<table>
<tr><td>Politechnika Wrocławska</td><td>Autor: Piotr Józefek 272311</td><td>Wydział Informatyki i Telekomunikacji
Rok akademicki: 3</td></tr>
<tr><td colspan="3">Grafika komputerowa i komunikacja człowiek – komputer</td></tr>
<tr><td>Data ćwiczenia:
29.10.2024</td><td rowspan="2">Temat ćwiczenia laboratoryjnego:
Modelowanie obiektów 3D</td><td>Ocena:</td></tr>
<tr><td>Nr ćwiczenia:
3</td><td>Prowadzący:
dr inż. arch. Tomasz Zamojski</td></tr>
</table>

1. Wstęp

Opis przy pomocy równań parametrycznych to technika, która pozwala na odwzorowanie skomplikowanych kształtów i powierzchni za pomocą funkcji zależnych od dwóch zmiennych, najczęściej oznaczanych jako u i v. W ramach tego podejścia, współrzędne punktów na powierzchni, np. jajka, są obliczane na podstawie funkcji parametrycznych, co pozwala na uzyskanie precyzyjnego odwzorowania kształtu. Parametryczne równań opisują sposób, w jaki zmieniają się współrzędne punktów w przestrzeni 3D w zależności od wartości parametrów, przy czym gęstość próbkowania tych parametrów (tj. podział przedziału) pozwala na kontrolowanie szczegółowości odwzorowania. W przykładzie jajka, współrzędne punktów powierzchni są określane przez układ równań parametrycznych, który zależy od parametrów u i v. Zmienna u opisuje jeden wymiar powierzchni, a zmienna v drugi, przy czym obie zmienne mieszczą się w przedziale od 0 do 1. Na tej podstawie, dla każdej kombinacji wartości u i v, obliczane są współrzędne punktów 3D, które tworzą model jajka. Mechanizm bufora głębi jest istotnym elementem renderowania przestrzeni 3D, który zapewnia prawidłowe wyświetlanie obiektów w scenie. Dzięki niemu, obiekty wirtualne mogą się poprawnie nakładać na siebie, a obiekt, który znajduje się bliżej kamery, zasłania te, które są dalej, co jest kluczowe w generowaniu realistycznych scen.

2. Cel ćwiczenia

- Zrozumienie różnych sposobów definiowania modeli 3D.
- Nabranie wprawy w definiowaniu brył przy pomocy wierzchołków.
- Poznanie zasady działania mechanizmu bufora głębi.

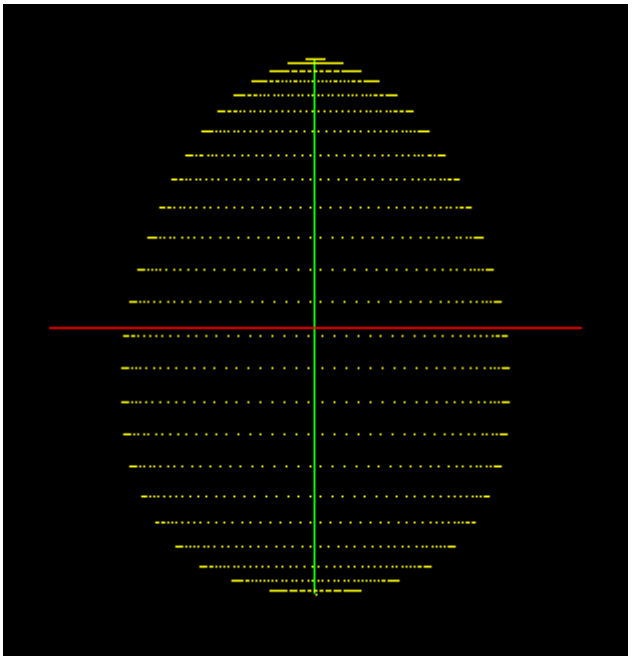
3. Wykonane zadania

Udało się zrealizować wszystkie zadania. W zadaniu na 5 wybrałem narysowanie Trójkąt Sierpińskiego – wersja 3-D.

