Przedmiot Grafika komputerowa - Projket 1
Wirtualna kamera

Wykonanie:
1. Ivan Prakapets

Data:23.03.2020-31.03.2020

Warszawa

Spis treści

1 Cel projektu

Celem projektu jest rozszerzyć swoje umiejętności oraz dowiedzieć się więcej o grafice komputerowej. Jednym słowem poczuć ją z punktu własnego doświadczenia. Zadanie projektu jest napisać program, który będzie umożliwiał poruszać się w dowolnym kierunku oraz powiększać i zmniejszyć rysunki, które dobrałem **4 prostopadłościany**. Do tego służy wirtualna kamera. Można powiedzieć to jest działanie podobne do zwykłej kamery. Projekt został zrealizowany na Python3 (używając biblioteki `tkinter` do rysowania, `numpy` do wyliczania macierzy.

2 Rysowanie prostopadłościanów

Dla poprawnego działania i napisania wirtualnej kamery, na początku trzeba zastanowić się co będziemy obserwować i w okół czego będziemy poruszać się. Najfajniejszym przykładem dla zilustrowania działania wirtualnej kamery jest zwykły prostopadłościan, bo widać wszystkie ściany. Dla jeszcze lepszego ilustrowania i pokazania poprawnego rzutowania są 4 protopadłościany.

2.1 Budowa prostopadłościanu

Budowa prostopodłościana:

- wierzchołki
- ściana
- krawędzi
- podstawa

2.2 Wierzchołki

Zacząłem od rysowania wierzchołków:

`(-1,-1,-1),(1,-1,-1),(1,1,-1),(-1,1,-1),(-1,-1,1),(1,-1,1),(1,1,1),(-1,1,1)`

Robimy to wokół nas, więc przesuńmy do przodu.

2.3 Renderowanie 3d to 2d

Można zauważyć, że na ekranie nie ma osi **Z**, więc użyjemy tylko **X** i oś **Y**, ale w zależności od osi **Z** punktu (czyli głębokości), będziemy kierować w kierunku odległości środka.

2.4 Krawędzi

Następnie jest rysowanie krawędzi:

`(0,1),(0,3),(0,4)`

`(2,1),(2,3),(2,6)`

`(5,1),(5,4),(5,6)`

`(7,3),(7,4),(7,6)`

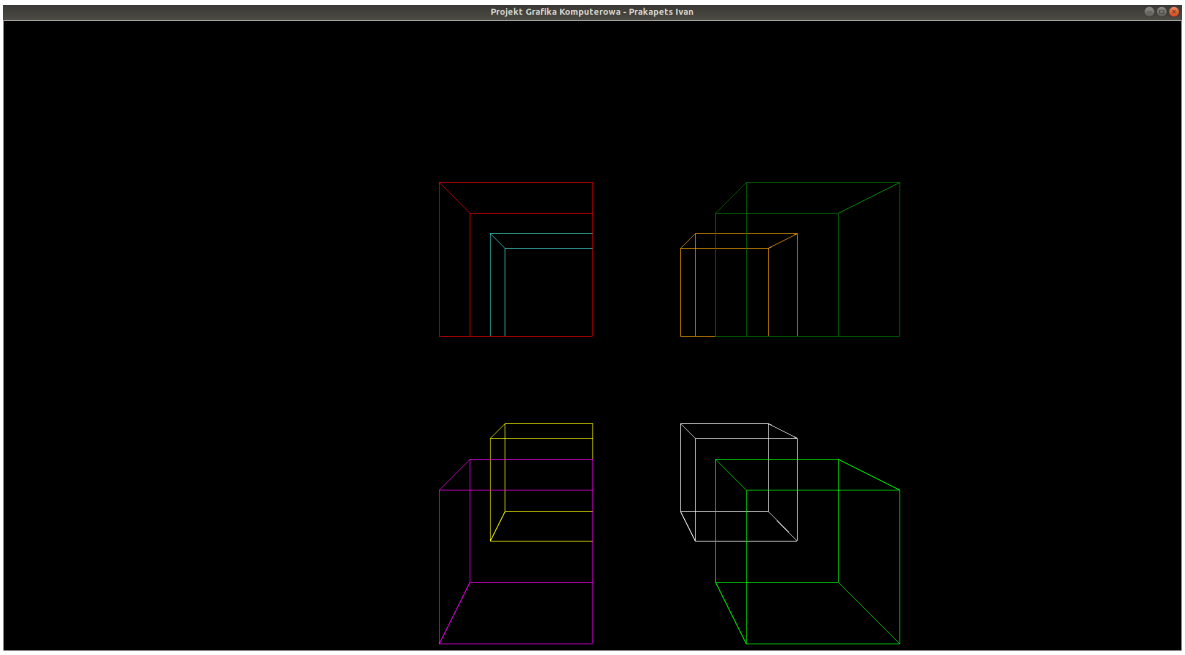
Kiedy mamy krawędzi oraz wierzchołki musimy uzyskać lokalizację 2D z nich obu umieścimy je na liscie.

