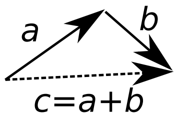
3d Maths Cheat Sheet

Vectors

Vector Addition

The sum of 2 vectors completes the triangle.



also $a = c - b$ and $b = c - a$

Unit Vectors - “Normalised” Vectors

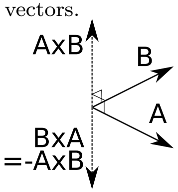
Used to represent a direction or **normal**. Length of 1.
$\hat{A} = \frac{\vec{A}}{||\vec{A}||}$
Where $||\vec{A}||$ is the length or magnitude of $\vec{A}$.

Dot Product of 2 Vectors

Can be used to get the **angle between 2 vectors**.
$\vec{A} \cdot \vec{B} = \sum_{i=1}^{n} A_i B_i = A_1 B_1 + A_2 B_2 + \cdots + A_n B_n$
The dot product returns a single **scalar** value. $\theta = arccos(\hat{A} \cdot \hat{B})$
$\theta = arccos(\frac{\vec{A} \cdot \vec{B}}{||\vec{A}||||\vec{B}||})$
Where $arccos$ is inverse cosine cos^{-1}.

Cross Product of 2 Vectors

Produces a vector perpendicular to the plane containing the 2 vectors.



$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

To compute **surface normals** from 2 edges:

```
N = normalize (cross (A, B));
```

Matrices

Identity Matrix

All 0, except the top-left to bottom-right diagonal.

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

if $AB = I$ then A is the inverse of B and *vice versa*.

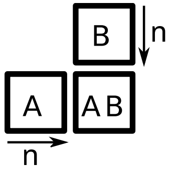
Matrix * Vector

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

Matrix * Matrix

Each cell (row, col) in AB is:
$\sum_{i=1}^{n} A(row, 1) * B(1, col) + \cdots + A(row, n) * B(n, col)$
Where n is dimensionality of matrix.

$$AB = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$



(rows of A with columns of B)

Matrix Determinant

For a 2x2 or 3x3 matrix use the Rule of Sarrus; add products of top-left to bottom-right diagonals, subtract products of opposite diagonals.

$M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ Its determinant $|M|$ is:

$|M| = aei + bfg + cdh - ceg - bdi - afh$
For 4x4 use Laplace Expansion; each top-row value * the 3x3 matrix made of all other rows and columns:
$|M| = aM_1 - bM_2 + cM_3 - dM_4$
See `http://www.euclideanspace.com/maths/algebra/matrix/functions/determinant/fourD/index.htm`

Matrix Transpose

Flip matrix over its main diagonal. In special case of orthonormal xyz matrix then inverse is the transpose. Can use to **switch between row-major and column-major matrices**.

$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad M^T = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

Matrix Inverse

Use an inverse matrix to **reverse its transformation**, or to transform **relative to another object**.
$MM^{-1} = I$ Where I is the identity matrix.
If the determinant of a matrix is 0, then there is no inverse. The inverse can be found by multiplying the determinant with a large matrix of cofactors. For the long formula see
`http://www.cg.info.hiroshima-cu.ac.jp/~miyazaki/knowledge/teche23.html`
Use the **transpose of an inverse model matrix** to transform normals: $n' = n(M^{-1})^T$

Homogeneous Matrices

Multiply affine transformations into a 4d matrix. Order is important.

Row-Order Homogeneous Matrix

Commonly used in **Direct3D maths libraries**

$$v' = \begin{bmatrix} V_x & V_y & V_z & 1 \end{bmatrix} \begin{bmatrix} X_x & X_y & X_z & 0 \\ Y_x & Y_y & Y_z & 0 \\ Z_x & Z_y & Z_z & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

$v' = vRT$ (change facing and position)
$v' = vTR$ (rotate around a point)

Column-Order Homogeneous Matrix

Commonly used in **OpenGL maths libraries**

$$v' = \begin{bmatrix} X_x & Y_x & Z_x & T_x \\ X_y & Y_y & Z_y & T_y \\ X_z & Y_z & Z_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \\ 1 \end{bmatrix}$$

$v' = TRv$ (change facing and position)
$v' = RTv$ (rotate around a point)

Translation, Scaling, and Rotation

$$\text{column order } T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

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

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

View Matrix

$$V = \begin{bmatrix} R_x & R_y & R_z & -P_x \\ U_x & U_y & U_z & -P_y \\ -F_x & -F_y & -F_z & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ (column-order)}$$

Where U is a vector pointing up, F forward, and P is world position of camera. Remember to multiply to combine transformations.

$$\text{Bird's-eye view } V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ (multipy } T^{-1} \text{ to move)}$$

$V = T^{-1} R^{-1}$ (column order: orbit camera around a point)
$V = R^{-1} T^{-1}$ (column order: orient camera and change position)

Projection Matrix

$$P = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & P_z \\ 0 & 0 & -1 & 0 \end{bmatrix} \text{ (column-order)}$$

$S_x = (2 * near)/(range * aspect + range * aspect)$
$S_y = near/range$
$S_z = -(far + near)/(far - near)$
$P_z = -(2 * far * near)/(far - near)$
$range = tan(fov/2) * near$

---

revision 6. 10 Jun 2014
Dr Anton Gerdelan, gerdela@scss.tcd.ie, Trinity College Dublin, Ireland.
LaTeX template from http://www.stdout.org/~winston/latex/
Thanks Michaël, Ken, Amoss, Veronica, and Stefan!