4. Prezentacja i omówienie funkcjonalności

4.1. Model jajka przy pomocy punktów

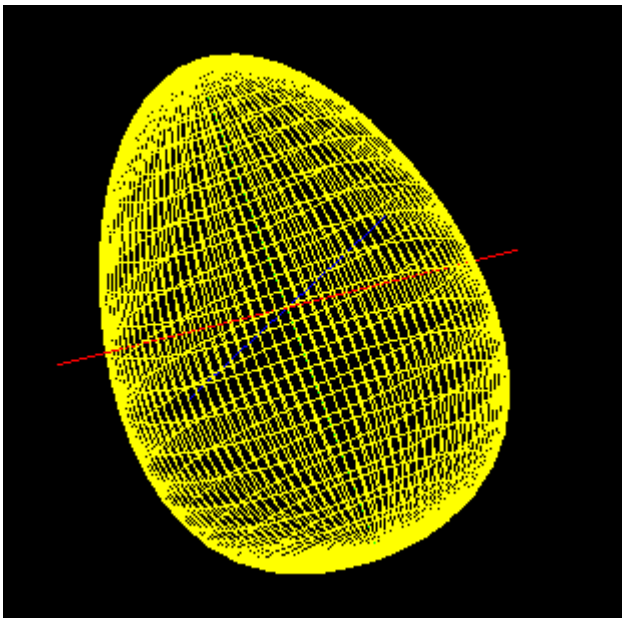
Celem tego zadania było obliczenie współrzędnych punktów modelu jajka i narysowanie ich w przestrzeni 3D przy użyciu OpenGL. W tym celu zostały stworzone dwie funkcje: calculate_egg_points() i draw_egg(). Funkcja calculate_egg_points() tworzy tablicę o wymiarach (N, N, 3), w której przechowywane są współrzędne punktów jajka. Zmienna u i v to wektory współrzędnych, które są generowane za pomocą funkcji np.linspace(). Następnie, w dwóch zagnieżdżonych pętlach for, obliczane są współrzędne x, y, z dla każdego punktu na powierzchni jajka na podstawie odpowiednich równań matematycznych. Wynikowe współrzędne są zapisywane w tablicy tab. Funkcja draw_egg() odpowiada za rysowanie modelu jajka. Najpierw wywoływana jest funkcja calculate_egg_points(), która oblicza współrzędne punktów. Następnie, za pomocą funkcji glColor3f(1.0, 1.0, 0.0), ustawiany jest kolor punktów na żółty. Funkcja glBegin(GL_POINTS) rozpoczyna rysowanie punktów, a w zagnieżdżonej pętli for każde współrzędne punktu są rysowane za pomocą funkcji glVertex3fv(). Na zakończenie, glEnd() kończy rysowanie, a model jajka jest wyświetlany jako zbiór żółtych punktów w przestrzeni 3D.



Rys 1 Narysowany model jajka przy pomocy punktów dla N=50

4.2. Model jajka przy pomocy linii

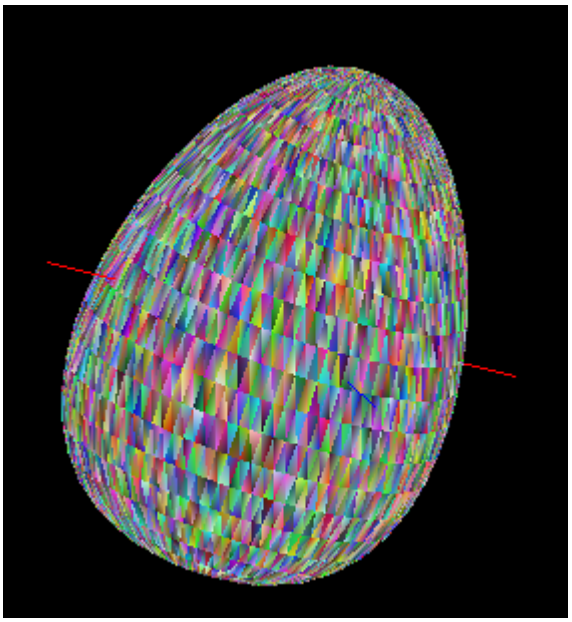
Celem tego zadania było obliczenie współrzędnych punktów jajka oraz narysowanie ich jako punkty i linie w przestrzeni 3D, a następnie dodanie funkcji umożliwiającej obracanie obiektu. W tym celu zostały stworzone dwie funkcje: draw_egg_lines() i spin(). Funkcja draw_egg_lines() odpowiada za wyświetlanie modelu jajka w dwóch częściach: jako punkty oraz linie. Najpierw, tak jak w poprzedniej funkcji, wywoływana jest funkcja calculate_egg_points(), która oblicza współrzędne punktów jajka. Następnie kolor punktów ustawiany jest na żółty za pomocą glColor3f(1.0, 1.0, 0.0). Rysowanie punktów odbywa się za pomocą funkcji glBegin(GL_POINTS) i glVertex3fv(), a zakończenie rysowania punktów następuje przez glEnd(). Kolejnym krokiem jest rysowanie linii, które łączą punkty jajka. Używa się do tego glBegin(GL_LINES) i glVertex3fv(). Dwie zagnieżdżone pętle for przechodzą przez wszystkie punkty w tablicy tab, rysując linie między punktami w poziomie i pionie, łącząc je w siatkę. Na końcu pętli dla kolumn i wierszy rysowane są linie, które łączą ostatni wiersz i kolumnę z odpowiednimi punktami. Rysowanie linii kończy glEnd(), a efektem jest wizualizacja jajka jako siatki punktów połączonych liniami. Dodatkowo, celem zadania było zaimplementowanie funkcji obracania obiektu. Funkcja spin(angle) używa funkcji glRotatef() do obrotu obiektu wokół trzech osi (X, Y, Z). Wartość kąta angle jest przekazywana do funkcji, a obrót realizowany jest poprzez trzykrotne wywołanie glRotatef(), jedno dla każdej z osi, co pozwala na obrót obiektu w przestrzeni 3D. Obrót odbywa się w wyniku zaktualizowania rzutni na każdej klatce animacji, dzięki czemu obiekt obraca się wokół tych trzech osi.