2.5 Sterowanie kamerą

Pamiętaj, że prostopadłościany sterujące kamerą, więc obserwowane efekty są przeciwne (np. jeżeli kamera obraca się w prawo, prostopadłościany obracają się w lewo) .

3 Pierwotny stan programu

Poniżej jest pierwotny stan programu(po uruchomieniu).



Rysunek 1: pierwotny stan

4 Funkcjonalności programu - poruszanie

Program umożliwia ruchy za pomocą klawiszy:

- W - ruch do góry wzdłuż osi Y
- S - ruch do dołu wzdłuż osi Y
- A - ruch w lewo wzdłuż osi X
- D - ruch w prawo wzdłuż osi X
- Q - ruch do tyłu wzdłuż osi Z
- E - ruch do przodu wzdłuż osi Z

Ruchy zostały zrealizowane za pomocą transformacji 3D, rzutowania translacji. Czyli zostało użyta funkcja transform, która zwraca iloczyn macierzy dwóch tablic.

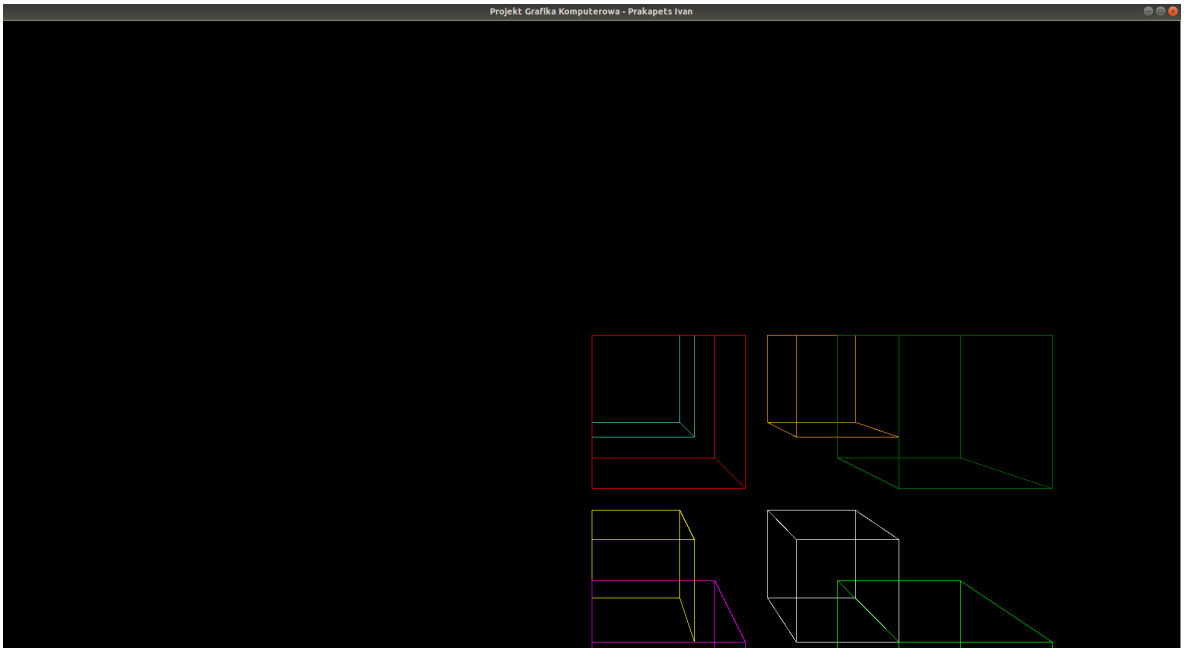
Transform: `[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]`

Oś X: transform[0,3]

Oś Y: transform[1,3]

