepts such as matrix operations, transformations, and perspective projection, this study will demonstrate how 3D models are flattened into 2D views while preserving depth and spatial relationships. Understanding these techniques not only sheds light on how tools like Desmos achieve their functionality but also provides a foundation for applications in computer graphics and visualization.

II. Theoretical Framework

A. Matrix

A matrix is a rectangular array of numbers. The horizontal lines of numbers form rows and the vertical lines of numbers form columns. A matrix with m rows and n columns is said to be an $m \times n$ matrix (“an m by n matrix”) [1].

The entries of an $m \times n$ matrix are indexed as follows:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}.$$

That is, a_{32} means "the number in the third row and second column."

Figure 1. Example representation of matrix and its elements [1]

1. Matrix Arithmetic

a. Matrix Addition

Let A and B be an m × n matrices. The sum of A and B is a single m × n with each of its element's value being the sum of elements of A and B that has the same location.

$$\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}.$$

Figure 2. Resulting matrix from the sum of matrix A and B [1]

b. Matrix Scalar Multiplication

Let k be a scalar and A be an m × n matrix. The resulting matrix from the multiplication A with k is the multiplication of each element of A with k.

$$\begin{bmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{bmatrix}.$$

Figure 3. Resulting matrix from the scalar multiplication of matrix A with scalar k [1]

c. Matrix Multiplication

Let A be an m × r matrix, and let B be an r × n matrix. The matrix product of A and B, denoted A · B, or simply AB, is the m × n matrix M whose entry in the i-th row and j-th column is the product of the i-th row of A and the j-th column of B.

$$AB = \begin{bmatrix} \vec{a}_1\vec{b}_1 & \vec{a}_1\vec{b}_2 & \cdots & \vec{a}_1\vec{b}_n \\ \vec{a}_2\vec{b}_1 & \vec{a}_2\vec{b}_2 & \cdots & \vec{a}_2\vec{b}_n \\ \vdots & \vdots & \ddots & \vdots \\ \vec{a}_m\vec{b}_1 & \vec{a}_m\vec{b}_2 & \cdots & \vec{a}_m\vec{b}_n \end{bmatrix}.$$

Figure 4. Resulting matrix from multiplication of matrix A and B, a_n is n-th row of A and b_n is n-th column of B [1]

2. Operations on Matrices

a. Transpose

Let A be an m × n matrices. The transpose of A, denoted A^T is the n × m matrix whose columns are the respective rows of A [1].

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.$$

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}.$$

Figure 5. Example matrix A (up) and its corresponding transpose (down)

3. Linear Transformations

a. Translation

Let x be an object in the n-th dimension. An object being composition of lines that connect points in the n-th dimension. A translation moves every point of x by adding an offset vector with dimension of n.

b. Rotation

A rotation turns the whole space around a pivot by a certain degree.

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

Figure 6. Standard matrix to rotate through an angle θ with the origin (0, 0) as a pivot for 2D

c. Reflection

A reflection matrix transforms every vector into its image on the opposite side of a mirror.

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Figure 7. Standard matrix to rotate about the y-axis.

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Figure 8. Standard matrix to rotate about the x-axis.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Figure 9. Standard matrix to rotate about the y = x line.

B. Transformations

Two fundamental ideas that are necessary to analyze this problem is view volume and canonical view volume. View volume refers to the space that the camera can see within the 3D scene. Anything outside this volume won't show up in the final image we generate. An example of this can be a 3D image of a character in an open field, the view volume of this scene will only consist of this character and the scenery near him and behind him that is not covered, anything else will be cut down from the view volume as it is unnecessary during that scene. On the other hand, canonical view volume is a view volume that has been transformed into a cube. This will make it easier to project 3D to screens.

The transformations that occur to project a 3D model to a 2D screen consists of three main transformations. These being model view transformation, projection transformation, and viewport transformation. The flow of these transformations can be summed up by the figure below.

