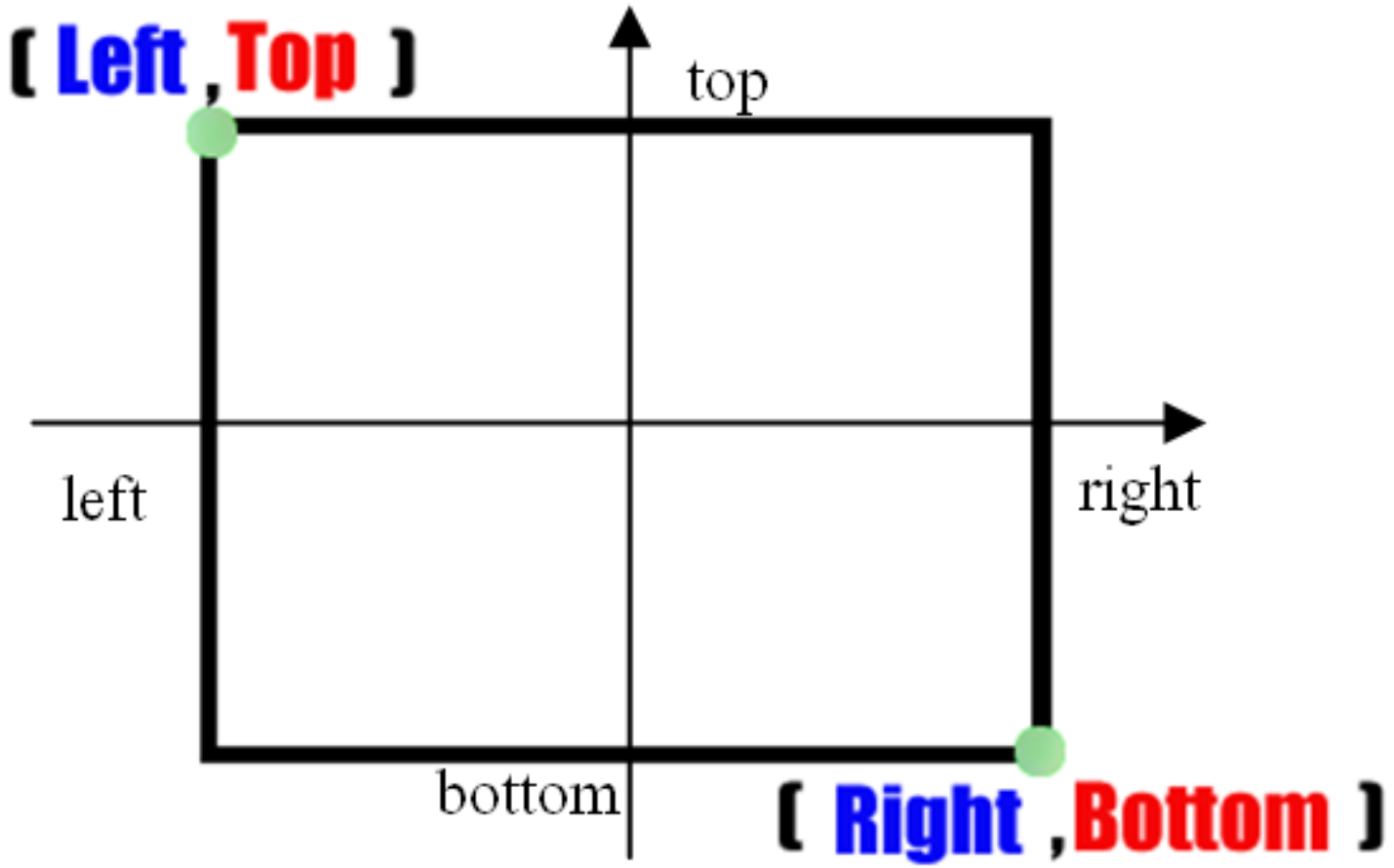
Vector & Matrix

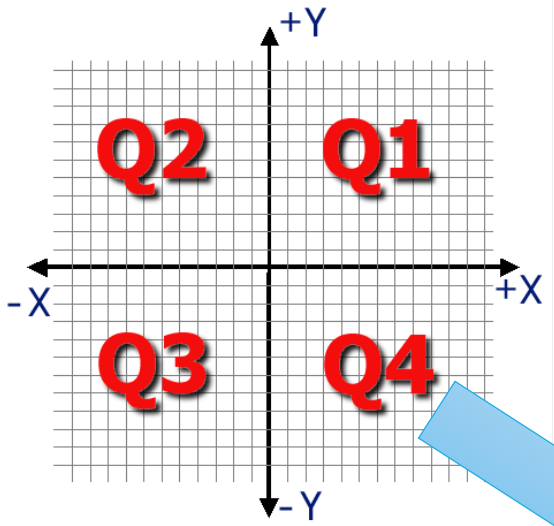
Jarut Busaratjid

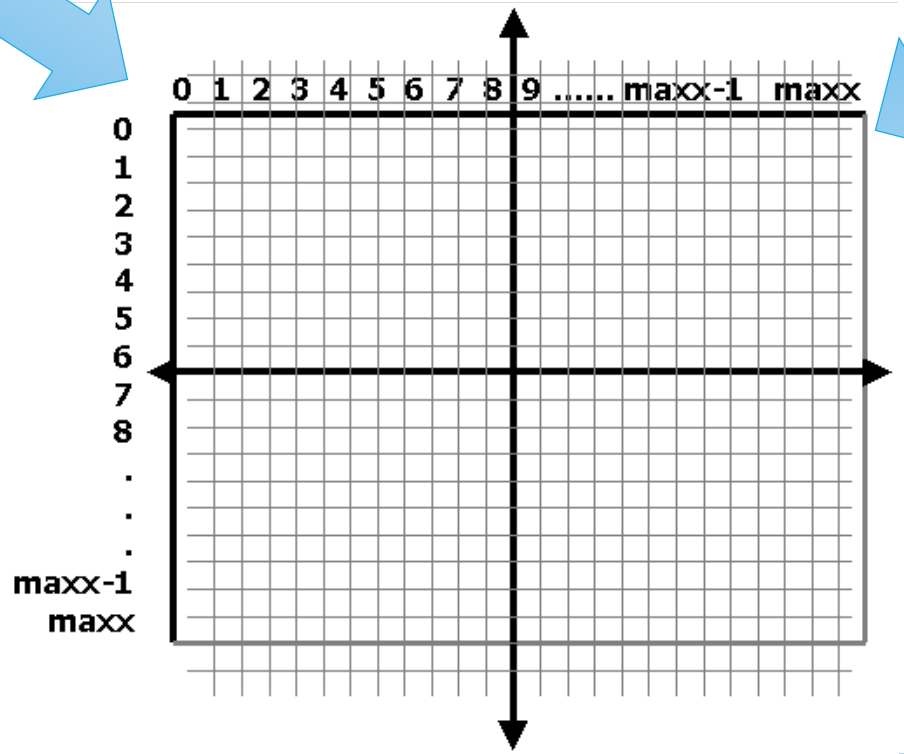
IT@PBRU

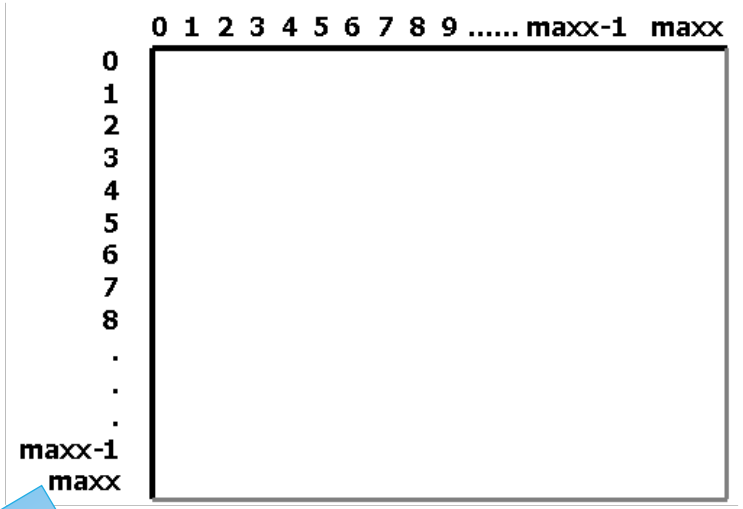
มุมมอง 2 มิติ

การแสดงผลของปริภูมิ 2 มิติอาศัยการอ้างอิงตำแหน่งของจุดของรูปทรงหรือวัตถุ โดยใช้ค่า ตำแหน่งของวัตถุบนแกน X กับแกน Y ซึ่งการกำหนดช่วงค่าของค่าทางซ้าย (left หรือค่าที่น้อยที่สุด ของแกน x), ขวา (right หรือค่าที่มากที่สุดของแกน x), ล่าง (bottom หรือค่าที่น้อยที่สุดของแกน y) และด้านบน (top หรือค่าที่มากที่สุดของแกน y)


(Left , Top)
top
left
right
bottom
(Right , Bottom)


+Y
Q2
Q1
-X
+X
Q3
Q4
-Y


0 1 2 3 4 5 6 7 8 9 ..... maxx-1 maxx
0
1
2
3
4
5
6
7
8
.
.
.
maxx-1
maxx


0 1 2 3 4 5 6 7 8 9 ..... maxx-1 maxx
0
1
2
3
4
5
6
7
8
.
.
.
maxx-1
maxx

วิธีการคำนวณการแปลงพิกัดจากปริภูมิ 2 มิติเป็นพิกัดจอแสดงผล

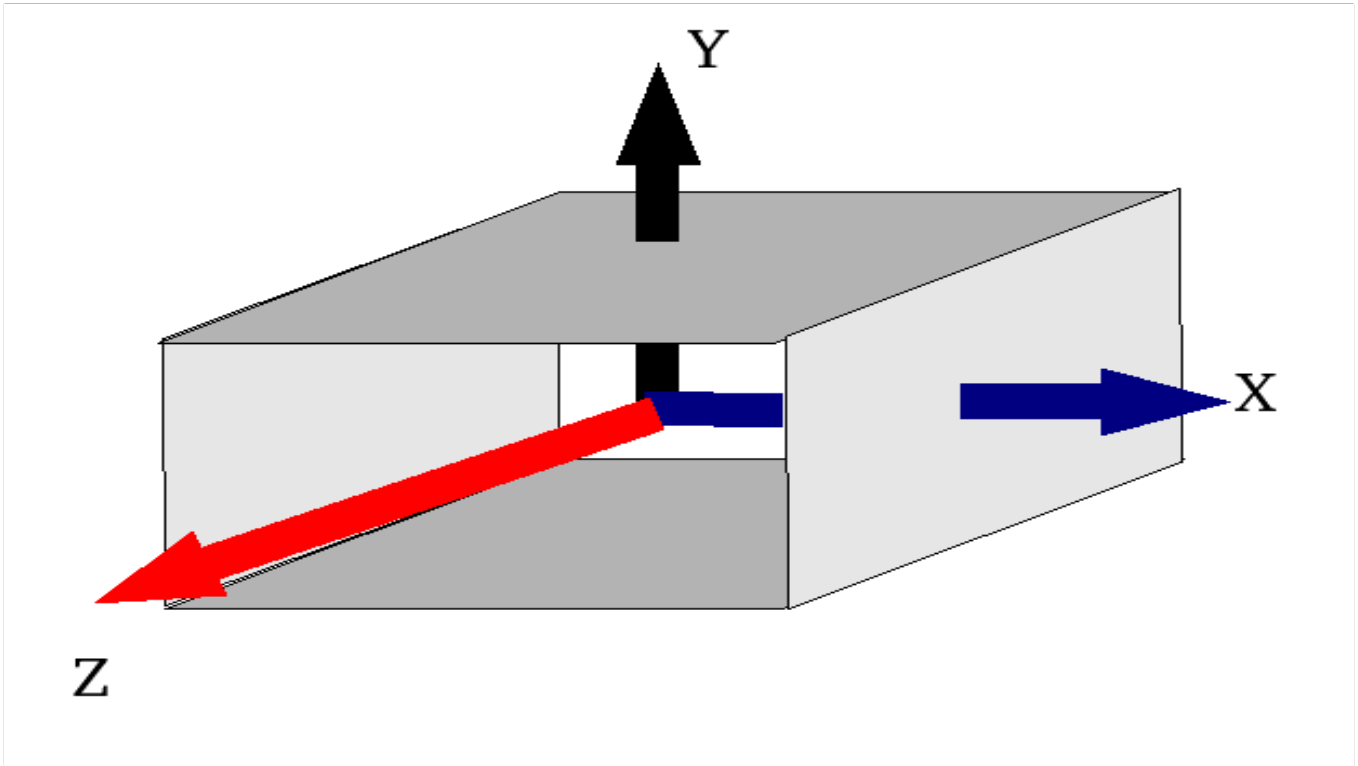
$$S_X = \left[\frac{\max_X}{2}\right] + X$$

