<table>
<tr><td>Politechnika Wrocławska</td><td>Autor:
Kyrylo Semenchenko
nr.indeksu: 273004</td><td>Wydział: Informatyki i telekomunikacji
Rok: 2024
Rok akadem.: III</td></tr>
<tr><td colspan="3">Grafika komputerowa i komunikacja człowiek-komputer</td></tr>
<tr><td>Data ćwiczenia:
25.10.2024</td><td rowspan="2">Temat ćwiczenia laboratoryjnego:
Modelowanie obiektów 3D</td><td rowspan="2">Prowadzący
Dr inż. Arch.
Tomasz Zamojski</td></tr>
<tr><td>Nr ćwiczenia:
3</td></tr>
</table>

Spis treści

1 Wstęp

Celem ćwiczenia było:

1. Zapoznanie się z modelowaniem obiektów 3D przy pomocy prymitywów graficznych.
2. Nabranie wprawy w definiowaniu brył przy pomocy wierzchołków.
3. Poznanie zasady działania mechanizmu bufora głębi.

2 Wykonane zadania

2.1 Zadanie 1 - Modelowanie jajka przy użyciu punktów (ocena 3.0)

2.1.1 Opis Programu

Program przedstawia implementację rysowania trójwymiarowego obiektu w kształcie jajka z wykorzystaniem biblioteki OpenGL oraz GLFW. Poniżej omówiono kluczowe elementy działania programu:

2.1.2 Generowanie punktów obiektu

Funkcja `egg_points(N)` generuje punkty opisujące kształt jajka w przestrzeni trójwymiarowej. Wykorzystano do tego parametryzację matematyczną, opartą na zmiennych u i v, które reprezentują współrzędne parametryczne w zakresie $[0, 1]$. Wynikiem są tablice współrzędnych x, y, z, obliczane dla $N \times N$ punktów.

2.1.3 Rysowanie obiektu

Obiekt jajka jest rysowany jako zbiór punktów w funkcji `draw_egg(N)`. Współrzędne punktów są przekazywane do OpenGL za pomocą funkcji `glVertex3f()`, a kolor punktów ustawiono na biały (`glColor3f(1.0, 1.0, 1.0)`).

2.1.4 Obrót sceny

W celu wizualizacji obiektu z różnych perspektyw, zastosowano funkcję `spin(angle)`, która obraca obiekt wokół trzech osi (X, Y, Z). Obrót realizowany jest za pomocą funkcji OpenGL `glRotatef()`.

2.1.5 Rysowanie osi układu współrzędnych

Funkcja `axes()` rysuje osie układu współrzędnych w trzech podstawowych kolorach: czerwonym (X), zielonym (Y) i niebieskim (Z). Ułatwia to orientację w przestrzeni trójwymiarowej.

2.1.6 Mechanizm widoku i projekcji

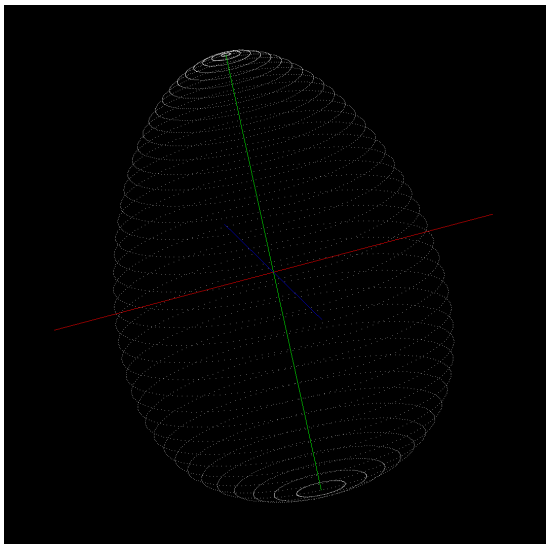
Funkcja `update_viewport()` odpowiada za konfigurację widoku i projekcji. Ustawia proporcje obrazu w zależności od rozmiarów okna oraz definiuje przestrzeń wyświetlania za pomocą projekcji ortogonalnej (`glOrtho()`).

