Sprawozdanie – System Wirtualnej Kamery

Adrian Rybaczuk 318483

8 kwietnia 2025

Spis treści

1 Wstęp

Projekt polega na implementacji systemu wirtualnej kamery. Celem jest przedstawienie sceny przy pomocy wirtualnej kamery. Użyłem do tego podejścia z wykorzystaniem 2 układów, układu kamery oraz układu sceny.

2 Transformacja obiektu 3D

W tej sekcji przedstawiamy szczegółowy opis procesu przekształcania obiektu 3D z jego pierwotnego umiejscowienia w scenie do ostatecznego obrazu na ekranie. Wszystkie operacje wykonujemy w przestrzeni jednorodnej (homogenicznej), co umożliwia łączenie transformacji translacji, rotacji oraz skalowania w jedną macierz.

2.1 Krok 1: Umiejscowienie Obiektu na Scenie (Modeling)

Obiekt w scenie jest pierwotnie opisany w swoim lokalnym układzie współrzędnych. Aby umiejscowić go w przestrzeni świata, stosujemy modelową macierz transformacji M,

$$M = T \cdot R \cdot S,$$

gdzie:

- T – macierz translacji,
- R – macierz rotacji,
- S – macierz skalowania.

Wszystkie współrzędne obiektu zapisujemy jako wektory jednorodne:

$$P_{model} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

W wyniku przekształcenia mamy:

$$P_{world} = M \cdot P_{model}.$$

2.2 Krok 2: Przelozenie Obiektu z Widoku Sceny na Widok Kamery (View Transformation)

Następnie, aby przejść z układu świata (sceny) do układu kamery, stosujemy macierz widoku V. Jest ona konstruowana na bazie pozycji kamery E (ang. eye), punktu, na który kamera patrzy, oraz wektorów definiujących orientację kamery. Metoda `lookAt` definiuje trzy ortonormalne wektory:

- **u** – wektor określający prawą stronę kamery,
- **v** – wektor skierowany do góry,

- $\mathbf{n}$ – wektor przeciwległy do kierunku patrzenia.

Macierz widoku ma postać:

$$V = \begin{bmatrix} u_x & u_y & u_z & -\mathbf{u} \cdot E \\ v_x & v_y & v_z & -\mathbf{v} \cdot E \\ n_x & n_y & n_z & -\mathbf{n} \cdot E \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Przekształcenie do układu kamery wykonujemy jako:

$$P_{camera} = V \cdot P_{world}.$$

2.3 Krok 3: Przelozenie Obiektu z Widoku Kamery na Widok 2D (Rzutowanie Perspektywiczne)

Aby odwzorować trójwymiarowy widok kamery na dwuwymiarowy obraz, stosujemy macierz rzutowania P korzystającą z zasad perspektywicznego rzutowania. Nowe współrzędne (x', y', z', w') są obliczane jako:

$$P' = P \cdot P_{camera}.$$

Następnie wykonujemy perspektywiczne dzielenie przez w':

$$(x_{ndc}, y_{ndc}, z_{ndc}) = \left(\frac{x'}{w'}, \frac{y'}{w'}, \frac{z'}{w'} \right),$$

gdzie $(x_{ndc}, y_{ndc}, z_{ndc})$ to współrzędne w przestrzeni znormalizowanej (ND). Macierz perspektywiczna P może mieć postać:

$$P = \begin{bmatrix} \frac{f}{a} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{N+F}{N-F} & \frac{2NF}{N-F} \\ 0 & 0 & -1 & 0 \end{bmatrix},$$

gdzie:

- $f = \frac{1}{\tan(\theta/2)}$ – zależny od kąta widzenia θ,
- a – współczynnik proporcji ekranu,
- N i F – odległości do płaszczyzny bliskiej (near) i dalszej (far).

2.4 Krok 4: Przelozenie Obiektu na Viewport (Viewport Transformation)

Ostatecznie, współrzędne z przestrzeni znormalizowanej, gdzie $x_{ndc}, y_{ndc} \in [-1, 1]$, są mapowane do rzeczywistych współrzędnych ekranu. Dla ekranu o szerokości W i wysokości H stosujemy następujące przekształcenie:

$$x_{screen} = \frac{W}{2}(x_{ndc} + 1), \quad y_{screen} = \frac{H}{2}(1 - y_{ndc}).$$

Zapewnia to, że punkt $(-1, -1)$ trafia do lewego dolnego rogu ekranu, a $(1, 1)$ do prawego górnego.

