

# GRAFIKA KOMPUTEROWA

## Projekt 1 - Sprawozdanie

### Opis zadania

Celem projektu jest zaimplementowanie kamery mogącej obserwować trójwymiarowe obiekty w przestrzeni. Kamera będzie posiadała możliwość poruszania się w przestrzeni, wykonywania obrotów oraz zmiany aktualnego przybliżenia.

[Repozytorium z projektem](#) jest dostępne na platformie Github. Projekt jest udostępniony publicznie na licencji MIT.

### Zmiany w stosunku do wstępnego planu

Ponieważ program wczytuje dane wyświetlanych obiektów z plików nie możliwa była implementacja w języku Javascript. Zamiast tego zdecydowałem się na język Python i bibliotekę pygame.

### Model matematyczny operacji

Program przechowuje wyświetlane modele jako macierz współrzędnych wierzchołków o wymiarach  $[n \times 4]$ , gdzie  $n$  to liczba punktów danej figury oraz listę krawędzi. Kolejne kolumny w macierzy wierzchołków to odpowiednio współrzędne  $x$ ,  $y$ ,  $z$  danego punktu oraz stała wartość 1. Pozwala to na sprawne wykonywanie transformacji wierzchołków. Krawędzie są przechowywane jako lista par numerów punktów, między którymi jest krawędź. Wyraźny podział między współrzędnymi wierzchołków a krawędziami pozwala na uniknięcie duplikacji danych i usprawnienie transformacji.

### Transformacje

Operacje ruchu i obrotów kamery są realizowane jako odpowiednie transformacje współrzędnych wierzchołków wszystkich figur. Transformacje są realizowane **jednocześnie** dla wszystkich punktów danej figury przy pomocy operacji macierzowych.

Przy implementacji transformacji wzorowałem się [artykułem napisanym przez Tobbiego Rufinusa](#).

### Translacje

Ruch kamery w osiach  $x$ ,  $y$ ,  $z$  jest wykonywany poprzez translacje wszystkich wierzchołków o stały krok w każdej klatce, w której użytkownik naciska klawisz ruchu.

W tym celu wykorzystana jest poniższa macierz:

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

## Rotacje

Obroty kamery w osiach x, y, z jest wykonywany poprzez zmianę pozycji wszystkich wierzchołków o stały krok w każdej klatce, w której użytkownik naciska klawisz ruchu.

Do rotacji w osi x została wykorzystana poniższa macierz:

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

Do rotacji w osi y została wykorzystana poniższa macierz:

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

Do rotacji w osi z została wykorzystana poniższa macierz:

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

## Mapowanie punktów 3d na płótno ekranu

W celu narysowania punktów na ekranie wartości 3d są rzutowane na płaszczyznę ekranu kamery. Dzięki temu, że zastosowany wzór uwzględnia wartość ogniskowej łatwe jest zaimplementowanie przybliżania i oddalania obrazu (zoom).

Do rzutowania punktów została wykorzystana poniższa funkcja.

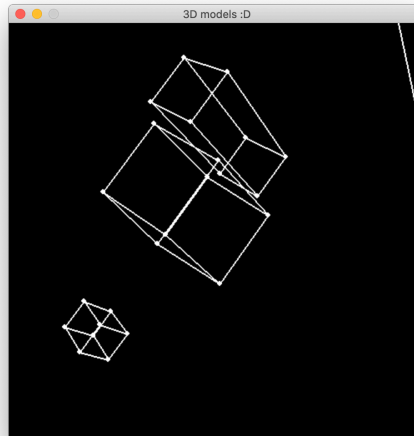
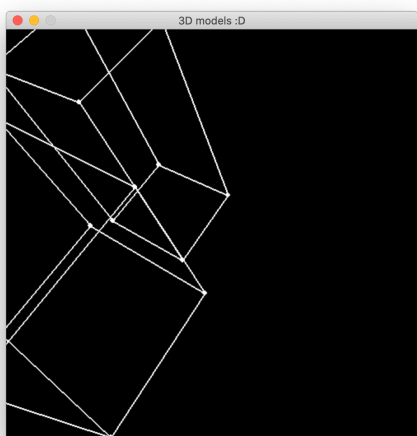
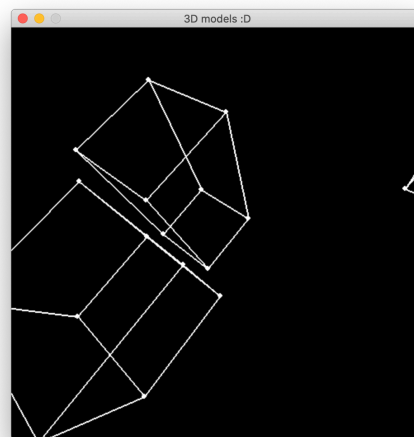
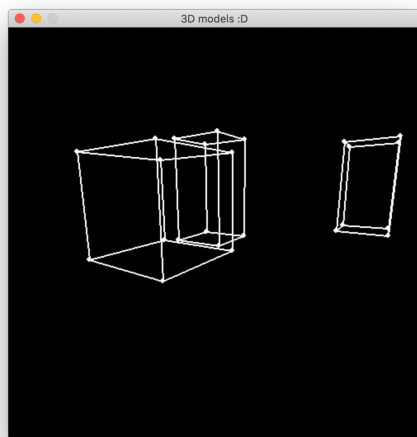
```
def translate_3d_to_2d(point_3d: np.array, view_width: float, view_height: float, focal: float) ->
    Tuple[float, float]:
    """Map 3D point to 2D value that can be displayed"""
    from_focal = focal / point_3d[1]
    x = from_focal * point_3d[0] + view_width / 2
    y = view_height / 2 - from_focal * point_3d[2]

    return x, y
```

## Przykłady działania programu

[Nagranie prezentujące działanie programu](#). Nie prezentuje ono niestety funkcji zoom, jednak jest ona widoczna na 3 grafice poniżej.

Jeśli wystąpiłby problem z nagraniem to poniższe zrzuty ekranu przedstawiają generowane grafiki.



## Obsługa programu

Do uruchomienia programu potrzeba jest Python3 z zainstalowanymi bibliotekami numpy (do operacji na macierzach) i pygame (do rysowania linii i okręgów). Następnie wystarczy uruchomić skrypt main.py.

Program po uruchomieniu wczyta modele z folderu *models*. Modele są w postaci listy linii, gdzie pierwsze 3 wartości to współrzędne punktu początkowego linii, a 3 kolejne to współrzędne końca linii. Na podstawie podanych linii tworzone są wierzchołki oraz krawędzie wyświetlanych modeli.

### Sterowanie

Za poruszanie kamerą odpowiadają klawisze w, a, s, d lub klawisze strzałek.

Za zmianę ogniskowej odpowiadają klawisze -, =.

Za rotację kamery w osi x odpowiadają klawisze i, k.

Za rotację kamery w osi y odpowiadają klawisze u, o.

Za rotacje kamery w osi z odpowiadają klawisze j, l.

## Generacja modeli

Do łatwego tworzenia modeli prostopadłościanów został napisany skrypt `generateModel.py`

Wymaga on podania współrzędnych jednego wierzchołka sześcianu oraz długości krawędzie w każdej z osi.