

Train Simulator Matrix

The basis of the mathematics behind the coordinate system is very complex

To keep the easy to understand in relation to train simulator we don't need to understand the mathematics only the application

If you're interested then just choose the key works and google search your interests

The **Euler angles** are three angles introduced by [Leonhard Euler](#) to describe the [orientation](#) of a [rigid body](#) with respect to a fixed [coordinate system](#).^[1] They can also represent the orientation of a mobile [frame of reference](#) in physics or the orientation of a general [basis](#) in [3-dimensional](#) linear algebra.

Any orientation can be achieved by composing three [elemental rotations](#), i.e. rotations about the axes of a [coordinate system](#). Euler angles can be defined by three of these rotations. They can also be defined by elemental [geometry](#) and the geometrical definition demonstrates that three rotations are always sufficient to reach any frame.

Information from https://en.wikipedia.org/wiki/Euler_angles - If you're looking for more information

Matrix

Train Simulator uses a 4 point Matrix which is the 3 point Matrix extended to include position relative to the origin/insertion point

A 4x4 matrix can be used to do both rotation and translation in a single matrix.

So to convert a 3x3 matrix to a 4x4, you simply copy in the values for the 3x3 upper left block, like so:

3x3 Matrix

a11	a12	a13
a21	a22	a23
a31	a32	a33

4x4 Matrix

a11	a12	a13	0
a21	a22	a23	0
a31	a32	a33	0
0	0	0	1

Notice in the 4x4 Matrix you can see the 3x3 matrix

The extra 0's values are used for position and global scale

Train Simulator uses the 4x4 matrix in two formats and the first format can be found in the class47.GeoPcDx

</Material>

<SourceLToPTransform>

```
<e d:numElements="16" d:elementType="sFloat32" d:precision="string">1.0000000 0.0000000 0.0000000 0.0000000
0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000
0.0000000 1.0000000</e>
```

```
<e d:numElements="16" d:elementType="sFloat32" d:precision="string">0.9999939 0.0000000 0.0034907 0.0000000
0.0000000 1.0000000 0.0000000 0.0000000 -0.0034907 0.0000000 0.9999939 0.0000000 -0.0001316 3.7562492
5.9099998 1.0000000</e>
```

</SourceLToPTransform>

You can see 16 values with 7 decimal places and is the 4x4 Matrix for the locomotive components

Two example of the many in the default Kuju class47.GeoPcDx file as an example

The second format is the child Matrix from the Default Kuju Class 47_br.bin

This example is the child object for the locomotive as a Train Driver

You can see 16 lines of data with the scale rotation and position data in the 4x4 Matrix for the locomotive below

The numbers assigned from 1 to 16 are to identify the 16 values to match the Matrix

```

<ContainerComponent>
  <cEntityContainerBlueprint>
    <TrackTerrainInEditor d:type="bool">0</TrackTerrainInEditor>
    <Children>
      <cEntityContainerBlueprint-sChild d:id="57419256">
        <ChildName d:type="cDeltaString">Driver</ChildName>
        <BlueprintID>
          <iBlueprintLibrary-cAbsoluteBlueprintID>
            <BlueprintSetID>
              <iBlueprintLibrary-cBlueprintSetID>
                <Provider d:type="cDeltaString">Kuju</Provider>
                <Product d:type="cDeltaString">RailSimulator</Product>
              </iBlueprintLibrary-cBlueprintSetID>
            </BlueprintSetID>
            <BlueprintID d:type="cDeltaString">Scenery\Characters\TrainDriver70_01.xml</BlueprintID>
          </iBlueprintLibrary-cAbsoluteBlueprintID>
        </BlueprintID>
        <Matrix>
          <cHcRMatrix4x4>
            <Element>
1      Scale X      <e d:type="sFloat32" d:alt_encoding="5072874D64E6EFBF" d:precision="string">-0.996874</e>
2                    <e d:type="sFloat32" d:alt_encoding="0000000000000000" d:precision="string">0</e>
3                    <e d:type="sFloat32" d:alt_encoding="EE940ED6FF39B4BF" d:precision="string">-0.07901</e>
4                    <e d:type="sFloat32" d:alt_encoding="0000000000000000" d:precision="string">0</e>
5                    <e d:type="sFloat32" d:alt_encoding="0000000000000000" d:precision="string">0</e>
6      Scale Y      <e d:type="sFloat32" d:alt_encoding="000000000000F03F" d:precision="string">1</e>
7                    <e d:type="sFloat32" d:alt_encoding="0000000000000000" d:precision="string">0</e>
8                    <e d:type="sFloat32" d:alt_encoding="0000000000000000" d:precision="string">0</e>
9                    <e d:type="sFloat32" d:alt_encoding="EE940ED6FF39B43F" d:precision="string">0.07901</e>
10                   <e d:type="sFloat32" d:alt_encoding="0000000000000000" d:precision="string">0</e>
11   Scale Z        <e d:type="sFloat32" d:alt_encoding="5072874D64E6EFBF" d:precision="string">-0.996874</e>
12                   <e d:type="sFloat32" d:alt_encoding="0000000000000000" d:precision="string">0</e>
13   Position X     <e d:type="sFloat32" d:alt_encoding="00E65AB400EDE6BF" d:precision="string">-0.716431</e>
14   Position Y     <e d:type="sFloat32" d:alt_encoding="EF8FF7AA9589FA3F" d:precision="string">1.65859</e>
15   Position Z     <e d:type="sFloat32" d:alt_encoding="40FB9122321C2040" d:precision="string">8.05507</e>
16   Scale XYZ      <e d:type="sFloat32" d:alt_encoding="000000000000F03F" d:precision="string">1</e>
            </Element>
          </cHcRMatrix4x4>
        </Matrix>
      </cEntityContainerBlueprint-sChild>
    </Children>
  </cEntityContainerBlueprint>
</ContainerComponent>