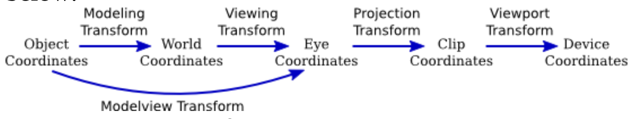

Figure 10. Transformations to project 3D objects to screens [2].

1. Model View Transformation

   This step consists of two smaller transformations: modeling transformation and viewing transformation.

   Modeling transformation changes object coordinates to match the world coordinates. Object coordinates are coordinates that are used for drawing the object. This object then is applied to the scene (world coordinate) by setting its size, orientation and position. This step can be neglected and be taken for granted in projecting the object to screens. As this step just sums up the process of building the 3D scene that will be shown on screen.

   On the other hand, viewing transformation is the process of transforming world coordinates to camera/eye coordinates. Camera/eye coordinates are the coordinate system for making a picture of the world as seen by a viewer. This means that we take the coordinate system of the world/scene and transform it to match the viewer's coordinate system. This is necessary as how the world is seen is completely dependent on the viewer, such as where the viewer is looking and the tilt of the viewers head for example.

2. Projection Transformation

   Projection is the process that converts the view volume into a canonical view volume. There are two types of projection: orthographic projection and perspective projection.

   a. Orthographic Projection

      This projection ensures that the size of an object is the same regardless of its distance from the camera.

   b. Perspective Projection

      This projection will scale objects based on its distance from the camera. Thus, objects that are further away from the camera be smaller and objects that are closer be bigger.

3. Viewport Transformation

   This is the last transformation to take place before the image is shown to the user. In this step, the canonical view volume generated from the projection transformation will be translated to the camera's viewport, or in other words, converting the canonical view volume into the screen coordinates.

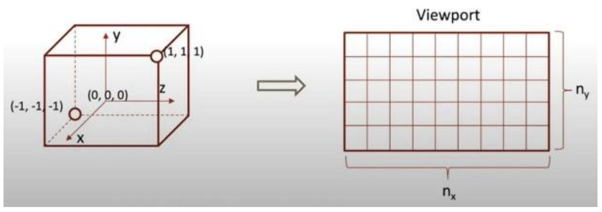

Figure 11. Transformation illustration from canonical view volume (CVV) to screen coordinates/viewport [3].

In fig 11., nx represents the screen's width and ny stands for the screen's height. To map from CVV to Viewport, points within the [-1, 1] range along the x-axis to fit within the range of [-0.5, nx-0.5], points within the [-1, 1] range along the y-axis to fit within [-0.5, ny-0.5], and points within the [-1, 1] range along the z-axis to fit within [0, 1]. The choice to start the range at 0.5 is because the origin of the screen's coordinate system is in the middle of the pixel located at the upper left corner. If the length of a pixel is 1, then the starting point of the screen's coordinates would be at -0.5.

III. Implementation of Theory in Desmos

A. View Transformation

As stated before, this step requires aligning the world coordinate with the camera coordinate. To do this transformation, a matrix called the ‘camera matrix’ is used.

$$M_{cam} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 12. Camera matrix [3].

This matrix that is used for alignment has two steps. The first step is translation (the right matrix of fig 12.) and the second is rotation (the left matrix of fig 12.). The multiplication product of those transformations creates the camera matrix. Another representation of the camera matrix is a simplifying it to become a 3 × 4 rather than a 4 × 4, which is done by multiplying the camera matrix above with the identity matrix.

$$\left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array} \right)$$

Figure 13. Camera matrix with different notation, R is the 3 × 3 matrix for rotation and t is the 3D translation vector.

$$\mathbf{y} \sim \mathbf{C}_0\, \mathbf{x} = \left(\begin{array}{c|c} \mathbf{I} & \mathbf{0} \end{array} \right) \left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array} \right) \mathbf{x}' = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \end{array} \right) \mathbf{x}'$$

Figure 14. Simplified camera matrix (C_0) with 3 × 4 dimension.

