**Linear Algebra**

Contents

[1. Point and Vector 4](#_Toc94283809)

[2. Vector 4](#_Toc94283810)

[2.1. Properties of Vector 4](#_Toc94283811)

[2.1.1. Commutative 4](#_Toc94283812)

[2.1.2. Associative 4](#_Toc94283813)

[2.1.3. Distributive 4](#_Toc94283814)

[2.2. Magnitude of a Vector 4](#_Toc94283815)

[2.3. Normalize Vector 5](#_Toc94283816)

[2.3.1. Magnitude of a normalize vector is always 1 5](#_Toc94283817)

[2.4. Dot Product 6](#_Toc94283818)

[2.4.1. General Formulae 6](#_Toc94283819)

[2.4.2. Geometric Definition 6](#_Toc94283820)

[2.4.3. Orthogonal 6](#_Toc94283821)

[2.4.4. Parallel 6](#_Toc94283822)

[2.4.5. Tips on the sign of dot product of two vectors 6](#_Toc94283823)

[2.5. Cross Product 7](#_Toc94283824)

[3. Matrix 8](#_Toc94283825)

[3.1. Type of Matrices 8](#_Toc94283826)

[3.1.1. Row Matrix (horizontal) 8](#_Toc94283827)

[3.1.2. Column Matrix (vertical) 8](#_Toc94283828)

[3.1.3. Square Matrix (Row = Column) 8](#_Toc94283829)

[3.1.4. Zero/Null Matrix 8](#_Toc94283830)

[3.1.5. Identity Matrix 8](#_Toc94283831)

[3.1.6. Diagonal Matrix 8](#_Toc94283832)

[3.1.7. Scalar Matrix 8](#_Toc94283833)

[3.2. Operation of Matrix 9](#_Toc94283834)

[3.2.1. Add/Subtract 9](#_Toc94283835)

[3.2.2. Multiply 9](#_Toc94283836)

[3.2.3. Scalar Multiply 9](#_Toc94283837)

[3.2.4. Transpose 9](#_Toc94283838)

[3.2.5. Inverse 9](#_Toc94283839)

[3.3. Properties of Matrix 10](#_Toc94283840)

[3.3.1. Commutative 10](#_Toc94283841)

[3.3.2. Associative 10](#_Toc94283842)

[3.3.3. Distributive 10](#_Toc94283843)

[3.4. Properties of Matrix Multiplication 10](#_Toc94283844)

[3.4.1. Not Commutative 10](#_Toc94283845)

[3.4.2. Associative 10](#_Toc94283846)

[3.4.3. Identity 10](#_Toc94283847)

[3.4.4. Null Matrix 10](#_Toc94283848)

[3.5. Find Inverse Matrix for n\*n matrix 11](#_Toc94283849)

[3.5.1. Find Determinant 11](#_Toc94283850)

[3.5.2. Gauss Jordan 12](#_Toc94283851)

[4. Coordinate system intro 14](#_Toc94283852)

[5. Linear/Affine Transformation – TRS (part 1) 14](#_Toc94283853)

[5.1. Translation 14](#_Toc94283854)

[5.1.1. 2D 14](#_Toc94283855)

[5.1.2. 3D 14](#_Toc94283856)

[5.2. Scaling 14](#_Toc94283857)

[5.2.1. 2D 14](#_Toc94283858)

[5.2.2. 3D 14](#_Toc94283859)

[5.3. Rotate 14](#_Toc94283860)

[5.3.1. 2D 14](#_Toc94283861)

[6. Coordinates System 15](#_Toc94283862)

[6.1. Example 1 16](#_Toc94283863)

[6.1.1. Give a point has a coordinate , what is the coordinate in system A? 16](#_Toc94283864)

[6.1.2. Give a point has a coordinate , what is the coordinate in system B? 18](#_Toc94283865)

[7. Lines 20](#_Toc94283866)

[7.1.1. Parametric Form 20](#_Toc94283867)

[7.1.2. Point-Normal Form 20](#_Toc94283868)

[7.1.3. Implicit Form 20](#_Toc94283869)

[8. Collision Detection 20](#_Toc94283870)

[8.1.1. Circle-Circle collision 20](#_Toc94283871)

[8.1.2. OBB-Point collision 20](#_Toc94283872)

[8.1.3. Ray-Point check 20](#_Toc94283873)

[8.1.4. Ray-Ray check 20](#_Toc94283874)

[8.1.5. Cone-Point check 20](#_Toc94283875)

[8.1.6. Circle-Line collision 20](#_Toc94283876)

[8.1.7. OBB-Line collision 20](#_Toc94283877)

[9. Collision Intersection 21](#_Toc94283878)

[10. Transformation 22](#_Toc94283879)

[10.2. Orthogonal Projections 22](#_Toc94283880)

[10.2.1. Example 1 How to find? 22](#_Toc94283881)

[10.2.2. General Formulae: 22](#_Toc94283882)

[10.3. Reflections 23](#_Toc94283883)

[10.3.1. Example 1 How to find? 23](#_Toc94283884)

