Grafika komputerowa
Projekt "Kamera Wirtualna"

Natalia Olszewska

Numer indeksu: 311368

Politechnika Warszawska

Wydział Elektryczny

Informatyka Stosowana

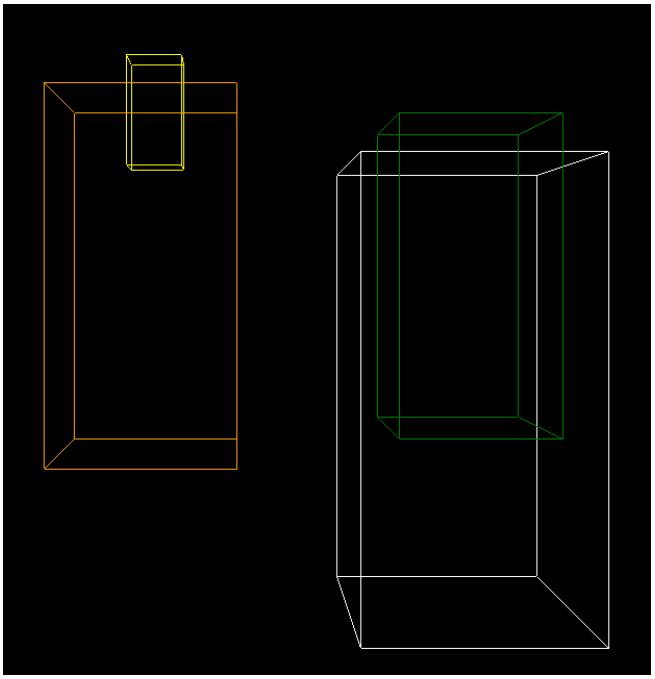
24.06.2023

Spis treści

1 Wstęp

Celem projektu "Kamera wirtualna" jest symulowanie działania kamery wirtualnej w przestrzeni trójwymiarowej. Głównym zadaniem projektu jest stworzenie systemu, który będzie w stanie renderować scenę trójwymiarową z perspektywy.

W ramach projektu zostanie stworzony kod w języku Python, który będzie symulował działanie kamery wirtualnej dla zdefiniowanej sceny, zawierającej 4 prostopadłościany. Implementacja będzie obejmować funkcje do przekształceń punktów oraz renderowania widocznych obiektów. Dodatkowo, projekt uwzględni możliwość obrotu, transformacji i skalowania sześcianów w przestrzeni.



Rysunek 1: Scena z 4 prostopadłościanów

2 Podstawy teoretyczne projektu

Projekt "Kamera wirtualna" opiera się na kilku podstawowych teoretycznych koncepcjach z dziedziny grafiki komputerowej i przetwarzania obrazu. Poniżej przedstawiam opis tych podstawowych pojęć:

1. Kamera wirtualna: Kamera wirtualna jest abstrakcyjnym modelem, który symuluje działanie prawdziwej kamery w przestrzeni trójwymiarowej.

Kamera definiuje perspektywę widoku, kąt widzenia, pozycję oraz orientację w przestrzeni. Dzięki temu możemy renderować sceny trójwymiarowe i tworzyć wizualne efekty.

2. Układ współrzędnych: Układ współrzędnych to system referencyjny, który służy do określania położenia i orientacji obiektów w przestrzeni. Najczęściej stosowanym układem współrzędnych w grafice komputerowej jest układ kartezjański, który składa się z trzech osi: x, y i z. Pozycje obiektów są określane jako kombinacje wartości na tych trzech osiach.

3. Renderowanie: Renderowanie to proces generowania obrazu trójwymiarowego na podstawie sceny, kamery i źródeł światła. W procesie renderowania obliczane są kolory i intensywności światła dla poszczególnych pikseli obrazu. Algorytmy renderowania wykorzystują informacje o geometrii, materiałach, teksturach oraz oświetleniu, aby generować realistyczne obrazy.

4. Rzutowanie: Rzutowanie jest procesem przekształcania obiektów trójwymiarowych na dwuwymiarowy obraz na płaszczyźnie ekranu. Rzutowanie perspektywiczne jest najczęściej stosowanym rodzajem rzutowania w grafice komputerowej, gdzie obiekty są rzutowane na płaszczyznę ekranu z uwzględnieniem perspektywy i oddalenia. Rzutowanie perspektywiczne odwzorowuje efekt głębi i perspektywy w renderowanych obrazach.

