Homogeneous Coordinates

Group 80

June 21, 2023

Linear Algebra

Karan Nijhawan
Daksh Shah
Sairam Babu

Abstract

This research report explores & dives deep into the topic of linear transformations in homogeneous coordinates within the realm of linear algebra. The report covers various types of linear transformations and their representation in homogeneous coordinates. Additionally, it discusses the integration of homogeneous coordinates in computer graphics, modeling, and perspective imaging. The report concludes with a comprehensive overview of the research conducted, including collaborative efforts & individual contributions.

What are Homogeneous Coordinates ?

Homogeneous coordinates are a mathematical framework used in projective geometry and computer graphics. They extend Euclidean coordinates by adding an extra coordinate, typically denoted as w. This allows for the representation of points at infinity and simplifies geometric transformations. They are widely used in computer graphics, computer vision, and robotics. In this research project, we will explore the theory, applications, and most importantly, practical implementations of homogeneous coordinates

The key idea behind homogeneous coordinates is that a point (x,y) in Euclidean space can be represented as (wx,wy,w), where w is a non-zero scalar. The coordinate (wx,wy,w) is called the homogeneous representation of the point. By varying the value of w, different points in Euclidean space can be represented using the same homogeneous coordinates. This concept is often referred to as projective equivalence.

Homogeneous coordinates allow for the representation of points at infinity by assigning $w = 0$. This property is particularly useful in computer graphics, where the representation of parallel lines and vanishing points becomes straightforward. Additionally, homogeneous coordinates simplify geometric transformations, such as translation, rotation, scaling, and perspective projection, by representing them as simple matrix multiplications.

The advantages of using homogeneous coordinates include:

- **Representation of points at infinity:** with homogeneous coordinates, it becomes possible to represent points at infinity as (x, y, 0) or (x, y, 0, w) in 2D and 3D, respectively. This representation is useful in projective geometry and computer graphics, as it allows for handling parallel lines and perspective effects.
- **Homogeneous Clipping:** Homogeneous Coordinates are used in clipping algorithms, which determine which parts of a geometric object are visible within a given viewing region. Clipping in homogeneous space simplifies the process and improves efficiency.
- **Simplified transformations:** geometric transformations can be expressed as simple matrix multiplications, making them more intuitive and easier to implement
- **Simplicity and versatility:** provide a unified representation for points, lines, and transformations, enabling concise and elegant mathematical formulations.

Aim of the Project

We aim to do research on variety of the fields in real world where HomoCoords find their usage.
It also focuses on the use of geometric transformations.

There are different kinds of geometric transformations:

- Scaling
- Rotation
- Translation
- Shearing
- Projection
- Reflection

We will be talking majorly about 3 geometric transformations used extensively in the above mentioned fields, namely: **scaling**, **rotation** & **translation**

TRANSLATION

Translation refers to moving an object in a certain direction without changing its size/shape. It is done by adding constant values to the coordinates of the object. Let's consider a 2D system where the object's coordinates are represented as (x, y). In homogeneous coordinates, we can represent it as (x, y, 1) denoted by V.
Suppose we want to translate this position vector by 2 units in the positive x-direction and 1 unit in the negative y-direction. We can achieve this by using a respective "translation matrix" T in homogeneous coordinates, such that:

$$V \cdot T = V'$$

where V' represents the final homogeneous coordinates after translation.
The translation matrix T can be represented as:

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

Here, tx = 2 and ty = -1 are the translation values. This leads to the final equation for a single translation:

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

Similarly, if we have to perform multiple translations, denoted as T1, T2, and T3, we can write the final homogeneous coordinates as V':

$$V \cdot Tf = V'$$

Here, Tf represents the combined translation matrix for all translations, given by $Tf = (T1 \cdot T2 \cdot T3)$. Each Ti represents the respective translation matrix for the ith translation.

