

1 Vector Mathematics

1 Vector Mathematics

A *vector* indicates a quantity, such as velocity or force, that has *direction* and *length*. Vectors in 3D coordinate systems are represented with an ordered set of three real numbers and look like:

$$\mathbf{v} = \langle a_1, a_2, a_3 \rangle$$

Vector representation

In this document, lower case bold letters will notate vectors. Vector components are also enclosed in angle brackets. Upper case letters will notate points. Point coordinates will always be enclosed by parentheses.

Using a coordinate system and any set of anchor points in that system, we can represent or visualize these vectors using a line-segment representation. An arrowhead shows the vector direction.

For example, if we have a vector that has a direction parallel to the x-axis of a given 3D coordinate system and a length of 5 units, we can write the vector as follows:

$$\mathbf{v} = \langle 5, 0, 0 \rangle$$

To represent that vector, we need an anchor point in the coordinate system. For example, all of the arrows in the following figure are equal representations of the same vector despite the fact that they are anchored at different locations.

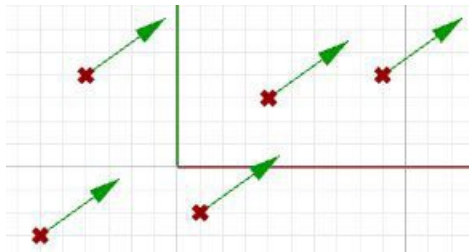


Figure (1): Vector representation in the 3-D coordinate system.

Given a 3D vector $\mathbf{v} = \langle a_1, a_2, a_3 \rangle$, all vector components a_1, a_2, a_3 are real numbers. Also all line segments from a point $A(x, y, z)$ to point $B(x+a_1, y+a_2, z+a_3)$ are equivalent representations of vector \mathbf{v} .

So, how do we define the end points of a line segment that represents a given vector?

Let us define an anchor point (A) so that:

$$A = (1, 2, 3)$$

And a vector:

$$\mathbf{v} = \langle 5, 6, 7 \rangle$$

The tip point (B) of the vector is calculated by adding the corresponding components from anchor point and vector \mathbf{v} :

$$B = A + \mathbf{v}$$

$$B = (1+5, 2+6, 3+7)$$

$$B = (6, 8, 10)$$

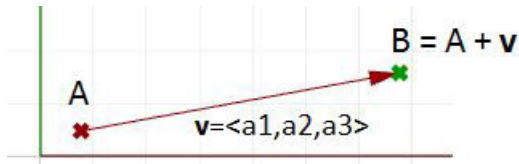


Figure (2): The relationship between a vector, the vector anchor point, and the point coinciding with the vector tip location.

Position vector

One special vector representation uses the origin point $(0,0,0)$ as the vector anchor point. The position vector $\mathbf{v} = \langle a_1, a_2, a_3 \rangle$ is represented with a line segment between two points, the origin and B , so that:

Origin point = $(0,0,0)$

$B = (a_1, a_2, a_3)$

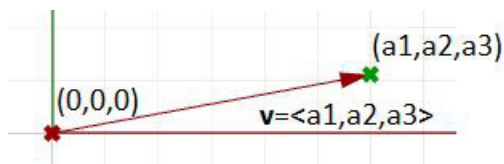


Figure (3): Position vector. The tip point coordinates equal the corresponding vector components.

A position vector for a given vector $\mathbf{v} = \langle a_1, a_2, a_3 \rangle$ is a special line segment representation from the origin point $(0,0,0)$ to point (a_1, a_2, a_3) .

Vectors vs. points

Do not confuse vectors and points. They are very different concepts. Vectors, as we mentioned, represent a quantity that has direction and length, while points indicate a location. For example, the North direction is a vector, while the North Pole is a location (point).

If we have a vector and a point that have the same components, such as:

$\mathbf{v} = \langle 3, 1, 0 \rangle$

$P = (3, 1, 0)$

We can draw the vector and the point as follows:

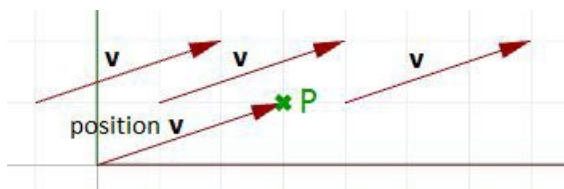


Figure (4): A vector defines a direction and length. A point defines a location.



Vector length

As mentioned before, vectors have length. We will use $|\mathbf{a}|$ to notate the length of a given vector \mathbf{a} . For example:

$$\mathbf{a} = \langle 4, 3, 0 \rangle$$

$$|\mathbf{a}| = \sqrt{4^2 + 3^2 + 0^2}$$

$$|\mathbf{a}| = 5$$

In general, the **length** of a vector $\mathbf{a} = \langle a_1, a_2, a_3 \rangle$ is calculated as follows:

$$|\mathbf{a}| = \sqrt{a_1^2 + a_2^2 + a_3^2}$$

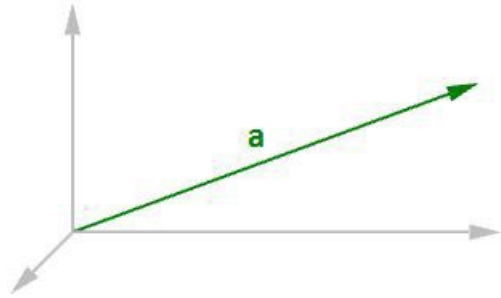


Figure (5): Vector length.



Unit vector

A unit vector is a vector with a length equal to one unit. Unit vectors are commonly used to compare the directions of vectors.

A unit vector is a vector whose length is equal to one unit.

To calculate a unit vector, we need to find the length of the given vector, and then divide the vector components by the length. For example:

$$\mathbf{a} = \langle 4, 3, 0 \rangle$$

$$|\mathbf{a}| = \sqrt{4^2 + 3^2 + 0^2}$$

$$|\mathbf{a}| = 5 \text{ unit length}$$

If \mathbf{b} = unit vector of \mathbf{a} , then:

$$\mathbf{b} = \langle 4/5, 3/5, 0/5 \rangle$$

$$\mathbf{b} = \langle 0.8, 0.6, 0 \rangle$$

$$|\mathbf{b}| = \sqrt{0.8^2 + 0.6^2 + 0^2}$$

$$|\mathbf{b}| = \sqrt{0.64 + 0.36 + 0}$$

$$|\mathbf{b}| = \sqrt{1} = 1 \text{ unit length}$$

In general:

$$\mathbf{a} = \langle a_1, a_2, a_3 \rangle$$

$$\text{The unit vector of } \mathbf{a} = \langle a_1/|\mathbf{a}|, a_2/|\mathbf{a}|, a_3/|\mathbf{a}| \rangle$$

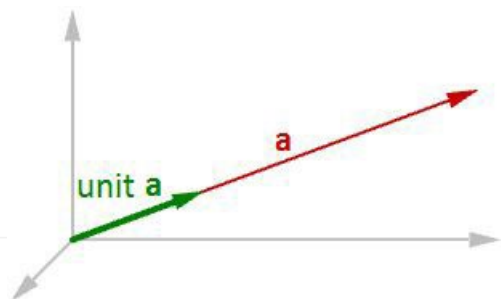


Figure (6): Unit vector equals one-unit length of the vector.

Vector operations

Vector scalar operation

Vector scalar operation involves multiplying a vector by a number. For example:

$$\mathbf{a} = \langle 4, 3, 0 \rangle$$

$$2 * \mathbf{a} = \langle 2 * 4, 2 * 3, 2 * 0 \rangle$$

$$2 * \mathbf{a} = \langle 8, 6, 0 \rangle$$

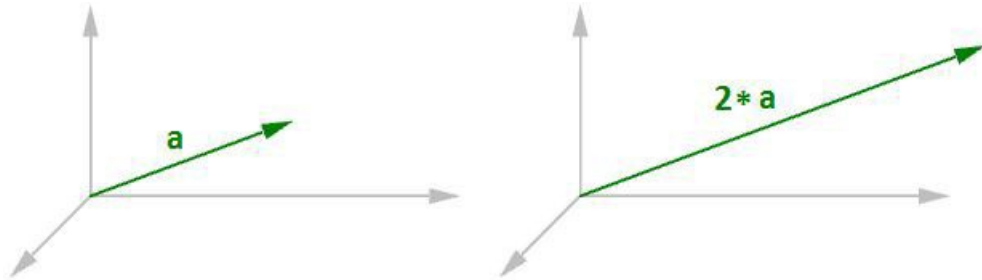


Figure (7): Vector scalar operation

In general, given vector $\mathbf{a} = \langle a_1, a_2, a_3 \rangle$, and a real number t

$$t * \mathbf{a} = \langle t * a_1, t * a_2, t * a_3 \rangle$$

Vector addition

Vector addition takes two vectors and produces a third vector. We add vectors by adding their components.

Vectors are added by adding their components.

For example, if we have two vectors:

$$\mathbf{a} \langle 1, 2, 0 \rangle$$

$$\mathbf{b} \langle 4, 1, 3 \rangle$$

$$\mathbf{a} + \mathbf{b} = \langle 1+4, 2+1, 0+3 \rangle$$

$$\mathbf{a} + \mathbf{b} = \langle 5, 3, 3 \rangle$$

Vector operations

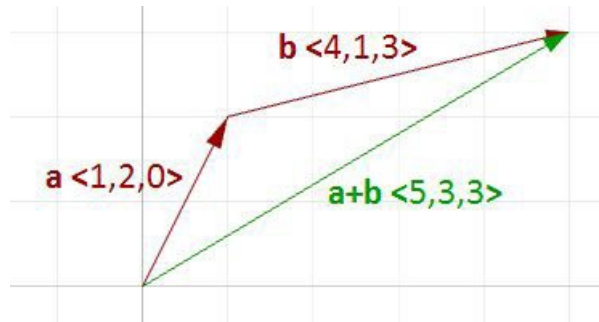


Figure (8): Vector addition.

In general, vector addition of the two vectors **a** and **b** is calculated as follows:

$$\mathbf{a} = \langle a_1, a_2, a_3 \rangle$$

$$\mathbf{b} = \langle b_1, b_2, b_3 \rangle$$

$$\mathbf{a} + \mathbf{b} = \langle a_1 + b_1, a_2 + b_2, a_3 + b_3 \rangle$$

Vector addition is useful for finding the average direction of two or more vectors. In this case, we usually use same-length vectors. Here is an example that shows the difference between using same-length vectors and different-length vectors on the resulting vector addition:

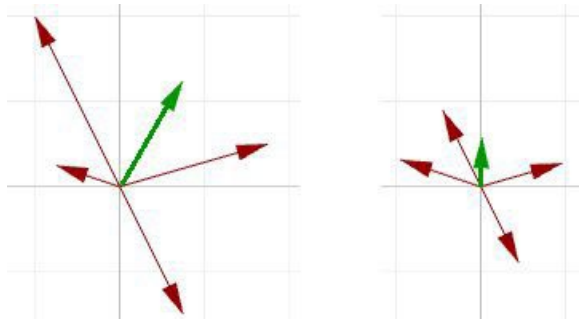


Figure (9): Adding various length vectors (left). Adding same length vectors (right) to get the average direction.

