Kamera wirtualna – sprawozdanie

Grafika komputerowa – projekt – część 1

Autor: Katarzyna Stankiewicz
Numer indeksu: 299264
Data: 22.03.2021r.

---

1. Cel ćwiczenia

Celem ćwiczenia było stworzenie kamery wirtualnej pozwalającej na wykonanie podstawowych operacji (translacja, rotacja, zoom), używając do tego celu odpowiednich przekształceń geometrycznych.

2. Opis projektu

W efekcie wykonanego ćwiczenia powstała kamera wirtualna, która widzi obraz 3D zrzutowany na płaszczyznę, tak, aby zachować zgodność perspektywy. Kamera obsługuje operacje translacji, rotacji oraz zoom. Obraz widziany przez kamerę przedstawia cztery bryły (budynki) ustawione w dwóch rzędach, imitując obraz ulicy, widziany w perspektywie. Używając przycisków klawiatury, do obsługi poszczególnych operacji (translacja, rotacja, zoom), obraz można obejrzeć pod różnymi kątami oraz z różnych stron.

3. Narzędzia wykorzystane do realizacji projektu

Do realizacji projektu zdecydowałam się użyć języka python, wykorzystując bibliotekę pyGame.

4. Szczegółowy opis realizacji

W celu zrealizowania konkretnych przekształceń geometrycznych użyłam zdefiniowanych dla nich operacji matematycznych. Poniżej szczegółowo zostały opisane metody wykorzystane dla poszczególnych operacji na przestrzeni 3D.

Rzutowanie – reprezentacja przestrzeni trójwymiarowej na płaszczyźnie

W projekcie wykorzystałam rzutowanie perspektywiczne, które sprawia, że obraz, na który patrzymy, wydaje się bardziej realistyczny, ponieważ linie zbiegają się w jednym punkcie. Aby osiągnąć taki efekt (przy założeniu, że kamera znajduje się w początku układu współrzędnych) wykorzystałam następującą operację macierzową:

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ \frac{d}{z_p} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \cdot \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

gdzie:

x_p, y_p, z_p - współrzędne rzutowanego punktu

d - odległość od rzutni

Uzyskany w ten sposób wektor poddałam normalizacji i w wyniku otrzymałam, wektor zawierający odpowiednio przekształcone współrzędne x oraz y, które posłużyły do narysowania rzutu na płaszczyźnie:

$$\begin{bmatrix} \frac{x_p \cdot d}{z_p} \\ \frac{y_p \cdot d}{z_p} \\ d \\ 1 \end{bmatrix}$$

Translacja

Aby uzyskać punkt (x'_p, y'_p, z'_p), przesunięty względem punktu początkowego (x_p, y_p, z_p) o wektor $[T_X, T_Y, T_Z]$, posłużyłam się poniższym równaniem macierzowym:

$$\begin{bmatrix} x'_p \\ y'_p \\ z'_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_X \\ 0 & 1 & 0 & T_Y \\ 0 & 0 & 1 & T_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

Dzięki zastosowaniu takiej operacji otrzymałam odpowiednio przesunięte współrzędne, które w czytelniejszy sposób możemy zapisać również:

$$x'_p = x_p + T_X$$

$$y'_p = y_p + T_Y$$

$$z'_p = z_p + T_Z$$

ponieważ mnożąc macierz translacji przez punkt, który chcemy przesunąć, tak naprawdę realizujemy operację dodawania.

Rotacja

Wykonując operację rotacji również posłużyłam się operacjami macierzowymi.

Znając macierze dla operacji rotacji względem każdej z osi, postanowiłam pomnożyć je, aby w wyniku otrzymać jedną macierz, która pozwoli mi wykonać operację rotacji względem dowolnej osi podając odpowiedni kąt rotacji dla osi, według której zostanie ona wykonana.

