Autor: **Krzysztof Dąbrowski 293101** 30 marca 2020

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](https://github.com/SiwyKrzysiek/perspective-camera) 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 x 4]**, gdzie n to liczba punktów danej figury oraz listę krawędzi. Kolejne kolumny w macierzy wieszołków 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](https://open.gl/transformations).

## 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:

A close up of a clock

Description automatically generated

## 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:

A picture containing clock, meter

Description automatically generated

Do rotacji w osi y została wykorzystana poniższa macierz:

A picture containing clock, meter

Description automatically generated

Do rotacji w osi z została wykorzystana poniższa macierz:

A picture containing clock, meter

Description automatically generated

# 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\_heigh: 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\_heigh / 2 - from\_focal \* point\_3d[2]

return x, y

# Przykłady działania programu

[Nagranie prezentujące działanie programu](https://youtu.be/ZblzQJ34KEE). 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.

A close up of a logo

Description automatically generatedA picture containing star

Description automatically generatedA picture containing star

Description automatically generatedA close up of a logo

Description automatically generated

# Obsługa programu

Do uruchomienia programu potrzeby 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 współrzędnie 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 dopowiadają klawisze -, =.

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

Za rotacje 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.