Input vectors are not likely to be same length. In order to find the average direction, you need to use the unit vector of input vectors. As mentioned before, the unit vector is a vector of that has a length equal to 1.

Vector subtraction

Vector subtraction takes two vectors and produces a third vector. We subtract two vectors by subtracting corresponding components. For example, if we have two vectors **a** and **b** and we subtract **b** from **a**, then:

$$\mathbf{a} = \langle 1, 2, 0 \rangle$$

$$\mathbf{b} = \langle 4, 1, 4 \rangle$$

$$\mathbf{a} - \mathbf{b} = \langle 1-4, 2-1, 0-4 \rangle$$

$$\mathbf{a} - \mathbf{b} = \langle -3, 1, -4 \rangle$$

If we subtract **b** from **a**, we get a different result:

$$\mathbf{b} - \mathbf{a} = \langle 4-1, 1-2, 4-0 \rangle$$

$$\mathbf{b} - \mathbf{a} = \langle 3, -1, 4 \rangle$$

Note that the vector $\mathbf{b} - \mathbf{a}$ has the same length as the vector $\mathbf{a} - \mathbf{b}$, but goes in the opposite direction.

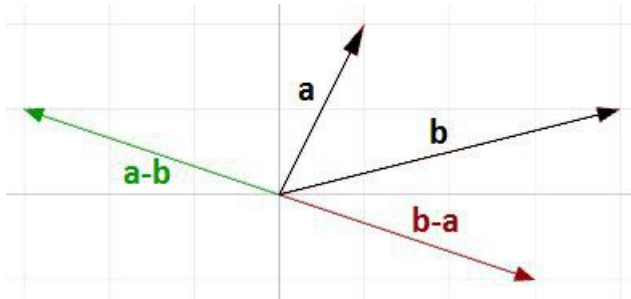


Figure (10): Vector subtraction.

In general, if we have two vectors, \mathbf{a} and \mathbf{b} , then $\mathbf{a} - \mathbf{b}$ is a vector that is calculated as follows:

$$\mathbf{a} = \langle a_1, a_2, a_3 \rangle$$

$$\mathbf{b} = \langle b_1, b_2, b_3 \rangle$$

$$\mathbf{a} - \mathbf{b} = \langle a_1 - b_1, a_2 - b_2, a_3 - b_3 \rangle$$

Vector subtraction is commonly used to find vectors between points. So if we need to find a vector that goes from the tip point of the position vector \mathbf{b} to the tip point of the position vector \mathbf{a} , then we use vector subtraction ($\mathbf{a} - \mathbf{b}$) as shown in Figure (11).

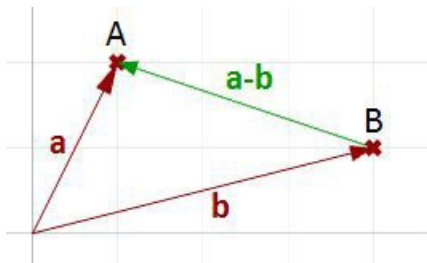


Figure (11): Use vector subtraction to find a vector between two points.

Vector properties

There are eight properties of vectors. If \mathbf{a} , \mathbf{b} , and \mathbf{c} are vectors, and s and t are numbers, then:

$$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$$

$$\mathbf{a} + \mathbf{0} = \mathbf{a}$$

$$s * (\mathbf{a} + \mathbf{b}) = s * \mathbf{a} + s * \mathbf{b}$$

$$s * t * (\mathbf{a}) = s * (t * \mathbf{a})$$

$$\mathbf{a} + (\mathbf{b} + \mathbf{c}) = (\mathbf{a} + \mathbf{b}) + \mathbf{c}$$

$$\mathbf{a} + (-\mathbf{a}) = \mathbf{0}$$

$$(s + t) * \mathbf{a} = s * \mathbf{a} + t * \mathbf{a}$$

$$1 * \mathbf{a} = \mathbf{a}$$

Vector dot product

The dot product takes two vectors and produces a number.

For example, if we have the two vectors **a** and **b** so that:

$$\mathbf{a} = \langle 1, 2, 3 \rangle$$

$$\mathbf{b} = \langle 5, 6, 7 \rangle$$

Then the dot product is the sum of multiplying the components as follows:

$$\mathbf{a} \cdot \mathbf{b} = 1 * 5 + 2 * 6 + 3 * 7$$

$$\mathbf{a} \cdot \mathbf{b} = 38$$

In general, given the two vectors **a** and **b**:

$$\mathbf{a} = \langle a_1, a_2, a_3 \rangle$$

$$\mathbf{b} = \langle b_1, b_2, b_3 \rangle$$

$$\mathbf{a} \cdot \mathbf{b} = a_1 * b_1 + a_2 * b_2 + a_3 * b_3$$

We always get a positive number for the dot product between two vectors when they go in the same general direction. A negative dot product between two vectors means that the two vectors go in the opposite general direction.

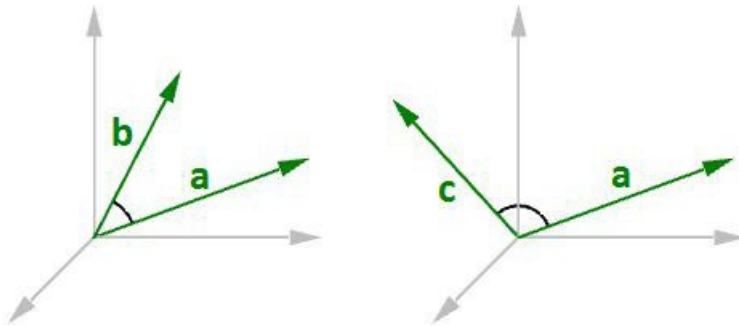


Figure (12): When the two vectors go in the same direction (left), the result is a positive dot product. When the two vectors go in the opposite direction (right), the result is a negative dot product.

When calculating the dot product of two unit vectors, the result is always between -1 and +1. For example:

$$\mathbf{a} = \langle 1, 0, 0 \rangle$$

$$\mathbf{b} = \langle 0.6, 0.8, 0 \rangle$$

$$\mathbf{a} \cdot \mathbf{b} = (1 * 0.6, 0 * 0.8, 0 * 0) = 0.6$$

In addition, the dot product of a vector with itself is equal to that vector's length to the power of two. For example:

$$\mathbf{a} = \langle 0, 3, 4 \rangle$$

$$\mathbf{a} \cdot \mathbf{a} = 0 * 0 + 3 * 3 + 4 * 4$$

$$\mathbf{a} \cdot \mathbf{a} = 25$$

Calculating the square length of vector **a**:

$$|\mathbf{a}| = \sqrt{4^2 + 3^2 + 0^2}$$

$$|\mathbf{a}| = 5$$

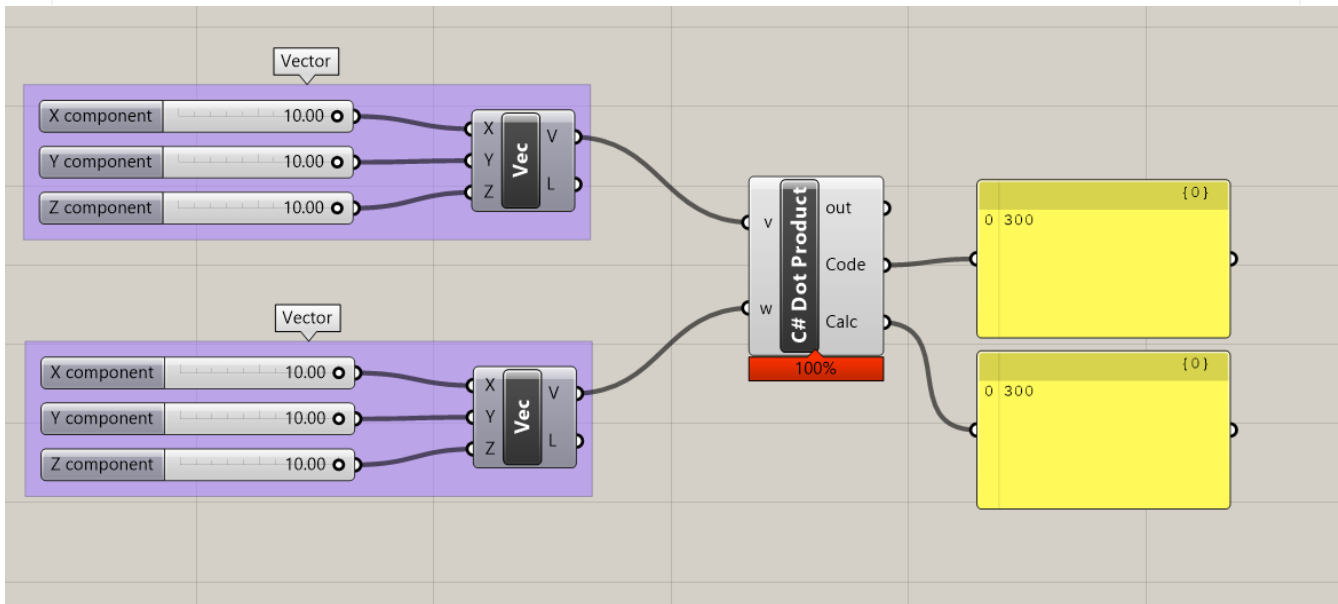
$$|\mathbf{a}|^2 = 25$$

Vector dot product



```
private void RunScript(Vector3d v, Vector3d w, ref object Code, ref object Calc)
{
    // Calculate Vector Length 2 ways
    Code = Vector3d.Multiply(v,w);

    Calc = v.X * w.X + v.Y * w.Y + v.Z * w.Z;
}
```



Vector dot product

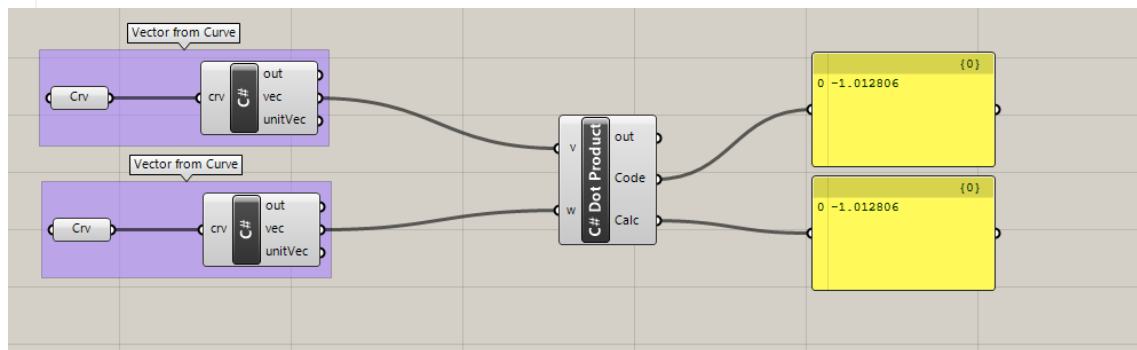


```
private void RunScript(Curve crv, ref object vec, ref object unitVec)
{
    Vector3d newVec = new Vector3d(crv.PointAtEnd) - new Vector3d(crv.PointAtStart);
    Vector3d newUnitVec = newVec;
    newUnitVec.Unitize();

    vec = newVec;
    unitVec = newUnitVec;
}
```

```
private void RunScript(Vector3d v, Vector3d w, ref object Code, ref object Calc)
{
    // Calculate Vector Length 2 ways
    Code = Vector3d.Multiply(v, w);

    Calc = v.X * w.X + v.Y * w.Y + v.Z * w.Z;
}
```



The dot product of two non-zero unit vectors equals the cosine of the angle between them.