5. Przekształcenia przestrzenne: Przekształcenia przestrzenne obejmują operacje takie jak obroty, translacje i skalowanie obiektów w przestrzeni trójwymiarowej. Przekształcenia te pozwalają na manipulację obiektami, zmianę ich pozycji, orientacji i skali. W projekcie wykorzystuje się te przekształcenia do symulacji ruchu kamery oraz manipulacji sześcianami.

6. Obrót: Obrót jest przekształceniem przestrzennym polegającym na zmianie orientacji obiektu wokół określonej osi. Obrót może być wykonywany wokół osi x, y i/lub z. Macierze obracające są używane do wykonywania operacji obrotu na punktach w przestrzeni trójwymiarowej. Macierze te określają kąt i oś obrótu oraz są stosowane do transformacji punktów.

7. Translacja: Translacja jest przekształceniem przestrzennym polegającym na przesunięciu obiektu wzdłuż osi x, y i/lub z. Przesunięcie obiektu o wektor przesunięcia zmienia jego położenie w przestrzeni. Translacja jest wykorzystywana do manipulacji obiektami w trójwymiarowej scenie, w tym do przemieszczania kamery wirtualnej.

8. Zoom: Zoom, zwany również skalowaniem, jest przekształceniem przestrzennym, które zmienia skalę obiektu. Może to oznaczać zarówno powiększanie, jak i pomniejszanie obiektu wzdłuż osi x, y i/lub z. Skalowanie jest wykorzystywane do zmiany perspektywy widoku oraz do dostosowywania rozmiarów obiektów w scenie.

Te podstawowe pojęcia teoretyczne umożliwiają manipulację obiektami w przestrzeni trójwymiarowej, pozwalają na precyzyjne określanie ich położenia, orientacji i skali oraz na wizualizację całej sceny.

3 Budowa programu

Plik ‘main.py‘ składa się z kilku elementów. Na początku importowane są niezbędne moduły, takie jak ‘numpy‘, ‘tkinter‘ i ‘collections‘. Następnie definiowana jest klasa ‘Scene‘, która zarządza sceną trójwymiarową. Klasa ta posiada różne metody, takie jak ‘load_scene()‘ do wczytywania sceny, ‘move()‘ do przesuwania obiektów, ‘zoom()‘ do zmiany odległości od kamery, ‘turn()‘ do obracania obiektów, ‘distance_from_camera()‘ do obliczania odległości obiektów od kamery, oraz ‘render()‘ do renderowania sceny na płótnie.

Następnie tworzony jest obiekt ‘Scene‘ i inicjalizowane są różne ustawienia początkowe. Plik ‘main.py‘ również definiuje funkcję ‘key()‘, która obsługuje interakcje z klawiatury. W zależności od wciśniętego klawisza, funkcja wykonuje odpowiednie operacje na scenie, takie jak przesuwanie, obracanie czy zoomowanie. Na końcu pliku znajduje się kod odpowiedzialny za uruchomienie interfejsu użytkownika i głównej pętli programu.

Plik ‘geometry.py‘ zawiera definicje dwóch klas: ‘Point‘ i ‘Cube‘. Klasa ‘Point‘ reprezentuje punkt w przestrzeni trójwymiarowej i udostępnia różne metody do manipulacji tym punktem, takie jak przesunięcie, transformacja macierzowa, projekcja na płaszczyznę dwuwymiarową oraz obliczanie odległości między punktami.

Klasa ‘Cube‘ reprezentuje prostopadłościan w przestrzeni trójwymiarowej i składa się z punktów oraz krawędzi. Udostępnia metody do tworzenia sześcianu na podstawie podanych parametrów, takich jak pozycja, rozmiar i kolor. Klasa ‘Cube‘ wykorzystuje również klasę ‘Point‘ do manipulacji punktami prostopadłościanu.

Oba pliki, ‘main.py‘ i ‘geometry.py‘, współpracują ze sobą w celu tworzenia, manipulacji i renderowania obiektów w przestrzeni trójwymiarowej. Plik ‘main.py‘ odpowiada za interakcję z użytkownikiem i renderowanie sceny na płótnie, natomiast plik ‘geometry.py‘ dostarcza niezbędne struktury danych i operacje geometryczne.