Macierz rotacji względem osi OX:

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

Macierz rotacji względem osi OY:

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

Macierz rotacji względem osi OZ:

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

Ogólna macierz rotacji:

$$R = \mathrm{R_x}(\alpha) \cdot \mathrm{R_y}(\beta) \cdot \mathrm{R_z}(\gamma) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{R} = \begin{bmatrix} \cos\beta \cdot \cos\gamma & -\cos\beta \cdot \sin\gamma & \sin\beta & 0 \\ \cos\alpha \cdot \sin\gamma + \sin\alpha \cdot \sin\beta \cdot \cos\gamma & \cos\alpha \cdot \cos\gamma - \sin\alpha \cdot \sin\beta \cdot \sin\gamma & -\sin\alpha \cdot \cos\beta & 0 \\ \sin\alpha \cdot \sin\gamma - \cos\alpha \cdot \sin\beta \cdot \cos\gamma & \cos\alpha \cdot \sin\beta \cdot \sin\gamma + \sin\alpha \cdot \cos\gamma & \cos\alpha \cdot \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Wykorzystując tak przygotowaną macierz oraz podając trzy kąty obrotu względem odpowiednich osi, mogłam za pomocą jednego mnożenia macierzowego obrócić punkt względem wszystkich osi.

Zoom

Operację zoom zrealizowałam dużo prościej, manewrując jedynie odległością od rzutni. W przypadku wykonania operacji zoom zwiększam odległość rzutni dodając pewną wartość, określoną w programie, w przypadku operacji odwrotne zmniejszam tę odległość, odejmując określoną w programie wartość.

5. Wczytywanie konstrukcji z pliku

Opis formy pliku

Program realizujący model wirtualnej kamery posiada moduł pozwalający na wczytanie konstrukcji z pliku. Aby wczytać podaną konstrukcję został zastosowany prosty szablon, w którym definiujemy bryły 3D posługując się definicją wierzchołków oraz krawędzi.

Szablon ten zakłada, że najpierw podajemy punkty, z których składa się dana bryła, poprzedzone słowem ‘vertices’. Współrzędne jednego punktu znajdują się w jednej linii i są oddzielone przecinkami. Następnie definiujemy połączenia między nimi (krawędzie), poprzedzając je słowem ‘edges’. W przypadku krawędzi również każda z nich znajduje się w osobnej linii, a wierzchołki, które łączy oddzielone są przecinkami. W pliku możemy dodać komentarze, ponieważ linie, które nie spełniają wymagań punktu wierzchołka lub krawędzi, są ignorowane.

6. Poruszanie się w wirtualnej przestrzeni

Konkretnym operacjom wykonywanym w ramach realizacji kamery (translacja, rotacja, zoom), przypisałam odpowiednio klawisze:

Translacja

Poruszanie się względem osi X

Strzałka w prawo – przemieszczenie w prawo
Strzałka w lewo – przemieszczenie w lewo