Każdy z tych kroków wykorzystuje macierze 4x4 oraz operacje w przestrzeni jednorodnej, co pozwala na efektywne łączenie transformacji oraz zapewnia spójność obliczeń podczas renderowania scen 3D. Metody te stanowią fundament nowoczesnych algorytmów renderowania w grafice komputerowej.

3 Sterowanie kamera w ukladzie 3D

Implementacja sterowania kamerą w układzie 3D opiera się na matematycznym modelu transformacji w przestrzeni jednorodnej. W systemie wirtualnej kamery zaimplementowano dwa główne typy ruchu: translację (przesunięcie) oraz rotację (obrót), które są realizowane za pomocą odpowiednich macierzy transformacji.

3.1 Translacja kamery

Translacja kamery jest realizowana w lokalnym układzie współrzędnych kamery, co zapewnia intuicyjne sterowanie niezależnie od aktualnej orientacji kamery. Wektor translacji $\mathbf{t} = [t_x, t_y, t_z]$ jest przekształcany do globalnego układu współrzędnych za pomocą macierzy rotacji kamery.

Dla kamery zorientowanej pod kątem θ_x (pitch) i θ_y (yaw), wektory kierunkowe w lokalnym układzie kamery są zdefiniowane jako:

$$\mathbf{forward} = \begin{bmatrix} -\sin(\theta_y)\cos(\theta_x) \\ \sin(\theta_x) \\ -\cos(\theta_y)\cos(\theta_x) \end{bmatrix}$$

$$\mathbf{right} = \begin{bmatrix} \cos(\theta_y) \\ 0 \\ -\sin(\theta_y) \end{bmatrix}$$

$$\mathbf{up} = \mathbf{right} \times \mathbf{forward}$$

Translacja w lokalnym układzie kamery jest realizowana jako kombinacja liniowa tych wektorów:

$$\mathbf{movement} = \mathbf{right} \cdot t_x + \mathbf{up} \cdot t_y + \mathbf{forward} \cdot t_z$$

Nowa pozycja kamery $\mathbf{E}_{new}$ jest obliczana jako:

$$\mathbf{E}_{new} = \mathbf{E}_{current} + \mathbf{movement}$$

gdzie $\mathbf{E}_{current}$ to aktualna pozycja kamery.

3.2 Rotacja kamery

Rotacja kamery jest realizowana za pomocą macierzy rotacji wokół lokalnych osi kamery. W systemie zaimplementowano trzy typy rotacji:

- **Pitch** - obrót wokół lokalnej osi X (góra-dół)

- **Yaw** - obrót wokół lokalnej osi Y (lewo-prawo)
- **Roll** - obrót wokół lokalnej osi Z (przechylenie)

Macierze rotacji dla poszczególnych osi są zdefiniowane jako:

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

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

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

Kombinacja rotacji jest realizowana jako iloczyn macierzy w odpowiedniej kolejności. W systemie zaimplementowano kolejność: yaw $\rightarrow$ pitch $\rightarrow$ roll, co zapewnia stabilne i intuicyjne sterowanie:

$$R_{combined} = R_z \cdot R_x \cdot R_y$$

Nowa macierz kamery C_{new} jest obliczana jako:

$$C_{new} = R_{combined} \cdot C_{current}$$

gdzie $C_{current}$ to aktualna macierz kamery.

3.3 Stabilizacja i ograniczenia

W systemie zaimplementowano mechanizmy stabilizacji i ograniczeń dla rotacji kamery:

- **Normalizacja yaw** - kąt yaw jest normalizowany do zakresu $[0, 360°)$
- **Stabilizacja** - funkcja stabilizacji zaokrągla kąty rotacji do najbliższych wartości 90-stopniowych, co ułatwia orientację w przestrzeni

3.3.1 Problem gimbal lock

Gimbal lock (zawieszenie kardana) to zjawisko występujące w systemach rotacji 3D, w którym traci się jeden stopień swobody. W przypadku kamery, problem ten pojawia się, gdy dwie osie rotacji stają się równoległe, co prowadzi do utraty jednego stopnia swobody. W układzie kardanicznym (eulerowskim), gdy pitch zbliża się do $\pm 90°$, osie yaw i roll stają się równoległe, co uniemożliwia niezależną kontrolę tych rotacji.

W systemie wirtualnej kamery zaimplementowano dwa podejścia do rozwiązania tego problemu:

1. **Ograniczenie kąta pitch** - kąt pitch jest ograniczony do zakresu $[-179°, 179°]$, co zapobiega osiągnięciu dokładnie $\pm 180°$, gdzie występuje pełne zawieszenie kardana.

2. **Kolejność rotacji** - rotacje są aplikowane w określonej kolejności (yaw $\rightarrow$ pitch $\rightarrow$ roll), co minimalizuje problem gimbal lock dla większości orientacji kamery.