2.1.7 Podsumowanie działania

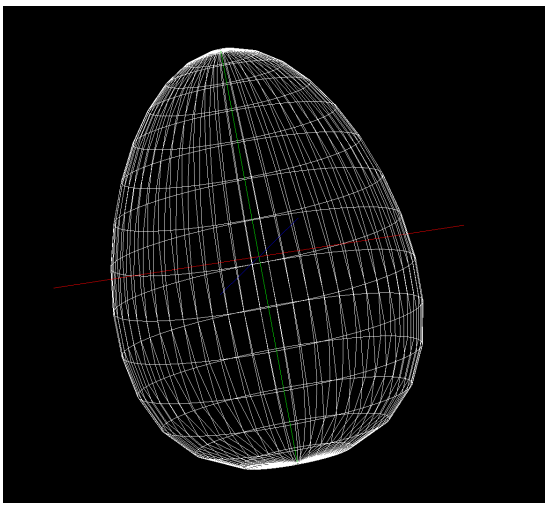
Program startuje przy pomocy funkcji `main()`, która inicjalizuje bibliotekę GLFW, tworzy okno renderowania i zarządza główną pętlą programu. W każdej iteracji pętli obiekt jest renderowany z aktualnym obrotem, a wynik wyświetlany jest w oknie aplikacji.

Program demonstruje podstawowe techniki rysowania trójwymiarowych obiektów oraz zarządzania sceną w OpenGL, jednocześnie wykorzystując matematyczne podejście do definiowania niestandardowych kształtów.

Wynik renderowania pierwszego kodu przedstawiono na Rysunku 1.



Rysunek 1: Wynik renderowania modelu jajka przy pomocy punktów.



Rysunek 2: Wynik renderowania modelu jajka, zbudowanego przy pomocy linii.

2.2 Zadanie 2 - Modelowanie jajka przy użyciu linii (ocena 3.5)

W tym zadaniu zmodyfikowano program z poprzedniego zadania, aby zamiast rysowania jajka za pomocą punktów (*GL_POINTS*), zastosować rysowanie za pomocą linii (*GL_LINES*), co pozwala na wizualizację struktury siatki parametrycznej obiektu.

2.2.1 Zmiany w programie

1. Dodano nową funkcję `egg_lines(N)`, która generuje siatkę złożoną z linii, łącząc sąsiadujące punkty obiektu jajka:

- Dla każdego punktu w siatce obliczono współrzędne sąsiadów w poziomie i w pionie.
- Użyto *GL_LINES* do rysowania linii pomiędzy kolejnymi punktami.

2. Funkcja `render(time)` została zmodyfikowana, aby wywoływać `egg_lines(N)` zamiast `draw_egg(N)`. 3. Wartość parametru N dla rysowania linii została ustawiona na 30, co pozwala na uzyskanie siatki o odpowiedniej szczegółowości.

2.2.2 Algorytm Rysowania Siatki Jajka

Algorytm rysowania siatki (`egg_lines`) działa w następujący sposób:

1. Wykorzystano funkcję `egg_points(N)`, aby obliczyć współrzędne punktów x, y, z obiektu jajka dla siatki $N \times N$.
2. Iteracyjnie przechodzono przez punkty siatki:
   - Dla każdego punktu połączono go liniami z jego sąsiadami w pionie oraz w poziomie.
   - Jeżeli punkt leży na brzegu siatki, pominięto połączenia wychodzące poza zakres.
3. Linie są rysowane za pomocą funkcji OpenGL `glVertex3f()` i wyświetlane w białym kolorze (`glColor3f(1.0, 1.0, 1.0)`).

2.2.3 Podsumowanie Działania

Modyfikacja programu pozwala na bardziej szczegółową wizualizację geometrii obiektu poprzez wyświetlanie jego struktury siatki. Linie lepiej odwzorowują kształt obiektu i ułatwiają analizę jego budowy matematycznej. Podobnie jak w poprzednim zadaniu, obiekt jest obracany w przestrzeni w celu lepszej prezentacji jego kształtu.