$$S_Y = \left[\frac{\max_y}{2}\right] - Y$$

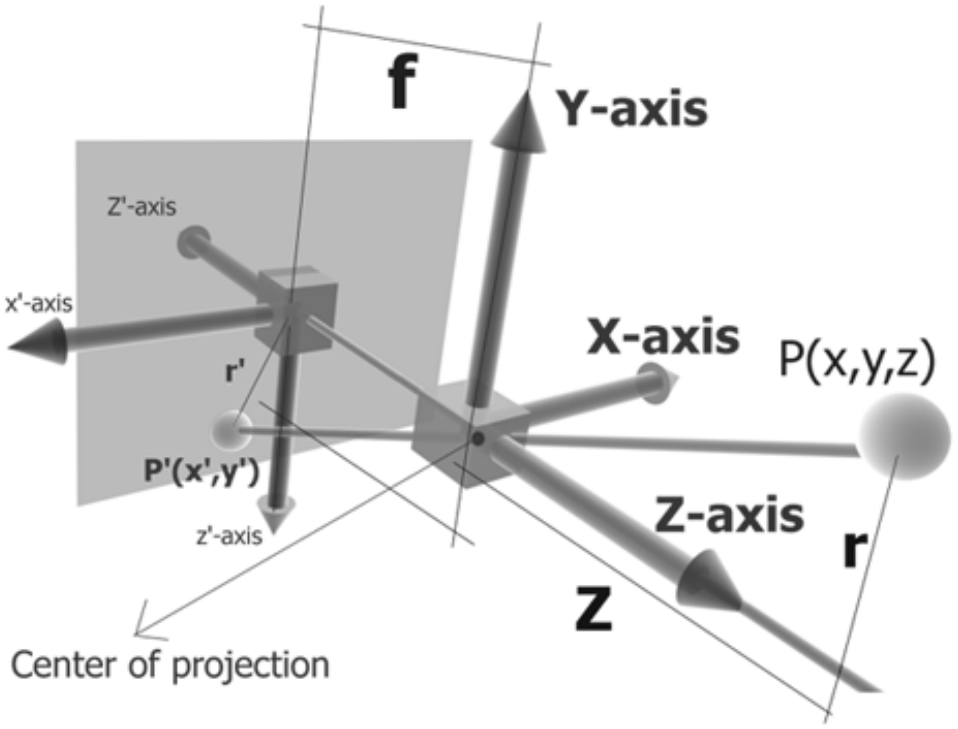
S_X คือ ค่าของจุดบนแกน X ของจอแสดงผล
S_Y คือ ค่าของจุดบนแกน Y ของจอแสดงผล
max_X คือ ค่าสูงสุดของจุดบนแกน X ของจอแสดงผล
max_Y คือ ค่าสูงสุดของจุดบนแกน Y ของจอแสดงผล
X คือ ค่าแกน X ของจุดบนปริภูมิ 2 มิติ
Y คือ ค่าแกน Y ของจุดบนปริภูมิ 2 มิติ

มุมมอง 3 มิติ

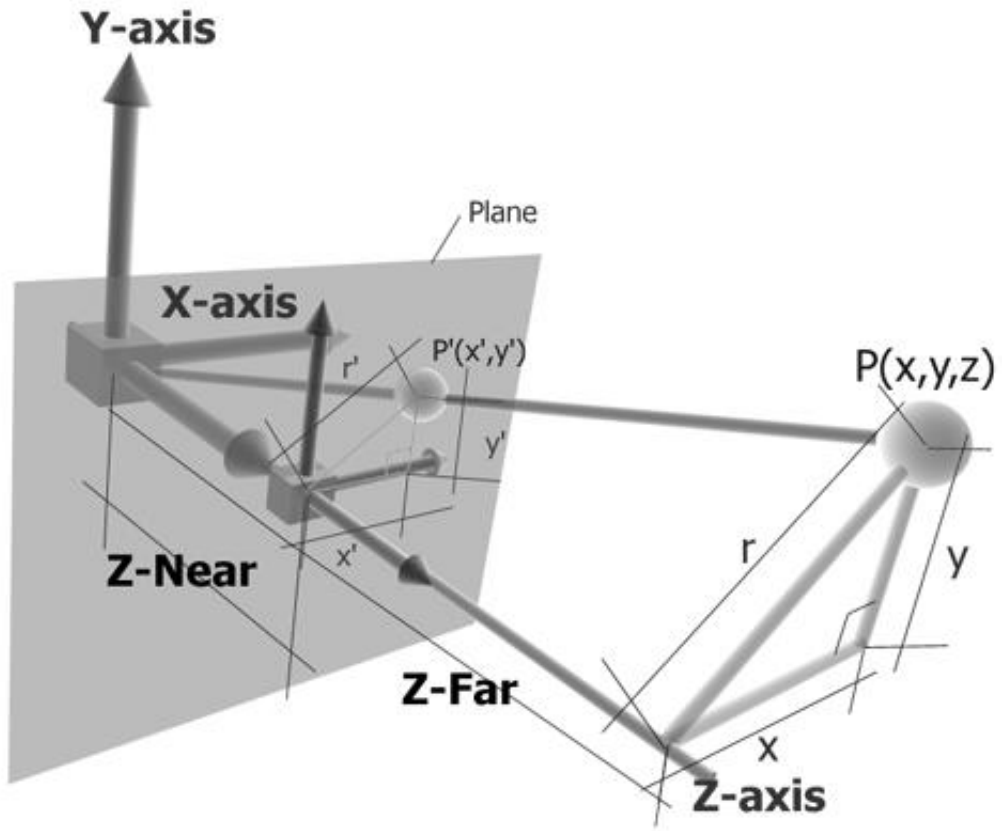
ปริภูมิ 3 มิติอ้างอิงพิกัดของจุดของรูปทรงหรือวัตถุโดยใช้ค่าตำแหน่งในแกน X ค่าตำแหน่ง ในแกน Y และค่าตำแหน่งในแกน Z เป็นค่าอ้างอิง


Y
X
Z

นำข้อมูลในปริภูมิ 3 มิติมาแสดงผลในจอแสดงผลที่เป็น 2 มิติ



จุด P(x,y,z) เป็นจุดในปริภูมิ 3 มิติ เมื่อทำการแปลงจากปริภูมิ 3 มิติเป็นปริภูมิ 2 มิติ หรือการโปรเจ็กชั่น (Projection) มาเป็นจุด P’(x, y) โดยมีจุดกึ่งกลางอยู่ที่ (0,0,0) มีค่า f เป็นระยะห่างระหว่างฉากรับ (จอแสดงผล) กับจุดกึ่งกลาง



วางฉากรับให้อยู่เข้ามาในแกน Z จะได้ว่า z-Near คือค่าความห่างของฉากรับกับจุดกึ่งกลางในแกน Z และ z-Far คือระยะห่างระหว่างจุด P(x,y,z) กับจุดกึ่งกลาง จะได้ระยะห่างของจุด P และระยะห่างของจุด P’ จากจุดกึ่งกลางภาพ

$$Dis\tan ceP = \sqrt{x^2 + y^2 + z^2}$$

$$Dis\tan ceP' = \sqrt{x'^2 + y'^2}$$

อัตราส่วน (Ratio) ของสามเหลี่ยมมุมฉากที่เกิดจากจุด P(x,y,z) ที่กระทำกับแกน Z และจากจุด P’(x’, y’) กับแกน Z เป็นอัตราส่วนของสามเหลี่ยมมุมฉากที่เท่ากัน และกำหนดให้ค่า เป็น z-Far และค่า f เป็นค่า z-Near

