|  |  |
| --- | --- |
| **Column-major:** | **Row-major :** |
| Array Indices:$\left[\begin{matrix}0&4&8&12\\1&5&9&13\\2&6&10&14\\3&7&11&15\end{matrix}\right]$ Subscript Indices:$$\left[\begin{matrix}m\_{00}&m\_{10}&m\_{20}&m\_{30}\\m\_{01}&m\_{11}&m\_{21}&m\_{31}\\m\_{02}&m\_{12}&m\_{22}&m\_{32}\\m\_{03}&m\_{13}&m\_{23}&m\_{33}\end{matrix}\right]$$Basis (Axis Direction) Vectors:$$\left[\begin{matrix}xx&yx&zx&tx\\xy&yy&zy&ty\\xz&yz&zz&tz\\0&0&0&1\end{matrix}\right]$$Multiplication order: **CBAx**$$\left[\begin{matrix}xx&yx&zx&tx\\xy&yy&zy&ty\\xz&yz&zz&tz\\0&0&0&1\end{matrix}\right]∙\left[\begin{matrix}x\\y\\z\\w\end{matrix}\right]$$ | Array Indices:$\left[\begin{matrix}0&1&2&3\\4&5&6&7\\8&9&10&11\\12&13&14&15\end{matrix}\right]$ Subscript Indices:$$\left[\begin{matrix}m\_{00}&m\_{01}&m\_{02}&m\_{03}\\m\_{10}&m\_{11}&m\_{12}&m\_{13}\\m\_{20}&m\_{21}&m\_{22}&m\_{23}\\m\_{30}&m\_{31}&m\_{32}&m\_{33}\end{matrix}\right]$$Basis (Axis Direction) Vectors:$$\left[\begin{matrix}xx&xy&xz&0\\yx&yy&yz&0\\zz&zy&zz&0\\0&0&0&1\end{matrix}\right]$$Multiplication order: **xABC**$$\left[\begin{matrix}x&y&z&w\end{matrix}\right]∙\left[\begin{matrix}xx&xy&xz&0\\yx&yy&yz&0\\zz&zy&zz&0\\tx&ty&tz&1\end{matrix}\right]$$ |

Notes:

* Array indices correspond to the same elements in both row and column-major matrices.
* This document uses column-major stuff from now on (transpose and reverse the order of multiplication to get row-major).

**Rotation:**

|  |  |  |
| --- | --- | --- |
| x-axis:$$\left[\begin{matrix}1&0&0\\0&cosθ&-sinθ\\0&sinθ&cosθ\end{matrix}\right]$$ | y-axis:$$\left[\begin{matrix}cosθ&0&sinθ\\0&1&0\\-sinθ&0&cosθ\end{matrix}\right]$$ | z-axis$$\left[\begin{matrix}cosθ&-sinθ&0\\sinθ&cosθ&0\\0&0&1\end{matrix}\right]$$ |
| arbitrary axis:$$c=1-cosθ$$$$\left[\begin{matrix}a\_{x}^{2}c+ cosθ&a\_{x}a\_{y}c-a\_{z}sinθ&a\_{x}a\_{z}c+a\_{y}sinθ\\a\_{x}a\_{y}c+a\_{z}sinθ&n\_{y}^{2}c+cosθ&a\_{y}a\_{z}c-a\_{x}sinθ\\a\_{x}a\_{z}c-a\_{y}sinθ&a\_{y}a\_{z}c+a\_{x}sinθ&n\_{z}^{2}c+ cosθ\end{matrix}\right]$$ |

**Scale:**

$$\left[\begin{matrix}x&0&0\\0&y&0\\0&0&z\end{matrix}\right]$$

**Matrix Multiplication:**

Take dot product of rows of the first matrix, and column of second matrix (same for both row and column-major matrices).