```

From left to right in that row of numbers, this is the position of each digit in the 4x4 matrix - the top left corner is a 3x3 rotation matrix, the last column, 13-14-15 is for translation (ie positioning). Scaling is along the top-left to bottom-right diagonal.

```

1 5 9 13
2 6 10 14
3 7 11 15
4 8 12 16

```

Scaling:

Fairly obvious, position 1 is scaling by X, 6 is scaling by Y, 11 is scaling by Z, and 16 is to scale on axis X Y Z
To scale an object using value 16 can give inconsistent results
But is usually an issue when rotation and scale are not applied correctly during the modelling process

$$S_v = \begin{bmatrix} v_x & 0 & 0 & 0 \\ 0 & v_y & 0 & 0 \\ 0 & 0 & v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Rotation:

As mentioned before, this is the top left corner of the 4x4 matrix above.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Top to bottom, rotation around X, around Y, around Z. You need to multiply the existing matrix with that, not just replace it, or you will lose any scaling.

To calculate the values using the algebra formula is theory we don't need to learn if we use a rotation calculator

The table below shows the common values required to rotate an object

Learning how to read the 3x3 Matrix and apply the values to the correct line in the 4x4 Matrix is required

Rotation for singular or multiple axis requires a **Matrix Calculator**

<https://www.andre-gaschler.com/rotationconverter/>

Link to the 3D Rotation Converter as seen below



3D Rotation Converter

Input

Input angle format ☐ Radians ☒ Degrees

Rotation matrix

1	0	0
0	1	0
0	0	1

Quaternion

x y z w (real part)

Axis-angle

Axis x y z Angle (degrees)

Axis with angle magnitude (degrees)

Axis x y z

Euler angles of multiple axis rotations (degrees)

XYZ x y z

Output

Output angle format ☐ Radians ☒ Degrees

Rotation matrix

```
[ 1.0000000, 0.0000000, 0.0000000;
 0.0000000, 1.0000000, 0.0000000;
 0.0000000, 0.0000000, 1.0000000 ]
```

Quaternion [x, y, z, w]

```
[ 0, 0, 0, 1 ]
```

Axis-Angle {[x, y, z], angle (degrees)}

```
{ [ 0, 0, 0 ], 0 }
```

Axis with angle magnitude (degrees) [x, y, z]

```
[ 0, 0, 0 ]
```

Euler angles (degrees) XYZ

```
[ x: 0, y: 0, z: 0 ]
```

Rotation in Train Simulator is usually in the horizontal plane, but multiple axis can be used for the calculation

First, we need to understand the axis of rotation to input the correct axis to rotate the object around