Poruszanie się względem osi Y

Strzałka w górę – przemieszczenie w górę
Strzałka w dół – przemieszczenie w dół

Poruszanie się względem osi Z

. (kropka) – przemieszczenie w przód
, (przecinek) – przemieszczenie w tył

Rotacja

Obrót względem osi X

Q – obrót w prawo
A – obrót w lewo

Obrót względem osi Y

W – obrót w prawo
S – obrót w lewo

Obrót względem osi Z

E – obrót w prawo
D – obrót w lewo

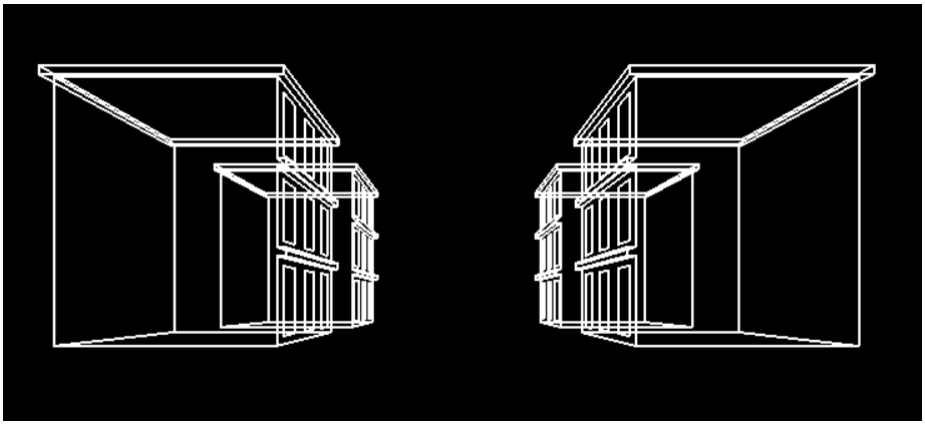
Zoom

Z – przybliżenie
X – oddalenie

7. Przykłady działania programu

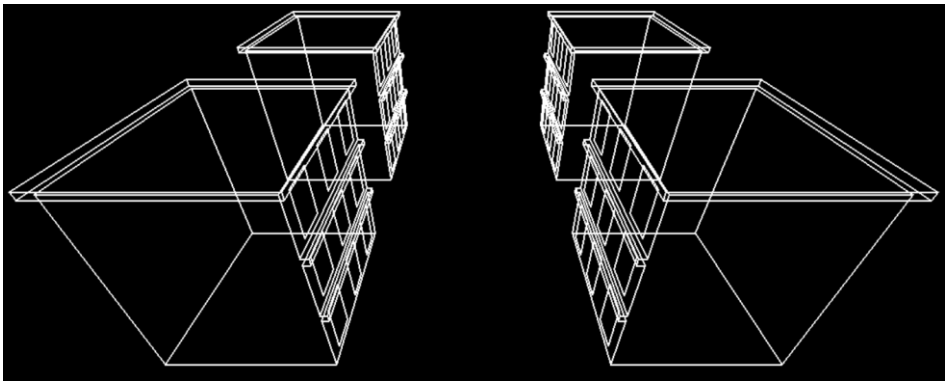
Widok podstawowy

Widok, który ukazuje się naszym oczom tuż po uruchomieniu programu, wygląda następująco:



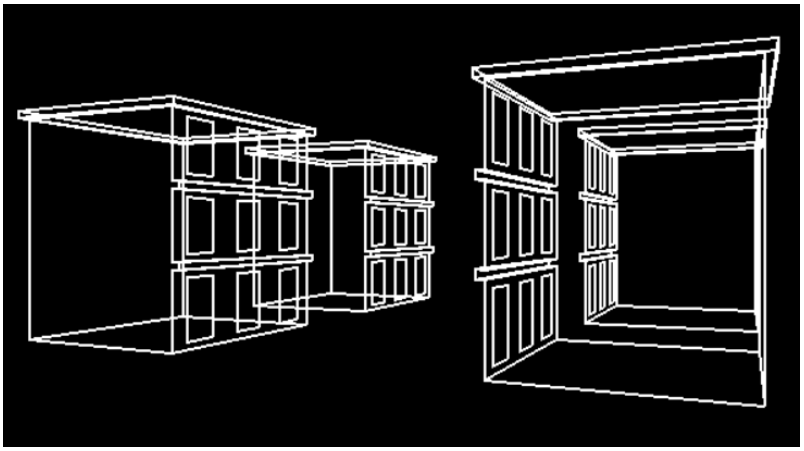
Rotacja względem osi OX + translacja w górę

W ten sposób uzyskujemy widok z góry.

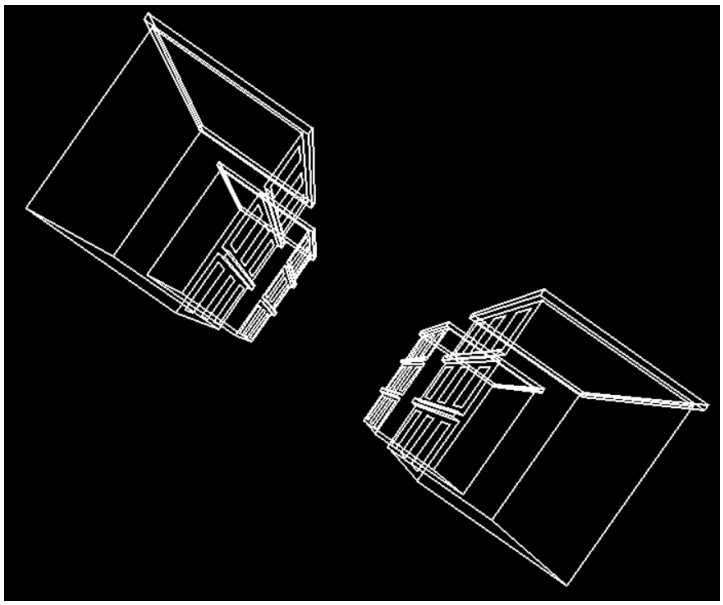


Rotacja w lewo względem osi OY + translacja w prawo

W ten sposób uzyskujemy lepszy widok frontu budynków z lewej strony.

