Spis treści

1 Literatura

- Wykład
- J. de Vries, "Learn OpenGL"
- Ćwiczenia

2 Wektory

$$\vec{v} \cdot \vec{w} = v_x \cdot w_x + v_y \cdot w_y + v_z \cdot w_z$$

$$\vec{v} \cdot \vec{w} = |\vec{v}| \cdot |\vec{w}| \cdot cos(\theta)$$

Jeśli $\vec{v}$ i $\vec{w}$ są prostopadłe to $\vec{v} \cdot \vec{w} = 0$. A z kolei jeśli $\vec{v} \cdot \vec{w}$ są normalne to $\vec{v} \cdot \vec{w} = cos(\theta)$.

3 Operacje

Wyróżniamy trzy rodzaje operacji na punktach w przestrzeni n-wymiarowej:

- Skalowanie

$$f(\begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} s_x * x \\ s_y * y \end{bmatrix}$$

- Rotacja

$$f(\begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} x * cos(\theta) - y * sin(\theta) \\ x * sin(\theta) + y * cos(\theta) \end{bmatrix}$$

- Translacja

$$f(\begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} x + 1 \\ y - 1 \end{bmatrix}$$

4 Operacje 2D

4.1 Skalowanie

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \text{ lub } \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.2 Rotacja

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \text{ lub } \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.3 Translacja

$$T = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ lub dla } \vec{V} = (T_x, T_y), T = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

5 Konkatenacja operacji

Operacje można łączyć w łańcuchy, np. $T * R * S$ Jako że to jest mnożenie macierzy, to kolejność ma znaczenie. Najpierw wykonuje się operację z prawej strony.

6 Operacje 3D

6.1 Skalowanie

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6.2 Rotacja

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

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

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

6.3 Translacja

$$\text{Dla wektora } \vec{V} = (T_x, T_y, T_z), T = \begin{bmatrix} 1 & 0 & 0, & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7 Kamery

Podczas patrzenia na obiekt w polu widzenia kamery można wyróżnić następujące płaszczyzny:

- Górna
- Dolna
- Lewa
- Prawa
- Bliska
- Daleka

7.1 Projekcje

Kluczowym aspektem kamery jest projekcja, czyli rzutowanie obiektów na płaszczyznę ekranu. Trzeba w jakiś sposób zamienić punkty w 3D na punkty w 2D.

7.1.1 Ortogonalna

- Wszystkie linie są równoległe
- Brak perspektywy
- Obiekty są przeskalowane
- Nie ma odczucia głębi, obiekty pozostają tego samego rozmiaru niezależnie od odległości
- Linie równoległe pozostają równoległe po projekcji

7.1.2 Perspektywiczna

- Linie zbiegają się w jednym punkcie
- Obiekty są przeskalowane
- Odczucie głębi
- Obiekty są mniejsze w miarę oddalania się od kamery
- "Normalna"projekcja

8 Pipeline

Renderowanie obrazu składa się z kilku etapów:

$$\text{Object Space} \to \text{World space} \overset{M_{cam}}{\to} \text{View space} \overset{M_{pro}}{\to} \text{Clip space} \overset{M_{vp}}{\to} \text{Screen space}$$

8.1 VP

Dla widoku wielkości $n_x * n_y$

$$M_{vp} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8.2 Pro

Dla przestrzeni widoku ortogonalnego stworzonego przez trzy punkty:
- (l, b, n) w lewym dolnym rogu - (r, b, n) w prawym dolnym rogu - (r, t, f) w prawym górnym rogu

$$M_{ort} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{l+r}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{b+t}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{per} = M_{ort} \cdot \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8.3 Cam

Dla wektorów określających kierunek patrzenia kamery: $\vec{g}$, $\vec{v}$, $\vec{u}$, $\vec{w} = -v$ oraz $\vec{e}$ określającym przesunięcie kamery

$$M_{cam} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8.4 Model

Matryca modelu to po prostu macierz transformacji obiektu. Jak obiekt jest obrócony, przesunięty, skalowany. To zależy od tego gdzie jest obiekt w przestrzeni, jako że zwykle punkty modelu są w przestrzeni obiektu (relatywnie do środka obiektu).

8.5 Złożenie

$$pipeline(\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}) = M_{vp} M_{ort} M_{cam} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

9 Algorytmy Renderowania

Jako, że w procesie przekształcenia pipeline zatracana jest informacja o koordynacie z punktów, konieczne jest renderowanie obiektów w konkretnej kolejności.

9.1 Algorytm malarza

Ten algorytm jest niezwykle prosty. Obiekty się sortuje wobec pozycji w z i najpierw renderuje się te w tle. Niestety, nie działa to dla obiektów przecinających się i innych o skomplikowanej topologii.

9.2 Z-buffer

Dla każdego obiektu zapisujesz na jakim z "jest"pixel i jakiego koloru jest. Jeśli w globalnym z bufferze ten pixel ma większe z to w bufferze pixel jest zastępowany. Efektem działania algorytmu jest buffer z pixelami "renderowany od tyłu".

10 Culling i Clipping

W procesie renderowania koniecznym jest czasami obsługiwanie co jeśli obiekt wychodzi poza kąt widzenia oraz co jeśli obiekt jest niewidoczny.

10.1 Clipping

Czyli obcinanie obiektów składa się z dwóch etapów:

- Wykrywanie czy obiekt przecina jedną z płaszczyzn
- Tworzenie nowych mniejszych obiektów

10.1.1 Wykrywanie

Jeśli dla n=normalny wektor płaszczyzny oraz q=punkt na płaszczyźnie $f(p) = n * (p - q) < 0$ to punkt p jest wewnątrz.

10.1.2 Tworzenie

Dla każdej krawędzi obiektu sprawdzamy czy przecina ona płaszczyznę. Dzieląc trójkąty na mniejsze wewnątrz pola widzenia, tworzymy nowe trójkąty na podstawie punktów przecięcia z płaszczyzną.

10.2 Culling

Odrzucanie trójkątów których wektor normalny jest skierowany od punktu widzenia to backface culling. Any zobaczyć czy wektor normalny $\vec{n}$ jest skierowany od punktu widzenia to wystarczy złożyć go z wektorem punktu widzenia $\vec{v}$ Jeśli $\vec{n} \cdot \vec{v} < 0$ to trójkąt jest skierowany do kamery.

11 Oświetlenie

$$I = f_{att}(D + S) + A$$

11.1 Diffuse

$$D = I_p \cdot K_d \cdot max(0, cos(\theta))$$

gdzie:

- I_p to intensywność światła
- K_d to współczynnik odbicia
- θ to kąt między wektorem światła a wektorem normalnym do powierzchni

11.2 Specular

$$S = I_p \cdot K_s \cdot max(0, cos(\alpha))^n$$

gdzie:

- I_p to intensywność światła
- K_s to współczynnik odbicia
- α to kąt między wektorem do kamery a wektorem odbicia
- n to współczynnik tłumienia

11.3 Ambient

$$A = I_a \cdot K_a$$

gdzie:

- I_a to intensywność światła
- K_a to współczynnik odbicia

11.4 Attenuation

$$f_{att} = 1 - \frac{d}{r}^2$$

gdzie:

- d to odległość od źródła światła
- r to promień światła

11.5 Shading

- Flat shading - kolor obiektu jest jednolity
- Gouraud shading - kolor obiektu jest interpolowany między wierzchołkami
- Phong shading - kolor obiektu jest interpolowany między pikselami