Vectors a and b are orthogonal if, and only if, $a \cdot b = 0$.

If a , b , and c are vectors and s is a number, then:

$$a \cdot a = |a|^2$$

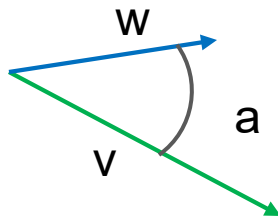
$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$0 \cdot a = 0$$

$$a \cdot b = b \cdot a$$

$$(s \cdot a) \cdot b = s \cdot (a \cdot b) = a \cdot (s \cdot b)$$

Dot product, cosine and angle

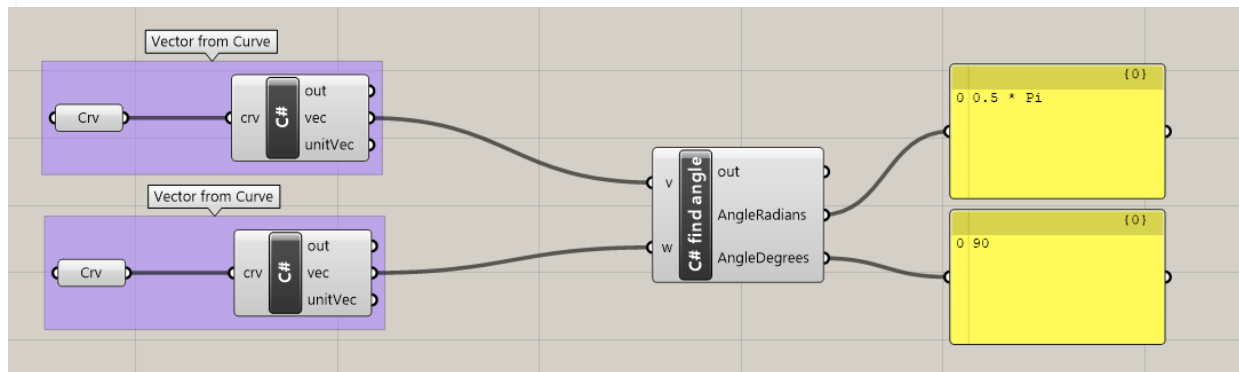


$$v \cdot w = |v| |w| \cos(a)$$

If $|v| = |w| = 1$ then

$$v \cdot w = \cos(a)$$

$$\text{angle} = \cos^{-1}(v \cdot w)$$



```
private void RunScript(Vector3d v, Vector3d w, ref object AngleRadians, ref object AngleDegrees)
{
    // Calculate Vector Length 2 ways
```

```
    v.Unitize(); w.Unitize();
    double dotp = Vector3d.Multiply(v, w);
    double angleRadians = Math.Acos(dotp);
    double angleDegrees = angleRadians * 180.0 / Math.PI;
```

```
    AngleRadians = angleRadians;
    AngleDegrees = angleDegrees;
```

Vector dot product, lengths, and angles

There is a relationship between the dot product of two vectors and the angle between them.

The dot product of two non-zero unit vectors equals the cosine of the angle between them.

In general:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| * |\mathbf{b}| * \cos(\theta), \text{ or}$$

$$\mathbf{a} \cdot \mathbf{b} / (|\mathbf{a}| * |\mathbf{b}|) = \cos(\theta)$$

Where:

θ is the angle included between the vectors.

If vectors \mathbf{a} and \mathbf{b} are unit vectors, we can simply say:

$$\mathbf{a} \cdot \mathbf{b} = \cos(\theta)$$

And since the cosine of a 90-degree angle is equal to 0, we can say:

Vectors \mathbf{a} and \mathbf{b} are orthogonal if, and only if, $\mathbf{a} \cdot \mathbf{b} = 0$.

For example, if we calculate the dot product of the two orthogonal vectors, World xaxis and yaxis, the result will equal zero.

$$\mathbf{x} = \langle 1, 0, 0 \rangle$$

$$\mathbf{y} = \langle 0, 1, 0 \rangle$$

$$\mathbf{x} \cdot \mathbf{y} = (1 * 0) + (0 * 1) + (0 * 0)$$

$$\mathbf{x} \cdot \mathbf{y} = 0$$

There is also a relationship between the dot product and the projection length of one vector onto another. For example:

$$\mathbf{a} = \langle 5, 2, 0 \rangle$$

$$\mathbf{b} = \langle 9, 0, 0 \rangle$$

$$\text{unit}(\mathbf{b}) = \langle 1, 0, 0 \rangle$$

$$\mathbf{a} \cdot \text{unit}(\mathbf{b}) = (5 * 1) + (2 * 0) + (0 * 0)$$

$$\mathbf{a} \cdot \text{unit}(\mathbf{b}) = 2 \text{ (which is equal to the projection length of } \mathbf{a} \text{ onto } \mathbf{b})$$

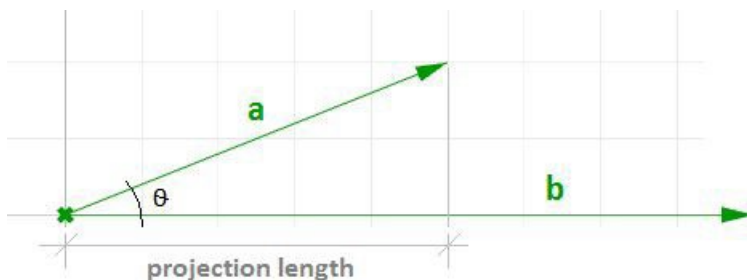


Figure (13): The dot product equals the projection length of one vector onto a non-zero unit vector.

In general, given a vector \mathbf{a} and a non-zero vector \mathbf{b} , we can calculate the projection length pL of vector \mathbf{a} onto vector \mathbf{b} using the dot product.

$$pL = |\mathbf{a}| * \cos(\theta)$$

$$pL = \mathbf{a} \cdot \text{unit}(\mathbf{b})$$

Vector cross product

The cross product takes two vectors and produces a third vector that is orthogonal to both.

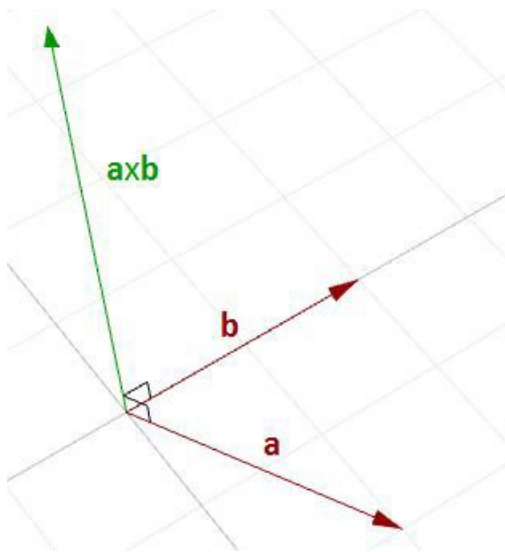


Figure (14): Calculating the cross product of two vectors.

For example, if you have two vectors lying on the World xy-plane, then their cross product is a vector perpendicular to the xy-plane going either in the positive or negative World z-axis direction. For example:

$$\mathbf{a} = \langle 3, 1, 0 \rangle$$

$$\mathbf{b} = \langle 1, 2, 0 \rangle$$

$$\mathbf{a} \times \mathbf{b} = \langle (1 * 0 - 0 * 2), (0 * 1 - 3 * 0), (3 * 2 - 1 * 1) \rangle$$

$$\mathbf{a} \times \mathbf{b} = \langle 0, 0, 5 \rangle$$

The vector $\mathbf{a} \times \mathbf{b}$ is orthogonal to both \mathbf{a} and \mathbf{b} .

You will probably never need to calculate a cross product of two vectors by hand, but if you are curious about how it is done, continue reading; otherwise you can safely skip this section. The cross product $\mathbf{a} \times \mathbf{b}$ is defined using *determinants*. Here is a simple illustration of how to calculate a determinant using the standard basis vectors:

$$\mathbf{i} = \langle 1, 0, 0 \rangle$$

$$\mathbf{j} = \langle 0, 1, 0 \rangle$$

$$\mathbf{k} = \langle 0, 0, 1 \rangle$$

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = \begin{vmatrix} \mathbf{i} & \mathbf{j} \\ a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} - \begin{vmatrix} \mathbf{i} & \mathbf{k} \\ a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} + \begin{vmatrix} \mathbf{j} & \mathbf{k} \\ a_2 & a_3 \\ b_2 & b_3 \end{vmatrix}$$

The cross product of the two vectors $\mathbf{a} \langle a_1, a_2, a_3 \rangle$ and $\mathbf{b} \langle b_1, b_2, b_3 \rangle$ is calculated as follows using the above diagram:

$$\mathbf{a} \times \mathbf{b} = \mathbf{i}(a_2 * b_3) + \mathbf{j}(a_3 * b_1) + \mathbf{k}(a_1 * b_2) - \mathbf{k}(a_2 * b_1) - \mathbf{i}(a_3 * b_2) - \mathbf{j}(a_1 * b_3)$$

$$\mathbf{a} \times \mathbf{b} = \mathbf{i}(a_2 * b_3 - a_3 * b_2) + \mathbf{j}(a_3 * b_1 - a_1 * b_3) + \mathbf{k}(a_1 * b_2 - a_2 * b_1)$$

$$\mathbf{a} \times \mathbf{b} = \langle a_2 * b_3 - a_3 * b_2, a_3 * b_1 - a_1 * b_3, a_1 * b_2 - a_2 * b_1 \rangle$$

Cross product and angle between vectors

There is a relationship between the angle between two vectors and the length of their cross product vector. The smaller the angle (smaller sine); the shorter the cross product vector will be. The order of operands is important in vectors cross product.