I will explain the details and how to use as we work the Practical Example below

Modelling in the Object creation software generally uses Euler XYZ

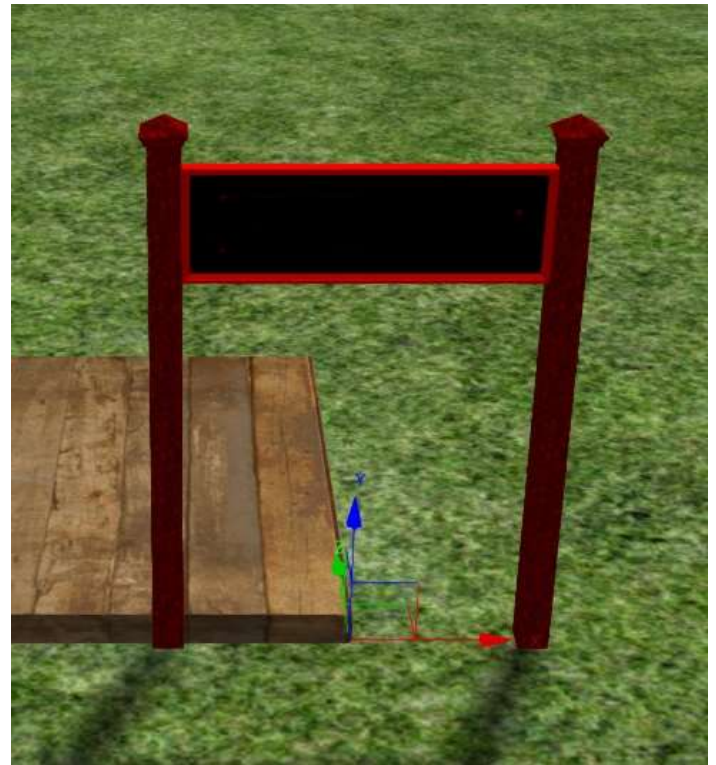
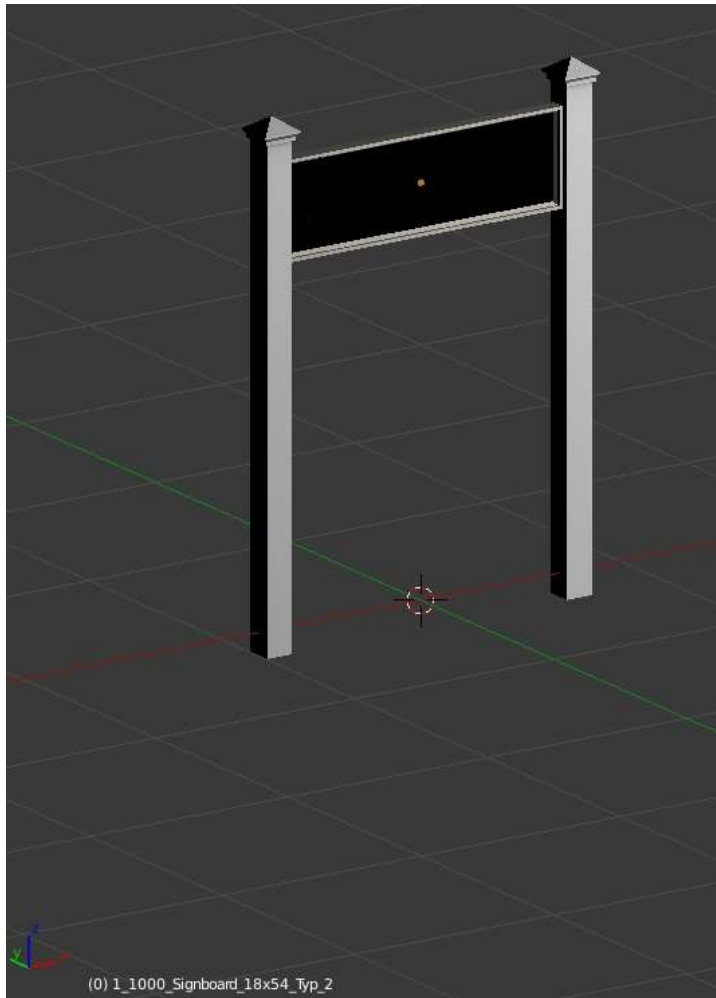
The Z axis is vertical

