Sprawozdanie projekt programowanie obiektowe i grafika komputerowa

Jakub Zakłada 198201 ACIR 4A

Kornel Wasilewski 197904 ACIR 4A

1. **Środowisko programowania i użyte biblioteki**

Język programowania: C++

Użyte biblioteki:

#include<iostream> - standardowa biblioteka C++

#include<fstream> - zapis, odczyt plików np. recording.csv

#include<glad/glad.h> - ładowanie funkcji OpenGL

#include<GLFW/glfw3.h> - tworzenie okna, obsługa klawiatury, myszy, renderowanie

#include<stb/stb\_image.h> - ładowanie tekstur

#include<glm/glm.hpp> - biblioteka matematyczna do 3D

#include<glm/gtc/matrix\_transform.hpp>

#include<glm/gtc/type\_ptr.hpp>

#include <cmath> - funkcje matematyczne

Własne klasy:

#include"EBO.h"

#include"shaderClass.h"

#include"VAO.h"

#include"VBO.h"

#include"Texture.h"

#include"Mesh.h"

#include"Model.h"

#include"Camera.h"

1. **Wykorzystanie obiektowości**

-Klasy do reprezentacji modeli 3D

Model, Mesh, Node – reprezentują hierarchiczne modele 3D (np. ramię robota).Każdy model składa się z siatek (Mesh), a te z wierzchołków, indeksów, tekstur

- Shader – klasa do obsługi shaderów

- Camera - klasa reprezentująca kamere 3D z możliwością zoomowania

- Bufory graficzne: VBO, VAO, EBO - dzięki nim kod w main() jest czytelniejszy, operujemy obiektami

- wykorzystanie struktur danych np. std::vector -przechowywania wierzchołków (Vertex)

1. **Korzyści zastosowania obiektowości**

-czytelność – każda klasa odpowiada za jeden typ danych

-możliwość wielokrotnego użycia – np. klasa Model może wczytać dowolny plik .glb, nie tylko ramię robota

-łatwe rozszerzanie – np. dodanie nowych typów obiektów czy logiki (fizyka, AI) nie wymaga przepisywania całości

-łatwa możliwość naprawy błędów poprzez debugowanie , ponieważ błędy są izolowane do konkretnej klasy np. Shader::Activate()

**4. Realizacja założeń projektu**

**4.1 Model robota i obiektu**

W ramach projektu został przygotowany własny model robota typu Puma 560, który został zamodelowany ręcznie w programie Blender. Robot został podzielony na logiczne segmenty (np. Base, Base\_rotator, Arm1, Arm2, Arm3, Grabber) i zapisany w formacie .glb, wspierającym hierarchię węzłów oraz transformacje.

Również obiekt manipulowany przez robota ( kulka ) został stworzony od podstaw w Blenderze jako osobny model 3D. Kulka otrzymała prostą geometrię sfery oraz została dodatkowo pokryta kolorem poprzez przypisanie kolorów do wierzchołków (Vertex Colors)

**4.2 Ruch robota**

Robot porusza się w 3 stopniach swobody oraz może zamykać i otwierać chwytak. Każda oś ruchu ma zdefiniowane granice (np. Arm3: −30° do 90°), co zapobiega nieprawidłowym pozycjom fizycznym i kolizjom między segmentami.

Ruch może być wykonywany:

ręcznie przez użytkownika za pomocą klawiszy (np. 1, 2, 3, 4, Q, E),

automatycznie – na podstawie wcześniej nagranej sekwencji (nagrywanie R, odtwarzanie P)

**4.3 Funkcjonalność**

Robot może chwycić kulkę, płynnie przenieść ją w dowolne miejsce w swoim obszarze roboczym i kontrolowanie ją uwolnić. Po zwolnieniu kulka podlega grawitacji i spada na powierzchnię. W przypadku upadku poza obszar roboczy robota pod przyciskiem **B** dodaliśmy możliwość zresetowania pozycji kulki do pozycji początkowej, a samo łapanie i opuszczanie kulki uzależnione jest od odległości ząbków chwytaka od środka kulki

**4.4 Tryb uczenia się**

Projekt oferuje prosty tryb „uczenia się” robota poprzez nagrywanie ruchu. Użytkownik, za pomocą klawiatury, steruje segmentami robota w czasie rzeczywistym i może nagrać całą sekwencję jego ruchów. Wciśnięcie klawisza R rozpoczyna nagrywanie, a T je zatrzymuje. Każda klatka zapisywana jest z dokładnymi parametrami kątów obrotu oraz pozycją chwytaka w czasie. Po zapisaniu takiej sekwencji użytkownik może ją odtworzyć (P), a robot dokładnie odwzoruje wcześniej wykonane ruchy

**4.5 Kamera**

W projekcie zastosowano kamerę typu free-look (FPS-style), która umożliwia swobodne przemieszczanie się po scenie 3D i obserwowanie działania robota z różnych perspektyw.