ROTATION

Rotation, as a transformation, involves rotating an object around a fixed point or axis while preserving its size and shape. Similar to translation, rotation can also be represented in homogeneous coordinates using a matrix.
The rotation matrix R can be represented as:

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

Here, θ represents the angle by which the object is rotated. If we want to rotate an object with homogeneous coordinates [x, y, 1] by an angle θ, we can use the following equation:

$$V \cdot R = V'$$

Here, V' represents the final homogeneous coordinates after rotation. Substituting the values, the equation becomes:

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Similar to translation, multiple rotations can be handled by finding a final rotation matrix that is multiplied with the homogeneous coordinates.

SCALING

Scaling leads to a change in the size of an object by either enlarging or shrinking it. This is achieved by multiplying the coordinates of the vector by a scalar value. Scaling can also be represented in matrix form.

The scaling factors can be denoted in matrix form as:

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

Here, Sx represents the scaling factor along the x-axis, and Sy represents the scaling factor along the y-axis. For example, if we have a position coordinate [x, y, 1] undergoing scaling, and we want to enlarge it to twice its size along the x-axis while halving it along the y-axis, we can set Sx = 2 and Sy = 0.5. Substituting the values into the equation, we get:

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

The above calculation provides us with the final homogeneous coordinates of the object after scaling.

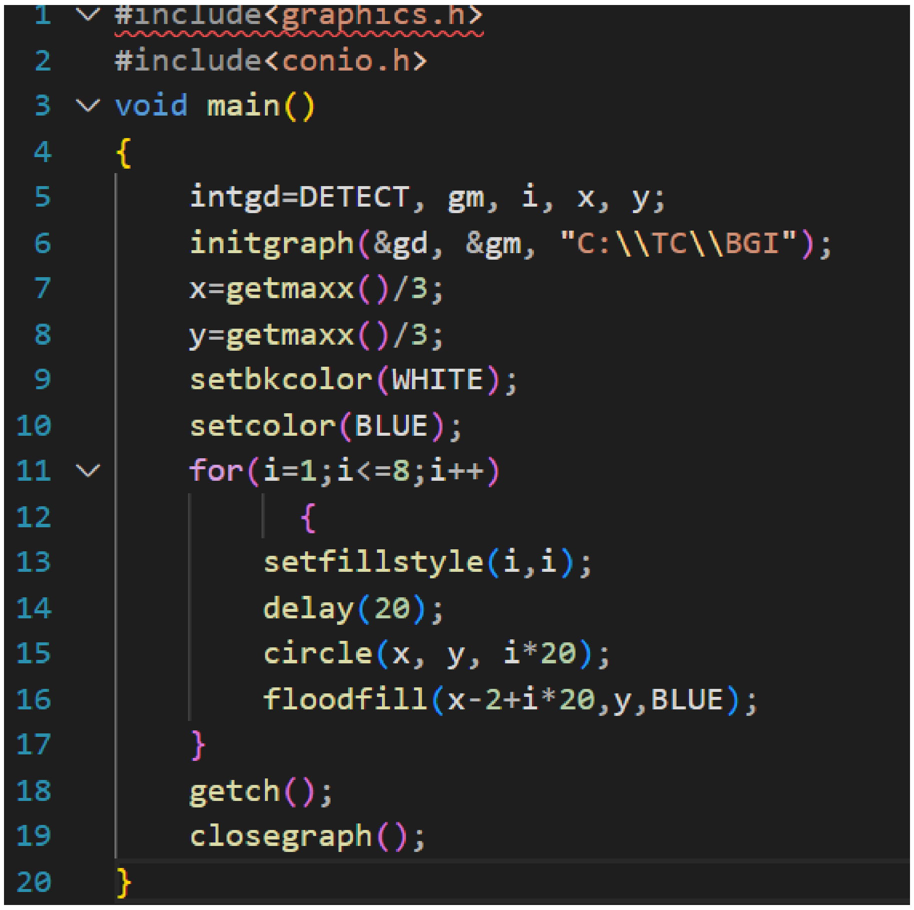
Computer Graphics

Homogeneous coordinates play a critical role in Computer Graphics(CG) by providing a unified framework for representing points, lines, and transformations. They enable efficient rendering of 3D objects, handling of perspective projection, and efficient computation of transformations such as translation, rotation, and scaling.
Transformations like scaling, rotation and translations are used to design graphics. Interesting colorful patterns & shapes are made with the help of these transformations.