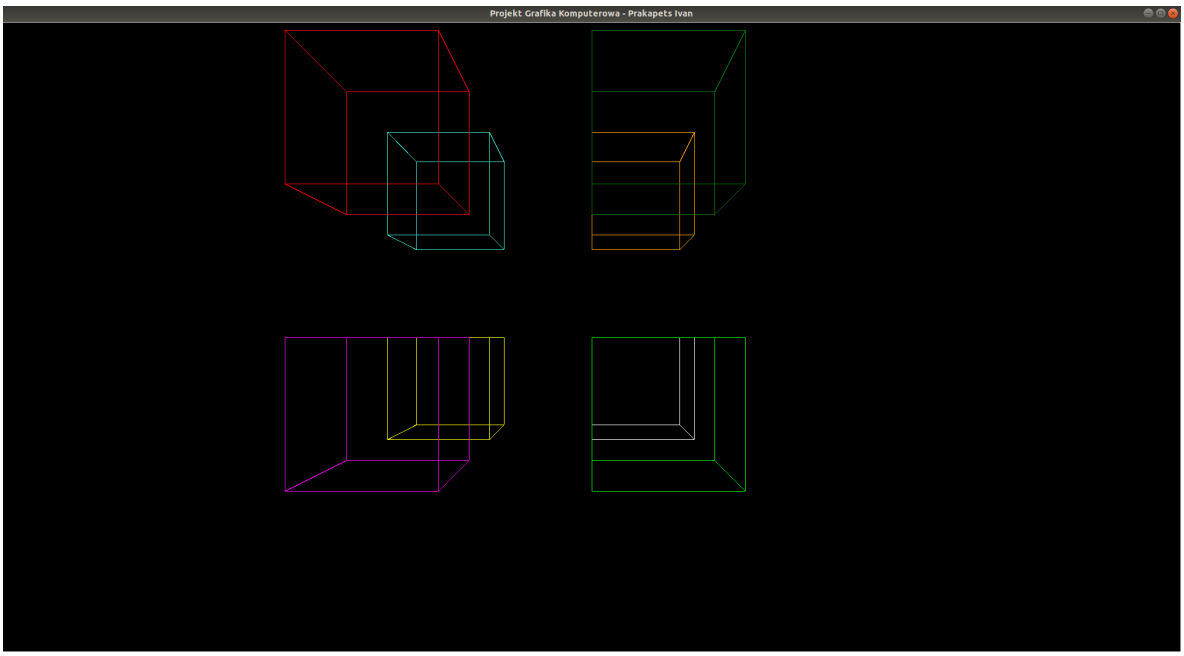
Oś Z: transform[2,3]

Ilustracje oraz wyniki macierzy poniżej:



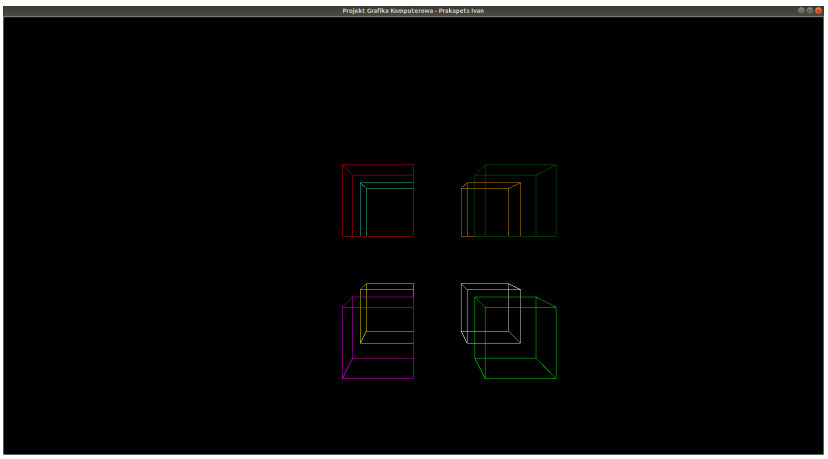
Rysunek 2: ruch do góry+lewo (10 kliknięć)

```
[[1. 0. 0. 0.] ruch do góry
 [0. 1. 0. 1.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

[[1. 0. 0. 1.] ruch w lewo
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

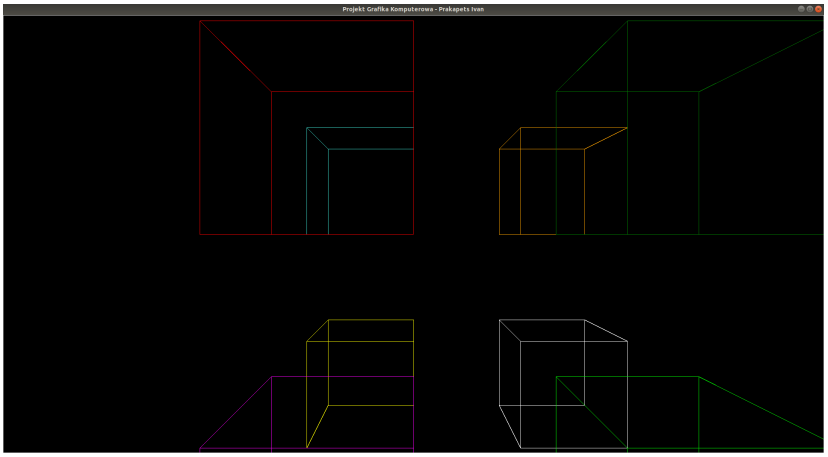


Rysunek 3: ruch do dołu+prawo (10 kliknięć)