Wynik renderowania jajka przedstawiono na Rysunku 2.

2.3 Zadanie 3 - Modelowanie jajka przy użyciu trójkątów (ocena 4.0)

W tym zadaniu zmodyfikowano poprzedni program tak, aby obiekt jajka został narysowany za pomocą trójkątów (*GL_TRIANGLES*), co pozwala na bardziej szczegółowe odwzorowanie jego powierzchni. Dodatkowo, każdemu trójkątowi przypisano losowy kolor, co zwiększa atrakcyjność wizualną obiektu.

2.3.1 Zmiany w programie

1. Wprowadzono funkcję `generate_colors(N)`, która generuje losowe kolory dla każdego trójkąta w siatce:
   - Funkcja generuje $2 \times (N-1) \times (N-1)$ kolorów, ponieważ dla każdego kwadratu w siatce potrzebne są dwa trójkąty.
   - Kolory są przechowywane w globalnej liście `colors`.

2. Dodano funkcję `egg_triangle(N)`, która rysuje obiekt jajka za pomocą trójkątów:
   - Dla każdego kwadratu w siatce obliczono wierzchołki dwóch trójkątów.
   - Każdemu trójkątowi przypisano losowy kolor z listy `colors`.
   - Wykorzystano funkcję OpenGL `glVertex3f()` do definiowania wierzchołków.

3. W funkcji `render(time)` wywołano `egg_triangle(N)` zamiast `egg_lines(N)`.
4. Ustawiono parametr $N = 30$

2.3.2 Algorytm Rysowania Powierzchni Jajka

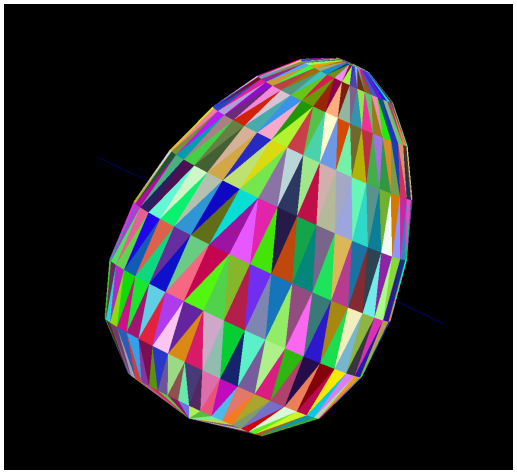
Algorytm rysowania trójkątów działa w następujący sposób:

1. Obliczono współrzędne punktów jajka (x, y, z) za pomocą funkcji `egg_points(N)`.
2. Iteracyjnie przechodzono przez punkty siatki:
   - Dla każdego punktu określono dwa trójkąty, które tworzą kwadrat:
     - Pierwszy trójkąt składa się z punktów $P(i, j)$, $P(i + 1, j)$, $P(i, j + 1)$.
     - Drugi trójkąt składa się z punktów $P(i + 1, j)$, $P(i + 1, j + 1)$, $P(i, j + 1)$.
   - Przypisano każdemu trójkątowi losowy kolor z wcześniej wygenerowanej listy `colors`.
3. Trójkąty są rysowane za pomocą *GL_TRIANGLES*.

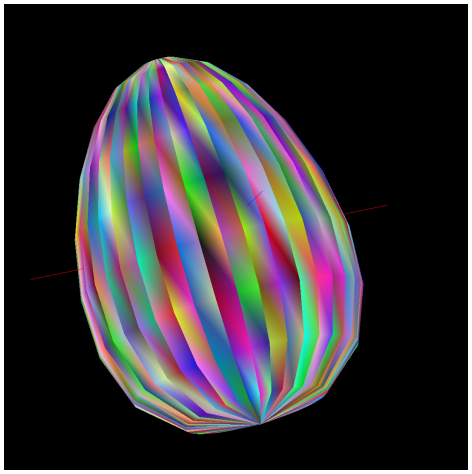
2.3.3 Podsumowanie Działania

Modyfikacja programu pozwala na wizualizację obiektu jajka w formie kolorowej, gładkiej powierzchni, co jest bardziej atrakcyjne wizualnie w porównaniu do poprzednich wersji programu. Dzięki zastosowaniu trójkątów jako elementów konstrukcyjnych powierzchni, możliwe jest dokładniejsze odwzorowanie geometrycznego kształtu obiektu. Losowe kolory zwiększają czytelność struktury obiektu.