Column-major:

$$\left[\begin{matrix}a\_{00}&a\_{10}&a\_{20}&a\_{30}\\a\_{01}&a\_{11}&a\_{21}&a\_{31}\\a\_{02}&a\_{12}&a\_{22}&a\_{32}\\a\_{03}&a\_{13}&a\_{23}&a\_{33}\end{matrix}\right]∙\left[\begin{matrix}b\_{00}&b\_{10}&b\_{20}&b\_{30}\\b\_{01}&b\_{11}&b\_{21}&b\_{31}\\b\_{02}&b\_{12}&b\_{22}&b\_{32}\\b\_{03}&b\_{13}&b\_{23}&b\_{33}\end{matrix}\right]=\left[\begin{matrix}c\_{00}&c\_{10}&c\_{20}&c\_{30}\\c\_{01}&c\_{11}&c\_{21}&c\_{31}\\c\_{02}&c\_{12}&c\_{22}&c\_{32}\\c\_{03}&c\_{13}&c\_{23}&c\_{33}\end{matrix}\right]$$

$$c\_{11}=a\_{01}∙b\_{10}+a\_{11}∙b\_{11}+a\_{21}∙b\_{12}+a\_{31}∙b\_{13}$$

$$c\_{5}=a\_{1}∙b\_{4}+a\_{5}∙b\_{5}+a\_{9}∙b\_{6}+a\_{13}∙b\_{7}$$

Row-major:

$$\left[\begin{matrix}b\_{00}&b\_{01}&b\_{02}&b\_{03}\\b\_{10}&b\_{11}&b\_{12}&b\_{13}\\b\_{20}&b\_{21}&b\_{22}&b\_{23}\\b\_{30}&b\_{31}&b\_{32}&b\_{33}\end{matrix}\right]∙\left[\begin{matrix}a\_{00}&a\_{01}&a\_{02}&a\_{03}\\a\_{10}&a\_{11}&a\_{12}&a\_{13}\\a\_{20}&a\_{21}&a\_{22}&a\_{23}\\a\_{30}&a\_{31}&a\_{32}&a\_{33}\end{matrix}\right]=\left[\begin{matrix}c\_{00}&c\_{01}&c\_{02}&c\_{03}\\c\_{10}&c\_{11}&c\_{12}&c\_{13}\\c\_{20}&c\_{21}&c\_{22}&c\_{23}\\c\_{30}&c\_{31}&c\_{32}&c\_{33}\end{matrix}\right]$$

$$c\_{11}=b\_{10}∙a\_{01}+b\_{11}∙a\_{11}+b\_{12}∙a\_{21}+b\_{13}∙a\_{31}$$

$$c\_{5}=b\_{4}∙a\_{1}+b\_{5}∙a\_{5}+b\_{6}∙a\_{9}+b\_{7}∙a\_{13}$$

**Projection:**

|  |  |
| --- | --- |
| **Orthographic:**Offsets from camera (in world space):* l, r, t, b – left, right, top, bottom of screen.
* n, f – near and far planes.

$$\left[\begin{matrix}\frac{2}{r-l}&0&0&-\frac{r+l}{r-l}\\0&\frac{2}{t-b}&0&-\frac{t+b}{t-b}\\0&0&\frac{-2}{f-n}&-\frac{f+n}{f-n}\\0&0&0&1\end{matrix}\right]$$ | **Perspective:**$$aspect=\frac{viewport\_{width}}{viewport\_{height}}$$$$f= \frac{1}{\tan(\left(\frac{fov}{2}\right))}$$$$\left[\begin{matrix}\frac{f}{aspect}&0&0&0\\0&f&0&0\\0&0&\frac{-n-f}{n-f}&\frac{2nf}{n-f}\\0&0&-1&0\end{matrix}\right]$$ |

**Transforms:**

* Model matrix – model local to world space.
* Camera matrix – camera local (view space) to world space.
* View matrix – world to view space (inverse camera matrix).
* Projection matrix – view space to clip space.