```
[[ 1.  0.  0.  0.] ruch do dołu
 [ 0.  1.  0. -1.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]

[[ 1.  0.  0. -1.] ruch w prawo
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
```



Rysunek 4: ruch do tyłu (20 kliknięć)



Rysunek 5: ruch do przodu (20 kliknięć)

```
[[1. 0. 0. 0.] do tyłu
 [0. 1. 0. 0.]
 [0. 0. 1. 1.]
 [0. 0. 0. 1.]]

[[ 1.  0.  0.  0.] do przodu
 [ 0.  1.  0.  0.]
 [ 0.  0.  1. -1.]
 [ 0.  0.  0.  1.]]
```

5 Funkcjonalności programu - obroty

Program umożliwia obroty za pomocą klawiszy:

- 8 - obrót do góry wzdłuż osi X
- 2 - obrót do dołu wzdłuż osi X
- 4 - obrót w lewo wzdłuż osi Y
- 6 - obrót w prawo wzdłuż osi Y
- 7 - obrót zgodnie z ruchem wskazówki zegara
- 9 - obrót odwrotnie z ruchem wskazówki zegara

W przestrzeni 3D możliwe są obroty wokół każdej z trzech osi (X ,Y,Z). Obrót wokół osi X to obrót na płaszczyźnie YZ. Zostało użyta funkcja transform, która zwraca iloczyn macierzy dwóch tablic.

Transform: `[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]`

Oraz angle = `direction * 10 * np.pi / 180.` (Kąt obrotu)

Oś X: transform[1:3, 1:3] =

```
[   [cos(angle), -sin(angle)],
    [sin(angle),  cos(angle)]   ]
```

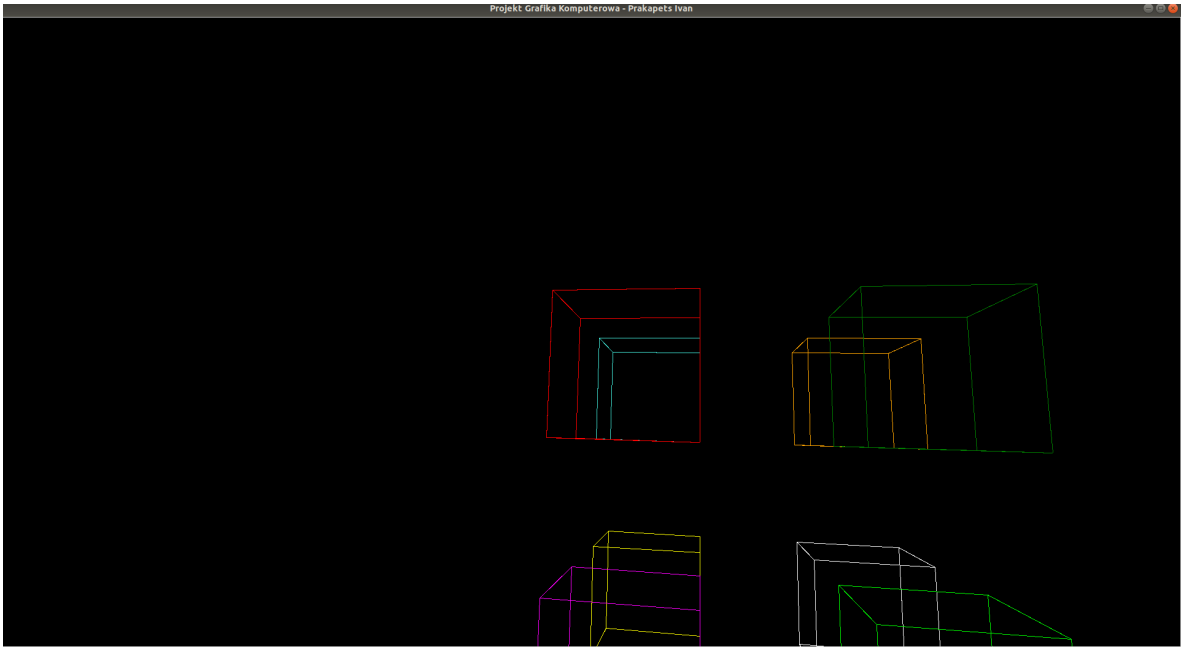
Oś Y: transform[0:3,0:3] =

