Sprawozdanie z projektu: Wirtualna kamera 3D

1 Streszczenie projektu

Projekt polegał na implementacji wirtualnej kamery 3D umożliwiającej swobodną nawigację w przestrzeni trójwymiarowej oraz wizualizację obiektów 3D za pomocą rzutowania perspektywicznego. Główne zadania obejmowały realizację transformacji geometrycznych w przestrzeni 3D (translacja, rotacja), implementację rzutowania perspektywicznego oraz rendering obiektów 3D z uwzględnieniem ich krawędzi. Projekt zrealizowano w Javie z wykorzystaniem biblioteki graficznej Swing.

2 Realizacja zadania

2.1 Układ współrzędnych i reprezentacja obiektów

W projekcie zastosowano prawoskrętny układ współrzędnych 3D, gdzie:

- Oś X wskazuje w prawo
- Oś Y wskazuje w górę
- Oś Z wskazuje w głąb (od obserwatora)

Obiekty 3D reprezentowane są jako zbiory wierzchołków (punktów) oraz krawędzi łączących te wierzchołki. Każdy punkt opisany jest za pomocą trzech współrzędnych (x, y, z), a krawędź definiowana jest przez dwa punkty (początkowy i końcowy).

Pierwotnie planowano użycie współrzędnych nieznormalizowanych, jednak finalnie zdecydowałem się na zastosowanie współrzędnych jednorodnych, które zapewniają bardziej spójną reprezentację przekształceń (translacji, rotacji, rzutowania) za pomocą mnożenia macierzy 4×4.

2.2 Transformacje geometryczne

2.2.1 Translacja kamery

Translacja kamery realizowana jest poprzez przesuwanie jej położenia wzdłuż trzech osi współrzędnych (X, Y, Z). Macierz translacji T ma postać:

$$T = \begin{bmatrix} 1 & 0 & 0 & -T_x \\ 0 & 1 & 0 & -T_y \\ 0 & 0 & 1 & -T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

gdzie (T_x, T_y, T_z) to współrzędne kamery. Znaki ujemne wynikają z tego, że przesuwamy scenę w kierunku przeciwnym do ruchu kamery.

2.2.2 Rotacja kamery

Rotacja kamery realizowana jest przy pomocy trzech macierzy obrotów wokół poszczególnych osi:

Rotacja wokół osi X (góra/dół):

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) & 0 \\ 0 & \sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Rotacja wokół osi Y (lewo/prawo):

$$R_y = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Rotacja wokół osi Z (przechylanie):

$$R_z = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Finalna macierz rotacji powstaje przez złożenie tych trzech rotacji. W implementacji zastosowano kolejność: $Y \rightarrow X \rightarrow Z$.

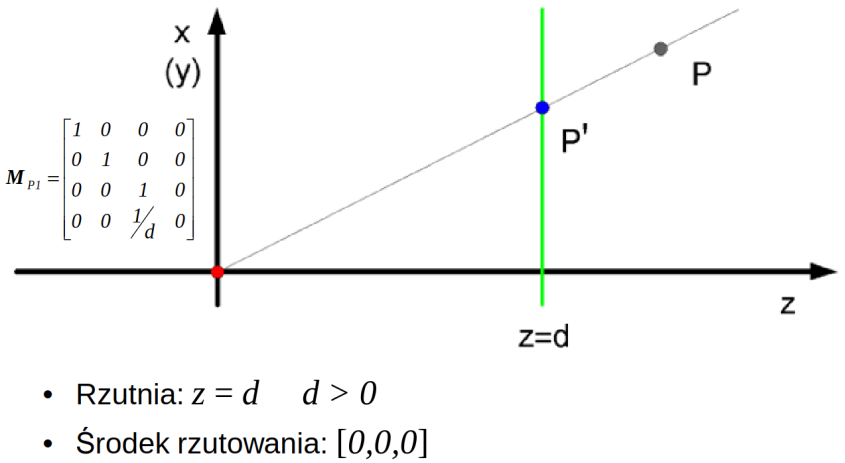
2.3 Rzutowanie perspektywiczne

Do rzutowania perspektywicznego wykorzystano macierz M o postaci:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \tag{5}$$

gdzie d to odległość rzutni od środka rzutowania. W tej implementacji przyjęto $d = 5$.

