**Linear Algebra**

Contents

[1. Coordinate system intro 3](#_Toc123625773)

[2. Linear/Affine Transformation – TRS (part 1) 3](#_Toc123625774)

[2.1. Translation 3](#_Toc123625775)

[2.1.1. 2D 3](#_Toc123625776)

[2.1.2. 3D 3](#_Toc123625777)

[2.2. Scaling 3](#_Toc123625778)

[2.2.1. 2D 3](#_Toc123625779)

[2.2.2. 3D 3](#_Toc123625780)

[2.3. Rotate 3](#_Toc123625781)

[2.3.1. 2D 3](#_Toc123625782)

[3. Coordinates System 4](#_Toc123625783)

[3.1. Example 1 5](#_Toc123625784)

[3.1.1. Give a point has a coordinate , what is the coordinate in system A? 5](#_Toc123625785)

[3.1.2. Give a point has a coordinate , what is the coordinate in system B? 7](#_Toc123625786)

[4. Lines 9](#_Toc123625787)

[4.1.1. Parametric Form 9](#_Toc123625788)

[4.1.2. Point-Normal Form 9](#_Toc123625789)

[4.1.3. Implicit Form 9](#_Toc123625790)

[5. Collision Detection 9](#_Toc123625791)

[5.1.1. Circle-Circle collision 9](#_Toc123625792)

[5.1.2. OBB-Point collision 9](#_Toc123625793)

[5.1.3. Ray-Point check 9](#_Toc123625794)

[5.1.4. Ray-Ray check 9](#_Toc123625795)

[5.1.5. Cone-Point check 9](#_Toc123625796)

[5.1.6. Circle-Line collision 9](#_Toc123625797)

[5.1.7. OBB-Line collision 9](#_Toc123625798)

[6. Collision Intersection 10](#_Toc123625799)

[7. Composing Transformation Matrix 11](#_Toc123625800)

[7.2. Orthogonal Projections 11](#_Toc123625801)

[7.2.1. Example 1 How to find? 11](#_Toc123625802)

[7.2.2. General Formulae: 11](#_Toc123625803)

[7.3. Reflections 12](#_Toc123625804)

[7.3.1. Example 1 How to find? 12](#_Toc123625805)

[7.3.2. General Formulae: 12](#_Toc123625806)

[7.4. Parallel Projections 12](#_Toc123625807)

[7.5. Shear 12](#_Toc123625808)

[8. 3D space 12](#_Toc123625809)

# 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.

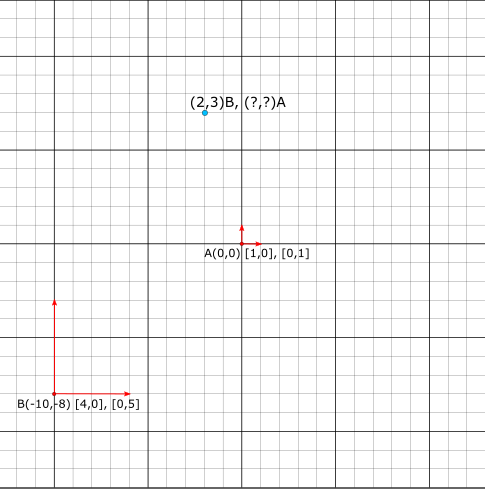
See below. Can you see it?

## 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?

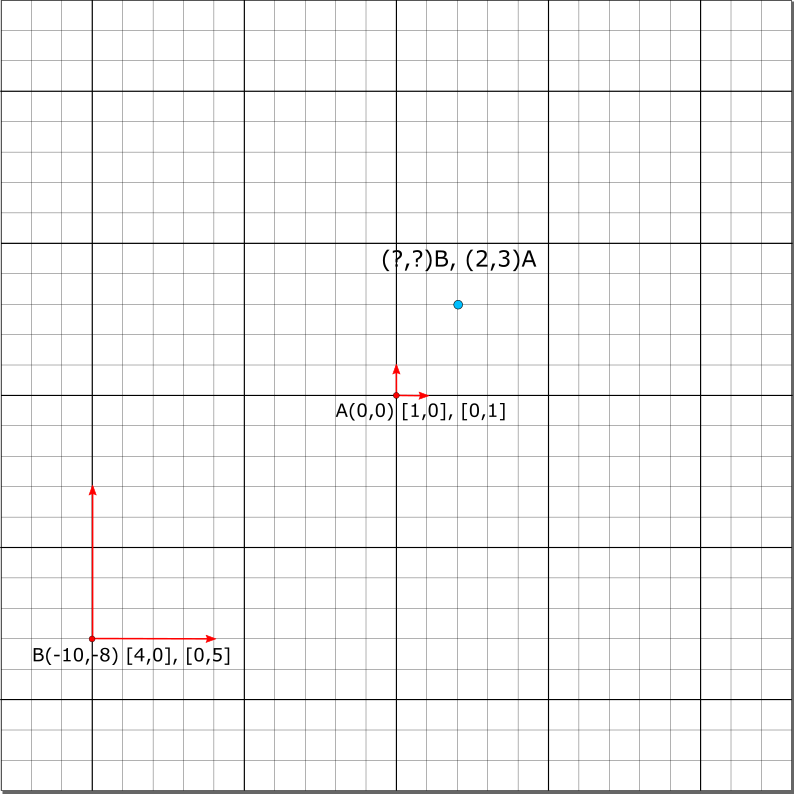


* + - * **Step 1: Form the coordinates system matrix**

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:

3rd row multiply \* then plus 1st row:

2nd row divide by 5:

3rd row multiply \* then plus 2nd row:

Check:

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

# Lines

### Parametric Form

### Point-Normal Form

### Implicit Form

# Composing Transformation Matrix

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 uses the same method

## Orthogonal Projections

Projecting a vector onto an arbitrary vector

* + - * Dot product between 2 vectors yields a scalar number, which comes in two scenario;
      * Scenario 1, Vector A is a non-unit vector; Vector B is a unit vector. Dot product yields a scalar numbers which tells us how many the ratio of Vector A : Vector B (s : 1) units
      * Scenario 2, Vector A is a non-unit vector; Vector B is a non-unit vector, normally dot product between these two vectors is use to check the direction of the vector, normalizing is not needed. The value is not really useful, except for the sign of the number which tells us the direction between these two vectors are facing. Positive, negative number, -1, 1 and 0 tells us the direction between these two vectors

### Steps to find

* + - * Normalize the arbitrary vector to a unit vector
      * Dot product to a scalar, s
      * Multiply scalar s to the unit vector

### General Formulae:

Rewrite:

Compose:

Expand:

## Reflections

Reflect a vector over an arbitrary vector

### How to find?

### General Formulae:

## Parallel Projections

## Shear

# 3D space