[10.3.2. General Formulae: 23](#_Toc94283885)

[10.4. Parallel Projections 23](#_Toc94283886)

[10.5. Shear 23](#_Toc94283887)

[11. 3D space 23](#_Toc94283888)

# Point and Vector

* A point is a physical location on a coordinates system
  + 2D point
  + 3D point
* A vector is just a scalar or a direction (non-physical)
  + 2D point
  + 3D point

# Vector

## Properties of Vector

### Commutative

### Associative

### Distributive

Where r is a scalar

## Magnitude of a Vector

* Also known as distance between 2 points
* Also known as length of vector
* Example:
  + Let
    - Vector have a length of 5

## Normalize Vector

* Get the unit vector of the vector
* Each value is between -1 to 1
* Example

### Magnitude of a normalize vector is always 1

## Dot Product

### General Formulae

* Dot product also known as scalar product
* Dot product also known as inner product
* Example

### Geometric Definition

**Prove:**

### Orthogonal

* Meaning two vectors are 90 degrees apart, or perpendicular to each other

### Parallel

* These two vectors are colinear to each other, points at the same direction

### Tips on the sign of dot product of two vectors

* When Negative value, two vectors lie on the different half
* When value is = 0, two vectors is orthogonal
* When value is = 1 or -1, two vectors is parallel to each other
  + 1 being facing exactly the same direction
  + -1 being facing the exact opposite direction
* When Positive value, two vectors lie on the same half

## Cross Product

* + - * 3D spaces only, Vector 3 only
      * Right-hand rule
      * Finding a 3rd vector that is both perpendicular to the 2 vectors if it exists

# Matrix

A matrix is denoted by

## Type of Matrices

### Row Matrix (horizontal)

### Column Matrix (vertical)

### Square Matrix (Row = Column)

### Zero/Null Matrix

### Identity Matrix

A square matrix contains 0 and 1, where 1 are diagonal across.

### Diagonal Matrix

A square matrix where only the diagonal has value, the rest is 0.

### Scalar Matrix

A square diagonal matrix where all the diagonal values are the same.

## Operation of Matrix

### Add/Subtract

### Multiply

### Scalar Multiply

### Transpose

### Inverse

## Properties of Matrix

### Commutative

### Associative

### Distributive

## Properties of Matrix Multiplication

### Not Commutative

### Associative

### Identity

### Null Matrix

## Find Inverse Matrix for n\*n matrix

### Find Determinant

* + - * 2x2 matrix
      * 3x3 matrix
      * 4x4 matrix

### Gauss Jordan

* + - * Start off with the matrix you are going to inverse on the left and an identity matrix on the right
      * Objective is to make the left side matrix into an identity matrix if determinant is not 0
      * Change the 1st row 1st digit into 1 by dividing by the value which is 4,
      * Divide the whole of 1st row
      * Next, change the other value in the same column to 0 by multiplying and then minus
      * 2nd column is 4, therefore multiply the first row by 4 temporary and then the 2nd column minus away the 1st.
      * Solve the next column:
      * At this point, the method on how to the rest of the column is the exact same method being used for the 1st row and column
      * Solve the rest:

Divide 2nd row, 2nd column by 3:

Change 1st row 2nd column to zero by -1/2 and minus away:

Change 3rd row 2nd column to zero by 4 and minus away:

Since 3rd row, 3rd column is already 1, no action needed, procced to multiply and minus step:

Check:

Summary: Divide, multiply, minus

# Coordinate system intro

A coordinate system consists of

* 1. An origin
  2. Axis Vectors
     1. 2 vectors (in 2D space)
     2. 3 vectors (in 3D space)

# Linear/Affine Transformation – TRS (part 1)

## Translation

### 2D

### 3D

## Scaling

### 2D

### 3D

## Rotate

### 2D

3D rotation is more complicated, look up for computer graphic, 3D rotation topics

# Coordinates System

* Local to Global
* Global to Local
* Local to Local

Technically, **you do not need to care whether it is from local to global or vice versa;** it is about **mapping from one general coordinate system to another general coordinate system**.

But in other topics such as Computer Graphics, you will learn terms and names such as Model/Local Space to represent a general coordinate system.

* A coordinate system consist of
  + An origin (translation)
  + Axis Vectors (scaling)
    - 2 vectors (in 2D space)
    - 3 vectors (in 3D space)

Be default, a UNIT coordinate system is defined as:

* + An origin of
  + Axis Vectors (scaling)
    - 2D = 2 vectors
    - 3D = 3 vectors

**Change of coordinates system formulae:**

Some tips and note if you want terms and names to understand:  
The equation of yields the coordinates in system A from system B.

As seen from below example, **if the coordinates System A is a Unit coordinate system**, the computation only needed **,** which yields the coordinates in system A.

The transformation of **,** can be seen as LOCAL to GLOBAL transformation, OR also known as MODEL to WORLD transformation in computer graphic.

The transformation of can be seen as LOCAL to GLOBAL to LOCAL transformation.