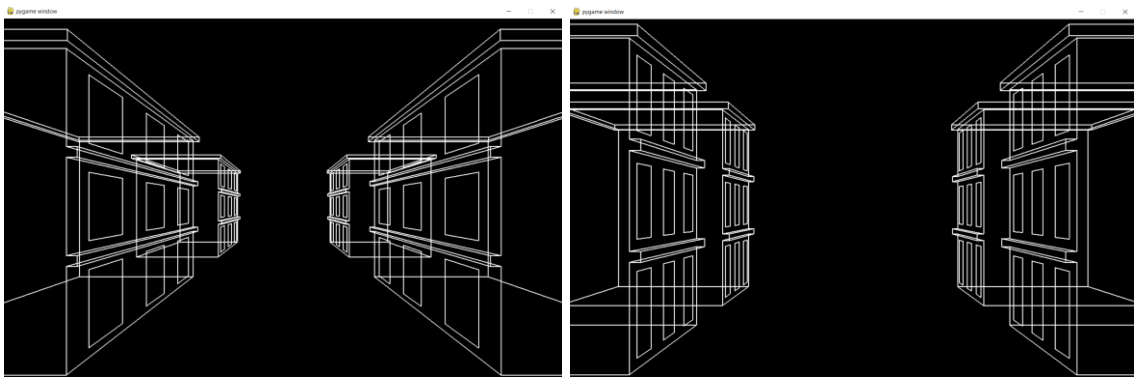


Rotacja względem osi OZ



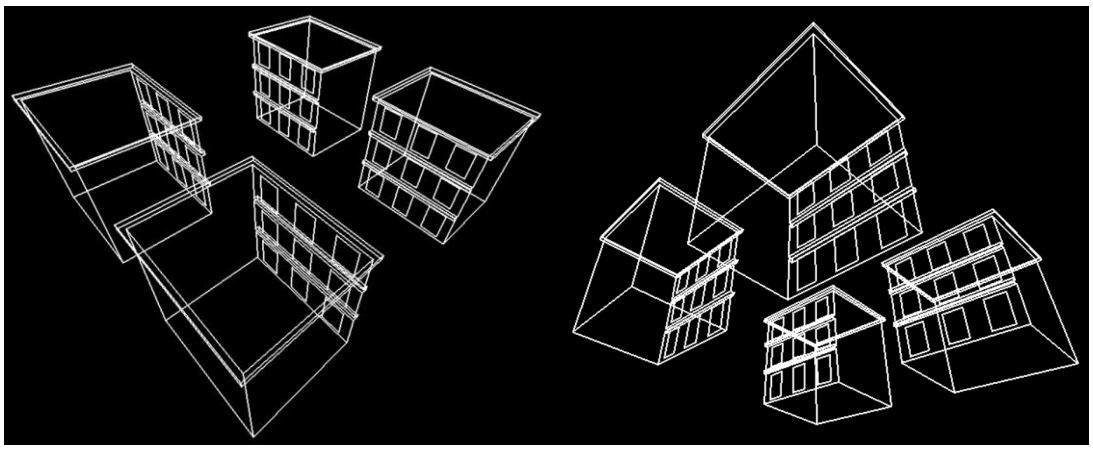
Porównanie translacji względem osi OZ do operacji zoom

Podczas wykonywania operacji zoom, perspektywa nie zmienia się. Możemy jedynie zauważyć, że oglądamy fragment obrazu w powiększeniu. Natomiast, wykonując translację względem osi OZ mamy wrażenie „podejścia bliżej”.



(a) translacja względem osi OZ (b) operacja zoom

Inne widoki uzyskane przez połączenie różnych operacji



8. Podsumowanie

W projekcie udało się zrealizować wszystkie założone na etapie planowania funkcjonalności. Największym wyzwaniem podczas realizacji projektu było zapoznanie się z zasadami rzutowania oraz przekształceń geometrycznymi i zrozumienie ich działania w praktyce, ponieważ w momencie rozpoczęcia realizacji projektu tematyka ta nie została jeszcze omówiona na wykładzie. Zrozumienie działania tych operacji pozwoliło na zaimplementowanie tego przy użyciu biblioteki pyGame, która po zapoznaniu się z jej działaniem, okazała się dość prosta i wygodna w użyciu.