Cały program działa w następujący sposób:

1. Wczytuje scenę za pomocą metody load_scene().
2. Tworzy obiekt sceny (Scene) i inicjalizuje go wczytanymi danymi.
3. Tworzy okno główne i płótno za pomocą modułu tkinter.
4. Renderuje scenę na płótnie przy użyciu metody render().
5. Oczekuje na interakcję użytkownika z klawiatury.
6. Obsługuje odpowiednie zdarzenia (np. przesunięcie, zoomowanie, obracanie) i aktualizuje scenę.

7. Ponownie renderuje scenę na płótnie, odzwierciedlając dokonane zmiany.
8. Powtarza kroki 5-7, aż do zakończenia programu.

W ten sposób program umożliwia manipulację obiektami w przestrzeni trójwymiarowej, określanie ich położenia, orientacji i skali oraz renderowanie całej sceny na płótnie.

4 Funkcjonalność programu

Program posiada szereg funkcjonalności, które umożliwiają interakcję użytkownika z obiektami w przestrzeni trójwymiarowej oraz renderowanie sceny. Oto opis wszystkich funkcjonalności programu:

1. Przesuwanie kamery wirtualnej: Użytkownik może przesuwać kamerę wirtualną wzdłuż trzech osi (x, y, z) w przestrzeni trójwymiarowej.
2. Obracanie kamery wirtualnej: Użytkownik może obracać kamerę wirtualną wokół trzech osi (x, y, z) w przestrzeni trójwymiarowej.
3. Skalowanie obiektów: Użytkownik może zmieniać skalę obiektów w scenie, powiększając lub pomniejszając je.
4. Renderowanie sceny: Program renderuje scenę w czasie rzeczywistym i wyświetla ją na ekranie.
5. Interakcja użytkownika: Program umożliwia interakcję użytkownika poprzez klawisze. Program pozwala na poruszanie się wokół obiektów sceny obracając kamerę i skalując obiekty.

To są główne funkcjonalności programu, które umożliwiają użytkownikowi manipulację obiektami w przestrzeni trójwymiarowej oraz renderowanie dynamicznej sceny.

Sterowanie

Przesuwanie kamery wirtualnej:

- Przemieszczenie kamery w górę (wciśnięcie klawisza "w")
- Przemieszczenie kamery w dół (wciśnięcie klawisza "s")
- Przemieszczenie kamery w lewo (wciśnięcie klawisza "a")
- Przemieszczenie kamery w prawo (wciśnięcie klawisza "d")
- Przemieszczenie kamery do tyłu (wciśnięcie klawisza "q")
- Przemieszczenie kamery do przodu (wciśnięcie klawisza "e")

Obracanie kamery wirtualnej:

- Obrót kamery wzdłuż osi x przeciwnie do ruchu wskazówek zegara (wciśnięcie klawisza "8")
- Obrót kamery wzdłuż osi x zgodnie z ruchem wskazówek zegara (wciśnięcie klawisza "2")
- Obrót kamery wzdłuż osi y zgodnie z ruchem wskazówek zegara (wciśnięcie klawisza "4")
- Obrót kamery wzdłuż osi y przeciwnie do ruchu wskazówek zegara (wciśnięcie klawisza "6")
- Obrót kamery wzdłuż osi z przeciwnie do ruchu wskazówek zegara (wciśnięcie klawisza "7")
- Obrót kamery wzdłuż osi z zgodnie z ruchem wskazówek zegara (wciśnięcie klawisza "9")

Zoom (zmiana skali obiektów):

- Powiększenie obiektów (wciśnięcie klawisza "+")
- Pomniejszenie obiektów (wciśnięcie klawisza "-")

Przy wciśnięciu klawisza "Enter" scena wraca do pozycji początkowej.

Kiedy użytkownik wciśnie odpowiedni klawisz, zostanie wywołana odpowiednia metoda sceny (move, turn, zoom) w zależności od klawisza naciśniętego przez użytkownika. Na przykład, wciśnięcie klawisza "w" spowoduje wywołanie metody move sceny, przekazując jako argumenty "y" (oś) i 1 (wartość przesunięcia w górę).