Basic Structures:

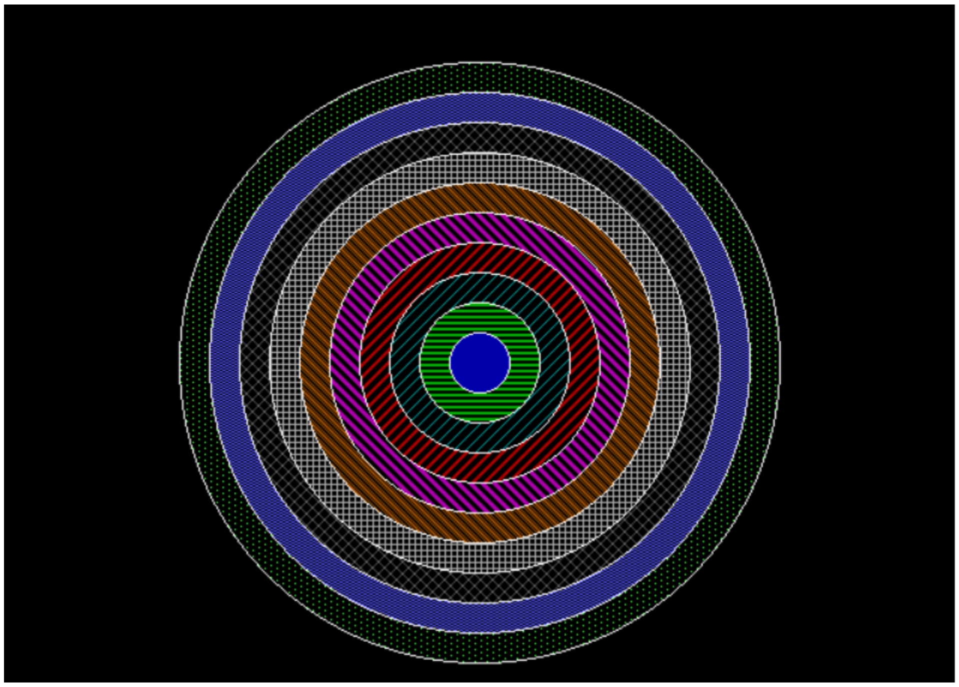
- *Line*: a point is translated and copied repeatedly to form a line
- *Polygons*: a line is rotated at different angles and its different copies are translated in such a way that vertices join
- *Circle*: similar to a line, the points are translated and copied such that it forms a circle

Making patterns: The following code prints different circle by scaling it by different factors and filling each circle with a different colour



```
#include<graphics.h>
#include<conio.h>
void main()
{
    intgd=DETECT, gm, i, x, y;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    x=getmaxx()/3;
    y=getmaxx()/3;
    setbkcolor(WHITE);
    setcolor(BLUE);
    for(i=1;i<=8;i++)
          {
        setfillstyle(i,i);
        delay(20);
        circle(x, y, i*20);
        floodfill(x-2+i*20,y,BLUE);
    }
    getch();
    closegraph();
}
```

The below image is the output of the above code and as we can see, it has produced multiple circles of different increasing sizes as well as filled them with different colors.

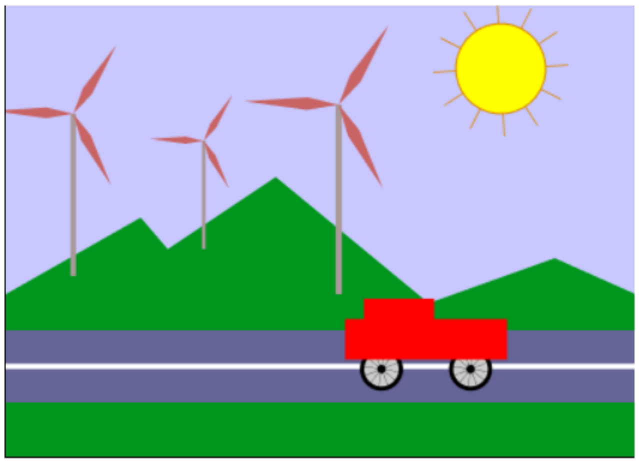


Using different patterns, objects are drawn and then combined together to make a complex pattern usually referred as a **complex scene**