The conversion of world coordinates is done by multiplying each point of the world with the camera matrix. Let x’ be a point P in the world coordinate, then the view transformation of point x’ will be as follows, with x as the world coordinate that is aligned with the camera coordinate system.

$$\mathbf{x} = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array} \right) \mathbf{x}'$$

Figure 15. View transformation of point x’.

In Desmos, this transformation is done with the center of the world coordinate always aligned with the center of the camera coordinate.

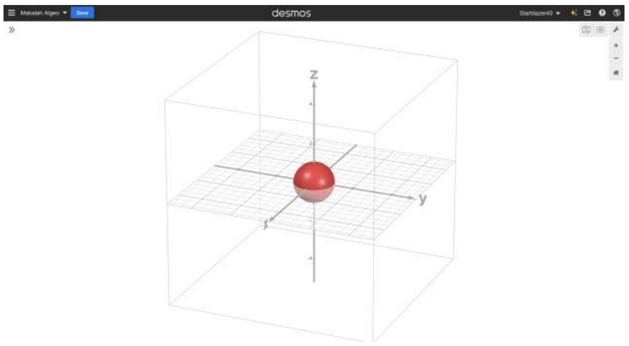

Figure 16. World coordinate center and camera coordinate center aligned in Desmos.

B. Projection Transformation

Here is demonstration of how Desmos projects the objects to the screen.

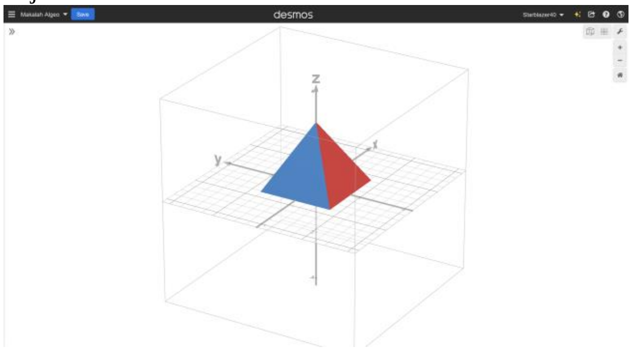

Figure 17. Desmos projection – normal view.

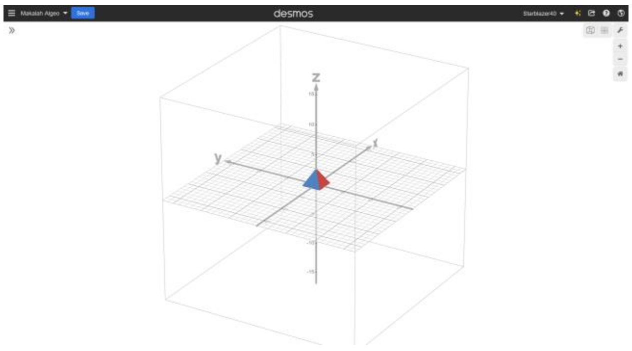

Figure 18. Desmos projection – zoomed out view.

From fig 17. and fig 18., it can be seen that Desmos is using perspective projection, as when it is zoomed out and effectively make the object further away from the screen, the smaller the object appears.

To get the final perspective transformation matrix, there are two transformation steps alongside their corresponding matrices. The first step is to convert the perspective view volume to an orthographic view volume, this can be done by using the orthogonal projection matrix. This step is needed to ensure farther object are compressed more than closer objects. Furthermore, the second step is to convert the orthographic view volume we got from the first step into the canonical view volume. This second step uses the same matrix for standard orthographic projection.

$$M_{p2o} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & f+n & fn \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Figure 19. Matrix to convert perspective view volume to orthographic view volume[3] .

$$M_{orth} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 20. Matrix to convert orthographic view volume to canonical view volume [3].