```
[   [cos(angle),  0, sin(angle)],
    [     0,      1,          0],
    [-sin(angle), 0,cos(angle) ]   ]
```

Oś Z: transform[0:2, 0:2]=

```
[   [cos(angle), -sin(angle)],
    [sin(angle),  cos(angle)]   ]
```

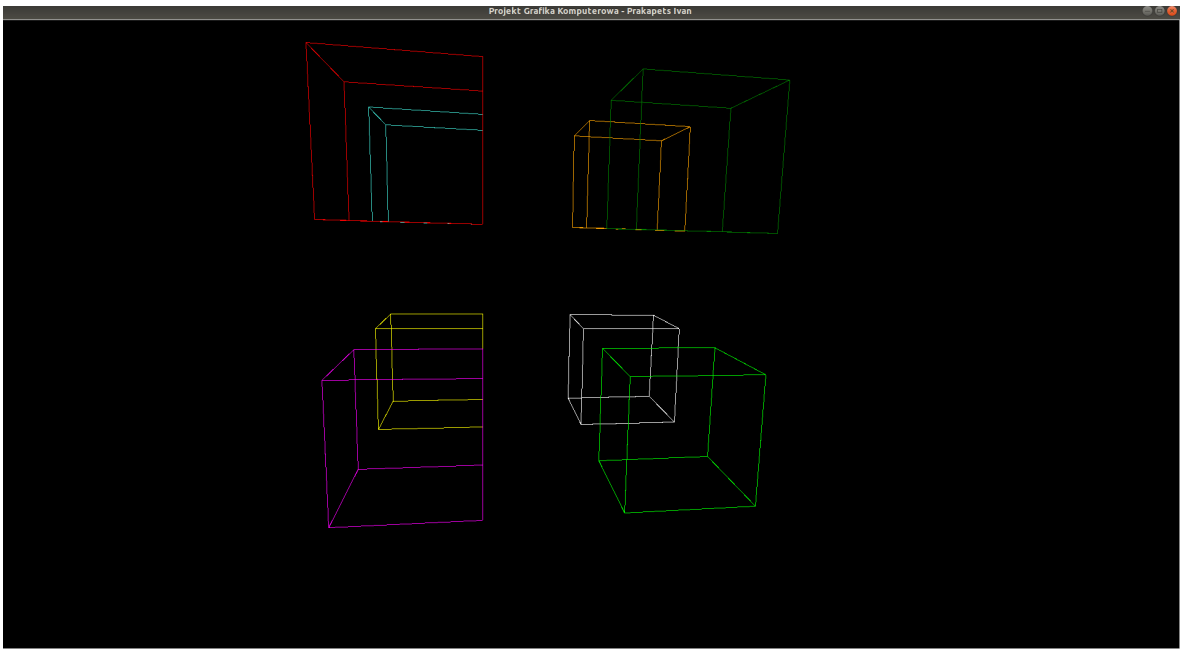
Ilustracje oraz wyniki macierzy poniżej:



Rysunek 6: obrót do góry+lewo (1 kliknięcie)

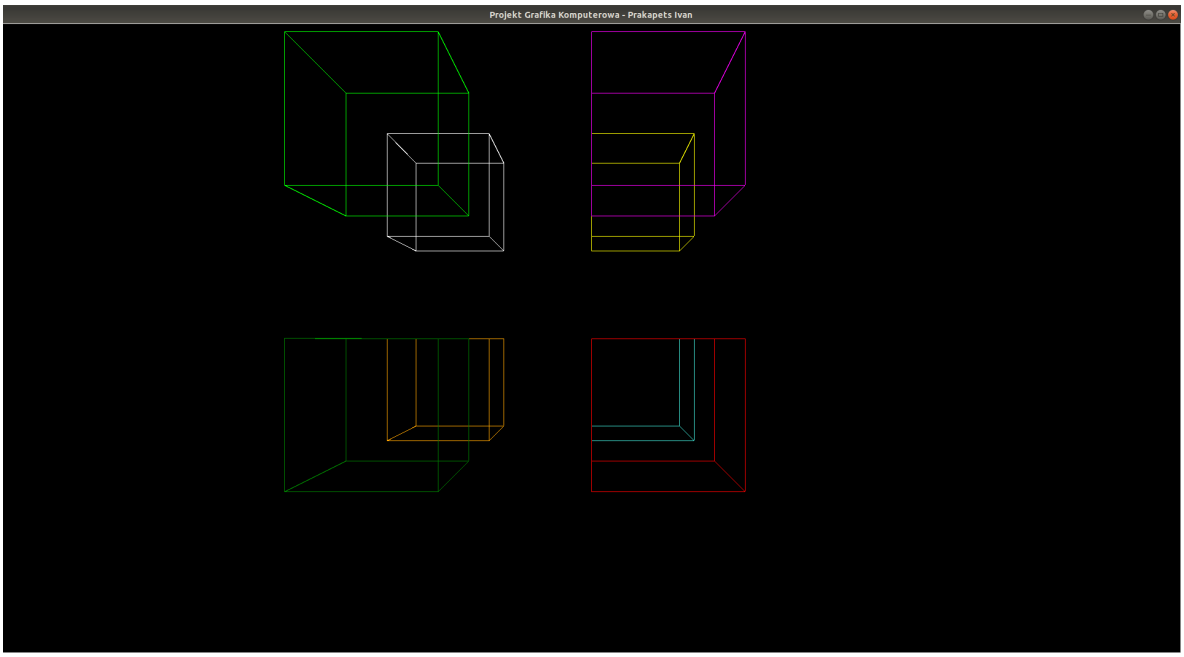
```
[[ 1.          0.          0.          0.        ] obroty do góry
 [ 0.          0.98480775  0.17364818  0.        ]
 [ 0.         -0.17364818  0.98480775  0.        ]
 [ 0.          0.          0.          1.        ]]