During the export and Blueprint process the object axis is rotated and requires the object to be rotated around the Y axis

I am using Blender to Train Simulator with the BRIAGE_G_2.79 export utility so may be a process of the Blender translation

Please check your 3D model to Blender process to ensure you're seeing a similar result

Note the most important check is in the Train Simulator Route Editor to check the Y axis is pointing up



Sign in the editor Note the blue Y axis is up and the red X axis is horizontal. I have placed the wood walkway at the pivot to prove the Insertion/Origin point is the pivot for the sign

The image to the left is in Blender and shows the Blue Z axis vertical and the Red X axis horizontal as modelled

Once you have checked the Y axis is the axis of rotation for the horizontal pivot we can look at how to create the calculation

Before we start let's look at the 3D Rotation Converter to show the correct settings

First up select the degrees for both Input and Output with the Input Angle Format radio buttons

Check the Rotation Matrix is set according to the XYZ Euler Angle Format

If you compare to the Scaling section above, you will see the same pattern and read the 1 values as the scale values

Not if the Euler is option at the bottom of the Input table is changed you will see a different matrix pattern

Note the Rotation Matrix Format at the Top of the Output table shows the same pattern as the 3x3 Matrix

This confirms we have the correct 3x3 Matrix format

To keep this simple for a single plane rotation we can use the Euler Angles fields at the bottom of the Input Table

Place 90 degrees in the Y field and you will see the output table populate

This will Rotate the X & Z value by 90 degrees

Rotation Matrix

Base Values

```
[ 1.0000000, 0.0000000, 0.0000000;
  0.0000000, 1.0000000, 0.0000000;
  0.0000000, 0.0000000, 1.0000000 ]
```

Rotated 90 degs on Y Axis

```
[ 0.0000000, 0.0000000, 1.0000000;
  0.0000000, 1.0000000, 0.0000000;
 -1.0000000, 0.0000000, 0.0000000 ]
```

Using the Converter, we can create a common angle rotation table as a quick reference using some standard angles

These will be the values used to confirm that we have the correct inputs and outputs and the 3x3 to 4x4 matrix conversion is correct

Matrix Rotation Table on Axis Y – Horizontal Plane

4x4 Matrix	Value 1	Value 2	Value 3	Value 5	Value 6	Value 7	Value 9	Value 10	Value 11
0 Degr	1.0000000	0.0000000	0.0000000	0.0000000	1.0000000	0.0000000	0.0000000	0.0000000	1.0000000
30 Degr	0.8660254	0.0000000	0.5000000	0.0000000	1.0000000	0.0000000	-0.5000000	0.0000000	0.8660254
-30 Degr	0.8660254	0.0000000	-0.5000000	0.0000000	1.0000000	0.0000000	0.5000000	0.0000000	0.8660254
45 Degr	0.7071068	0.0000000	0.7071068	0.0000000	1.0000000	0.0000000	-0.7071068	0.0000000	0.7071068
-45 Degr	0.7071068	0.0000000	-0.7071068	0.0000000	1.0000000	0.0000000	0.7071068	0.0000000	0.7071068
60 Degr	0.5000000	0.0000000	0.8660254	0.0000000	1.0000000	0.0000000	-0.8660254	0.0000000	0.5000000
-60 Degr	0.5000000	0.0000000	-0.8660254	0.0000000	1.0000000	0.0000000	0.8660254	0.0000000	0.5000000
90 Degr	0.0000000	0.0000000	1.0000000	0.0000000	1.0000000	0.0000000	-1.0000000	0.0000000	0.0000000
-90 Degr	0.0000000	0.0000000	-1.0000000	0.0000000	1.0000000	0.0000000	1.0000000	0.0000000	0.0000000
180 Degr	-1.0000000	0.0000000	0.0000000	0.0000000	1.0000000	0.0000000	0.0000000	0.0000000	-1.0000000
3x3 Matrix	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6	Value 7	Value 8	Value 9

Practical Examples

GD Ide Halt.GeoPcDx Matrix 30 Deg Rotation on Y Axis

```
</Material>
  <SourceLToPTransform>
    <e d:numElements="16" d:elementType="sFloat32" d:precision="string">
  </SourceLToPTransform
```