Rys 2 Narysowany model jajka przy pomocy linii dla N=50

4.3. Model przy pomocy trójkątów

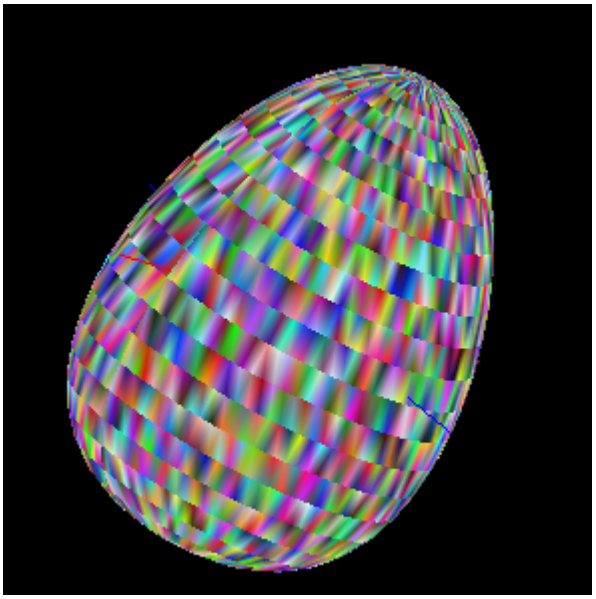
Celem tego zadania było obliczenie współrzędnych punktów jajka, a następnie narysowanie go jako trójkąty, gdzie każdy wierzchołek ma przypisany losowy kolor. Dodatkowo celem było zapewnienie, by losowanie kolorów nie powodowało migotania podczas renderowania. W tym celu została zaimplementowana funkcja draw_egg_triangles() oraz sposób przekazywania parametru random_seed. Funkcja draw_egg_triangles(random_seed) zaczyna się od ustawienia ziarna generatora liczb losowych za pomocą random.seed(random_seed), co gwarantuje, że kolory wierzchołków nie będą się zmieniać w już uruchomionym programie, eliminując efekt migotania, ale będą się losować przy każdym uruchomieniu. Następnie wywoływana jest funkcja calculate_egg_points(), która oblicza współrzędne punktów jajka. Dla każdego wierzchołka trójkąta przypisywany jest losowy kolor. Używa się funkcji random.random(), która generuje losową wartość z zakresu [0, 1] dla każdego składnika koloru (R, G, B). Dzięki temu każdy wierzchołek trójkąta ma przypisany inny, losowy kolor. Rysowanie trójkątów odbywa się przy użyciu funkcji glBegin(GL_TRIANGLES) i glVertex3fv(), gdzie kolejno dodawane są wierzchołki trójkątów. Dwa trójkąty są rysowane w każdej iteracji podwójnej pętli for, łącząc odpowiednie punkty w tablicy tab w celu utworzenia siatki trójkątów na powierzchni jajka. Każdy trójkąt jest tworzony przez trzy punkty, które są połączone liniami, a kolory wierzchołków są losowane przy każdym wywołaniu glColor3f(). Na końcu glEnd() kończy rysowanie trójkątów, a wynikiem jest model jajka, którego powierzchnia składa się z trójkątów o losowych kolorach.