Complex scenes

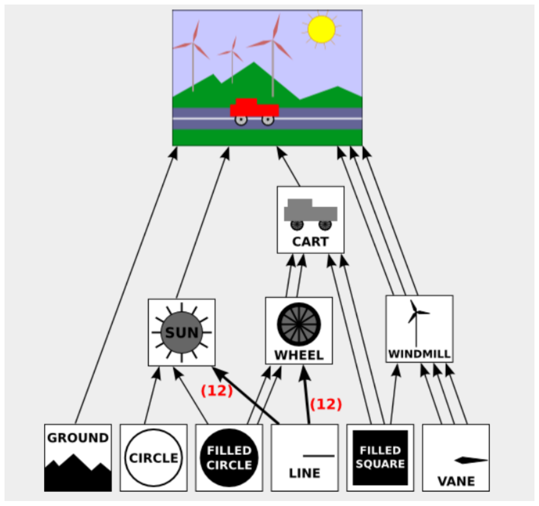
Using linear transformation with the help of homogeneous co-ordinates, we can even apply 3D animations to the CG.

A sample scene:



- Logically, the components of a complex scene form a structure. In this structure, each object is associated with the sub-objects that it contains
- If the scene is hierarchical, then the structure is hierarchical. This structure is known as a scene graph
- A scene graph is a tree-like structure, with the root representing the entire scene, the children of the root representing the top-level objects in the scene, and so on

We can visualize the scene graph for our sample scene:

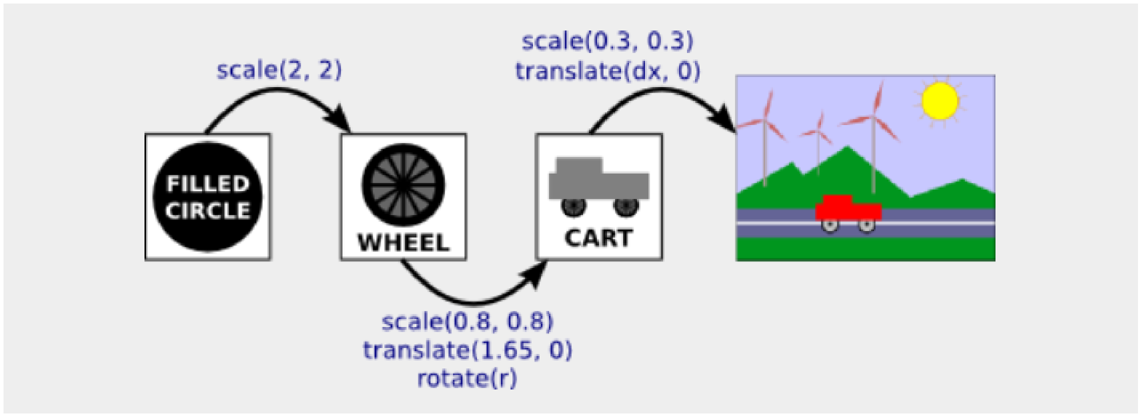


Adding Animations

The Transformation Stack:

- Suppose that you write a subroutine to draw an object. At the beginning of the subroutine, you use a routine such as **saveTransform()** to save a copy of the current transform.
- At the end of the subroutine, you call **restoreTransform()** to reset the current transform back to the value that was saved.
- Now, in order for this to work correctly for hierarchical graphics, these routines actually use a stack of transforms.
- Further, one subroutine can call other subroutines. This means that several drawing subroutines can be active at the same time, each with its own saved transform.
- When restoreTransform() is called, it is the most recently saved transform that should be restored.

The Transformation stack works as follows:



- We first make the shape that is required in its natural coordinate system centred at Origin
- Then we first scale it to whatever actual size we require
- We also rotate it according to the angles required before translating as these 2 operations don't require any movement with reference to the centre
- Further, we finally translate it to the required position in the final scene

3D Modelling

Modeling: HomoCoords provide a powerful mathematical framework for modeling, allowing for accurate representation, efficient geometric transformations, perspective projection, and intersection calculations.
Using this we can transform 3-D space to 2-D linearly and then work accordingly.