For example:

$$\mathbf{a} = \langle 1, 0, 0 \rangle$$

$$\mathbf{b} = \langle 0, 1, 0 \rangle$$

$$\mathbf{a} \times \mathbf{b} = \langle 0, 0, 1 \rangle$$

$$\mathbf{b} \times \mathbf{a} = \langle 0, 0, -1 \rangle$$

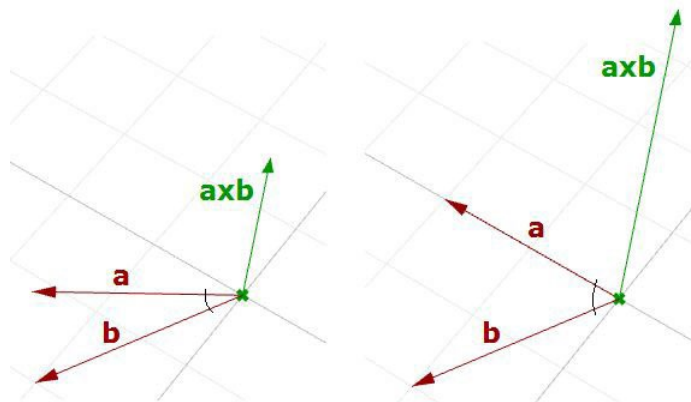


Figure (15): The relationship between the sine of the angle between two vectors and the length of their cross product vector.

Vector cross product

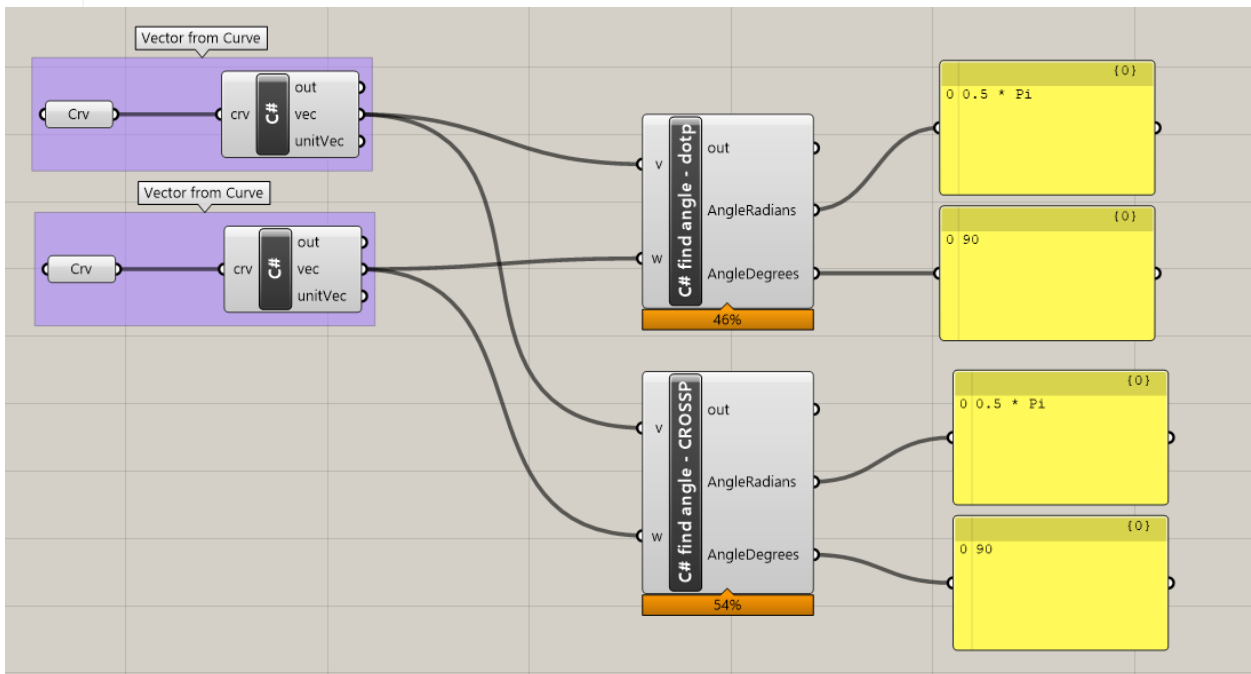


```
private void RunScript(Vector3d v, Vector3d w, ref object AngleRadians, ref object AngleDegrees)
{
    // Calculate Vector Length 2 ways

    v.Unitize(); w.Unitize();
    double dotp = Vector3d.Multiply(v, w);
    double angleRadians = Math.Acos(dotp);
    double angleDegrees = angleRadians * 180.0 / Math.PI;

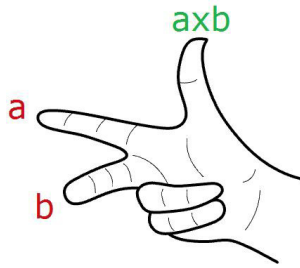
    v.Unitize(); w.Unitize();
    Vector3d crosspVector = Vector3d.CrossProduct(v, w);
    double angleRadians = Math.Asin(crosspVector.Length);
    double angleDegrees = angleRadians * 180.0 / Math.PI;

    AngleRadians = angleRadians;
    AngleDegrees = angleDegrees;
}
```



Right hand coordinate system

In Rhino's right-handed system, the direction of $\mathbf{a} \times \mathbf{b}$ is given by the right-hand rule (where \mathbf{a} = index finger, \mathbf{b} = middle finger, and $\mathbf{a} \times \mathbf{b}$ = thumb).



In general, for any pair of 3-D vectors \mathbf{a} and \mathbf{b} :

$$|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}| |\mathbf{b}| \sin(\theta)$$

Where:

θ is the angle included between the position vectors of \mathbf{a} and \mathbf{b}

If \mathbf{a} and \mathbf{b} are unit vectors, then we can simply say that the length of their cross product equals the sine of the angle between them. In other words:

$$|\mathbf{a} \times \mathbf{b}| = \sin(\theta)$$

The cross product between two vectors helps us determine if two vectors are parallel. This is because the result is always a zero vector.

Vectors \mathbf{a} and \mathbf{b} are parallel if, and only if, $\mathbf{a} \times \mathbf{b} = \mathbf{0}$.

Cross product properties

If \mathbf{a} , \mathbf{b} , and \mathbf{c} are vectors, and s is a number, then:

$$\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$$

$$(s * \mathbf{a}) \times \mathbf{b} = s * (\mathbf{a} \times \mathbf{b}) = \mathbf{a} \times (s * \mathbf{b})$$

$$\mathbf{a} \times (\mathbf{b} + \mathbf{c}) = \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c}$$

$$(\mathbf{a} + \mathbf{b}) \times \mathbf{c} = \mathbf{a} \times \mathbf{c} + \mathbf{b} \times \mathbf{c}$$

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}$$

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c}) * \mathbf{b} - (\mathbf{a} \cdot \mathbf{b}) * \mathbf{c}$$

Vector equation of line

The vector line equation is used in 3D modeling applications and computer graphics.

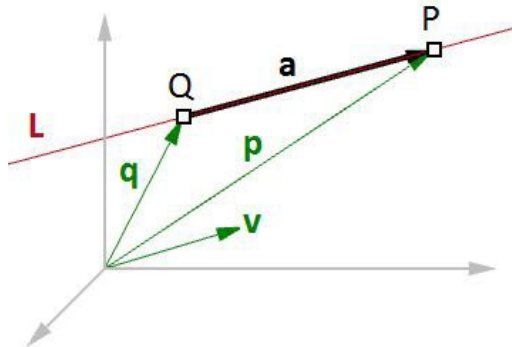


Figure (16): Vector equation of a line.

For example, if we know the direction of a line and a point on that line, then we can find any other point on the line using vectors, as in the following:

L = line

$\mathbf{v} = \langle a, b, c \rangle$ line direction unit vector

$Q = (x_0, y_0, z_0)$ line position point

$P = (x, y, z)$ any point on the line

We know that:

$$\mathbf{a} = t * \mathbf{v} \quad \text{--- (2)}$$

$$\mathbf{p} = \mathbf{q} + \mathbf{a} \quad \text{--- (1)}$$

From 1 and 2:

$$\mathbf{p} = \mathbf{q} + t * \mathbf{v} \quad \text{--- (3)}$$

However, we can write (3) as follows:

$$\langle x, y, z \rangle = \langle x_0, y_0, z_0 \rangle + \langle t * a, t * b, t * c \rangle$$

$$\langle x, y, z \rangle = \langle x_0 + t * a, y_0 + t * b, z_0 + t * c \rangle$$

Therefore:

$$x = x_0 + t * a$$

$$y = y_0 + t * b$$

$$z = z_0 + t * c$$

Which is the same as:

$$P = Q + t * \mathbf{v}$$

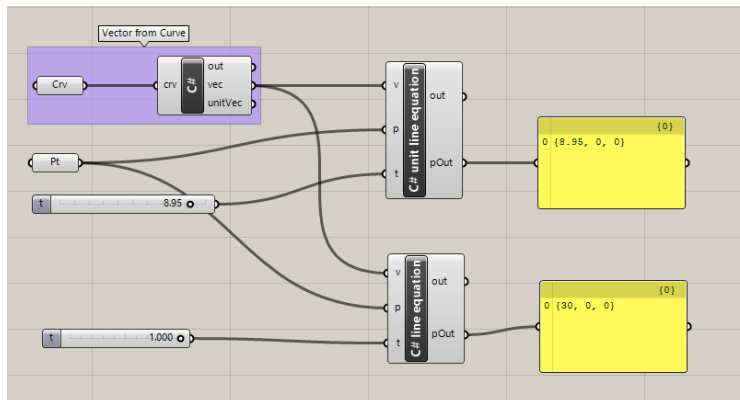
Given a point Q and a direction \mathbf{v} on a line, any point P on that line can be calculated using the vector equation of a line $P = Q + t * \mathbf{v}$ where t is a number.

Vector equation of line



The vector line equation is used in 3D modeling applications and computer graphics.

Given a point Q and a direction v on a line, any point P on that line can be calculated using the vector equation of a line $P = Q + t * v$ where t is a number.



Unit line equation

```
private void RunScript(Vector3d v, Point3d p, double t, ref object pOut)
{
    // Calculate Vector Length 2 ways
    Vector3d vnorm = v;
    vnorm.Unitize();
    pOut = p + vnorm * t;
}
```

line equation

```
private void RunScript(Vector3d v, Point3d p, double t, ref object pOut)
{
    // Calculate Vector Length 2 ways
    Vector3d vnorm = v;
    //vnorm.Unitize();
    pOut = p + vnorm * t;
}
```

Midpoint of a line

Another common example is to find the midpoint between two points. The following shows how to find the midpoint using the vector equation of a line:

q is the position vector for point Q

p is the position vector for point P

a is the vector going from Q to P

From vector subtraction, we know that:

$$\mathbf{a} = \mathbf{p} - \mathbf{q}$$

From the line equation, we know that:

$$\mathbf{M} = \mathbf{Q} + t * \mathbf{a}$$

And since we need to find midpoint, then:

$$t = 0.5$$

Hence we can say:

$$\mathbf{M} = \mathbf{Q} + 0.5 * \mathbf{a}$$

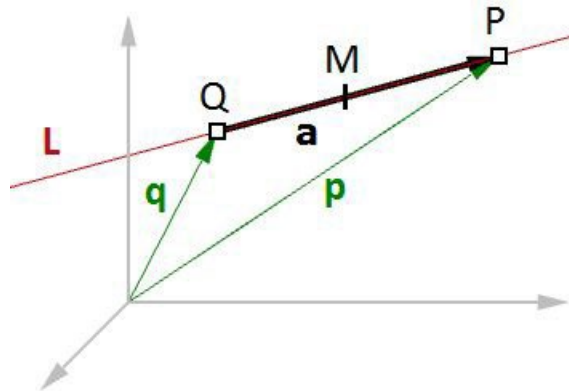


Figure (17): Find the midpoint between two input points.