Rys 3 Narysowany model jajka przy pomocy trójkątów dla N=50

4.4. Model za pomocą prymitywu paskowego

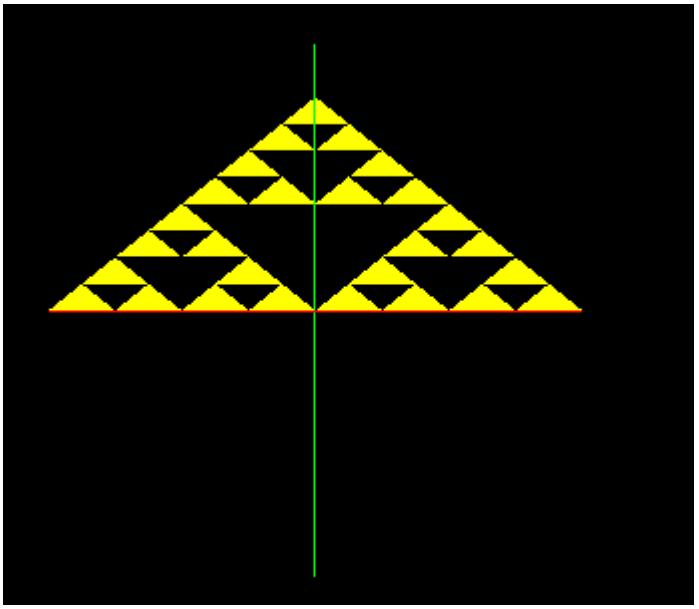
Celem tego zadania było obliczenie współrzędnych punktów jajka oraz narysowanie go przy użyciu pasma trójkątów, gdzie każdy wierzchołek ma przypisany losowy kolor. Dodatkowo, zadaniem było zapewnienie, że kolory będą spójne i nie będą migotać, co osiągnięto poprzez użycie parametru random_seed. Funkcja draw_egg_triangle_strip(random_seed) rozpoczyna się od ustawienia ziarna generatora liczb losowych za pomocą random.seed(random_seed), co zapewnia, że kolory wierzchołków nie będą się zmieniać w już uruchomionym programie eliminując efekt migotania, ale będą się losować przy każdym uruchomieniu. Następnie, wywoływana jest funkcja calculate_egg_points(), która oblicza współrzędne punktów jajka i przechowuje je w tablicy tab. W funkcji rysowania jajka, pętla for przechodzi przez każdy wiersz poza ostatnim, a dla każdego wiersza, za pomocą glBegin(GL_TRIANGLE_STRIP) rozpoczyna się rysowanie pasma trójkątów. Dla każdego wiersza, w zagnieżdżonej pętli for, dla każdego punktu w tym wierszu i następnego, generowany jest losowy kolor dla każdego wierzchołka przy pomocy glColor3f(random.random(), random.random(), random.random()). Następnie, dla każdego punktu, funkcja glVertex3fv() dodaje wierzchołek do pasma trójkątów: najpierw punkt z aktualnego wiersza, a potem punkt z następnego wiersza, tworząc trójkąt w przestrzeni 3D. Na końcu glEnd() kończy rysowanie pasma trójkątów, a efektem jest model jajka, którego powierzchnia składa się z pasków trójkątów, z losowymi kolorami wierzchołków.



Rys 4 Narysowany model jajka za pomocą prymitywu paskowego dla N=50

4.5. Trójkąt Sierpińskiego – wersja 3-D

Celem zadania było narysowanie piramidy Sierpińskiego z możliwością określenia poziomu rekurencji, co tworzy efekt fraktalu. Do tego celu została utworzona funkcja draw_sierpinski_pyramid(vertices, depth). Jeśli wartość depth jest równa 0 funkcja rysuje piramidę przy użyciu glBegin(GL_TRIANGLES), co rozpoczyna definiowanie kształtu. Funkcja glVertex3fv() służy do ustalania pozycji wierzchołków, tworząc cztery ściany piramidy:
Ściana 1 składa się z wierzchołków vertices[0], vertices[1], vertices[2].
Ściana 2 składa się z wierzchołków vertices[0], vertices[1], vertices[3].
Ściana 3 składa się z wierzchołków vertices[1], vertices[2], vertices[3].
Ściana 4 składa się z wierzchołków vertices[0], vertices[2], vertices[3].
Na zakończenie glEnd() kończy rysowanie, a piramida zostaje wyświetlona. Jeśli depth jest większe niż 0 funkcja oblicza punkty środkowe między parami wierzchołków i zapisuje je w liście mid_points. Nowe zestawy wierzchołków są tworzone dla każdej z mniejszych piramid, a funkcja rekurencyjnie wywołuje samą siebie z mniejszą wartością depth, co pozwala na podział pierwotnej piramidy na mniejsze części. Efektem działania funkcji jest wyświetlenie piramidy Sierpińskiego, której struktura składa się z wielu mniejszych piramid, tworząc skomplikowany wzór fraktalny.



Rys 5 Narysowany Trójkąt Sierpińskiego – wersja 3-D

5. Wnioski

- Udało się zrealizować wszystkie zadania
- Dla wartości N = 500 obracanie jajka przebiegało w sposób znacząco niepłynny, charakteryzując się niską liczbą klatek na sekundę, co powodowało zauważalne opóźnienia i spadki wydajności podczas renderowania obrazu

6. Literatura

- Szymon Datko [GK] Lab3 :: Modelowanie 3D