[[ 0.98480775  0.          0.17364818  0.        ] obroty w lewo
 [ 0.          1.          0.          0.        ]
 [-0.17364818  0.          0.98480775  0.        ]
 [ 0.          0.          0.          1.        ]]


[[ 1.          0.          0.          0.        ] obroty do dołu
 [ 0.          0.98480775 -0.17364818  0.        ]
 [ 0.          0.17364818  0.98480775  0.        ]
 [ 0.          0.          0.          1.        ]]

[[ 0.98480775  0.         -0.17364818  0.        ] obroty w prawo
 [ 0.          1.          0.          0.        ]
 [ 0.17364818  0.          0.98480775  0.        ]
 [ 0.          0.          0.          1.        ]]
```

Poniżej są obroty dołu oraz na prawo

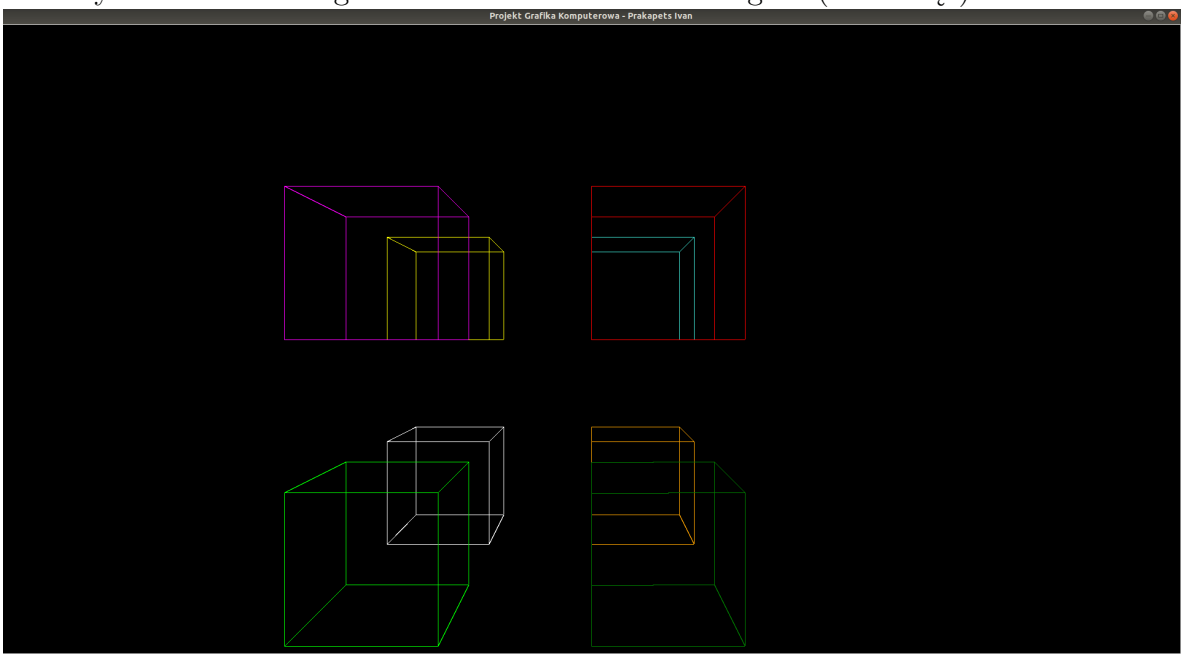


Rysunek 7: obrót do dołu+prawo (1 kliknięcie)