In general, you can find any point between Q and P by changing the t value between 0 and 1 using the general equation:

$$\mathbf{M} = \mathbf{Q} + t * (\mathbf{P} - \mathbf{Q})$$

Given two points Q and P, any point M between the two points is calculated using the equation $\mathbf{M} = \mathbf{Q} + t * (\mathbf{P} - \mathbf{Q})$ where t is a number between 0 and 1.

Vector equation of a plane

One way to define a plane is when you have a point and a vector that is perpendicular to the plane. That vector is usually referred to as *normal* to the plane. The normal points in the direction above the plane.

One example of how to calculate a plane normal is when we know three non-linear points on the plane.

In Figure (16), given:

A = the first point on the plane

B = the second point on the plane

C = the third point on the plane

And:

a = a position vector of point A

b = a position vector of point B

c = a position vector of point C

We can find the normal vector **n** as follows:

$$\mathbf{n} = (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})$$

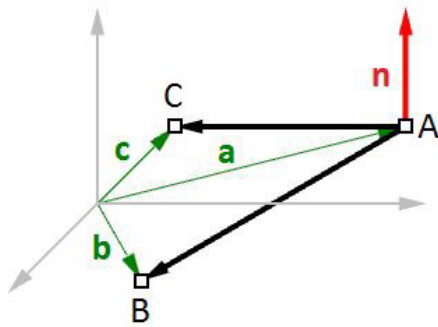


Figure (18): Vectors and planes

We can also derive the scalar equation of the plane using the vector dot product:

$$\mathbf{n} \cdot (\mathbf{b} - \mathbf{a}) = 0$$

If:

$$\mathbf{n} = \langle a, b, c \rangle$$

$$\mathbf{b} = \langle x, y, z \rangle$$

$$\mathbf{a} = \langle x_0, y_0, z_0 \rangle$$

Then we can expand the above:

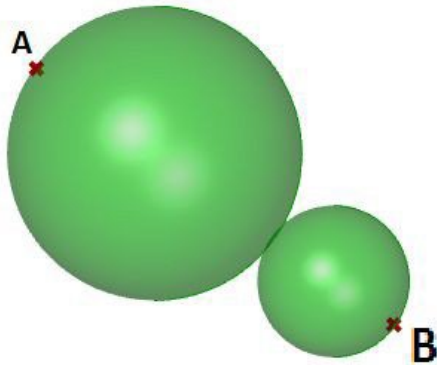
$$\langle a, b, c \rangle \cdot \langle x - x_0, y - y_0, z - z_0 \rangle = 0$$

Solving the dot product gives the general scalar equation of a plane:

$$a * (x - x_0) + b * (y - y_0) + c * (z - z_0) = 0$$

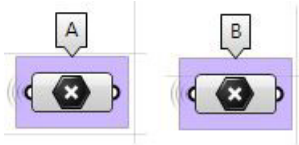
Tangent spheres

This tutorial will show how to create two tangent spheres between two input points. This is what the result looks like:



Input:

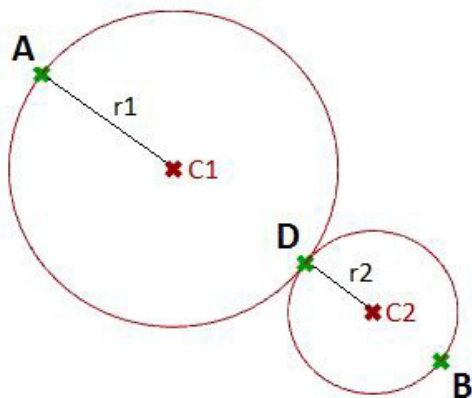
Two points (A and B) in the 3-D coordinate system.



Parameters:

The following is a diagram of the parameters that we will need in order to solve the problem:

- A tangent point D between the two spheres, at some t parameter (0-1) between points A and B.
- The center of the first sphere or the midpoint C1 between A and D.
- The center of the second sphere or the midpoint C2 between D and B.
- The radius of the first sphere (r_1) or the distance between A and C1.
- The radius of the second sphere (r_2) or the distance between D and C2.



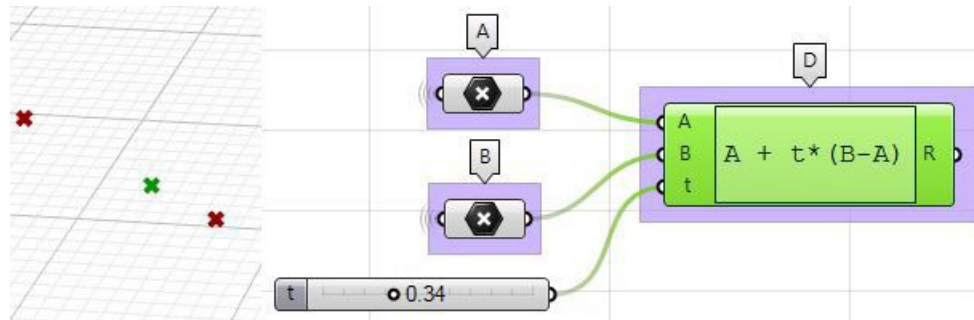
Solution:

1. Use the **Expression** component to define point **D** between **A** and **B** at some parameter **t**. The expression we will use is based on the vector equation of a line:
 $D = A + t*(B-A)$.

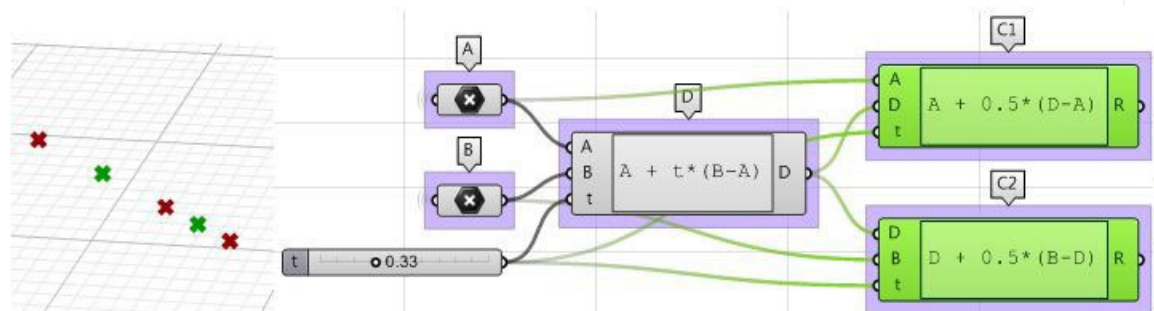
$B-A$: is the vector that goes from B to A using the vector subtraction operation.

$t*(B-A)$: where t is between 0 and 1 to get us a location on the vector.

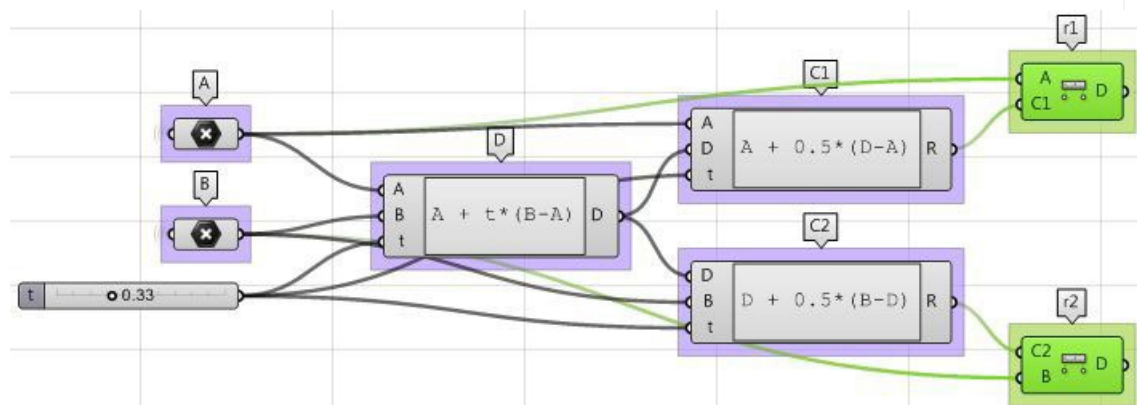
$A+t*(B-A)$: gets a point on the vector between A and B.



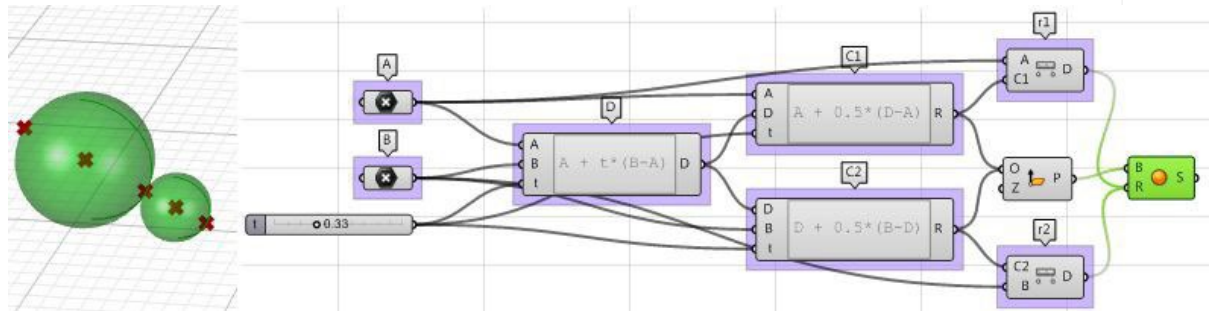
2. Use the **Expression** component to also define the mid points **C1** and **C2**.



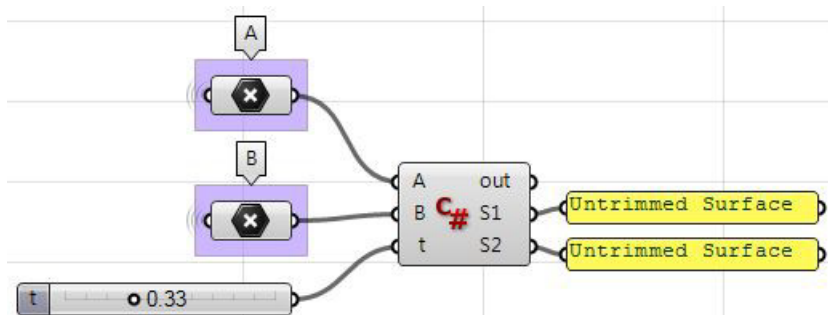
3. The first sphere radius (**r1**) and the second sphere radius (**r2**) can be calculated using the **Distance** component.



