Cheat Sheet for Computer Graphics

Winter 2019/Summer 2020

This is a rudimentary cheat sheet for the lecture *Computer Graphics*, Winter Term 2019. It is by no means complete, rather it is optimized to fit with the challenge set by the exam. That is, it contains the formulas and matrices you are expected to recall during the exam.

1 Basic Math

1.1 Scalar Product

$$a \circ b = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \circ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_1 b_1 + a_2 b_2 + a_3 b_3 \tag{1}$$

1.2 Cross Product

$$a \times b = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_2 b_2 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix} \tag{2}$$

2 Bresehenham Algorithm

Given two points $A = (x_0, y_0)$ and $B = (x_1, y_1)$, we want to rasterize a line from A to B. For this, we first compute updates

$$\Delta DE = 2\Delta y \qquad \text{and} \qquad \Delta DNE = 2(\Delta x - \Delta y) \tag{3}$$

and initialize

$$D := \Delta x - 2\Delta y. \tag{4}$$

Then keep incrementing $x := x_0$ up to x_1. In each iteration, as long as $x \geq 0$, we move to the east (right) and update with ΔDE. If $x < 0$, we move north-east (increment y) and update with ΔDNE.

3 Homogeneous Coordinates

- Given a point (x, y, z) in 3D space, the homogeneous coordinate that represents that point is $(x, y, z, w = 1)$.
- To undo this conversion, compute

$$\frac{1}{w}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5}$$

4 Affine Transformations

4.1 Shearing

Horizontal sheering keeps y fixed. On paper, it looks like the y-axis is moving. It maps the axis vector $(0, 1)$ to $(s_h, 1)$.

$$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Vertical sheering keeps x fixed. On paper, it looks like the x-axis is moving. It maps the axis vector $(1, 0)$ to $(1, s_v)$.

$$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

4.2 Rotation

Rotation by angle ϕ from origin $(0, 0)$, anti-clockwise is implemented by

$$R_\phi = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

In particular we get

$$R_{\frac{\pi}{2}} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ R_\pi = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ R_{\frac{3\pi}{2}} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{9}$$

4.3 Reflection

To reflect on the x-axis, invert y with

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{10}$$

To reflect on the y-axis, invert x with

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

5 Multiple Transformations

5.1 Matrix Application

Given are transformations M_1 and M_2. To first compute M_1 and then M_2, multiply a given point with the matrix product

$$M_2\, M_1, \tag{12}$$

that is transformations are applied right to left.

5.2 Rotation and Transformation

Matrix M first translates by (t_x, t_y) and then rotates by angle ϕ.

$$M = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & t_x\cos\phi - t_y\sin\phi \\ \sin\phi & \cos\phi & t_x\cos\phi + t_y\sin\phi \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

Matrix N first rotates by angle ϕ and then translates by (t_x, t_y).

$$N = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & t_x \\ \sin\phi & \cos\phi & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

6 Rotations in 3D Space

Anti-Clockwise rotation around an axis m by angle ϕ is achievable with matrix R_m as follows.

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$R_y = \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

$$R_z = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

7 Camera

A camera is often defined by eye vector e, gaze direction g and view-up vector t. However, in practice it is useful to construct the (u, v, w) coordinate system for a given camera.

$$w = -\frac{1}{||g||}\, g \qquad u = \frac{1}{||t \times w||}\,(t \times w) \qquad v = w \times u \tag{18}$$

8 Viewing Transformation

From camera coordinate system (u, v, w), we can construct viewing matrix

$$M_{view} = \begin{bmatrix} u_x & u_y & u_z & -u^T e \\ v_x & v_y & v_z & -v^T e \\ w_x & w_y & w_z & -w^T e \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

which moves the camera into the origin.

9 Phong Lighting Model

The Phong Lighting Model is based around three components, namely ambient light L_{amb}, diffuse reflection L_{diff} and specular reflection L_{spec}.

9.1 Ambient Light

Ambient light is constant, parameterized by two arguments.

$$L_{amb} = k_{amb} \cdot I_{in} \tag{20}$$

9.2 Diffuse Reflection

Diffuse reflection imitates micro structure and random reflection. It is dependent on the angle ϕ between surface normal n and light vector l. Both n and l have to be normalized.

$$\begin{aligned} L_{diff} &= k_{diff} \cdot I_{in} \cdot \cos\phi & (21)\\ &= k_{diff} \cdot I_{in} \cdot (n \circ l) & (22) \end{aligned}$$

9.3 Specular Reflection

Specular reflection creates spotlight effects. It takes the eye position into account. To compute it, we need the perfect reflection vector r and view vector v, both of which need to be normalized. Angle φ is the angle between vectors v and r. Material constant m introduces a cutoff for the highlight, higher values for m result in smaller highlights.

$$\begin{aligned} L_{spec} &= k_{spec} \cdot I_{in} \cdot \cos^m \varphi & (23)\\ &= k_{spec} \cdot I_{in} \cdot (v \circ r)^m & (24) \end{aligned}$$

If perfect reflection r is not known, it can be computed in terms of

$$r = 2 \cdot (n \circ l) \cdot n - l. \tag{25}$$

10 Ray Tracing

10.1 Eye Ray

An eye ray r is a ray starting at eye position e in direction d.

$$r(t) = e + t \cdot d \tag{26}$$

10.2 Intersection Point

An intersection point x of an eye ray with some geometry is defined by distance t_x the ray had to travel from eye to geometry.

$$x = e + t_x \cdot d \tag{27}$$

10.3 Intersection With a Plane

Given a plane defined by a normal n and some point c on the plane, we can compute intersection point x by finding the right value for t_x.

$$t_x = \frac{(c - e) \circ n}{d \circ n} \tag{28}$$

We only care about real solutions for t_x. We also only care for solutions $t_x > 0$ as negative values mean that we traveled in opposite of direction d.

10.4 Intersection With a Sphere

Given a sphere with center c and radius r, find out t_x for which an eye ray $e + t_x \cdot d$ intersects the sphere. The solution can be found by solving the quadratic equation in terms of

$$t_x = \frac{-b \pm \sqrt{b^2 - 4c}}{2} \tag{29}$$

with

$$b = 2 \cdot d \circ (e - c) \qquad \text{and} \qquad c = (e - c) \circ (e - c) - r^2. \tag{30}$$

If there is no real solution for t_x, there is no intersection of the ray with the sphere.

11 Fin

Keine Macht für Niemand.