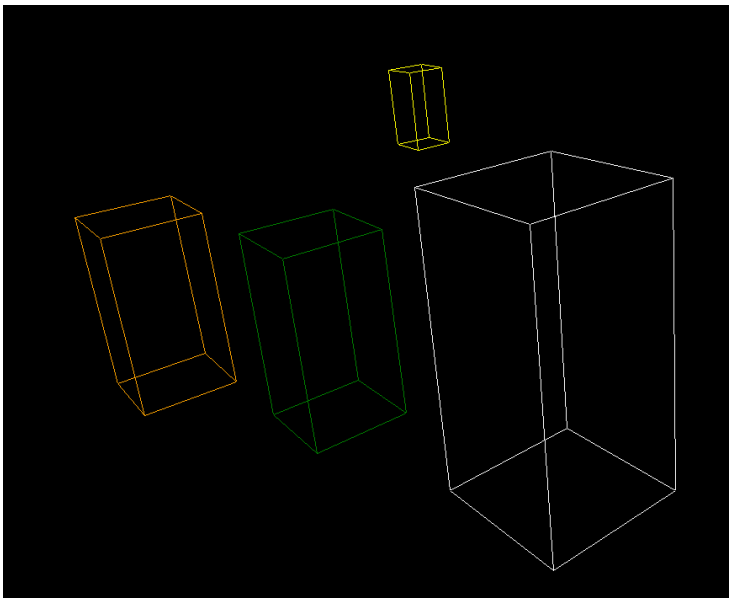
Dzięki tym funkcjonalnościom użytkownik będzie mógł kontrolować ruch, obrót i skalę kamery oraz interaktywnie eksplorować scenę trójwymiarową.

5 Podsumowanie

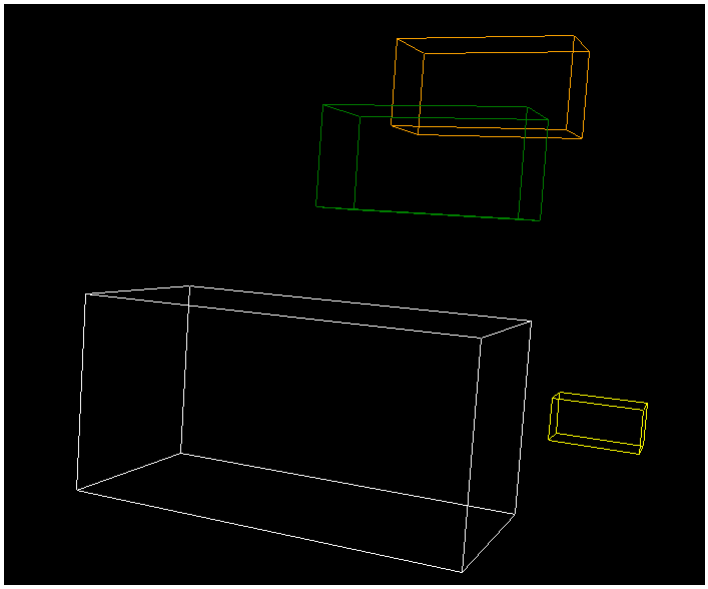
Podsumowując, program umożliwia manipulację obiektami w przestrzeni trójwymiarowej, takie jak przesuwanie, obracanie i przybliżanie. Użytkownik może wczytywać sceny, renderować je w czasie rzeczywistym i interaktywnie eksplorować. Dzięki prostemu interfejsowi i wykorzystaniu biblioteki Tkinter, program zapewnia użytkownikowi przyjemne doświadczenie w trójwymiarowej wizualizacji.