$$Ratio = \frac{f}{Z}$$

เมื่อนำอัตราส่วนของสามเหลี่ยมมุมฉากที่เท่ากันมาเขียนใหม่ในจะได้ดังสมการ

$$\frac{x'}{x} = \frac{y'}{y} = \frac{r'}{r}$$

อัตราส่วนของพิกัดในแกน X และ Y ของจุด P และ P’ เป็นดังสมการ

$$\frac{x'}{x} = \frac{f}{z}$$

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

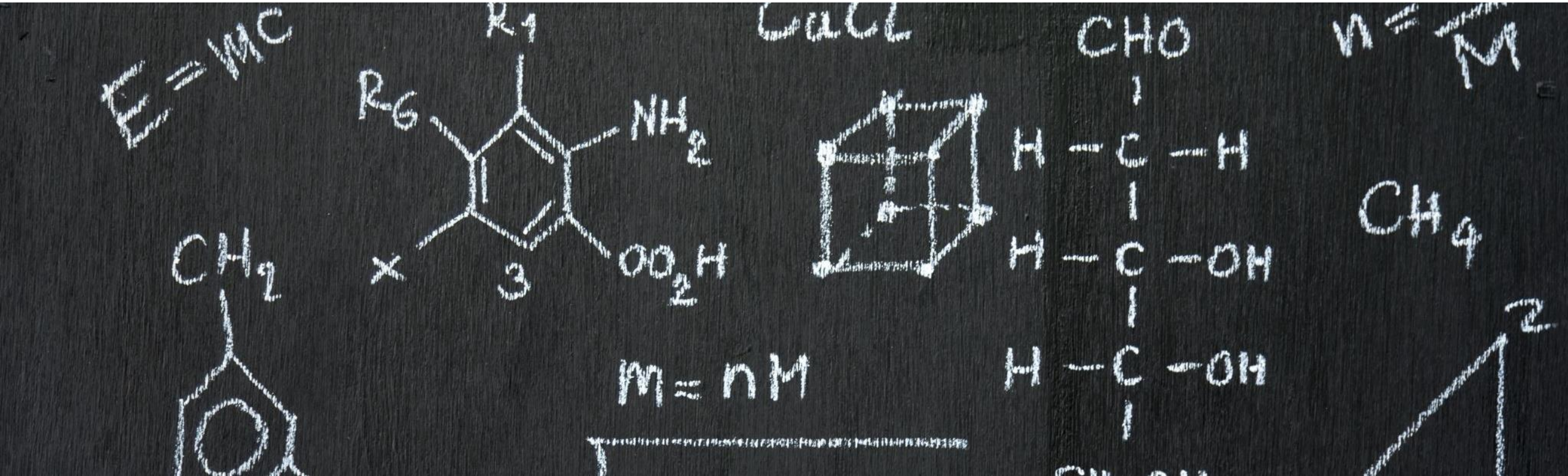
ดังนั้น ค่าของ P′(x′, y′) หาได้จากสมการต่อไปนี้

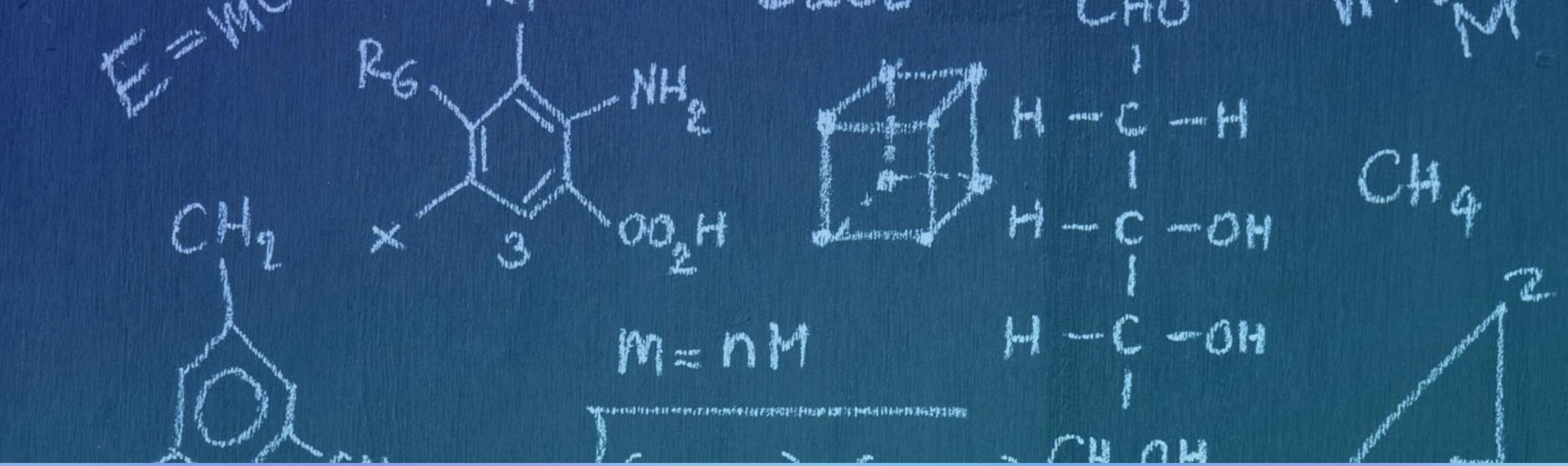
$$x' = \frac{f}{z}x$$

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

เวกเตอร์
(Vector)

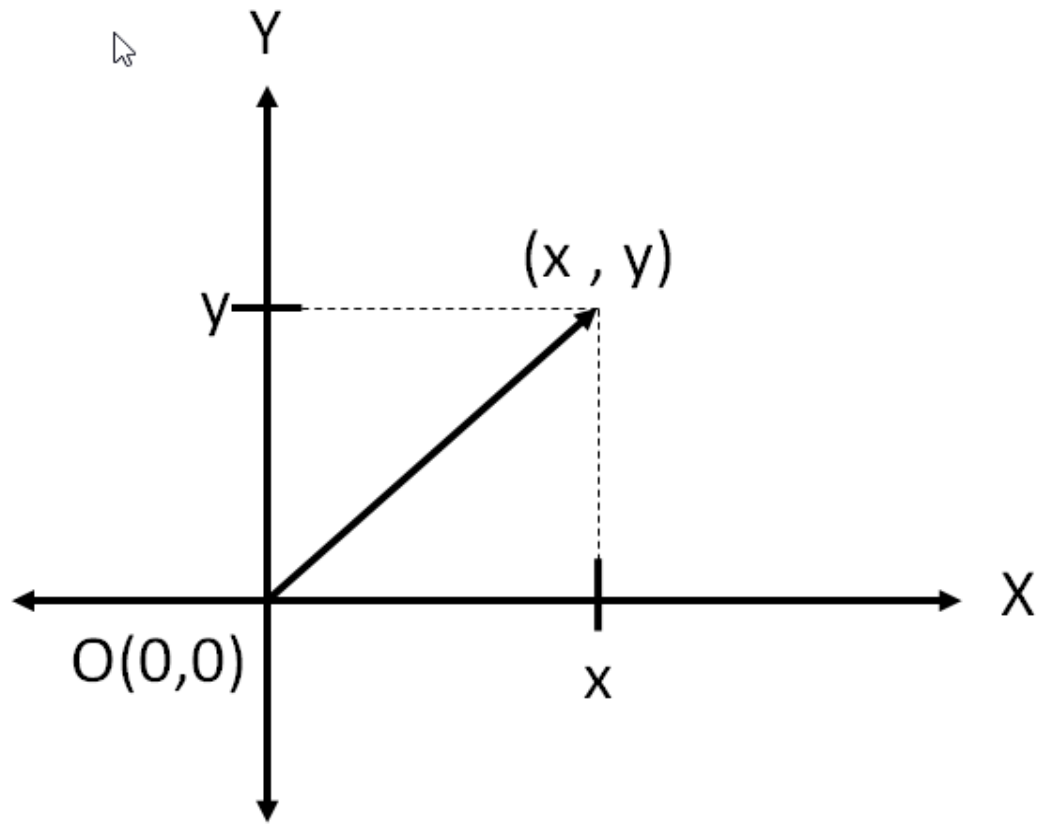
เวกเตอร์เป็นข้อมูลที่บ่งบอกถึง ขนาด และทิศทาง จะตรงข้ามกับข้อมูลแบบสเกลาร์(Scalars) ซึ่งจะบอกเพียงขนาด (เป็นเลขจำนวนจริง) ตัวอย่างของเวกเตอร์ได้แก่ ค่าอัตราเร่ง, แรง, โมเมนทัม (Momentum) เป็นต้น เวกเตอร์ที่นำมาใช้กับคอมพิวเตอร์กราฟิกส์มีทั้งแบบ 2 มิติ และ 3 มิติ