Wynik renderowania jajka przedstawiono na Rysunku 3.



Rysunek 3: Wynik renderowania modelu jajka, zbudowanego przy pomocy trójkątów.



Rysunek 4: Wynik renderowania modelu jajka, zbudowanego przy pomocy prymitywu paskowego.

2.4 Zadanie 4 - Modelowanie jajka przy użyciu prymitywu paskowego (ocena 4.5)

W tej wersji programu obiekt jajka jest rysowany za pomocą pasków trójkątów (*GL_TRIANGLE_STRIP*), co jest alternatywnym sposobem reprezentacji powierzchni 3D w OpenGL. Zastosowano również losowe kolory dla poszczególnych wierzchołków, co dodaje atrakcyjności wizualnej.

2.4.1 Zmiany w programie

1. Zaktualizowano funkcję `generate_colors(N)`, aby generować kolory dla wszystkich wierzchołków w paskach trójkątów:

- Funkcja generuje $(N-1) \times (2 \times N)$ kolorów, ponieważ dla każdego paska trójkątów potrzebne są dwa kolory na każdy wierzchołek.
- Kolory są przechowywane w globalnej liście `colors`.

2. Dodano funkcję `egg_triangle_strip(N)`, która rysuje obiekt jajka za pomocą pasków trójkątów:

- Dla każdego paska trójkątów obliczane są wierzchołki z dwóch kolejnych rzędów siatki punktów jajka.
- Dla każdego wierzchołka przypisywany jest losowy kolor z listy `colors`.
- Użyto funkcji OpenGL `glVertex3f()` do definiowania wierzchołków, a `glColor3f()` do przypisywania kolorów.

3. W funkcji `render(time)` wywołano `egg_triangle_strip(20)` zamiast wcześniejszej wersji `egg_lines(N)`.

4. Parametr $N = 20$ został wybrany w celu uzyskania wyraźnej struktury jajka z paskami trójkątów.

2.4.2 Algorytm Rysowania Powierzchni Jajka z Paskami Trójkątów

Algorytm rysowania powierzchni jajka za pomocą pasków trójkątów działa w następujący sposób:

1. Obliczamy współrzędne punktów jajka (x, y, z) za pomocą funkcji `egg_points(N)`.
2. Iteracyjnie przechodzimy przez wiersze siatki (od $i = 0$ do $N - 1$):
    - Dla każdego wiersza dodawane są dwa wierzchołki z wiersza i oraz z wiersza $i + 1$.
    - Wierzchołkom przypisywane są kolory z listy `colors`.
    - Każdy pasek trójkątów rysowany jest za pomocą *GL_TRIANGLE_STRIP*.
3. Paski trójkątów są rysowane za pomocą funkcji OpenGL `glBegin(GL_TRIANGLE_STRIP)` oraz `glEnd()`.

2.4.3 Podsumowanie Działania

Ta wersja programu generuje bardziej zaawansowaną wizualizację obiektu jajka, wykorzystując paski trójkątów, które są wydajniejszym sposobem reprezentacji powierzchni 3D w OpenGL. Losowe kolory przypisane do każdego wierzchołka zwiększają atrakcyjność wizualną rysowanego obiektu. Dzięki tej metodzie możliwe jest uzyskanie płynniejszych przejść między trójkątami i lepszego odwzorowania powierzchni jajka.

Na Rysunku Rysunkku 4 przedstawiono model jajka, zbudowanego przy pomocy prymitywu paskowego.

