Rafał Gulewski 307348 09.05.2022r.

Grafika Komputerowa – Projekt 1
Kamera Wirtualna

Opis zadania

Celem projektu było utworzenie kamery wirtualnej pozwalającej na obserwowanie obiektów rysowanych krawędziowo w przestrzeni. Kamera miała pozwalać na poruszanie się, obracanie się i przybliżanie oraz oddalanie widoku.

Dane techniczne programu

Program powstał zgodnie z założeniami w języku Java z wykorzystaniem biblioteki Swing i AWT. Zrezygnowałem jednak z biblioteki EJML. Zamiast tego utworzona została zwykła klasa odpowiedzialne za obliczenia macierzowe.

Działanie

Program działa domyślnie w następujących ustawieniach:

- liczba wyświetlanych klatek na sekundę: 2
- pojedyncze przemieszczenie kamery: 0.1
- pojedynczy kąt obrotu kamery: 0.01 stopnia
- domyślna ogniskowa: -10
- pojedyncza zmiana ogniskowej: 0.025

Obiekty są wczytywane z pliku tekstowego. Każda linia pliku odpowiada krawędzi obiektu zaś puste linie oddzielają obiekty od siebie.

Następnie wyświetlany jest obraz zaś użytkownik może zacząć poruszać kamerą zgodnie z poniższym sterowaniem:

- W/S – przesunięcie oś Z
- A/D – przesunięcie oś X
- Q/E – przesunięcie oś Y
- I/O – zoom in/out
- Y/H – rotacja oś X
- T/U – rotacja oś Z
- G/J – rotacja oś Y

Transformacje widoku kamery

Współrzędne wierzchołków figur są aktualizowane na bieżąco, za pomocą odpowiednich obliczeń opierających się na macierzach przekształceń.

Translacja:

Za translację odpowiedzialna jest macierz przemieszczenia:

$$\begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + X \\ y + Y \\ z + Z \\ 1 \end{bmatrix}$$

Rotacja:

Do implementacji rotacji użyłem macierzy:

Rotacja osi X:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & sin\,\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ \cos\theta * y - \sin\theta * z \\ \sin\theta * y + \cos\theta * z \\ 1 \end{bmatrix}$$

Rotacja osi Y:

$$\begin{bmatrix} \cos\theta & 0 & sin\,\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta * x + sin\,\theta * z \\ y \\ -\sin\theta * x + \cos\theta * z \\ 1 \end{bmatrix}$$

Rotacja osi Z:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ sin\,\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta * x - sin\,\theta * y \\ sin\,\theta * y + \cos\theta * y \\ z \\ 1 \end{bmatrix}$$

Przybliżanie:

Do implementacji funkcji zoom-owania użyłem odpowiedniego przekształcania punktów z przestrzeni 3D na 2D w oparciu o ogniskową. Zmiana ogniskowej pozwala na osiągnięcie efektu przybliżania i oddalania obrazu.

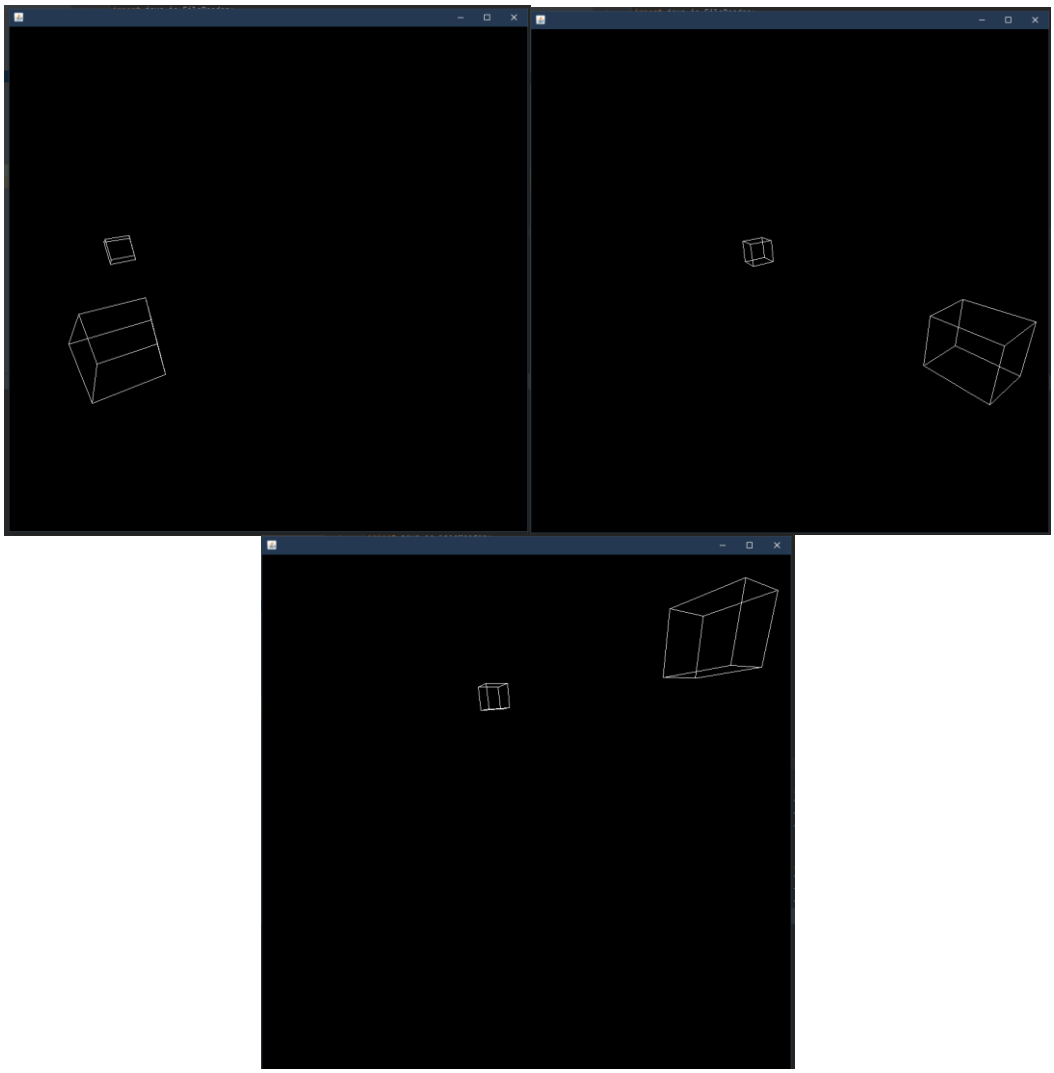
Funkcja rzutująca punktu z przestrzeni 3D na ekran:

```
x = ogniskowa/z * x + szerokość_ekranu / 2;
y = wysokość_ekranu / 2 - ogniskowa/z * y;
```