W obecnej implementacji, ograniczenie kąta pitch do zakresu $[-179°, 179°]$ zapewnia dobry kompromis między swobodą ruchu kamery a stabilnością systemu. Pozwala to na prawie pełny obrót kamery w pionie, jednocześnie unikając problemu gimbal lock.

4 Podsumowanie Matematyczne

System wirtualnej kamery opiera się na fundamentalnych zasadach algebry liniowej:

- **Homogeniczne współrzędne:** Pozwalają na łączenie różnych transformacji (translacji, rotacji, skalowania) w jedną operację.
- **Macierze transformacji:** Umożliwiają precyzyjne operacje na punktach poprzez mnożenie macierzy.
- **Twierdzenie Eulera:** Każdy obrót w przestrzeni można sprowadzić do pojedynczego obrotu wokół ustalonej osi, co upraszcza modelowanie ruchu kamery.
- **Rzutowanie perspektywiczne:** Dzielenie przez w oraz przekształcenie do układu widoku umożliwiają realistyczną projekcję 3D na 2D.

5 Mapowanie sterowania

System wirtualnej kamery oferuje intuicyjne sterowanie poprzez kombinację klawiatury i myszy. Poniżej przedstawiono szczegółowe mapowanie kontroli:

5.1 Sterowanie klawiaturą

5.1.1 Translacja (Przesunięcie)

- **W** - przesunięcie w przód (oś Z)
- **S** - przesunięcie w tył (oś Z)
- **A** - przesunięcie w lewo (oś X)
- **D** - przesunięcie w prawo (oś X)
- **Shift** - przesunięcie w górę (oś Y)
- **Ctrl** - przesunięcie w dół (oś Y)

5.1.2 Rotacja (Obrót)

- **Strzałka w górę** - obrót w górę (pitch)
- **Strzałka w dół** - obrót w dół (pitch)
- **Strzałka w lewo** - obrót w lewo (yaw)
- **Strzałka w prawo** - obrót w prawo (yaw)
- **Alt** + **Strzałka w lewo** - obrót w lewo (roll)
- **Alt** + **Strzałka w prawo** - obrót w prawo (roll)

5.1.3 Dodatkowe kontrolki

- + - przybliżenie (zmniejszenie pola widzenia)
- **-** - oddalenie (zwiększenie pola widzenia)
- **R** - reset kamery do pozycji początkowej
- **E** - stabilizacja kamery (wyprostowanie)
- **ESC** - wyjście z aplikacji

5.2 Sterowanie myszą

5.2.1 Rotacja (Obrót)

- **Lewy przycisk myszy** + **ruch** - obrót kamery w kierunku ruchu myszy
  - Ruch w poziomie - obrót wokół osi Y (yaw)
  - Ruch w pionie - obrót wokół osi X (pitch)
- **Środkowy przycisk myszy** + **ruch w poziomie** - obrót wokół osi Z (roll)

5.2.2 Zoom

- **Kółko myszy w górę** - przybliżenie (zmniejszenie pola widzenia)
- **Kółko myszy w dół** - oddalenie (zwiększenie pola widzenia)

5.3 Parametry czułości

W systemie zaimplementowano następujące parametry czułości:

- **Czułość myszy w poziomie** - określa, jak szybko kamera obraca się w poziomie w odpowiedzi na ruch myszy (w stopniach na piksel)
- **Czułość myszy w pionie** - określa, jak szybko kamera obraca się w pionie w odpowiedzi na ruch myszy (w stopniach na piksel)
- **Czułość roll** - określa, jak szybko kamera obraca się wokół osi Z w odpowiedzi na naciśnięcie klawiszy Alt+strzałki (w stopniach na sekundę)

- **Prędkość ruchu** - określa, jak szybko kamera przesuwa się w przestrzeni w odpowiedzi na naciśnięcie klawiszy WASD (w jednostkach na sekundę)
- **Prędkość zoomu** - określa, jak szybko zmienia się pole widzenia kamery w odpowiedzi na naciśnięcie klawiszy +/- lub kółka myszy

6 Literatura i Źródła

1. Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1990). *Computer Graphics: Principles and Practice* (2nd ed.). Addison-Wesley.
2. Takahashi, S. (2006). *The Manga Guide to Linear Algebra*. No Starch Press.
3. Lengyel, E. (2004). *Mathematics for Game Programming and Computer Graphics*. CRC Press.
4. Shirley, P., Marschner, S. (2013). *Fundamentals of Computer Graphics* (4th ed.). A K Peters/CRC Press.
5. Pygame Documentation.
6. NumPy Documentation.