- The final step involves creating the sphere from a base plane and radius. We need to make sure the origins are hooked to **C1** and **C2** and the radius from **r1** and **r2**.



Using the Grasshopper C# component:



```
private void RunScript(Point3d A, Point3d B, double t, ref object S1, ref object S2)
{
    //declare variables
    Rhino.Geometry.Point3d D, C1, C2;
    double r1, r2;

    //find a point between A and B
    D = A + t * (B - A);

    //find mid point between A and D
    C1 = A + 0.5 * (D - A);

    //find mid point between D and B
    C2 = D + 0.5 * (B - D);

    //find spheres radius
    r1 = A.DistanceTo(C1);
    r2 = B.DistanceTo(C2);

    //create spheres and assign to output
    S1 = new Rhino.Geometry.Sphere(C1, r1);
    S2 = new Rhino.Geometry.Sphere(C2, r2);
}
```

Basic Matrix operations

Basic Matrix – vector multiplication

$M * v$

The diagram illustrates the first row of the matrix-vector multiplication $M * v$. It shows a 2x2 matrix M with elements M_{00} , M_{01} , M_{10} , and M_{11} . A vector v is shown with elements v_0 and v_1 . A blue arrow points from the v_0 element to the M_{00} element, indicating the dot product of the first row of M with v . The result is shown as a 2x1 vector with elements $M_{00} v_0 + M_{01} v_1$ and $M_{10} v_0 + M_{11} v_1$.

v_0	v_1
-------	-------

M_{00}	M_{01}
M_{10}	M_{11}

 *

v_0
v_1

 =

$M_{00} v_0 + M_{01} v_1$
$M_{10} v_0 + M_{11} v_1$

Basic Matrix operations

Basic Matrix – vector multiplication

$M * v$

v_0	v_1
-------	-------

M_{00}	M_{01}
M_{10}	M_{11}

 $*$

v_0
v_1

 $=$

$M_{00} v_0 + M_{01} v_1$
$M_{10} v_0 + M_{11} v_1$

Transformation matrix – in 2D

v_x	v_y
-------	-------

M_{xx}	M_{xy}
M_{yx}	M_{yy}

 $*$

v_x
v_y

 $=$

$M_{xx} v_x + M_{xy} v_y$
$M_{yx} v_x + M_{yy} v_y$

Basic Matrix operations

Identity matrix transformation

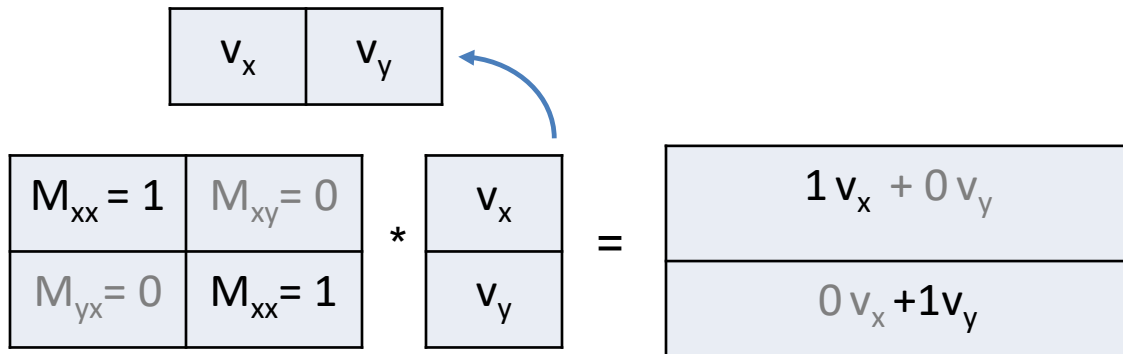


Diagram illustrating the Identity matrix transformation. A vector $\begin{bmatrix} v_x & v_y \end{bmatrix}$ is transformed by the Identity matrix $\begin{bmatrix} M_{xx}=1 & M_{xy}=0 \\ M_{yx}=0 & M_{xx}=1 \end{bmatrix}$ to produce the same vector $\begin{bmatrix} v_x & v_y \end{bmatrix}$.

$$\begin{bmatrix} M_{xx}=1 & M_{xy}=0 \\ M_{yx}=0 & M_{xx}=1 \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} 1v_x + 0v_y \\ 0v_x + 1v_y \end{bmatrix}$$

Scaling transformation

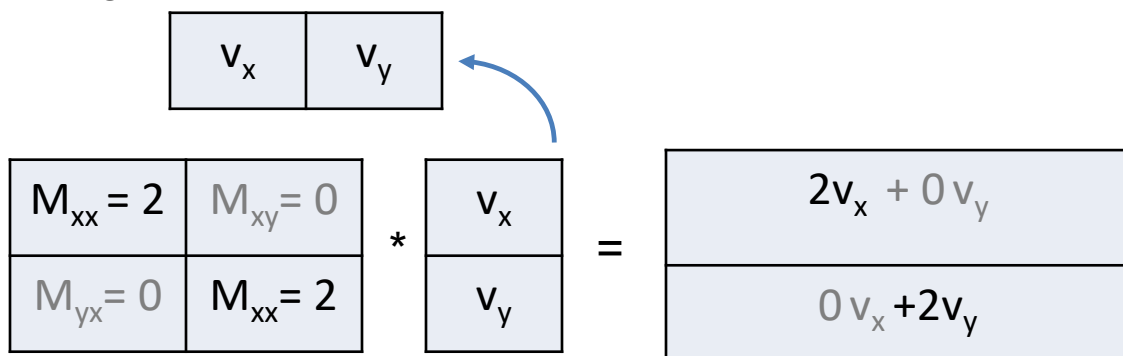


Diagram illustrating the Scaling transformation. A vector $\begin{bmatrix} v_x & v_y \end{bmatrix}$ is transformed by the Scaling matrix $\begin{bmatrix} M_{xx}=2 & M_{xy}=0 \\ M_{yx}=0 & M_{xx}=2 \end{bmatrix}$ to produce the scaled vector $\begin{bmatrix} 2v_x & 2v_y \end{bmatrix}$.

$$\begin{bmatrix} M_{xx}=2 & M_{xy}=0 \\ M_{yx}=0 & M_{xx}=2 \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} 2v_x + 0v_y \\ 0v_x + 2v_y \end{bmatrix}$$

Project to x axis

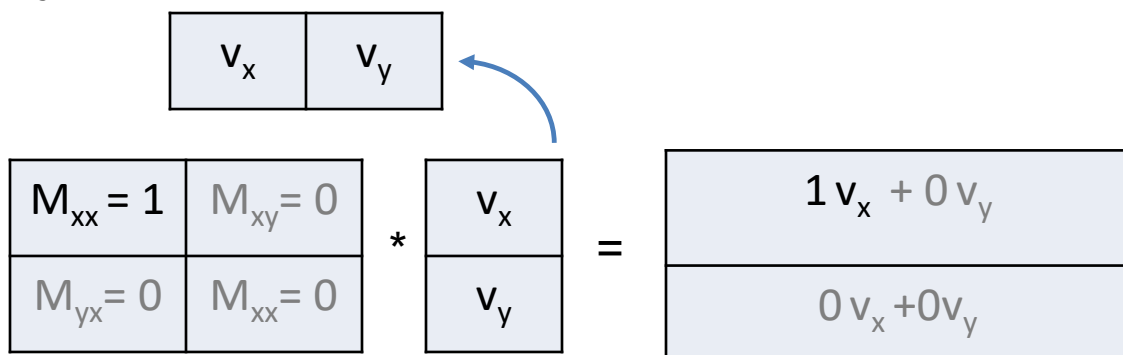


Diagram illustrating the Project to x axis transformation. A vector $\begin{bmatrix} v_x & v_y \end{bmatrix}$ is transformed by the Projection matrix $\begin{bmatrix} M_{xx}=1 & M_{xy}=0 \\ M_{yx}=0 & M_{xx}=0 \end{bmatrix}$ to produce the projected vector $\begin{bmatrix} 1v_x + 0v_y \\ 0v_x + 0v_y \end{bmatrix}$.

$$\begin{bmatrix} M_{xx}=1 & M_{xy}=0 \\ M_{yx}=0 & M_{xx}=0 \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} 1v_x + 0v_y \\ 0v_x + 0v_y \end{bmatrix}$$

Basic Matrix operations – homogeneous coordinates

Transformation matrix – in 2D

The diagram illustrates the matrix multiplication of a 2D transformation matrix with a 2D vector. At the top, a box contains the original vector components v_x and v_y . A blue curved arrow points from this box to the second vector box in the equation below. The equation shows a 2x2 matrix with elements M_{xx} , M_{xy} , M_{yx} , and M_{yy} multiplied by a 2x1 vector with elements v_x and v_y . The result is a 2x1 vector with elements $M_{xx} v_x + M_{xy} v_y$ and $M_{yx} v_x + M_{yy} v_y$.

$$\begin{bmatrix} M_{xx} & M_{xy} \\ M_{yx} & M_{yy} \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} M_{xx} v_x + M_{xy} v_y \\ M_{yx} v_x + M_{yy} v_y \end{bmatrix}$$

Basic Matrix operations – homogeneous coordinates

Transformation matrix – in 2D

$$\begin{bmatrix} M_{xx} & M_{xy} \\ M_{yx} & M_{yy} \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} M_{xx} v_x + M_{xy} v_y \\ M_{yx} v_x + M_{yy} v_y \end{bmatrix}$$

Homogeneous coordinates transformation matrix – in 2D

$$\begin{bmatrix} M_{xx} & M_{xy} & t_x \\ M_{yx} & M_{yy} & t_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} M_{xx} v_x + M_{xy} v_y + 1 * t_x \\ M_{yx} v_x + M_{yy} v_y + 1 * t_y \\ 0 v_x + 0 v_y + 1 * 1 = 1 \end{bmatrix}$$

2 Matrices and Transformations

Transformations refer to operations such as moving (also called *translating*), rotating, and scaling objects. They are stored in 3D programming using matrices, which are nothing but rectangular arrays of numbers. Multiple transformations can be performed very quickly using matrices. It turns out that a [4x4] matrix can represent all transformations. Having a unified matrix dimension for all transformations saves calculation time.

$$\begin{array}{c} \text{col(1)} \quad \text{col(2)} \quad \text{col(3)} \quad \text{col(4)} \\ \begin{array}{l} \text{row(1)} \\ \text{row(2)} \\ \text{row(3)} \\ \text{row(4)} \end{array} \begin{bmatrix} + & + & + & + \\ + & + & + & + \\ + & + & + & + \\ + & + & + & + \end{bmatrix} \end{array}$$

Matrix operations

The one operation that is most relevant in computer graphics is *matrix multiplication*. We will explain it with some detail.

Matrix multiplication

Matrix multiplication is used to apply transformations to geometry. For example if we have a point and would like to rotate it around some axis, we use a rotation matrix and multiply it by the point to get the new rotated location.

$$\begin{array}{c} \text{rotate matrix} \quad \text{input point} \quad \text{rotated point} \\ \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \end{array}$$

Most of the time, we need to perform multiple transformations on the same geometry. For example, if we need to move and rotate a thousand points, we can use either of the following methods.