```
obrót zgodnie z ruchem wskazówki zegara
[[ 0.98480775  0.17364818  0.          0.        ]
 [-0.17364818  0.98480775  0.          0.        ]
 [ 0.          0.          1.          0.        ]
 [ 0.          0.          0.          1.        ]]

obrót odwrotnie z ruchem wskazówki zegara
[[ 0.98480775 -0.17364818  0.          0.        ]
 [ 0.17364818  0.98480775  0.          0.        ]
 [ 0.          0.          1.          0.        ]
 [ 0.          0.          0.          1.        ]]
```



Rysunek 8: obrót zgodnie z ruchem wskazówki zegara (9 kliknięć)



Rysunek 9: obrót odwrotnie z ruchem wskazówki zegara (9 kliknięć)

6 Funkcjonalności programu - ZOOM

Program umożliwia powiększania obrazu oraz zmniejszanie za pomocą klawiszy:

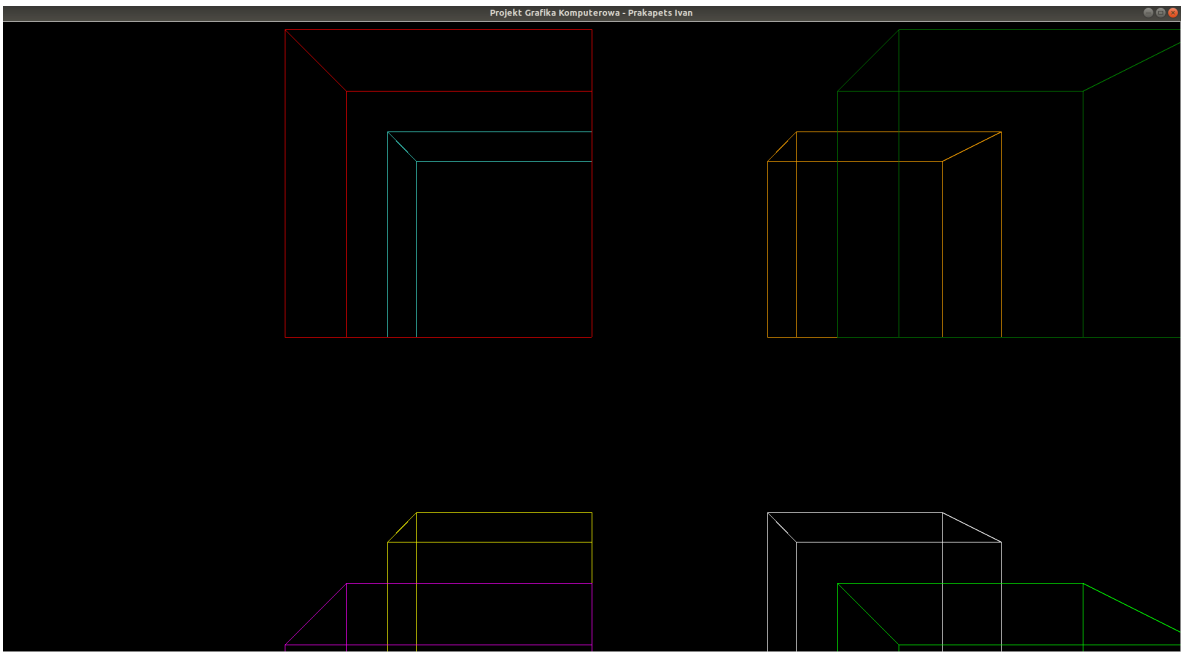
- + powiększanie
- − zmniejszanie

Zoom bez zmiany położenia obserwatora: nie zmienia wzajemnych relacji między punktami obiektu i obserwatorem. Napisałem funkcję distance, która określa położenia punktów. Za pomocą formuły:

$$distance = \sqrt{(point1[0] - point2[0])^2 + (point1[1] - point2[1])^2 + (point1[2] - point2[2])^2}$$