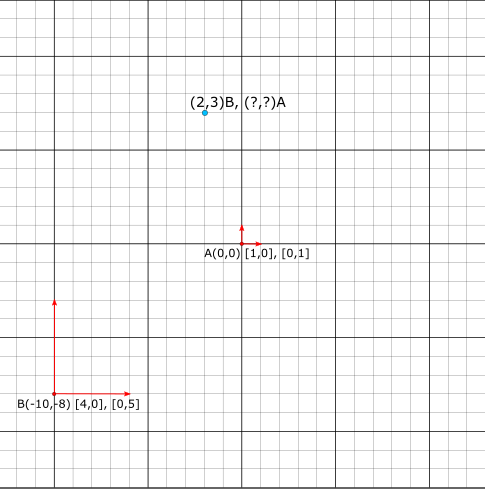
Can you see it?

## Example 1

A coordinate system A by the point of origin at

A coordinate system B by the point of origin at

### Give a point has a coordinate , what is the coordinate in system A?

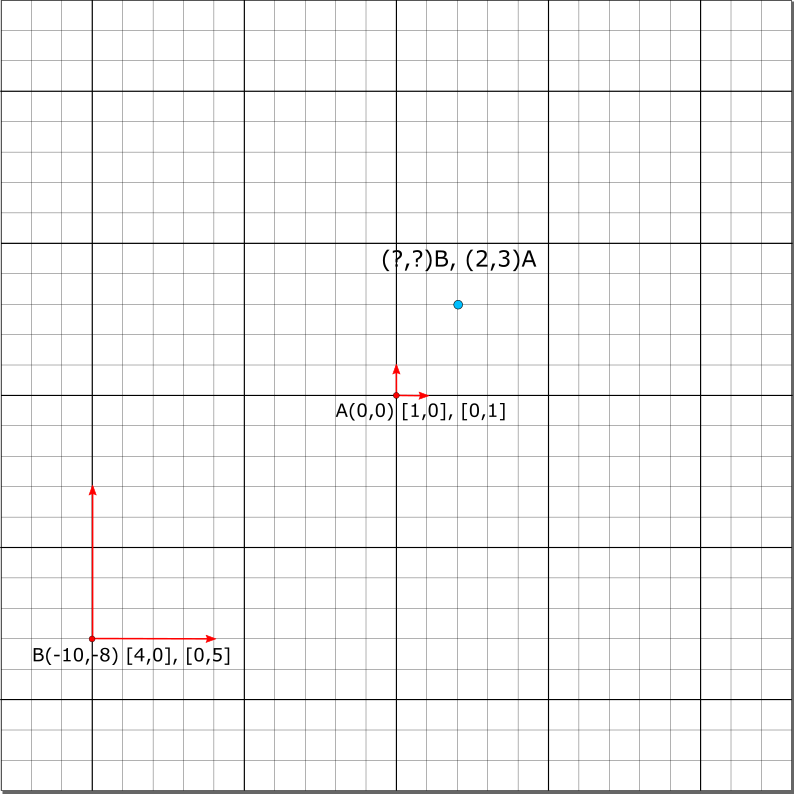


* + - * **Step 1: Form the coordinates system matrix**

Find unknown v, A, B, u is known

* + - * **Step 2: Find the Inverse Matrix**
      * In this example A, the inverse matrix of A is itself since it is an identity matrix
      * **Step 3: Solve the Matrix Multiplication**

### Give a point has a coordinate , what is the coordinate in system B?

****

* + - * **Step 1: Form the coordinates system matrix**

Find unknown v, A, B, u is known

* + - * **Step 2: Find the Inverse Matrix**
      * In this example A, the inverse matrix of A is itself since it is an identity matrix
      * **Step 2.5: Find the Inverse Matrix using Gauss Jordon**

1st row divide by 4:

Flip the sign:

Check:

* + - * **Step 3: Solve the Matrix Multiplication**

# Lines

### Parametric Form

### Point-Normal Form

### Implicit Form

# Collision Detection

### Circle-Circle collision

* + - * 1st Get the distance between the 2 circles, which is also the length of the vector between the 2 centres of the 2 circles
      * Lastly, compare the distance by the sum of the radius of the 2 circles
      * If is the distance is less than the value, 2 circles have collided else not

Optimizing

* + - * In practise, finding the length/distance of a vector requires square root, which is computation expensive.
      * We learn that dot product of itself yield the square length of the vector, using this facts, we can optimize the equation

### OBB-Point collision

### Ray-Point check

### Ray-Ray check

### Cone-Point check

### Circle-Line collision

### OBB-Line collision

# Collision Intersection

# Transformation

Steps in composing the transformation matrix

* + - * Always use the unit vector
      * Try working out with 1 vector first
      * Then compute each vector e respectively to get each column for the matrix
      * 2D and 3D is the same method

## Orthogonal Projections

Project a vector onto an arbitrary vector

### Example 1 How to find?

### General Formulae:

Rewrite:

Compose:

Expand:

## Reflections

Reflect a vector over an arbitrary vector

### Example 1 How to find?

### General Formulae:

## Parallel Projections

## Shear

# 3D space