Method 1

1. Multiply the move matrix by 1000 points to move the points.
2. Multiply the rotate matrix by the resulting 1000 points to rotate the moved points.

Number of operations = **2000**.

Method 2

1. Multiply the rotate and move matrices to create a combined transformation matrix.
2. Multiply the combined matrix by 1000 points to move and rotate in one step.

Number of operations = **1001**.

Notice that method 1 takes almost twice the number of operations to achieve the same result. While method 2 is very efficient, it is only possible if both the move and rotate matrices are $[4 \times 4]$. This is why in computer graphics a $[4 \times 4]$ matrix is used to represent all transformations, and a $[4 \times 1]$ matrix is used to represent points.

Three-dimensional modeling applications provide tools to apply transformations and multiply matrices, but if you are curious about how to mathematically multiply matrices, we will explain a simple example. In order to multiply two matrices, they have to have matching dimensions. That means the number of columns in the first matrix must equal the number of rows of the second matrix. The resulting matrix has a size equal to the number of rows from the first matrix and the number of columns from the second matrix. For example, if we have two matrices, **M** and **P**, with dimensions equal to $[4 \times 4]$ and $[4 \times 1]$ respectively, then their resulting multiplication matrix **M · P** has a dimension equal to $[4 \times 1]$ as shown in the following illustration:

$$\begin{array}{c} \mathbf{M} \end{array} \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{array}{c} \mathbf{P} \\ \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{array} = \begin{array}{c} \mathbf{P}' \\ \begin{bmatrix} x' = a \cdot x + b \cdot y + c \cdot z + d \cdot 1 \\ y' = e \cdot x + f \cdot y + g \cdot z + h \cdot 1 \\ z' = i \cdot x + j \cdot y + k \cdot z + l \cdot 1 \\ 1 = 0 \cdot x + 0 \cdot y + 0 \cdot z + 1 \cdot 1 \end{bmatrix} \end{array}$$

Identity matrix

The *identity matrix* is a special matrix where all diagonal components equal 1 and the rest equal 0.

1.0	0.0	0.0	0.0
0.0	1.0	0.0	0.0
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0

The main property of the identity matrix is that if it is multiplied by any other matrix, the values multiplied by zero do not change.

$$\begin{array}{c} \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \end{array} \times \begin{array}{c} \begin{bmatrix} 2.0 \\ 3.0 \\ 1.0 \\ 1.0 \end{bmatrix} \end{array} = \begin{array}{c} \begin{bmatrix} 1.0 \times 2.0 + 0.0 \times 3.0 + 0.0 \times 1.0 + 0.0 \times 1.0 \\ 0.0 \times 2.0 + 1.0 \times 3.0 + 0.0 \times 1.0 + 0.0 \times 1.0 \\ 0.0 \times 2.0 + 0.0 \times 3.0 + 1.0 \times 1.0 + 0.0 \times 1.0 \\ 0.0 \times 2.0 + 0.0 \times 3.0 + 0.0 \times 1.0 + 1.0 \times 1.0 \end{bmatrix} \end{array} = \begin{array}{c} \begin{bmatrix} 2.0 \\ 3.0 \\ 1.0 \\ 1.0 \end{bmatrix} \end{array}$$

Transformation operations

Most transformations preserve the parallel relationship among the parts of the geometry. For example collinear points remain collinear after the transformation. Also points on one plane stay coplanar after transformation. This type of transformation is called an *affine transformation*.

Translation (move) transformation

Moving a point from a starting position by certain a vector can be calculated as follows:

$$P' = P + V$$

Suppose:

$P(x,y,z)$ is a given point

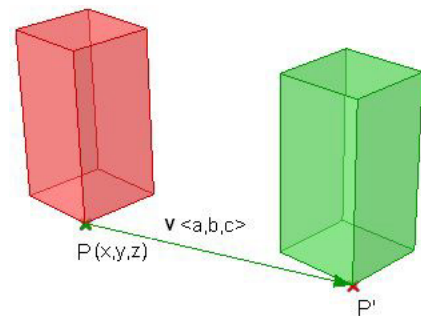
$\mathbf{v}\langle a,b,c\rangle$ is a translation vector

Then:

$$P'(x) = x + a$$

$$P'(y) = y + b$$

$$P'(z) = z + c$$



Points are represented in a matrix format using a $[4 \times 1]$ matrix with a 1 inserted in the last row. For example the point $P(x,y,z)$ is represented as follows:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Using a $[4 \times 4]$ matrix for transformations (what is called a homogenous coordinate system), instead of a $[3 \times 3]$ matrices, allows representing all transformations including translation. The general format for a translation matrix is:

$$\begin{bmatrix} 1 & 0 & 0 & a1 \\ 0 & 1 & 0 & a2 \\ 0 & 0 & 1 & a3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For example, to move point $P(2,3,1)$ by vector $\mathbf{v}\langle 2,2,2\rangle$, the new point location is:

$$P' = P + \mathbf{v} = (2+2, 3+2, 1+2) = (4, 5, 3)$$

If we use the matrix form and multiply the translation matrix by the input point, we get the new point location as in the following:

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} (1*2 + 0*3 + 0*1 + 2*1) \\ (0*2 + 1*3 + 0*1 + 2*1) \\ (0*2 + 0*3 + 1*1 + 2*1) \\ (0*2 + 0*3 + 0*1 + 1*1) \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 3 \\ 1 \end{bmatrix}$$

Similarly, any geometry is translated by multiplying its construction points by the translation matrix. For example, if we have a box that is defined by eight corner points, and we want to move it 4 units in the x-direction, 5 units in the y-direction and 3 units in the z- direction, we must multiply each of the eight box corner points by the following translation matrix to get the new box.

$$\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

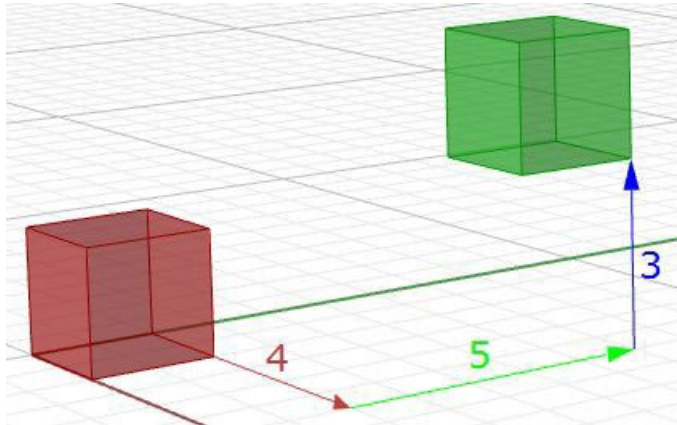


Figure (19): Translate all box corner points.



Rotation transformation

This section shows how to calculate rotation around the z-axis and the origin point using trigonometry, and then to deduce the general matrix format for the rotation. Take a point on x,y plane P(x,y) and rotate it by angle(b).

From the figure, we can say the following:

$$x = d \cos(a) \quad \text{---(1)}$$

$$y = d \sin(a) \quad \text{---(2)}$$

$$x' = d \cos(b+a) \quad \text{---(3)}$$

$$y' = d \sin(b+a) \quad \text{--- (4)}$$

Expanding x' and y' using trigonometric identities for the sine and cosine of the sum of angles:

$$x' = d \cos(a)\cos(b) - d \sin(a)\sin(b) \quad \text{---(5)}$$

$$y' = d \cos(a)\sin(b) + d \sin(a)\cos(b) \quad \text{---(6)}$$

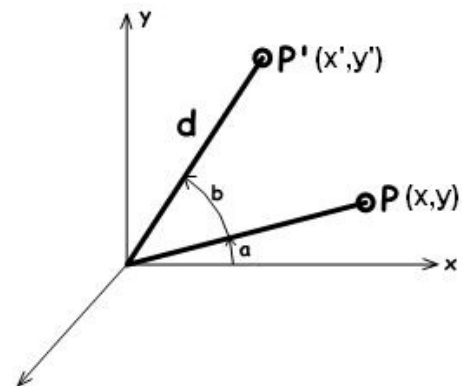
Using Eq 1 and 2:

$$x' = x \cos(b) - y \sin(b)$$

$$y' = x \sin(b) + y \cos(b)$$

The rotation matrix around the **z-axis** looks like:

$$\begin{bmatrix} \cos(b) & -\sin(b) & 0 & 0 \\ \sin(b) & \cos(b) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



The rotation matrix around the **x-axis** by angle **b** looks like:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(b) & -\sin(b) & 0 \\ 0 & \sin(b) & \cos(b) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotation matrix around the **y-axis** by angle **b** looks like:

$$\begin{bmatrix} \cos(b) & 0 & \sin(b) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(b) & 0 & \cos(b) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For example, if we have a box and would like to rotate it 30 degrees, we need the following:

1. Construct the 30-degree rotation matrix. Using the generic form and the cos and sin values of 30-degree angle, the rotation matrix will look like the following:

$$\begin{bmatrix} 0.87 & -0.5 & 0 & 0 \\ 0.5 & 0.87 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Multiply the rotation matrix by the input geometry, or in the case of a box, multiply by each of the corner points to find the box's new location.

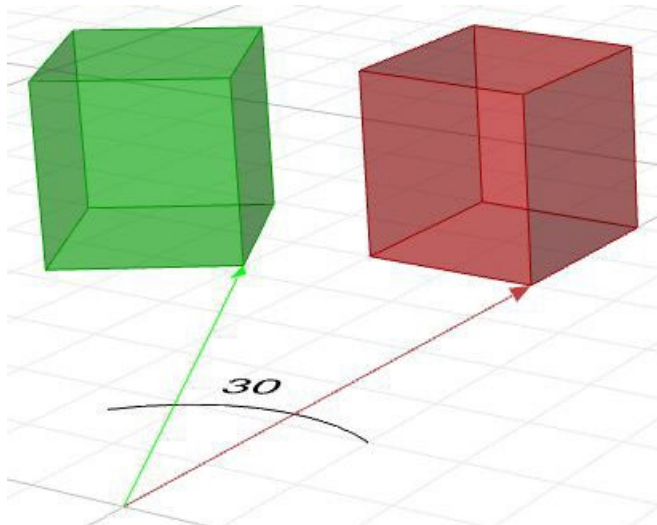


Figure (20): Rotate geometry.