Modeling here refers to usage in several domains:

- Medical Industry: *Medical Industry uses the transformations to get a scanned 3D body organ on a 2D screen, still looking as 3D, making it easy to be analysed by the doctor*
- 3D Printing: *In 3D printing, transformations and homogeneous coordinates are used to accurately position, resize, and manipulate 3D models before they are printed to keep the model within the printing volume. Additionally, they are used in mesh manipulation to deform and align the model as needed.*

Homogeneous coordinates represent a point (x,y,z) in 3D space using a four-component vector (x',y',z',w), where x', y', and z' are the scaled versions of x, y, and z respectively, and w is a scaling factor. The relationship between Cartesian coordinates (x,y,z) and homogeneous coordinates (x',y',z',w) can be expressed as:

$$x' = w \cdot x$$
$$\text{z'=w}\cdot z$$
$$\text{y'=w}\cdot y$$

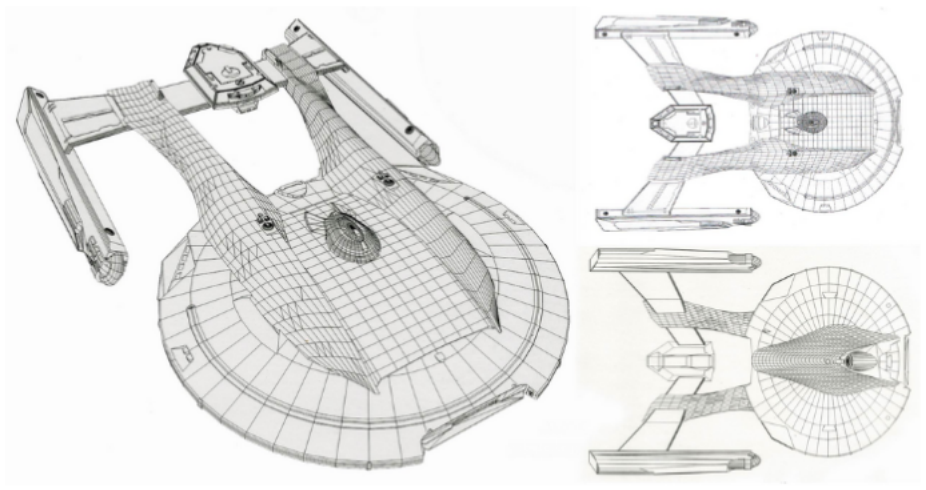
The advantage of using homogeneous coordinates is that they allow for the efficient representation and manipulation of points, vectors, and transformations through matrix operations.

1. **Translation:** Homogeneous coordinates simplify translation operations by representing a translation vector as $(tx,ty,tz,1)$ and multiplying it with the homogeneous coordinate vector (x',y',z',w). The resulting vector $(x'+tx, y'+ty, z'+tz, w)$ represents the translated point.
2. **Scaling:** Homogeneous coordinates make scaling operations straightforward. By multiplying the homogeneous coordinate vector (x',y',z',w) with a scaling matrix, the resulting vector $(s\cdot x', s\cdot y', s\cdot z', w)$ represents the scaled point.
3. **Rotation:** Homogeneous coordinates allow for efficient rotation operations as well. A rotation matrix can be multiplied with the homogeneous coordinate vector to obtain the rotated point.
4. **Projection:** Homogeneous coordinates are particularly useful for perspective projection, which converts 3D points into 2D coordinates. By dividing the homogeneous coordinate vector (x',y',z',w) by its w-component, we obtain the projected 2D point $(x'' = \frac{x'}{w}, y'' = \frac{y'}{w}, z'' = \frac{z'}{w})$. This process allows for perspective effects, such as objects appearing smaller as they move away from the viewer.

