

# New Page

- ☐ **Vectors describe positions or directions within a coordinate space.**
- ☐ **Matrices describe how to transform vectors from one space to another.**

“ [ Xaxis.x Yaxis.x Zaxis.x Tx ]  
[ Xaxis.y Yaxis.y Zaxis.y Ty ]  
[ Xaxis.z Yaxis.z Zaxis.z Tz ]  
[ 0 0 0 1 ]

A 4x4 transformation matrix is made up of:

- **3 direction vectors** (X, Y, Z axes — orientation)
- **1 position vector** (T — translation)
- **1 extra row** that supports **homogeneous coordinates**

Action	Result
Set <input type="text" value="w = 1"/>	Vector becomes a <b>position</b> → gets full matrix transform (T * R * S)
Set <input type="text" value="w = 0"/>	Vector becomes a <b>direction</b> → skips translation
Divide by <input type="text" value="w"/>	Converts <b>clip space</b> → <b>NDC</b> (perspective divide)
Store something in <input type="text" value="w"/>	Use <input type="text" value="w"/> as a <b>packed value</b> (for instancing, motion, etc.)

YES — asking *what is a basis* means you’re ready to **understand all of 3D spacebuilding and matrix logic**. Let’s break it down clearly, intuitively, and practically for tech art ☐☐

## What Is a **Basis**?

“ A **basis** is a set of vectors that define a coordinate space.  
It tells you:

- Where the **origin** is
- What direction **X, Y, Z** go
- What "**1 unit**" means in each direction

It's like building a mini coordinate system — your own local grid.

---

## □ Example: Standard Basis in 3D

The default basis in world space is:

$X = (1, 0, 0)$

$Y = (0, 1, 0)$

$Z = (0, 0, 1)$

This is called an **orthonormal basis**:

- Each axis is **unit length**
- Each axis is **perpendicular** to the others

This is what Unity uses for world space, object space (by default), and more.

---

## □□ Why Is a Basis Useful?

“ A basis lets you **build your own space** inside another space.

You can:

- Rotate it (to align to a surface)
  - Scale it (to stretch/compress space)
  - Offset it (with a position = origin)
  - Pack it into a matrix → for transforming vectors
- 

## □□ Basis = The Building Blocks of a Matrix

In a transformation matrix, the 3 main columns **are your basis vectors**:

[ Right Up Forward Position ]

[ X Y Z T ]

So:

- The **first 3 columns** = your basis vectors (local X, Y, Z)
- The **4th column** = your new origin (translation vector)

□ A matrix is just **abasis with a location**.

## □ Real Tech Art Examples

Use Case	Basis Involved?	What It Does
Tangent space normal maps	TBN basis	Converts normals from UV space to world space
Custom pivot control	Custom basis	Reorients local X/Y/Z axes
Surface alignment shader	Basis from world normals	Builds a local space aligned to geometry
Procedural animation	Create and apply basis	Bones, aim constraints, etc.
Billboarding	Rebuild basis using camera	Rotate object to face view direction

“□ **Abasis** is a set of vectors that define a space's axes.

A **matrix** = **basis** + **position**.

You use it to **move**, **rotate**, or **remap** things between spaces.

- **Identity Matrix**

- Does nothing when applied to a vector.
- Basis for understanding other transformations.
- Represents standard world coordinates.

- **Uniform Scaled Matrix**

- Modify all diagonal values equally.

- Scales objects equally in all directions.
- E.g., `0.5` on all diagonal entries = mesh is scaled down by half.
- **Axis Scaled Matrix**
  - Change individual diagonal values independently.
  - Scales the object along specific axes.
  - E.g., a 3 in the X-axis = stretches object 3x in X only.
- **Mirror (Reflection) Matrix**
  - Multiply a diagonal entry by `-1` to mirror on that axis.
  - E.g., `-1` in X = flips mesh on the X-axis.
- **Matrix with Normalized Basis**
  - Basis vectors (matrix columns) are unit length and orthogonal.
  - Often used for pure rotation without scaling.
  - Each axis remains perpendicular and consistent in size.
- **Matrix with Non-Orthogonal Basis**
  - Columns are not perpendicular.
  - Causes skewing—object axes are no longer at right angles.
- **Matrix with Non-Normalized Basis**
  - Basis vectors may differ in length.
  - Can scale along arbitrary axes (not just X, Y, Z).
- **Rotation Matrices**
  - Represented by changing off-diagonal values (not just diagonal).
  - Rotate vectors in space while maintaining their magnitude.
- **Matrix Interpretation Tip**
  - Think of each matrix column as defining a new space axis.
  - Multiplying a vector by a matrix expresses it in that new space.

Let me know if you'd like a diagram or Unity version of any of these!

---

Revision #5

Created 23 March 2025 21:54:12 by victor

Updated 31 March 2025 00:40:59 by victor