Kamera została zaimplementowana jako oddzielna klasa (Camera) i posiada następujące możliwości:

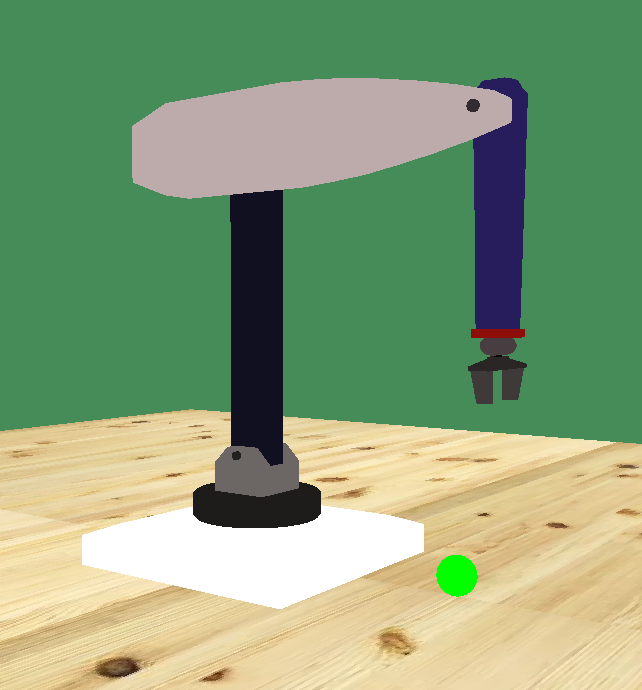
- obrót myszką – użytkownik może rozglądać się w dowolnym kierunku

- zoom (scroll) – przybliżenie i oddalenie obrazu

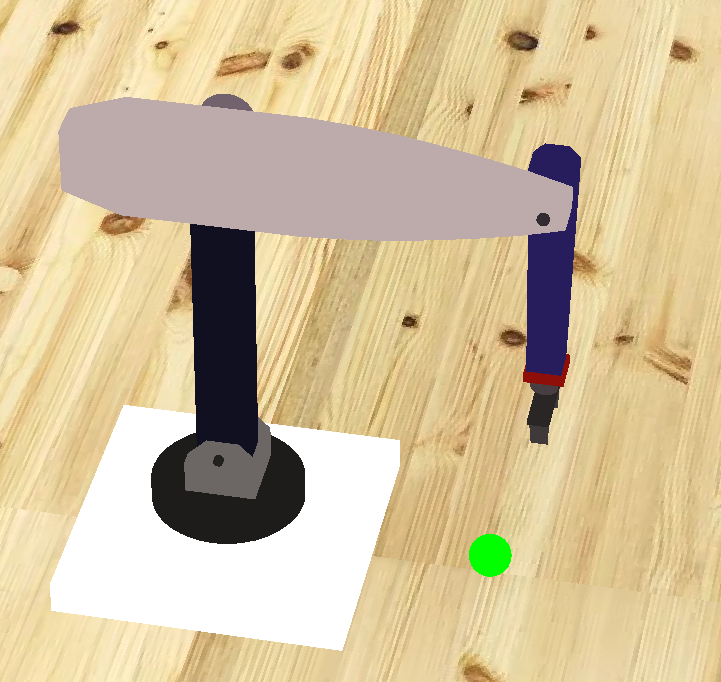
-ruch klawiaturą (W, A, S, D) – umożliwia przemieszczanie się kamery do przodu, do tyłu oraz na boki

1. **Wygląd projektu**

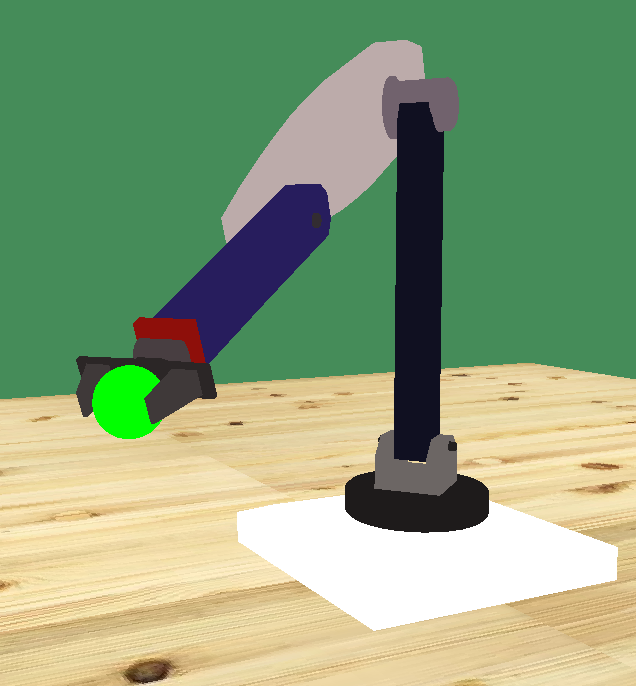
- Wygląd modelu robota, obiektu , podłogi oraz otoczenia



- działanie kamery



-łapanie oraz przenoszenie kulki



-upuszczanie oraz grawitacja kulki