Working:

- A 3D model of the objects in the scene is created
- The model is converted into many small polygons in 3D that approximate the surfaces of the model
- The polygons are transformed via a linear transformation to yield a 2D representation that can be shown on a flat screen

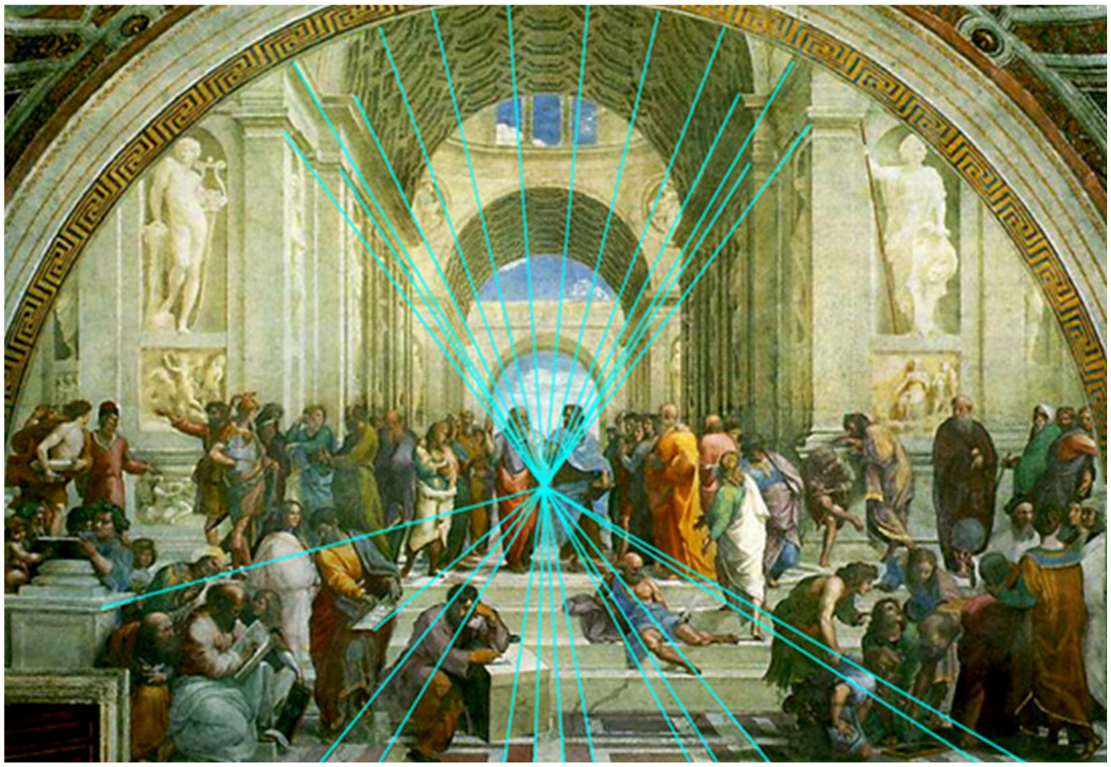
These steps led to a blueprint of the 3d object in the 2d space, where it can be easily designed and then converted back to 3d space. One such blueprint is as given below:



- Initially, object models may be expressed in terms of smooth functions like polynomials.However the first step is to convert those smooth functions into a set of discrete pieces – coordinates and line segments.
- All subsequent processing is done in terms of the discrete coordinates that approximate the shape of the original model.The reason for this conversion is that most transformations needed in graphics are linear.
- Expressing the scene in terms of coordinates is equivalent to expressing it in terms of vectors, eg, in R^3. And linear transformations on vectors are always matrix multiplications, so implementation is simple and uniform.
- The resulting representation consists of lists of 3D coordinates called faces. Each face is a polygon.The lines drawn between coordinates are implied by the way that coordinates are grouped into faces

Perspective Imaging(a link back to computer graphics):

- There is another nonlinear transformation that is important in computer graphics. Homogeneous coordinates allow us to capture this too as a linear transformation in R^4.
- The eye, or a camera, captures light (essentially) in a single location, and hence gathers light rays that are converging.So, to portray a scene with realistic appearance, it is necessary to reproduce this effect.The effect can be thought of as "nearer objects are larger than further objects."