2.5 Zadanie 5 (zadanie domowe)

2.5.1 Wstęp

Celem zadania było zaimplementowanie programu, który rysuje siatkę punktów kontrolnych w postaci małych kuleczek połączonych cienkimi liniami. Po naciśnięciu klawisza na rysunku miała się pojawić aproksymacja powierzchni Beziera złożona z wypełnionych przez interpolację kolorów trójkątów. Program został napisany w języku Python z wykorzystaniem bibliotek OpenGL i GLFW.

2.5.2 Metodologia

Program został zaimplementowany w kilku krokach:

1. Inicjalizacja środowiska graficznego za pomocą bibliotek OpenGL i GLFW.
2. Definicja siatki punktów kontrolnych.
3. Implementacja funkcji do rysowania punktów kontrolnych i linii łączących te punkty.
4. Implementacja funkcji do obliczania punktów na powierzchni Beziera.
5. Implementacja funkcji do rysowania powierzchni Beziera z wypełnionymi kolorami trójkątami.
6. Obsługa zdarzeń klawiatury do przełączania między trybami rysowania.

2.5.3 Inicjalizacja

Funkcja `startup` ustawia podstawowe parametry OpenGL, takie jak kolor tła i włączenie testu głębokości.

2.5.4 Rysowanie punktów kontrolnych i linii

Funkcje `draw_control_points` i `draw_control_lines` rysują punkty kontrolne oraz linie łączące te punkty.

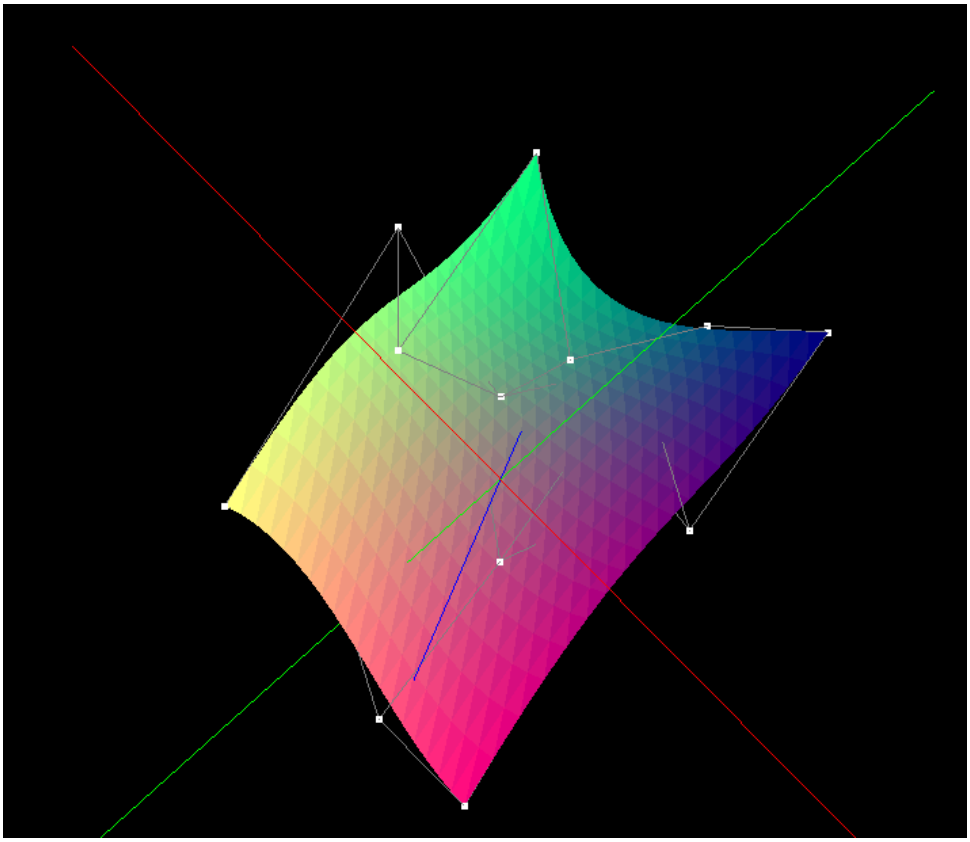
Powierzchnia Beziera Funkcje `bernstein` i `bezier_surface` obliczają wartości wielomianów Bernsteina oraz współrzędne punktów na powierzchni Beziera. Funkcja `draw_bezier_surface` rysuje powierzchnię Beziera z wypełnionymi kolorami trójkątami.