For fig 19. matrix, n is the focal length or the distance from the viewpoint to the near plane. Viewpoint is the camera or viewer, while near plane is the nearest boundary of the view volume (in this case, the perspective view volume). Meanwhile, f is the distance from the viewpoint to the far plane. The far plane is the farthest end of the view volume from the viewpoint or the farthest distance from the camera where objects are rendered. The matrix works by scaling x, y values by n and scaling z or depth by f+n and additionally adding fn. While the -1 is used to for perspective divide, that is preparing each axis to be in the normalized device coordinate [4].

On the other hand, for fig 20. Matrix, same as before, n refers to the distance from the camera to the near plane and f refers to the distance from the camera to the far plane. The other variables, r, l, t, and b refer to the boundaries of the orthographic view volume, with l = left or the minimum x-coordinate, r = right or the maximum x-coordinate, b = bottom or the minimum y-coordinate, t = top or the maximum y-coordinate. Note that the values of l and b are usually negative because they are on the other side of the origin. This matrix works by scaling each axis x, y, and z with the values of the diagonal of the matrix. While the three-dimensional vector on the right of the matrix shifts the x, y, and z axis to the center of the x, y, and depth range of the screen [4].

The product of the two matrices mentioned above will produce the final matrix to do perspective projection transformation

$$M_{per} = M_{orth} M_{p2o}$$

Figure 21. How the perspective projection matrix is formed [3].

$$M_{per} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Figure 22. Perspective projection matrix [3].

Please note that the references at the end of this document are in the preferred referencing style. Give all authors' names; do not use "*et al*." unless there are six authors or more. Use a space after authors' initials. Papers that have not been published should be cited as "unpublished" [4]. Papers that have been submitted for publication should be cited as "submitted for publication" [5]. Papers that have been accepted for publication, but not yet specified for an issue should be cited as "to be published" [6]. Please give affiliations and addresses for private communications [7].

C. Viewport Transformation

The final step is to convert the canonical view volume produced by the projection transformation into a format that is mapped on 2D screens. This is done with the viewport matrix. What the matrix does is scale x, y axis to be normalized device coordinates which ranges [-1, 1] and also translates it with the right-most vector, in the image below the image width is nx and the screen height is ny. The same goes for the z-axis, but here z is fixed to be 1 as z or depth will be lost in projecting the 3D object to the screen.

$$M_{vp} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x - 1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y - 1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

Figure 23. Viewport matrix [3].

In Desmos, it can be seen that the entire view volume is fitted on the screen meaning that the viewport transformation was completed. In addition to that, pay attention to the loss of elements due to the compression of the z axis. This is also a characteristic that is brought by the projection of 3D to 2D and is implemented in the viewport matrix.

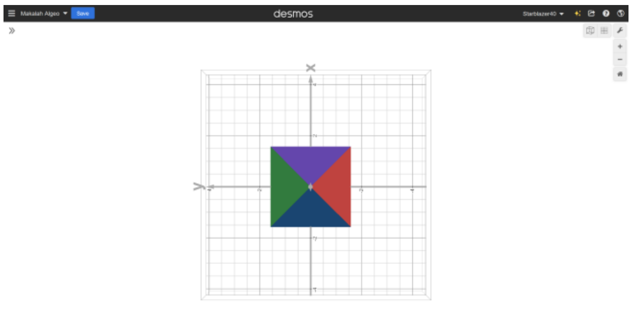

Figure 24. Top view of the graph reveals four sides of the pyramid.

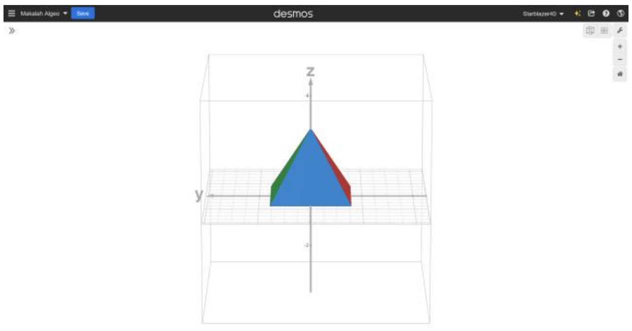

Figure 25. Camera shifts to the side of the graph, hides the purple side of the pyramid because z-axis compression.