Scale transformation



In order to scale geometry, we need a scale factor and a center of scale. The scale factor can be uniform scaling equally in x-, y-, and z-directions, or can be unique for each dimension. Scaling a point can use the following equation:

$$P' = \text{ScaleFactor}(S) * P$$

Or:

$$P'.x = S_x * P.x$$

$$P'.y = S_y * P.y$$

$$P'.z = S_z * P.z$$

This is the matrix format for scale transformation, assuming that the center of scale is the World origin point (0,0,0).

$$\begin{bmatrix} \text{Scale-x} & 0 & 0 & 0 \\ 0 & \text{Scale-y} & 0 & 0 \\ 0 & 0 & \text{Scale-z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For example, if we would like to scale a box by 0.25 relative to the World origin, the scale matrix will look like the following:

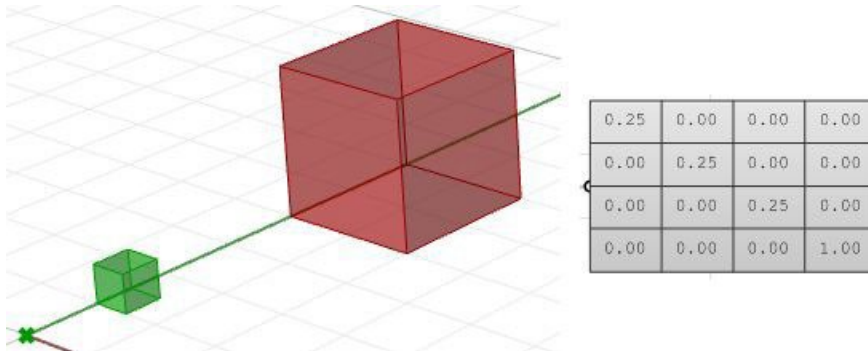


Figure (21): Scale geometry

Shear transformation

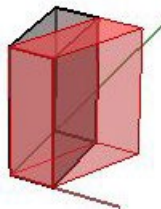


Shear in 3D is measured along a pair of axes relative to a third axis. For example, a shear along a z-axis will not change geometry along that axis, but will alter it along x and y. Here are few examples of shear matrices:

1. Shear in x and z, keeping the y-coordinate fixed:

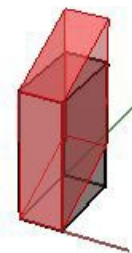
Shear x-axis

1.0	0.5	0.0	0.0
0.0	1.0	0.0	0.0
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0



Shear z-axis

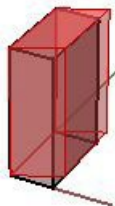
1.0	0.0	0.0	0.0
0.0	1.0	0.0	0.0
0.0	0.5	1.0	0.0
0.0	0.0	0.0	1.0



2. Shear in y and z, keeping the x-coordinate fixed:

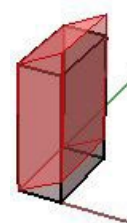
Shear y-axis

1.0	0.0	0.0	0.0
0.5	1.0	0.0	0.0
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0



Shear z-axis

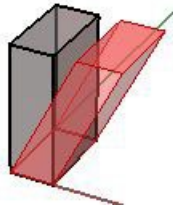
1.0	0.0	0.0	0.0
0.0	1.0	0.0	0.0
0.5	0.0	1.0	0.0
0.0	0.0	0.0	1.0



3. Shear in x and y, keeping the z-coordinate fixed:

Shear x-axis

1.0	0.0	0.5	0.0
0.0	1.0	0.0	0.0
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0



Shear y-axis

1.0	0.0	0.0	0.0
0.0	1.0	0.5	0.0
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0

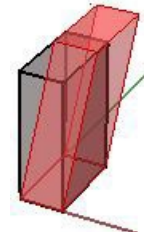
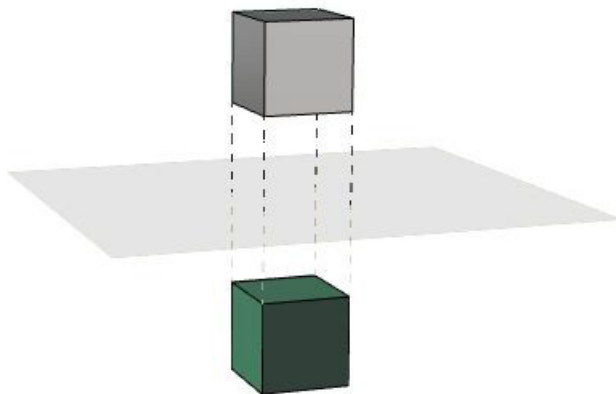


Figure (22): Shear Matrices.

Mirror or reflection transformation

The mirror transformation creates a reflection of an object across a line or a plane. 2-D objects are mirrored across a line, while 3-D objects are mirrored across a plane. Keep in mind that the mirror transformation flips the normal direction of the geometry faces.



1.000	0.000	0.000	0.000
0.000	1.000	0.000	0.000
0.000	0.000	-1.000	0.000
0.000	0.000	0.000	1.000

Figure (23): Mirror matrix across World xy-plane. Face directions are flipped.



Planar Projection transformation

Intuitively, the projection point of a given 3-D point $P(x,y,z)$ on the world xy -plane equals $P_{xy}(x,y,0)$ setting the z value to zero. Similarly, a projection to xz -plane of point P is $P_{xz}(x,0,z)$. When projecting to yz -plane, $P_{yz} = (0,y,z)$. Those are called orthogonal projections¹.

If we have a curve as an input, and we apply the planar projection transformation, we get a curve projected to that plane. The following shows an example of a curve projected to xy -plane with the matrix format.

Note: NURBS curves (explained in the next chapter) use control points to define curves. Projecting a curve amounts to projecting its control points.

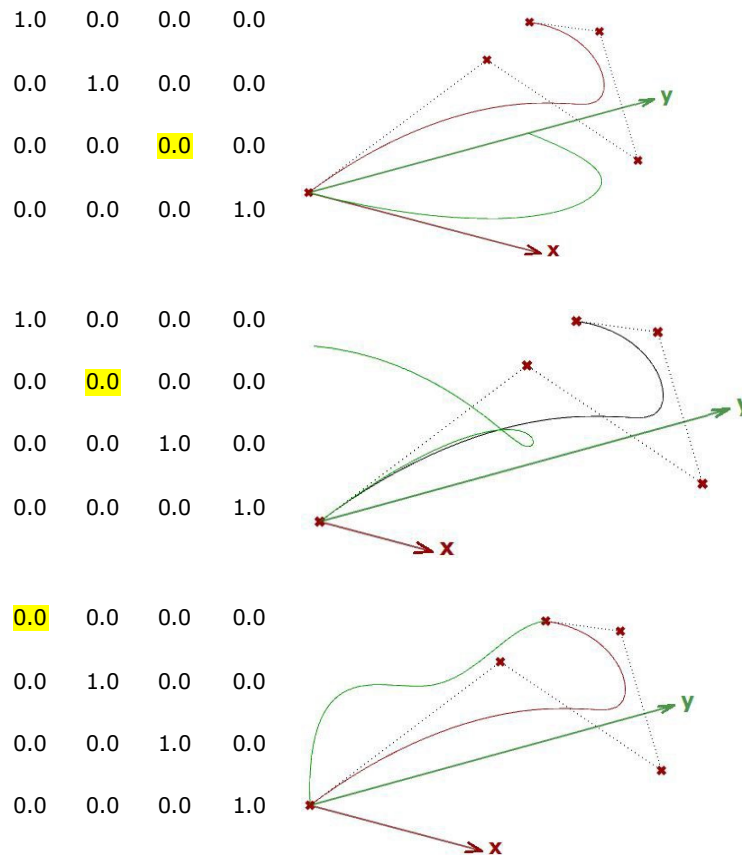
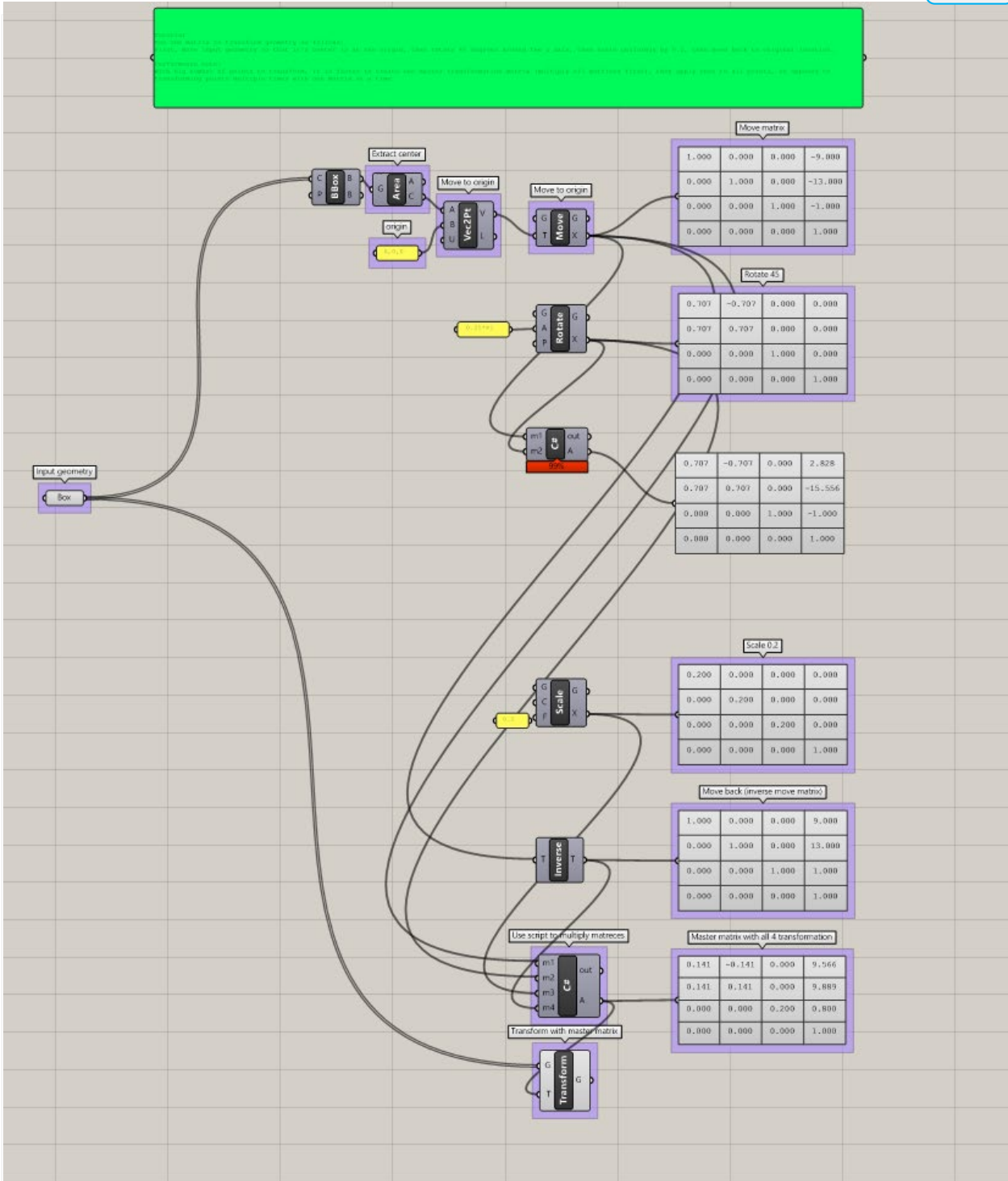


Figure (24): Projection matrices.

¹ [Wikipedia: Bézier curve](#).

Summary - all transformations & multiply



Make your own Matrix