**Spaces:**

* Clip space – geometry outside of the frustum is clipped here. Then vertices are normalized (divided by w – the “perspective divide”) to get normalized device coordinates (NDC).
* NDC – coordinates between -1.0 and 1.0 (D3D uses 0.0 to 1.0 for z instead). The viewport transformation is then applied to get screen space.
* Screen Space – pixel positions (i.e. gl\_FragCoord).

**Inverse:**

// from glm

mat4 matrix\_inverse(mat4 m)

{

 float c00 = m[2][2] \* m[3][3] - m[3][2] \* m[2][3];

 float c02 = m[1][2] \* m[3][3] - m[3][2] \* m[1][3];

 float c03 = m[1][2] \* m[2][3] - m[2][2] \* m[1][3];

 float c04 = m[2][1] \* m[3][3] - m[3][1] \* m[2][3];

 float c06 = m[1][1] \* m[3][3] - m[3][1] \* m[1][3];

 float c07 = m[1][1] \* m[2][3] - m[2][1] \* m[1][3];

 float c08 = m[2][1] \* m[3][2] - m[3][1] \* m[2][2];

 float c10 = m[1][1] \* m[3][2] - m[3][1] \* m[1][2];

 float c11 = m[1][1] \* m[2][2] - m[2][1] \* m[1][2];

 float c12 = m[2][0] \* m[3][3] - m[3][0] \* m[2][3];

 float c14 = m[1][0] \* m[3][3] - m[3][0] \* m[1][3];

 float c15 = m[1][0] \* m[2][3] - m[2][0] \* m[1][3];

 float c16 = m[2][0] \* m[3][2] - m[3][0] \* m[2][2];

 float c18 = m[1][0] \* m[3][2] - m[3][0] \* m[1][2];

 float c19 = m[1][0] \* m[2][2] - m[2][0] \* m[1][2];

 float c20 = m[2][0] \* m[3][1] - m[3][0] \* m[2][1];

 float c22 = m[1][0] \* m[3][1] - m[3][0] \* m[1][1];

 float c23 = m[1][0] \* m[2][1] - m[2][0] \* m[1][1];

 vec4 f0 = vec4(c00, c00, c02, c03);

 vec4 f1 = vec4(c04, c04, c06, c07);

 vec4 f2 = vec4(c08, c08, c10, c11);

 vec4 f3 = vec4(c12, c12, c14, c15);

 vec4 f4 = vec4(c16, c16, c18, c19);

 vec4 f5 = vec4(c20, c20, c22, c23);

 vec4 v0 = vec4(m[1][0], m[0][0], m[0][0], m[0][0]);

 vec4 v1 = vec4(m[1][1], m[0][1], m[0][1], m[0][1]);

 vec4 v2 = vec4(m[1][2], m[0][2], m[0][2], m[0][2]);

 vec4 v3 = vec4(m[1][3], m[0][3], m[0][3], m[0][3]);

 vec4 inv0 = vec4(v1 \* f0 - v2 \* f1 + v3 \* f2);

 vec4 inv1 = vec4(v0 \* f0 - v2 \* f3 + v3 \* f4);

 vec4 inv2 = vec4(v0 \* f1 - v1 \* f3 + v3 \* f5);

 vec4 inv3 = vec4(v0 \* f2 - v1 \* f4 + v2 \* f5);

 vec4 signA = vec4(+1, -1, +1, -1);

 vec4 signB = vec4(-1, +1, -1, +1);

 mat4 inverse = mat4(inv0 \* signA, inv1 \* signB, inv2 \* signA, inv3 \* signB);

 vec4 row0 = vec4(inverse[0][0], inverse[1][0], inverse[2][0], inverse[3][0]);

 vec4 dot0 = vec4(m[0] \* row0);

 float d = (dot0.x + dot0.y) + (dot0.z + dot0.w);

 return inverse \* (1.0 / d);

}