เวกเตอร์ 2 มิติ

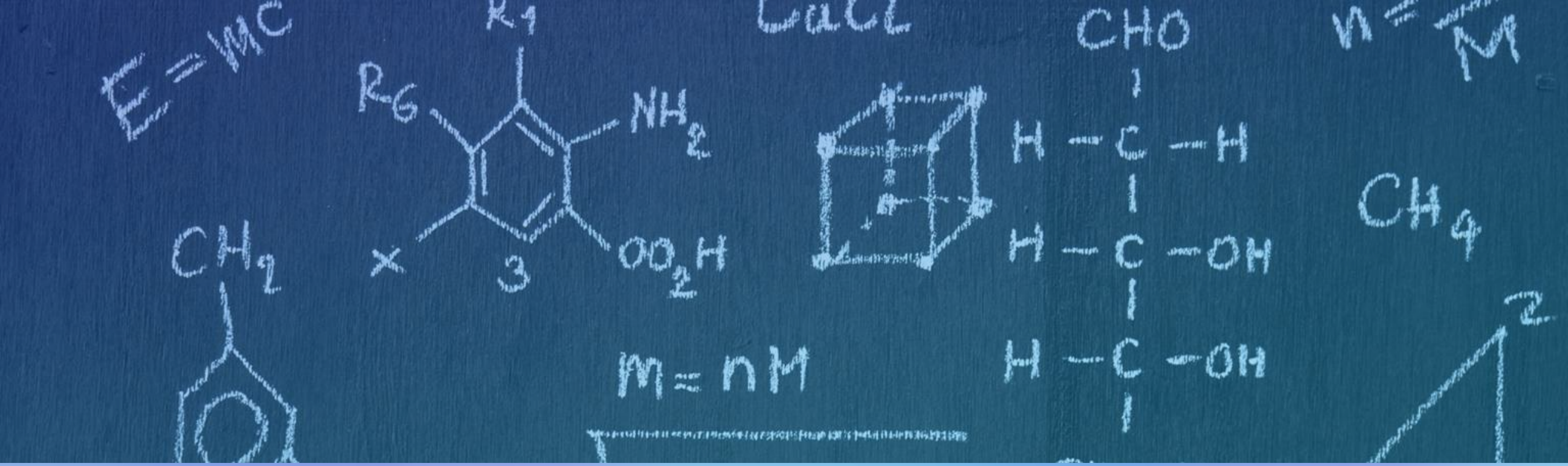
คือ เวกเตอร์ในระนาบ XY โดยแทนเวกเตอร์ v ด้วยส่วนของเส้นตรงที่มีทิศทางจากจุดเริ่มต้น O(0,0) ไปยังจุดสิ้นสุด (x,y)