Macierz ta realizuje rzutowanie środkowe, w którym wszystkie linie rzutujące przechodzą przez środek rzutowania (punkt $(0, 0, 0)$ w układzie kamery). Płaszczyzna rzutni znajduje się na pozycji $z = d$.



Rysunek 1: Rzutowanie perspektywiczne

Dla punktu $P = (x, y, z, 1)$ w przestrzeni kamery, po zastosowaniu macierzy rzutowania i normalizacji współrzędnych jednorodnych, otrzymujemy punkt $P' = (x', y', d, 1)$, gdzie:

$$x' = x \cdot \frac{d}{z} \tag{6}$$

$$y' = y \cdot \frac{d}{z} \tag{7}$$

Efekt perspektywy wynika z zależności współczynnika skalowania $\frac{d}{z}$ od odległości punktu od kamery: obiekty bliższe są większe, dalsze mniejsze.

2.4 Pole widzenia (FOV)

Pole widzenia (FOV - Field of View) zaimplementowano jako modyfikację standardowej macierzy rzutowania, wprowadzając współczynnik skalowania $fovScale$:

$$fovScale = \frac{\tan(FOV/2)}{\tan(60^\circ/2)} \tag{8}$$

Współczynnik ten jest następnie uwzględniany w macierzy rzutowania przez zastąpienie $\frac{1}{d}$ przez $\frac{fovScale}{d}$, co pozwala na regulację "siły" efektu perspektywy bez zmiany pozycji rzutni.

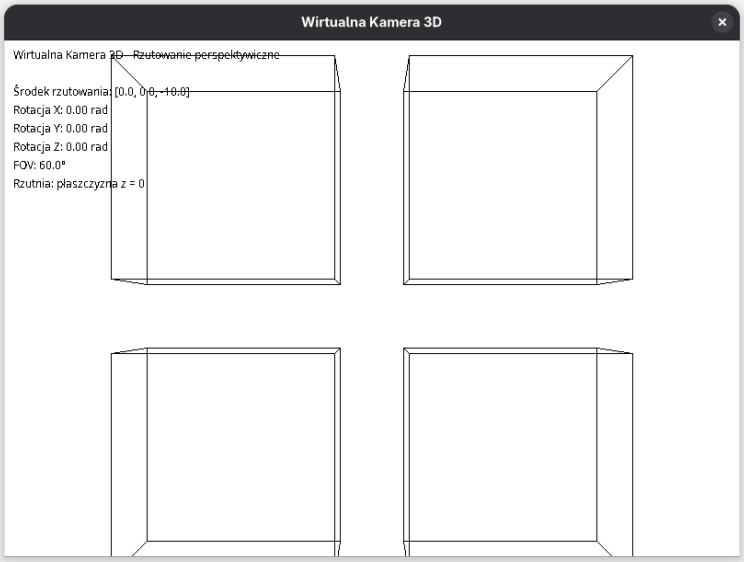
2.5 Pełny proces renderingu

Pełny proces renderingu sceny 3D składa się z następujących kroków:

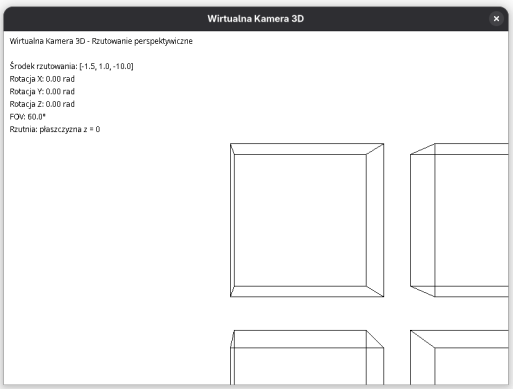
1. **Transformacja punktów do przestrzeni kamery:**
   - Zastosowanie macierzy widoku (View Matrix), która jest złożeniem macierzy translacji i rotacji
   - Dla każdego punktu P_{world} w przestrzeni świata obliczamy $P_{camera} = ViewMatrix \times P_{world}$
2. **Odrzucanie punktów poza polem widzenia:**
   - Punkty o $z \leqslant 0$ (znajdujące się za kamerą) są odrzucane
3. **Rzutowanie perspektywiczne:**
   - Zastosowanie macierzy rzutowania M do punktów w przestrzeni kamery
   - Dla każdego punktu P_{camera} obliczamy $P_{projected} = M \times P_{camera}$
4. **Normalizacja współrzędnych jednorodnych:**
   - Dzielenie współrzędnych x, y, z przez współrzędną w (która po rzutowaniu nie jest już równa 1)
   - $P_{normalized} = (\frac{P_{projected}.x}{P_{projected}.w}, \frac{P_{projected}.y}{P_{projected}.w}, \frac{P_{projected}.z}{P_{projected}.w}, 1)$
5. **Transformacja do współrzędnych ekranowych:**
   - Przeskalowanie i przesunięcie znormalizowanych współrzędnych do przestrzeni ekranu
   - $scale = screenHeight/2$
   - $screenX = screenWidth/2 + P_{normalized}.x \times scale$
   - $screenY = screenHeight/2 - P_{normalized}.y \times scale$
6. **Renderowanie krawędzi:**
   - Narysowanie linii łączących przetransformowane punkty
   - Krawędzie są filtrowane na podstawie położenia ich punktów końcowych:
     - Krawędzie, których oba punkty znajdują się w rozsądnej odległości od obszaru widoczności (z wartościami współrzędnych powyżej progu -500), są przekazywane do renderingu
     - Krawędzie z punktami daleko poza ekranem lub z punktami znajdującymi się za kamerą (reprezentowanymi przez arbitralne współrzędne (-1000, -1000)) są całkowicie pomijane
     - Dla widocznych krawędzi, których fragment wychodzi poza ekran, wykorzystywane jest automatyczne przycinanie (clipping) zapewniane przez bibliotekę graficzną

