3D Points

- OpenGL and Unity
  - 4 element column vector
  \[
  \begin{bmatrix}
  x \\
  y \\
  z \\
  1
  \end{bmatrix}
  \]

- Direct3D and XNA
  - 4 element row vector
  \[
  [x \ y \ z \ 1]
  \]

- Note:
  - Points in Unity are typically expressed as 3 element vectors
  - Transformations in Unity are typically expressed as specific fields in the UI and functions in code
  - Rotations are represented internally in Unity as *quaternions*
## 3D Transformations
(for column vectors)

\[
T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]

\[
R_s(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]

\[
S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]

\[
R_r(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]

## 3D Rigid Body Transformations

- **Upper left 3×3 submatrix rows**
  - are unit vectors (length = 1)
  - are mutually perpendicular (dot product = 0)
  - rotate into the x, y, and z axes

- **Upper left 3×3 submatrix columns**
  - are unit vectors (length = 1)
  - are mutually perpendicular (dot product = 0)
  - are vectors into which the x, y, and z axes rotate