Vector2

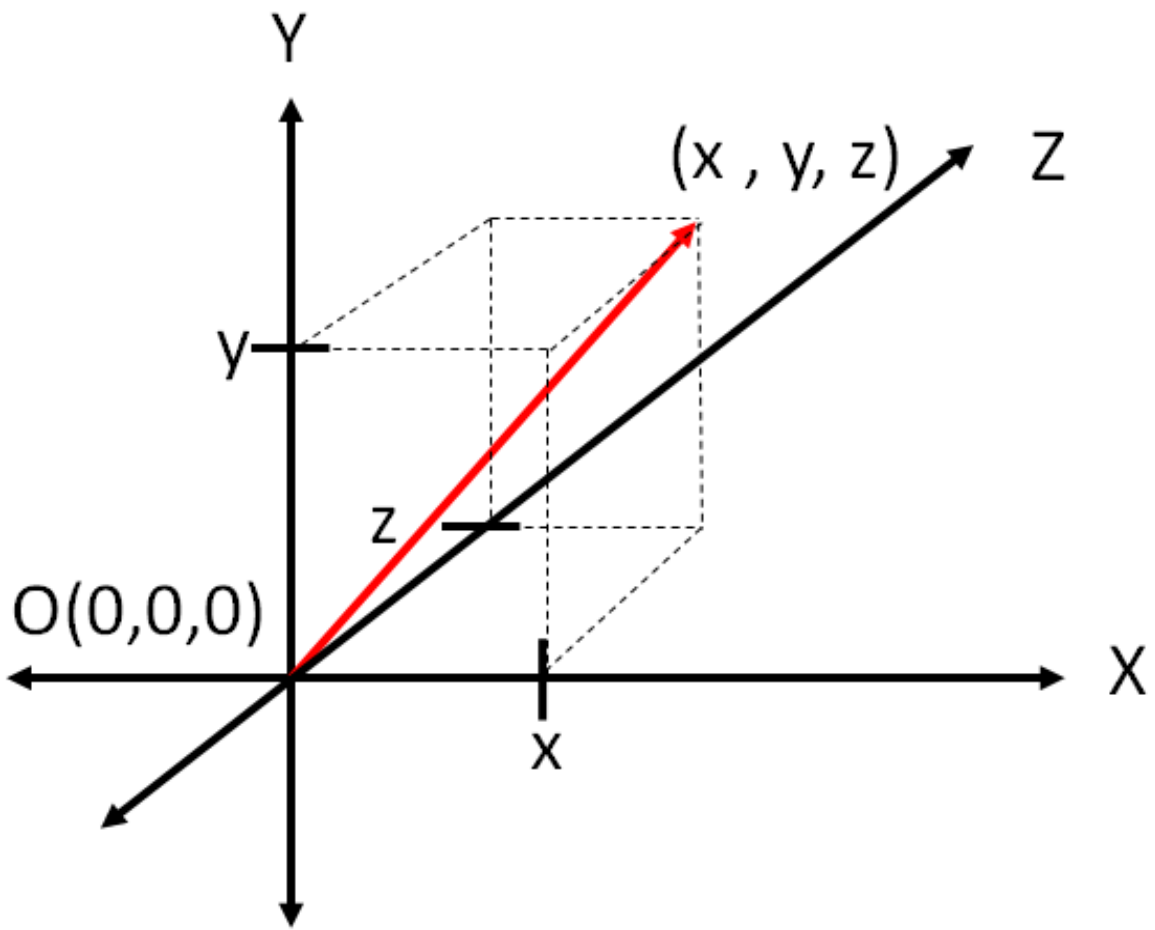
float x

float y



เวกเตอร์ 3 มิติ

คือเวกเตอร์ในระนาบ XYZ โดยแทนเวกเตอร์ v ด้วยส่วนของเส้นตรงที่มีทิศทางจากจุดเริ่มต้น O(0,0,0) ไปยังจุดสิ้นสุด (x,y,z)



Vector3

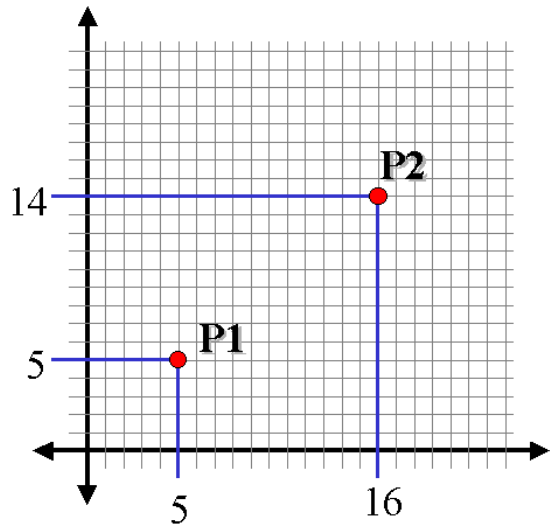
float x

float y

float z

การสร้างเวกเตอร์

กรณีที่กำหนดจุด P1 และ P2 เป็นจุดเริ่มต้น และจุดสิ้นสุดของเวกเตอร์ v จะสามารถคำนวณหาเวกเตอร์ v ได้ตามสมการ



$$v = P2 - P1$$

ขนาดของเวกเตอร์

การหาขนาดของเวกเตอร์ (Magnitude) คำนวณได้ดังสมการต่อไปนี้

$$M = |V|$$

$$= \sqrt{V_x^{\,2} + V_y^{\,2}}$$

$$= \sqrt{V_x^{\,2} + V_y^{\,2} + V_z^{\,2}}$$

การบวกและลบเวกเตอร์

$$
\begin{aligned}
V + U &= \left(V_x + U_x, V_y + U_y\right) \\
&= \left(V_x + U_x, V_y + U_y,, V_z + U_z\right) \\
V - U &= \left(V_x - U_x, V_y - U_y\right) \\
&= \left(V_x - U_x, V_y - U_y, V_z - U_z\right)
\end{aligned}
$$

การคูณและหารเวกเตอร์ด้วยค่าสเกลาร์

การคูณเวกเตอร์ด้วยค่าแบบสเกลาร์เป็นการนำค่าสเกลาร์ (S) ไปคูณกับค่า x และ y ของเวกเตอร์ V

$$
\begin{aligned}
U &= S \times V \\
&= \left(S \times V_x, S \times V_y\right) \\
&= \left(S \times V_x, S \times V_y, S \times V_z\right)
\end{aligned}
$$

การทำผลคูณจุด

การทำผลคูณจุด (Dot Product) เป็นวิธีการนำเวกเตอร์มาคูณกัน ระหว่างค่าในแต่ละแกน หลังจากนั้นนำมาหาผลรวมของผลคูณ

$$V \bullet U = (V_x \times U_x) + (V_y \times U_y)$$

$$= (V_x \times U_x) + (V_y \times U_y) + (V_z \times U_z)$$

การหาค่ามุมระหว่างเวกเตอร์ V และ U

$$V \bullet U = \cos(\theta)$$

$$\theta = \cos^{-1}(\theta)$$

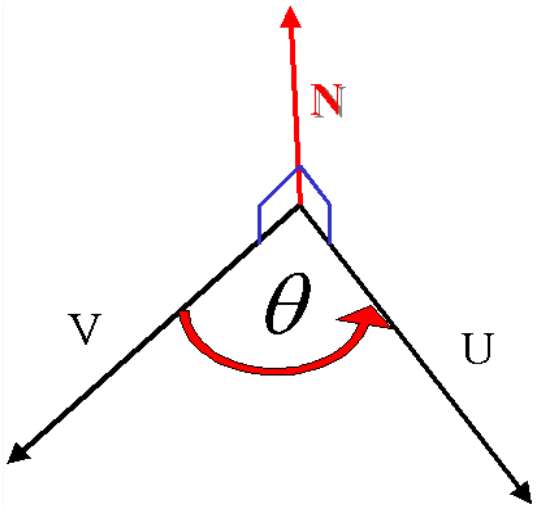
$$\therefore \theta = \frac{1}{V \bullet U}$$

การทำผลคูณไขว้

การทำผลคูณไขว้ (Cross Product) เป็นการคูณกันระหว่าง 2 เวกเตอร์ที่มีจุดเริ่มต้นเป็นจุดเดียวกัน โดย สมการหาผลคูณไขว้ของเวกเตอร์ U กับเวกเตอร์ V

$$U \times V = (U_x \times V_y) - (U_y \times V_x)$$

เมื่อนำเวกเตอร์ 3 มิติ U และ V ที่มีจุดเริ่มต้นอยู่จุดเดียวกันมาคูณไขว้กัน จะได้เวกเตอร์ N ที่เป็นเวกเตอร์ที่ตั้งฉากกับเวกเตอร์ U และ V ดังภาพที่ 2.4 และวิธีคำนวณหา Nx, Ny และ Nz เป็นดังสมการ



$$N = U \times V$$

$$(N_x) = (U_y \times V_z) - (U_z \times V_y)$$

$$(N_y) = (U_z \times V_x) - (U_x \times V_z)$$

$$(N_z) = (U_x \times V_y) - (U_y \times V_x)$$

การแปลงเป็นเวกเตอร์หนึ่งหน่วย

การแปลงเวกเตอร์ขนาดใดๆ ให้เป็นเวกเตอร์ขนาดหนึ่งหน่วย (Normalize vector) หรือ ยูนิตเวกเตอร์ (Unit vector) กระทำเพื่อสร้างเวกเตอร์ที่มีหน้าที่กำหนดทิศทางของวัตถุ ซึ่งวิธีคำนวณเป็นดังสมการต่อไปนี้

$$N = \frac{V}{|V|}$$

เวกเตอร์ใน Unity

เวกเตอร์ (Vector) ใน Unity เป็นตัวแทนของปริมาณที่มีทั้งขนาดและทิศทาง เช่น แรง ความเร็ว หรือตำแหน่งในพื้นที่ 3 มิติ เป็นต้น