5.1 Screeny z eksploracji

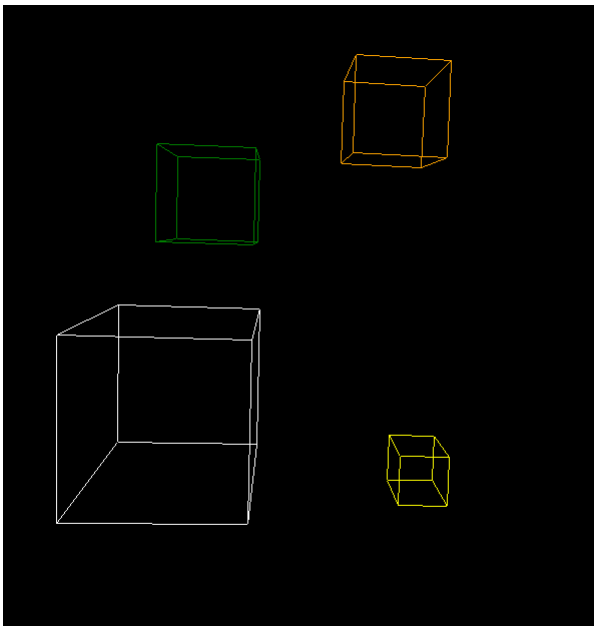
Przykładowe widoki podczas używania programu:



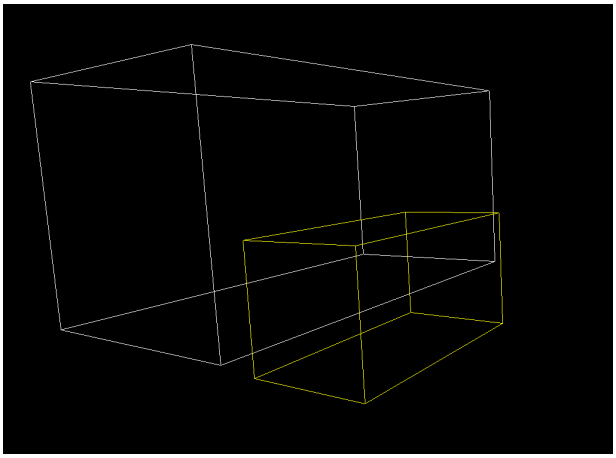
Rysunek 2: Przykład 1



Rysunek 3: Przykład 2



Rysunek 4: Przykład 3



Rysunek 5: Przykład 4

5.2 Wnioski

Wnioskiem z tego projektu jest to, że tworzenie interaktywnych trójwymiarowych aplikacji graficznych jest możliwe przy użyciu odpowiednich bibliotek i narzędzi programistycznych. Program umożliwia manipulację obiektami w trójwymiarowej przestrzeni, co pozwala na tworzenie różnorodnych interaktywnych

wizualizacji i prezentacji. Projekt ten pokazuje, jak za pomocą prostych transformacji geometrycznych można zmieniać położenie, orientację i skalę obiektów, a następnie renderować sceny w czasie rzeczywistym. Dzięki takim rozwiązaniom można tworzyć interaktywne aplikacje graficzne o dużej użyteczności i estetyce, które mogą być wykorzystane w wielu dziedzinach, takich jak architektura, projektowanie gier, edukacja czy wizualizacja danych.

Praca z trójwymiarowymi obiektami i przestrzenią wymaga zrozumienia podstawowych pojęć geometrycznych. Wiedza ta jest niezbędna do manipulacji obiektami w przestrzeni trójwymiarowej i renderowania ich na ekranie. Transformacje geometryczne, takie jak translacja, rotacja i skalowanie, są podstawowymi narzędziami do manipulacji obiektami w trójwymiarowej przestrzeni. Pozwalają one na zmianę położenia, orientacji i skali obiektów, co daje możliwość animacji i interakcji w grafice komputerowej.

Biblioteki graficzne i narzędzia programistyczne, takie jak numpy, tkinter i inne, są niezwykle przydatne przy implementacji grafiki komputerowej. Umożliwiają one łatwą manipulację danymi, tworzenie interfejsu użytkownika i renderowanie grafiki w sposób efektywny i wydajny. Wnioski te podkreślają wagę zrozumienia podstawowych pojęć i technik grafiki komputerowej oraz wykorzystanie odpowiednich narzędzi i bibliotek programistycznych. Pozwalają one na tworzenie zaawansowanych aplikacji graficznych, wizualizacji danych i interaktywnych doświadczeń użytkownika w dziedzinie grafiki komputerowej.

Bibliografia

[1] `https://developer.nvidia.com/`

[2] `https://realpython.com/`

[3] `https://www.pygame.org/docs/`

[4] `https://realpython.com/tutorials/graphics/`

[5] `https://computergraphics.stackexchange.com/`

[6] `https://numpy.org/`