Prezentacje działania

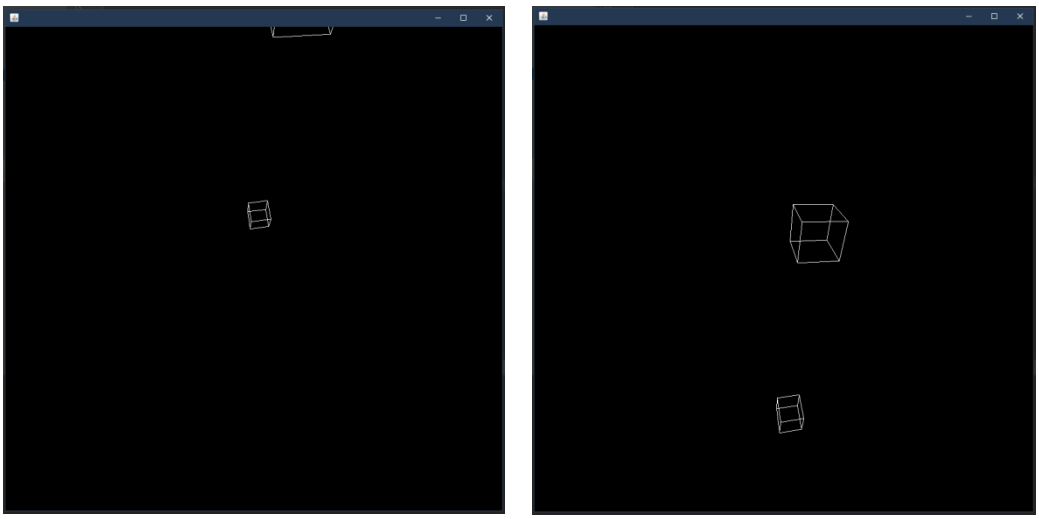
Translacja:

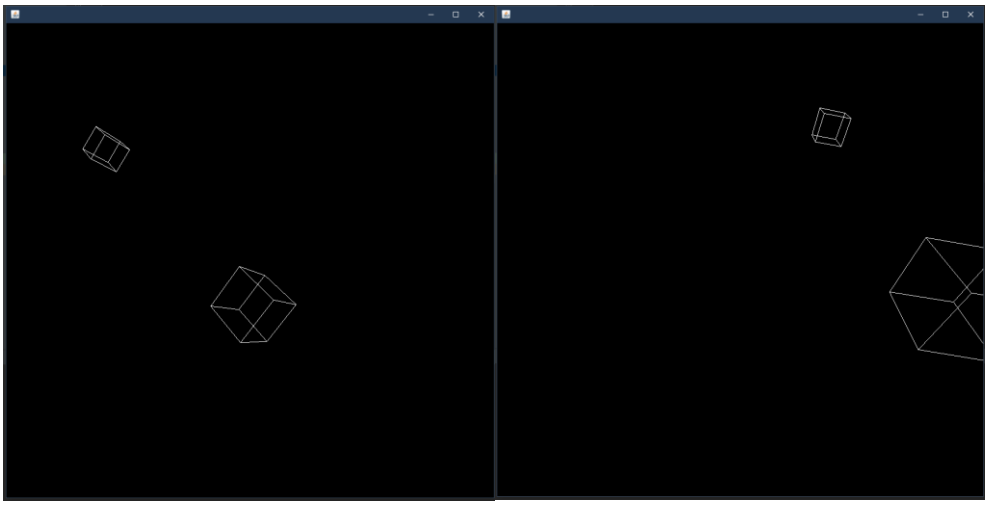
Poniższe zrzuty ekranu przedstawiają translację wzdłuż osi X oraz wzdłuż osi Y



Rotacja:

Poniższe zrzuty ekranu przedstawiają kolejno: rotację wokół osi X, osi Z, oraz osi Y





Przybliżenie:

Widok programu przed i po przybliżeniu obiektów