2.5.5 Wyniki

Na Rysunku 5 przedstawiono efekt renderowania powierzchni Beziera.

2.5.6 Podsumowanie

Program rysuje siatkę punktów kontrolnych i linie łączące te punkty. Po naciśnięciu klawisza rysuje powierzchnię Beziera z wypełnionymi kolorami trójkątami. Wykorzystano biblioteki OpenGL i GLFW do renderowania grafiki oraz bibliotekę NumPy do obliczeń matematycznych. Program umożliwia interaktywne wyświetlanie modelu powierzchni Beziera oraz osi współrzędnych.



Rysunek 5: Powierzchnia Beziera.

3 Wnioski Ogólne

W ramach laboratorium zrealizowano kilka zadań, które miały na celu zapoznanie się z technikami renderowania grafiki 3D za pomocą bibliotek OpenGL i GLFW. Poniżej przedstawiono najważniejsze wnioski z przeprowadzonych ćwiczeń:

- **Inicjalizacja środowiska graficznego:** W każdym zadaniu kluczowym elementem była poprawna inicjalizacja środowiska graficznego za pomocą bibliotek OpenGL i GLFW. Umożliwiło to tworzenie okien, ustawianie kontekstu renderowania oraz obsługę zdarzeń.
- **Rysowanie osi współrzędnych:** Rysowanie osi współrzędnych w trzech kolorach (czerwonym, zielonym i niebieskim) pozwoliło na lepszą orientację w przestrzeni 3D i ułatwiło wizualizację renderowanych obiektów.
- **Modelowanie powierzchni:** W zadaniach wykorzystano różne techniki modelowania powierzchni, takie jak rysowanie jajka za pomocą punktów oraz aproksymacja powierzchni Beziera. Pozwoliło to na zrozumienie, jak różne metody interpolacji i aproksymacji wpływają na kształt i wygląd renderowanych obiektów.
- **Interaktywność:** Implementacja obsługi zdarzeń klawiatury umożliwiła interaktywne przełączanie między trybami rysowania oraz manipulację widokiem sceny. Dzięki temu użytkownik może dynamicznie zmieniać sposób prezentacji danych i lepiej zrozumieć ich strukturę.

- **Wykorzystanie bibliotek:** Biblioteki OpenGL i GLFW okazały się być potężnymi narzędziami do renderowania grafiki 3D. Biblioteka NumPy była nieoceniona w obliczeniach matematycznych, takich jak generowanie punktów na powierzchni jajka czy obliczanie wartości wielomianów Bernsteina.
- **Wyzwania:** Podczas realizacji zadań napotkano na kilka wyzwań, takich jak poprawne połączenie punktów kontrolnych w siatce czy zapewnienie płynnej interpolacji kolorów na powierzchni Beziera. Rozwiązanie tych problemów wymagało dokładnej analizy i testowania różnych podejść.
- **Wnioski praktyczne:** Praktyczne doświadczenie zdobyte podczas realizacji zadań pozwoliło na lepsze zrozumienie zasad renderowania grafiki 3D oraz technik modelowania powierzchni. Umiejętności te są kluczowe w wielu dziedzinach, takich jak grafika komputerowa, wizualizacja danych czy projektowanie gier.

Podsumowując, laboratorium pozwoliło na zdobycie cennych umiejętności w zakresie renderowania grafiki 3D oraz modelowania powierzchni. Wykorzystanie bibliotek OpenGL, GLFW i NumPy umożliwiło realizację złożonych zadań i uzyskanie satysfakcjonujących wyników.