GD Ide Halt.GeoPcDx Matrix 45 Deg Rotation on Y Axis

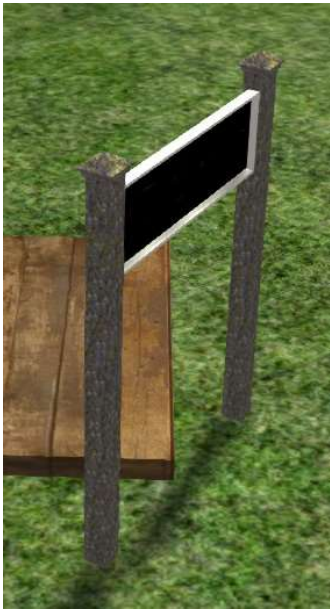
```
</Material>
  <SourceLToPTransform>
    <e d:numElements="16" d:elementType="sFloat32" d:precision="string">0.7071068 0.0000000 0.7071068
    0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 -0.7071068 0.0000000 0.7071068 0.0000000 0.0000000
    1.8300000 0.0000000 1.0000000</e>
  </SourceLToPTransform
```



Left 30 degree rotation on Y Axis

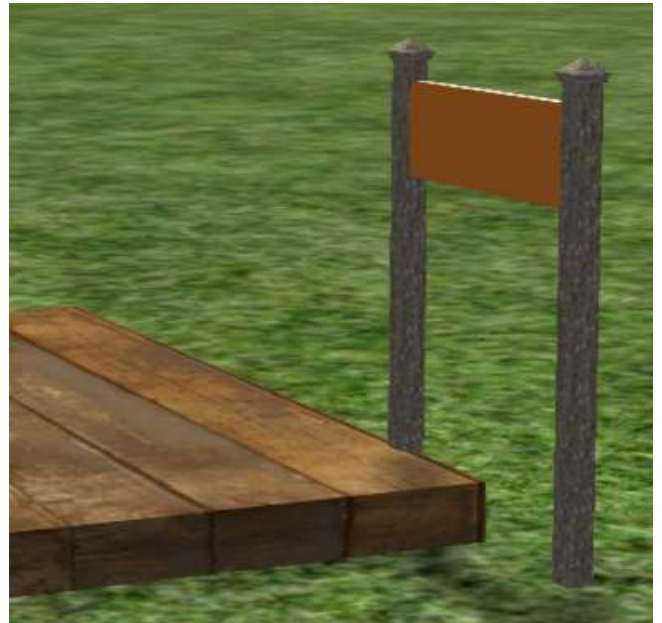
Right 45 degree rotation on Y Axis





Left 60 degree rotation on Y Axis

Right 90 degree rotation on Y Axis
Scale Value 2 for a half scale Sign



GD Ide Halt.GeoPcDx Matrix 60 Deg Rotation on Y Axis

```
</Material>
  <SourceLToPTransform>
    <e d:numElements="16" d:elementType="sFloat32" d:precision="string">0.5000000 0.0000000 0.8660254
    0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 -0.8660254 0.0000000 0.5000000 0.0000000 0.0000000
    1.8300000 0.0000000 1.0000000</e>
  </SourceLToPTransform>
```

GD Ide Halt.GeoPcDx Matrix 90 Deg Rotation on Y Axis with 2 Global Scale = ½ Scale Sign

```
</Material>
  <SourceLToPTransform>
    <e d:numElements="16" d:elementType="sFloat32" d:precision="string">0.0000000 0.0000000 1.0000000
    0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 -1.0000000 0.0000000 0.0000000 0.0000000 0.0000000
    1.8300000 0.0000000 2.0000000</e>
  </SourceLToPTransform>
```

The information sources used:

https://en.wikipedia.org/wiki/Euler_angles
<http://forums.uktrainsim.com/viewtopic.php?f=308&t=124596>
<https://www.andre-gaschler.com/rotationconverter/>
<https://stackoverflow.com/questions/4833380/how-to-convert-a-3x3-rotation-matrix-into-4x4-matrix>

Please review the Matrix Guide and compare with your own 3D editing software to Train Simulator to ensure the coordinate system give the same Y axis for Horizontal rotation in the Train Simulator Route Editor

If you find any errors or omission that could be added to the guide

Message Auscgu at UK TrainSim