IV. Analysis

First is to take a look into the first transformation, the view transformation. This transformation is to align the world/scene to the camera coordinate system. Desmos does this by first translating the world so that the center origin of the graph aligns with the center of the origin of the camera. It then rotates the world according to the position of the camera, whether the camera is looking from below, above, and any other perspective.

The second transformation is the perspective projection transformation. It’s clear that Desmos uses this type of projection from how the scale of the object changing depending on how zoomed in or zoomed out the camera is. This feature to allow users to zoom is definitely a necessary feature to allow users to look at the finer details of the graph by zooming in or look at the general shape of the graph by zooming out. Interestingly the process to get the perspective projection requires two steps. One of which uses the orthographic projection matrix, this means that in order to get the perspective projection, it is required to go through the orthographic projection transformation. The other step to this perspective projection transformation is transforming the perspective view to the orthogonal view. This was done by scaling the perspective view with the conversion matrix.

The last step was to conform the canonical view volume to the camera’s dimension of 2D. This was achieved by rely on a transformation matrix that compressed the third dimension that was not present in the camera’s dimension. This also allowed for realistic 3D representation by hiding structures that are behind other objects and make objects that take up the third dimension seems to gradually shrink smoothly, which made for the perspective effect.

In short, this study focuses on examining how Desmos effectively implements the projection of 3D graphs onto 2D screens, bridging the gap between complex mathematical functions and user-friendly visualization. By analyzing the underlying principles of perspective projection and the mathematical techniques employed, such as matrix transformations and linear algebra, this study aims to shed light on the processes that make such visualizations possible. Future studies may delve deeper into the mathematical frameworks and algorithms that play a critical role in rendering 3D models, particularly in applications where mathematical precision and theoretical rigor are prioritized over implementation.

V. Conclusion

In conclusion, using basic linear algebra, specifically using matrix transformation, the mechanism that Desmos uses for projecting its 3D graph to screens can be solved. In short, Desmos applies all three steps of 3D object projection to 2D screens. Using viewing transformation to match the world with the camera’s coordinates. Perspective projection transformation to differentiate object scales based on their distance from the screen. Lastly, Desmos uses viewport transformation to correctly compresses the third dimension that is not present on the screen, allowing realistic representation of 3D object to 2D screens.

VI. Acknowledgment

The completion of this paper could not have been possible without the help of all IF2123 lecturers, especially Dr. Ir. Rinaldi Munir, M.T. whom has taught class K02 2024 for linear and geometric algebra and provided a great amount of learning resources for students to study from. The author hopes this paper to serve as a useful reference, not only for others interested in the relevant field of study but also as a resource for the author's future work.

References

[1] G. Hartman, *Fundamentals of Matrix Algebra*, 3rd ed. Virginia Military Institute, Department of Mathematics and Computer Science, 2011, pp. 1–136.
[2] D. J. Eck, *Introduction to Computer Graphics*, Aug. 2023, sec. 3.3, "Projection and Viewing." [Online]. Available: https://math.hws.edu/graphicsbook/c3/s3.html
[3] Skann.ai, "Projecting 3D Points into a 2D Screen," *Medium*, 2023. [Online]. Available: https://skannai.medium.com/projecting-3d-points-into-a-2d-screen-58db65609f24.
[4] Clemson University, "Chapter 10: Orthographic and Perspective Projection," *Course 405 Notes*. [Online]. Available: https://people.computing.clemson.edu/~dhouse/courses/405/notes/projections.pdf.

[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-18-Ruang-vektor-umum-Bagian4-2023.pdf, accessed at 27 December 2024.

STATEMENT

Hereby, I declare that this paper I have written is my own work, not a reproduction or translation of someone else's paper, and not plagiarized.

Bandung, 27 December 2024

[illegible]

Aryo Wisanggeni 13523100