

Contents

1 Literatura 1

2 Operacje 1

3 Operacje 2D 2

 3.1 Skalowanie 2

 3.2 Rotacja 2

 3.3 Translacja 2

4 Konkatenacja operacji 2

5 Operacje 3D 2

 5.1 Skalowanie 2

 5.2 Rotacja 2

 5.3 Translacja 3

6 Kamery 3

 6.1 Projekcje 3

 6.1.1 Ortogonalna 3

 6.1.2 Perspektywiczna 3

7 Pipeline 3

 7.1 VP 4

 7.2 Pro 4

 7.3 Cam 4

 7.4 Model 4

 7.5 Złożenie 4

8 Algorytmy Renderowania 4

 8.1 Algorytm malarza 4

 8.2 Z-buffer 5

9 Culling i Clipping 5

 9.1 Clipping 5

 9.1.1 Wykrywanie 5

 9.1.2 Tworzenie 5

 9.2 Culling 5

1 Literatura

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

2 Operacje

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

- Skalowanie

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

- Rotacja

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

- Translacja

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

3 Operacje 2D

3.1 Skalowanie

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

3.2 Rotacja

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

3.3 Translacja

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

4 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.

5 Operacje 3D

5.1 Skalowanie

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

5.2 Rotacja

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

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

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

5.3 Translacja

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

6 Kamery

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

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

6.1 Projekcje

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

6.1.1 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

6.1.2 Perspektywiczna

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

7 Pipeline

Renderowanie obrazu składa się z kilku etapów:

$$\text{Object Space} \rightarrow \text{World space} \xrightarrow{M_{cam}} \text{View space} \xrightarrow{M_{proj}} \text{Clip space} \xrightarrow{M_{vp}} \text{Screen space}$$

7.1 VP

Dla widoku wielkości $n_x * n_y$

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

7.2 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

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

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

7.3 Cam

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

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

7.4 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).

7.5 Złożenie

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

8 Algorytmy Renderowania

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

8.1 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.

8.2 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".

9 Culling i Clipping

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

9.1 Clipping

Czyli obcinanie obiektów składa się z dwóch etapów:

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

9.1.1 Wykrywanie

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

9.1.2 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.

9.2 Culling

Odrzucanie trójkątów których wektor normalny jest skierowany od punktu widzenia to backface culling. Aby 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.