In the above image, the 3D look created by the artist is due to perspective imaging. Closer objects look bigger and the farther, smaller has been kept in mind while drawing the picture.
Similar techniques are used by designers while designing computer graphics and creating 3D avatars & objects

The way to compute the projection is using similar triangles.The triangle in the xz-plane shows the lengths of corresponding line segments.Similar triangles show that

$$(\tfrac{x*}{d} = \tfrac{x}{d-z})$$
$$(x* = \tfrac{dx}{d-z} = \tfrac{x}{1-\frac{z}{d}})$$

Using homogeneous coordinates, we can construct a linear version of T inR^4 To do so, we establish the following convention: we will allow the fourth coordinate to vary away from 1.
The matrix that implements this transformation is quite simple:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{d} & 1 \end{bmatrix} I$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{d} & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ 1 - \frac{z}{d} \end{bmatrix}$$

Using these techniques we can create 3D effects and 3D characters on the computer screen.

3D Modeling in Medical Science

Medical Science and its various other research fields have had huge applications for Linear Algebra. Fields that involve image visualization and especially **Modern Medical fields** like radiology work with processes that are primarily based on linear algebraic transformations.

In this project we will delve into the methodology of Tomographic Reconstruction used in Computed Tomographic (CT) scans.

Computed Tomography

What is Tomography?

- It is the idea or concept of reconstruction of an object, via obtaining the object in terms of slices.
- The way we do this procedure with humans is by passing photons through a human body part and upon doing so, we can study the rates absorption of those photons.
- Its analogue would be akin to studying an object from its shadow from various heights. The phenomenon used here is something called *"attenuation"*.
- We pass X-rays through the human body and at every point, we measure the intensities of these X-rays after they pass out through the body part.

Homogeneous Co-ordinates in Tomography

- HomoCoords are simply the very basis of tomography. We obtain various projections of an image and from those projections, we apply a linear combination of them to reconstruct something similar to the original image.
- So two major homo-cord applications are present in tomography(scaling,projections).

Additionally one of the major facets of tomographic reconstructions is Volume rendering , which transforms a set of 2D images into a 3D object(Transformations). Reconstructions can also be used to transform the dimensions of objects obtained.(Using a combination of lower dimensional slices to reconstruct a higher dimensional object).

Mathematical understanding behind CT scans

Now CT constructions are obtained via obtaining multiple 'slices', as illustrated in the above image , we simply obtain multiple of these 'slices'. So lets study each slice. In the below diagram , we see how each slice (in this case the transverse projection) can be visualised.

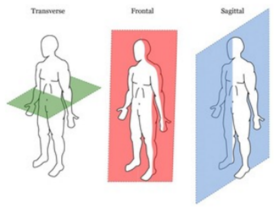


Figure 1: A row of 'pixels' CT-scans

Now lets study each of these slices , each slice is a grid/matrix , each image would be a grid of pixels , with each pixel having a certain intensity. Now we can start constructing mathematical equations to do so. Lets take an example of a beam of photons passing straight through a row of pixels. It would look like this:

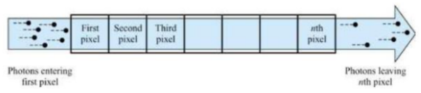


Figure 2: Slices in CT-scans

The photons constituting the X-ray beam are absorbed by the tissue within the pixel at a rate proportional to the X-ray density of the tissue. Quantitatively, the X-ray density of the jth pixel is denoted by x_j and is defined by

$$x_j = \log\left(\text{fraction of photons passing through the } j\text{th pixel without being absorbed}\right)$$

Since logarithmic functions are additive, we have

$$x_1 + x_2 + x_3 + \ldots + x_n = \log(\text{total fraction of photons passing through a row of } n \text{ photons})$$

Now let's look at each beam. In a CT scan machine, we can have two main things: emitters and receptors. There's an array of receptors, each of which obtains the intensity of a beam. Since we obviously know the intensity of the beam when it's produced, we can obtain the relative change in intensity of the beam as some constant b_i.

Now, these are values that we obtain, and hence we can use these as solutions to these equations.