3 Zdjęcia

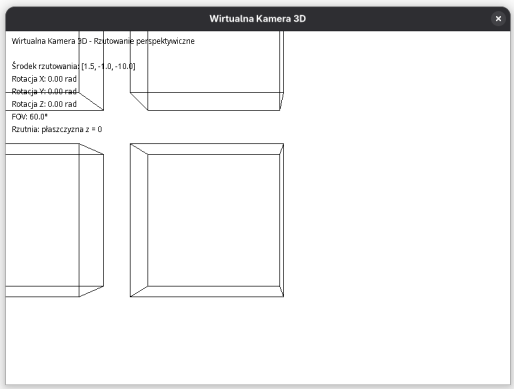
3.1 Translacja



Rysunek 2: Widok początkowy

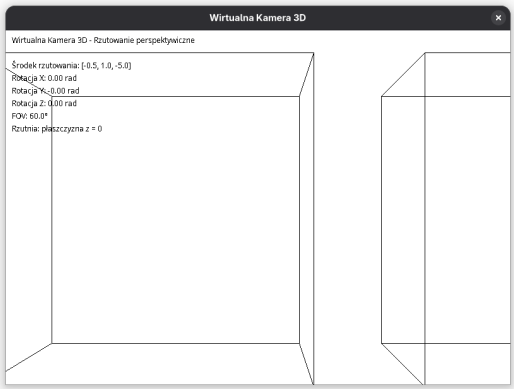


(a) Translacja w lewo i górę

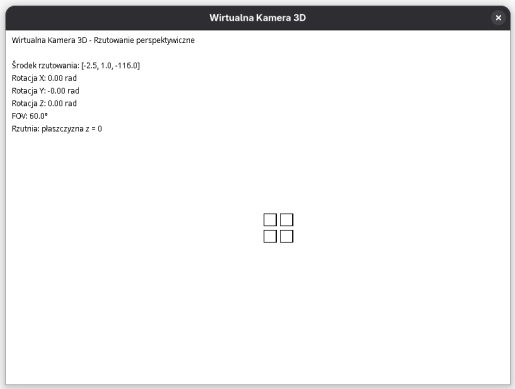


(b) Translacja w prawo i w dół

Rysunek 3: Translacja w płaszczyźnie XY



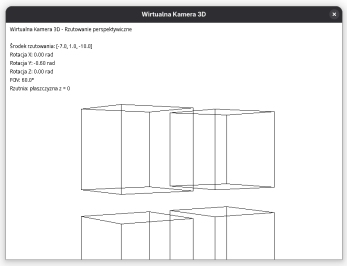
(a) Translacja w przód



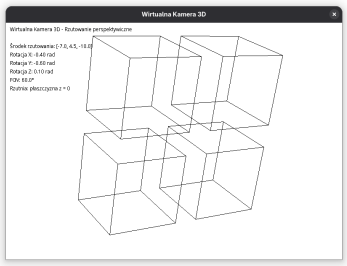
(b) Translacja w tył

Rysunek 4: Translacja wzdłuż osi Z

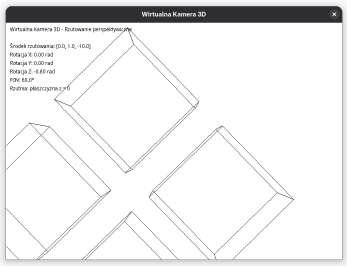
3.2 Obrót



(a) Obrót w prawo



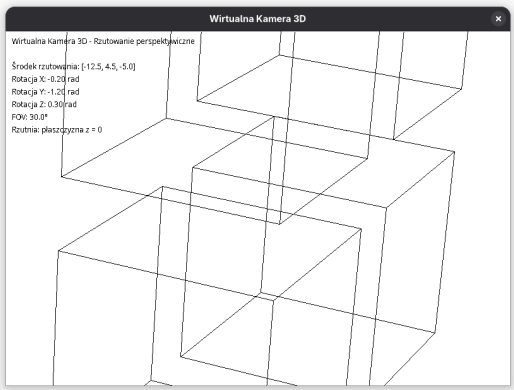
(b) Obrót w dół



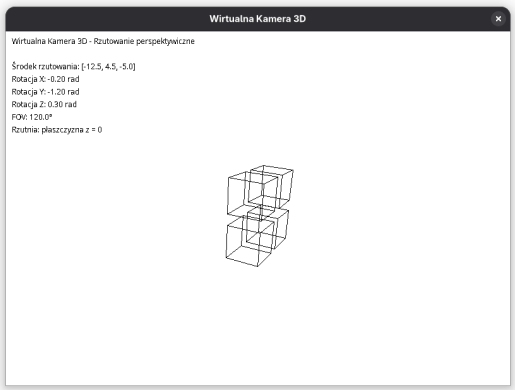
(c) Obrót wzdłuż osi Z

Rysunek 5: Różne rodzaje obrotów kamery

3.3 Zoom



(a) Zoom in



(b) Zoom out

Rysunek 6: Efekty zmiany pola widzenia (FOV)

4 Wnioski

Implementacja wirtualnej kamery 3D pozwoliła na realizację wszystkich założonych celów projektu. Zastosowanie współrzędnych jednorodnych i macierzy 4×4 znacząco uprościło implementację przekształceń geometrycznych oraz rzutowania perspektywicznego, zapewniając jednocześnie poprawne działanie efektów perspektywy.

Problem "gimbal lock" występujący przy rotacjach Eulera został częściowo złagodzony przez odpowiedni dobór kolejności rotacji ($Y \rightarrow X \rightarrow Z$), choć pełne rozwiązanie tego problemu wymagałoby zastosowania bardziej zaawansowanych metod reprezentacji rotacji, np. kwaternionów.