เวกเตอร์ถูกนำมาใช้ในการคำนวณต่าง ๆ ใน Unity เช่น การเคลื่อนที่ของวัตถุ การคำนวณแรง การตรวจจับการชน และการสร้างเอฟเฟกต์ต่าง ๆ

ประเภทของเวกเตอร์ใน Unity

- Vector2
  ใช้สำหรับแทนตำแหน่งหรือเวกเตอร์ในระนาบ 2 มิติ (x, y)
- Vector3
  ใช้สำหรับแทนตำแหน่งหรือเวกเตอร์ในพื้นที่ 3 มิติ (x, y, z)
- Quaternion
  ใช้สำหรับแทนการหมุนของวัตถุใน 3 มิติ

การใช้งานเวกเตอร์ใน Unity

- กำหนดตำแหน่งของวัตถุในฉาก
- กำหนดทิศทางการเคลื่อนที่ของวัตถุ
- คำนวณระยะห่างระหว่างสองจุด
- คำนวณแรงที่กระทำต่อวัตถุ
- ตรวจสอบว่าวัตถุสองชิ้นชนกันหรือไม่

```
using UnityEngine;

public class VectorExample : MonoBehaviour
{
    public Transform target; // วัตถุเป้าหมาย

     void Update()
    {
        // หาเวกเตอร์ที่ชี้จากวัตถุปัจจุบันไปยังวัตถุเป้าหมาย
        Vector3 direction = target.position - transform.position;

        // นอร์มัลไลซ์เวกเตอร์เพื่อให้มีความยาวเป็น 1
        direction.Normalize();

        // เคลื่อนที่ไปยังเป้าหมาย
        transform.position += direction * speed * Time.deltaTime;
    }
}
```

เมทริกซ์
(Matrix)

เมทริกซ์ (Matrix) เป็นโครงสร้างข้อมูลชนิดแถวลำดับแบบสี่เหลี่ยม (Rectangular array) หรือ เรียกว่าเป็นแถวลำดับแบบ 2มิติ

การสร้างเมทริกซ์ต้องบอกระบุขนาดของเมทริกซ์ด้วยจำนวนของแถวและจำนวนคอลัมน์ในแต่ละแถวมีจำนวนเท่าใด

กรณีที่เมทริกซ์มีจำนวนแถวและจำนวนคอลัมน์เท่ากัน เราเรียกว่าเมทริกซ์นั้นว่า เมทริกซ์ด้านเท่า (Square matrix)

กรณีที่เมทริกซ์มีจำนวนแถวเพียงแถวเดียว จะเรียกเมทริกซ์นั้นว่า เมทริกซ์แถว (Row matrix)

และกรณีที่เมทริกซ์มีจำนวนคอลัมน์เพียงคอลัมน์เดียว เมทริกซ์นั้นจะเรียกว่า คอลัมน์เมทริกซ์ (Column Matrix)

ในด้านคอมพิวเตอร์กราฟิกส์ต้องนำเมทริกซ์มาประยุกต์ใช้ในการคำนวณเกี่ยวกับการย้ายตำแหน่ง การปรับขนาด การหมุน และการเปลี่ยนมุมมอง

การจัดเก็บโครงสร้างข้อมูลเมทริกซ์

เนื่องด้วยเมทริกซ์ต้องระบุจำนวนแถวและคอลัมน์และเก็บข้อมูลชนิดตัวเลขทศนิยม การออกแบบโครงสร้างข้อมูลเมทริกซ์จึงอาศัยโครงสร้างข้อมูลแบบแถวลำดับ ตัวอย่างเช่น ต้องการจัดเก็บข้อมูลของเมทริกซ์ขนาด 3x3 และ 4x4 สามารถออกแบบโครงสร้างการจัดเก็บข้อมูลได้ดังนี้

```
float mat3x3[3][3];
float mat4x4[4][4];
```

การบวกเมทริกซ์

การบวกเมทริกซ์กระทำได้เฉพาะกรณีที่เมทริกซ์ทั้งสองมีขนาด หรือจำนวนของแถวและคอลัมน์ที่เท่ากัน และผลลัพธ์จากการบวกเป็นการหาผลรวมของแต่ละตำแหน่งของสมาชิกในเมทริกซ์

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

$$N = \begin{bmatrix} 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{bmatrix}$$

$$M + N = \begin{bmatrix} 1+2 & 1+2 & 1+2 \\ 2+3 & 2+3 & 2+3 \\ 3+4 & 3+4 & 3+4 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 3 & 3 \\ 5 & 5 & 5 \\ 7 & 7 & 7 \end{bmatrix}$$

การลบเมทริกซ์

การลบกันของเมทริกซ์ 2 เมทริกซ์กระทำได้ต่อเมื่อเมทริกซ์ทั้งสองมีขนาดหรือจำนวนแถวและคอลัมน์ที่เท่ากัน และผลลัพธ์จากการเมทริกซ์ตัวตั้งลบด้วยเมทริกซ์ตัวลบของแต่ละตำแหน่งในเมทริกซ์

$$M = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

$$N = \begin{bmatrix} 7 & 1 & 4 \\ 8 & 2 & 5 \\ 9 & 3 & 6 \end{bmatrix}$$

$$M - N = \begin{bmatrix} 1-7 & 4-1 & 7-4 \\ 2-8 & 5-2 & 8-5 \\ 3-9 & 6-3 & 9-6 \end{bmatrix}$$

$$= \begin{bmatrix} -6 & 3 & 3 \\ -6 & 3 & 3 \\ -6 & 3 & 3 \end{bmatrix}$$

การคูณและหารเมทริกซ์ด้วยสเกลาร์

การคูณและหารเมทริกซ์ด้วยค่าแบบสเกลาร์เป็นการนำค่าคงที่มาทำการคูณหรือหารกับสมาชิกทุกตัวในเมทริกซ์

$$M = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

$$M \times 2 = \begin{bmatrix} 1 \times 2 & 4 \times 2 & 7 \times 3 \\ 2 \times 2 & 5 \times 2 & 8 \times 2 \\ 3 \times 2 & 6 \times 2 & 9 \times 2 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 8 & 14 \\ 4 & 10 & 16 \\ 6 & 12 & 18 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

$$M \div 2 = \begin{bmatrix} 1 \div 2 & 4 \div 2 & 7 \div 3 \\ 2 \div 2 & 5 \div 2 & 8 \div 2 \\ 3 \div 2 & 6 \div 2 & 9 \div 2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 & 2 & 3.5 \\ 1 & 2.5 & 4 \\ 1.5 & 3 & 4.5 \end{bmatrix}$$