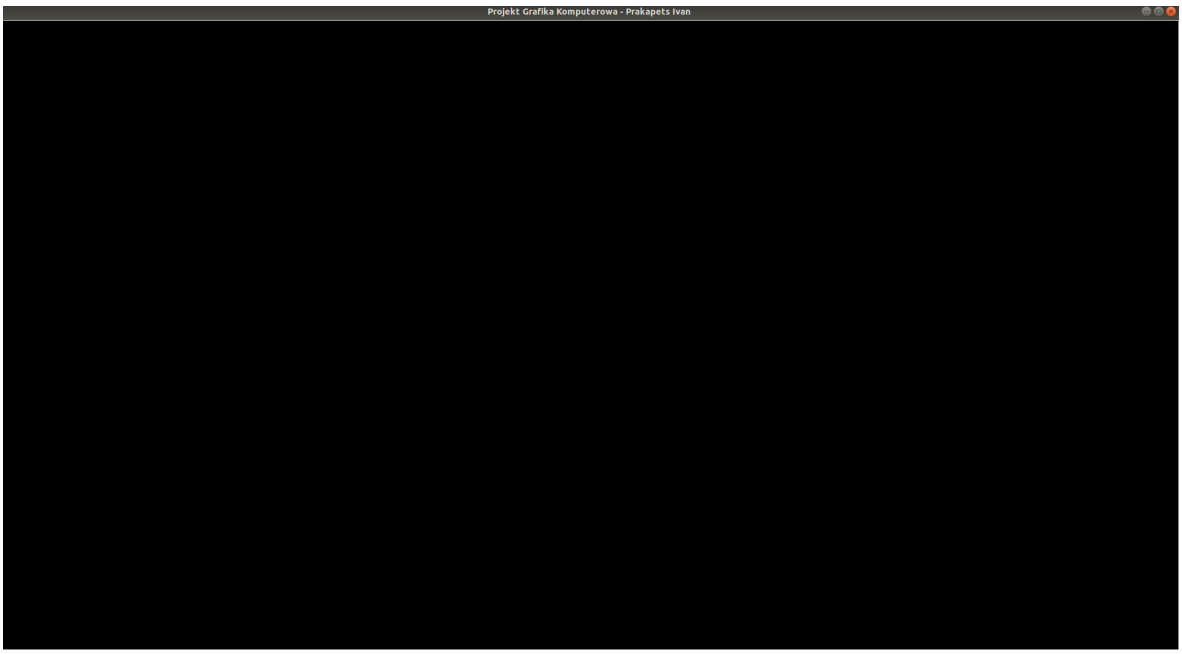
Pierwotna wartość to 1000.

To są wyniki `distance` dla powiększania:



Rysunek 10: powiększanie (20 kliknięć)

```
Pierwotna wartość: 1000
1050 1100 1150 1200 1250 1300 1350 1400 1450 1500
1550 1600 1650 1700 1750 1800 1850 1900 1950 2000
```



Rysunek 11: zmniejszanie (20 kliknięć)

Powyżej można zaobserwować, że obrazku w ogóle niema (czyli znikł) bo bardzo zmniejszyliśmy na wartość 0. To są wyniki `distance` dla zmniejszania:

```
Pierwotna wartość: 1000
950 900 850 800 750 700 650 600 550 500
450 400 350 300 250 200 150 100 50 0
```

7 Wnioski

Wszystkie wymagania zostały zrealizowane, program działa w nieskończonej pętli, czeka póki użytkownik naciśnie odpowiedni przycisk. Program przelicza współrzędne za pomocą macierzy wraz z kątem obrotu i wyświetla nowe położenie obiektów. Widać wszystkie krawędzi prostopadłościanów. Jestem zadowolony swojej pracą, i mam nadzieję, że wszystkie wymagania zostały zrealizowane poprawnie i tak jak powinni być.

8 Różne rzuty ekranu

Poniżej są przykładowe różne rzuty ekranu.