Therefore, for the ith beam:

$$x_1 + x_2 + x_3 + \ldots + x_n = b_i$$

However, the projections we obtain are for various angles and are most definitely not only linear. So what we actually obtain is a linear combination of these pixels.

$$a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + \ldots + a_{in}x_n = b_i$$

So we can have m such equations for M projections.

This is simply a system of linear equations !.

After this , we obtain the images in a slice .These images are then fit together using whats known as a Radon transform.

As usual we could simply use decomposition methods to decrease computational time taken to solve this set of linear equations.

However there lies another major point, the dimensions of the matrix obtained , does have physical significance in estimating the efficiency of tomographic reconstructions. With M>N , we have something called an over-determined system , essentially the number of beams outweigh the number of pixels/receptors.

These situations do arise and there exists computational methods to solve said systems(method of least squares).So linear algebra does have a huge weight in computerized tomography .

Conclusion

In conclusion, this project explored the concept of homogeneous coordinates in computer graphics and 3D modeling. Homogeneous coordinates provide an efficient mathematical representation for performing various transformations and calculations in 3D space. By extending Cartesian coordinates with an additional component, homogeneous coordinates allow for translation, scaling, rotation, and perspective projection operations to be easily performed using matrix operations.

Throughout this project, we examined how homogeneous coordinates simplify and streamline transformations in 3D modeling. We demonstrated how translation vectors, scaling factors, rotation matrices, and perspective projection can be applied to points represented in homogeneous coordinates to achieve desired effects.

Moreover, we highlighted the advantages of using homogeneous coordinates, such as efficient blending of transformations, interpolation, and clipping. By leveraging the power of matrix operations, complex calculations in 3D modeling can be executed more effectively.

This project serves as a comprehensive overview of the role of homogeneous coordinates in computer graphics and 3D modeling. By understanding and implementing homogeneous coordinate transformations, one gains a deeper insight into the underlying mathematical principles behind these applications. Homogeneous coordinates prove to be a valuable tool for representing, manipulating, and rendering objects in three-dimensional space, making them an essential concept for any practitioner or enthusiast in the field.

Overall, this project sheds light on the importance and practicality of homogeneous coordinates in computer graphics and 3D modeling, providing a solid foundation for further exploration and advancement in the field.

Work Distribution

For this research project on homogeneous coordinates, we divided the work among our team members based on our individual expertise and interests. The distribution of work was as follows:

- **Daksh Shah:** Led the mathematical aspect of the project, focusing on the understanding and application of linear algebra in homogeneous coordinates.
  He explored the theoretical foundations, derived equations and matrices for transformations as well as worked out on custom-made examples that could be incorporated in the project for better understanding.
  He also built a python notebook respective to each topic, the notebook is attached to the report.
- **Karan Nijhawan:** Conducted extensive research in the field of computer graphics, investigating techniques and algorithms used in rendering, shading, and lighting calculations.
  He explored how homogeneous coordinates integrate with computer graphics, creating visually appealing 3D scenes and animations, some of which are also included in the project as examples. He also researched about the perspective imaging and how it is used in computer graphics and 3d modelling.
- **Sairam Babu:** Explored the intersection of homogeneous coordinates with medical sciences and electrical component design. He researched over the applications of homogeneous coordinates in medical imaging, focusing on 3D modeling and visualization.

Throughout the project, all team members collaborated & worked very closely, sharing their findings, insights, and code implementations.
This collaborative approach ensured the accuracy and coherence of the overall research project. In this way, we are aiming to produce a comprehensive research project that encompasses the mathematical foundations as well as highly useful & trending real-world applications of HOMOGENOUS COORDINATES!

References

1. https://www.intechopen.com/chapters/43595
2. Role of Medical Imaging and it's Applications in Health Care Monitoring. -Dr Hamsapriye
3. https://www.cs.bu.edu/fac/crovella/cs132-book/L13ComputerGraphics-Spring2021.html
4. https://www.math.utah.edu/ treiberg/Perspect/Perspect.htm
5. https://math.hws.edu/graphicsbook/c2/s3.html