การคูณเมทริกซ์

การคูณเมทริกซ์ด้วยแมทริกซ์มีความแตกต่างจากการคูณเมทริกซ์ด้วยค่าแบบสเกลาร์ คือ เมทริกซ์ที่เป็นตัวตั้งจะต้องมีจำนวนคอลัมน์เท่ากับจำนวนแถวของเมทริกซ์ที่นำมาคูณ เช่น เมทริกซ์ M ขนาด R แถว x C คอลัมน์ จะต้องใช้เมทริกซ์ที่มี C แถว มาคูณจึงสามารถดำเนินการคูณเมทริกซ์ด้วยเมทริกซ์ได้

ตัวอย่างเช่น มีเมทริกซ์ M ขนาด 3 แถวx 3 คอลัมน์ นำมาคูณกับเมทริกซ์ N ขนาด 3 แถว x 1 คอลัมน์

$$M = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix}$$

$$N = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

$$M \times N = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} (1 \times 3) + (3 \times 2) + (0 \times 1) \\ (0 \times 3) + (1 \times 2) + (0 \times 1) \\ (2 \times 3) + (0 \times 2) + (1 \times 1) \end{bmatrix}$$

$$= \begin{bmatrix} 9 \\ 2 \\ 7 \end{bmatrix}$$

ขั้นตอนวิธีของการคูณเมทริกซ์ด้วยเมทริกซ์เป็นดังนี้

กำหนด M เป็นเมทริกซ์ตัวตั้งขนาด m x n

N เป็นเมทริกซ์ตัวคูณขนาด n x o และ

R เป็นเมทริกซ์ผลลัพธ์ขนาด m x o

```
FOR row=0 TO m
      FOR col =0 TO n
            FOR k=0 TO o
                  R(row, k) += M(row, col) x N(col, k)
```

การทำเมทริกซ์สลับเปลี่ยน

การทำเมทริกซ์สลับเปลี่ยน (Transpose Matrix) เป็นการเปลี่ยนค่าของสมาชิกภายในเมทริกซ์โดยนำสมาชิกของแต่ละแถวมาเป็นสมาชิกของแต่ละคอลัมน์ของอีกเมทริกซ์หนึ่ง

$$M = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

$$M^T = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

เมทริกซ์เอกลักษณ์

เมทริกซ์เอกลักษณ์ (Identity Matrix) คือ เมทริกซ์ด้านเท่า (Square matrix) ที่สมาชิกในแนวทแยงมุม (ค่าแถว=ค่าคอลัมน์) มีค่าเป็น 1 และสมาชิกอื่นมีค่าเป็น 0 คุณสมบัติของเมทริกซ์เอกลักษณ์ก็คือ เมื่อนำเมทริกซ์นี้คูณกับเมทริกซ์ใด ๆ ที่เป็นเมทริกซ์ด้านเท่า ผลลัพธ์ที่ได้ไม่เปลี่ยนแปลง ซึ่งในระนาบ 2 มิติแบบโฮโมจีเนียส (Homogenous) นิยมใช้เมทริกซ์เอกลักษณ์ขนาด 3 แถวx 3 คอลัมน์ และในระนาบ 3 มิติแบบโฮโมจีเนียสจะใช้เมทริกซ์ขนาด 4 แถว x 4 คอลัมน์

เมทริกซ์เอกลักษณ์ 2 มิติ =

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

เมทริกซ์เอกลักษณ์ 3 มิติ =

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

การแทนเวกเตอร์ด้วยเมทริกซ์

เพื่อความสะดวกในการคำนวณด้านคอมพิวเตอร์กราฟิกส์จึงจัดรูปแบบโครงสร้างข้อมูลของเวกเตอร์ให้อยู่ในลักษณะของเมทริกซ์ขนาด 3 แถวx 1 คอลัมน์ สำหรับเวกเตอร์ในปริภูมิ 2 มิติ และเมทริกซ์ขนาด 4 แถว x 1 คอลัมน์สำหรับเวกเตอร์ในปริภูมิ 3 มิติ

ต้องการเก็บข้อมูลเวกเตอร์ 2 มิติที่มีค่าเป็น (100,100)

$$M = \begin{bmatrix} 100 \\ 100 \\ 1 \end{bmatrix}$$

ต้องการเก็บข้อมูลเวกเตอร์ 3 มิติที่มีค่าเป็น (200, 150,-50)

$$M = \begin{bmatrix} 200 \\ 150 \\ -50 \\ 1 \end{bmatrix}$$

การใช้เมทริกซ์ใน Unity

- การแปลงวัตถุ:
  - ตำแหน่ง (Translation): ใช้เมทริกซ์การเลื่อนเพื่อย้ายวัตถุไปยังตำแหน่งใหม่
  - การหมุน (Rotation): ใช้เมทริกซ์การหมุนเพื่อหมุนวัตถุรอบแกนต่างๆ
  - การปรับขนาด (การเลื่อนScaling): ใช้เมทริกซ์การปรับขนาดเพื่อเปลี่ยนขนาดของวัตถุ

การใช้เมทริกซ์ใน Unity

- การคำนวณทางคณิตศาสตร์:
  - การหาจุดตัดของเส้นตรง: ใช้เมทริกซ์ในการแก้ระบบสมการเชิงเส้น
  - การแปลงพิกัด: ใช้เมทริกซ์ในการแปลงพิกัดจากระบบพิกัดหนึ่งไปยังอีกระบบพิกัดหนึ่ง
  - การสร้างเอฟเฟกต์ต่างๆ: ใช้เมทริกซ์ในการสร้างเอฟเฟกต์ เช่น การบิดเบือน การสะท้อน และการหักเห

เมทริกซ์ใน Unity

เมทริกซ์ถูกแทนด้วยคลาส Matrix4x4 ซึ่งเป็นโครงสร้างข้อมูลที่ใช้ในการจัดเก็บเมทริกซ์ขนาด 4x4 ซึ่งเป็นขนาดที่ใช้บ่อยในการแปลงวัตถุใน 3 มิติ

```
using UnityEngine;
public class MatrixExample : MonoBehaviour {
    void Start() {
        // สร้างเมทริกซ์เอกลักษณ์ (Identity Matrix)
        Matrix4x4 identityMatrix = Matrix4x4.identity;

        // สร้างเมทริกซ์การเลื่อน
        Vector3 translation = new Vector3(2, 1, 0);
        Matrix4x4 translationMatrix = Matrix4x4.Translate(translation);

        // สร้างเมทริกซ์การหมุนรอบแกน Y
        Quaternion rotation = Quaternion.Euler(0, 45, 0);
        Matrix4x4 rotationMatrix = Matrix4x4.Rotate(rotation);

        // คูณเมทริกซ์
        Matrix4x4 resultMatrix = translationMatrix * rotationMatrix;

        // นำเมทริกซ์ไปใช้กับวัตถุ
        transform.localToWorldMatrix = resultMatrix;
    }
}
```

เวกเตอร์และเมทริกซ์เป็นวิธีการที่ถูกนำประยุกต์ใช้สำหรับการคำนวณทางด้านคอมพิวเตอร์กราฟิกส์ เช่น การย้ายตำแหน่ง การปรับขนาด การหมุนและการแปลงพิกัดของวัตถุและการแสดงผลของภาพ

Q&A