<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:
7.01.2024</td><td rowspan="2">Temat ćwiczenia laboratoryjnego:
Teksturowanie obiektów</td><td>Ocena:</td></tr>
<tr><td>Nr ćwiczenia:
6</td><td>Prowadzący:
dr inż. arch. Tomasz Zamojski</td></tr>
</table>

1. Wstęp

Tekstury w grafice komputerowej pełnią funkcję map kolorów, które nakłada się na powierzchnie modeli 3D w celu nadania im bardziej realistycznego wyglądu. Można je intuicyjnie porównać do obrazów rozciąganych pomiędzy wierzchołkami obiektów geometrycznych. Stanowią one zarówno źródło danych wejściowych dla operacji graficznych, jak i przestrzeń do zapisu wyników renderowania. W OpenGL tekstury są często wykorzystywane w shaderach fragmentów, gdzie pozwalają na precyzyjne określenie koloru każdego fragmentu obiektu na podstawie współrzędnych tekstury. Do tego celu używa się obiektów typu uniform sampler, które odnoszą się bezpośrednio do bufora tekstury. Podstawowe rodzaje tekstur to tekstury jednowymiarowe, dwuwymiarowe oraz trójwymiarowe. W przeszłości tekstury były ograniczone do kwadratowych kształtów, a ich wymiary musiały być potęgami liczby dwa. Choć współczesne standardy graficzne złagodziły te wymagania, nadal zaleca się przestrzeganie tych zasad w celu optymalizacji wydajności. Aby poprawnie odwzorować teksturę na modelu, mapuje się punkty z przestrzeni UV tekstury na odpowiednie wierzchołki obiektu. W starszych wersjach OpenGL, takich jak Legacy OpenGL, do wskazywania współrzędnych tekstury dla poszczególnych wierzchołków stosuje się funkcję glTexCoord(). Tekstury pozwalają również na tworzenie bardziej zaawansowanych efektów wizualnych, takich jak mapowanie nierówności (bump mapping) czy mapowanie paralaksy. Dzięki filtrowaniu mipmap można skutecznie rozwiązać problemy związane z rysowaniem małych, odległych obiektów. Ponadto mechanizm Face Culling umożliwia eliminację niewidocznych powierzchni obiektów, co pozwala na oszczędność obliczeń GPU oraz poprawę wydajności renderowania scen trójwymiarowych.

2. Cel ćwiczenia

- Zapoznać się i zrozumieć zagadnienie teksturowania.
-. Poznać sposób aplikowania tekstury na zdefiniowane powierzchnie.
- Nauczyć się jak stworzyć własną teksturę dla własnego obiektu.

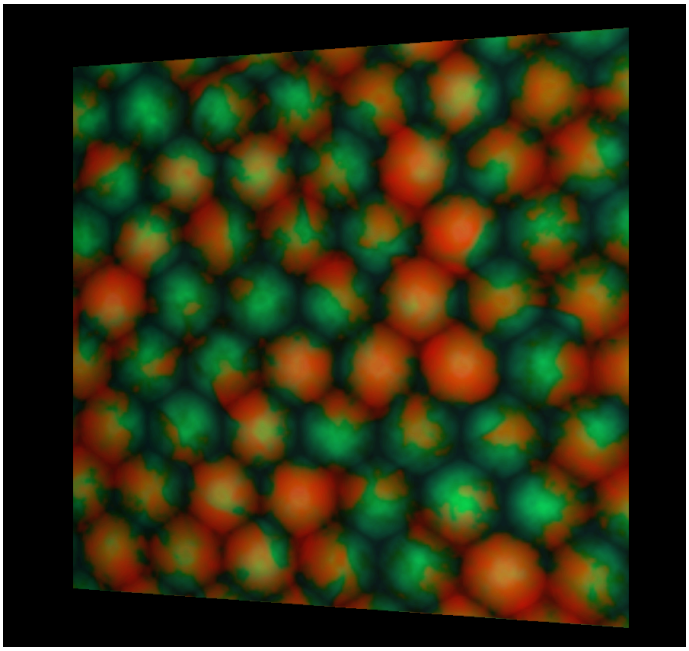
3. Wykonane zadania

Udało się zrealizować wszystkie zadania.

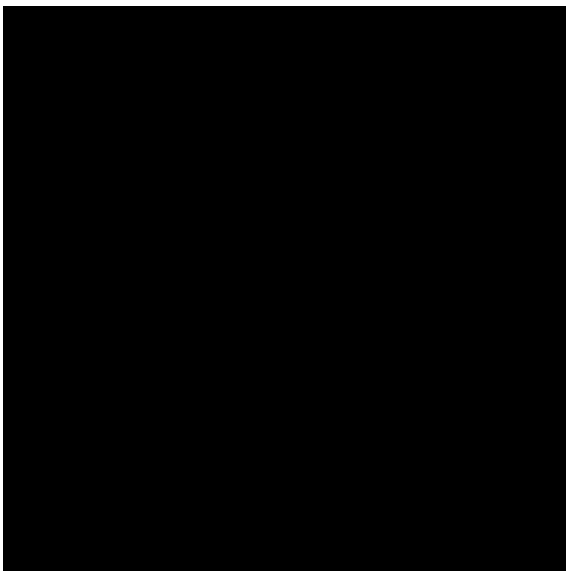
4. Prezentacja i omówienie funkcjonalności

4.1 Oteksturowany kwadrat

Celem tego zadania było stworzenie kwadratu z nałożoną teksturą, wykorzystując funkcje OpenGL. Do jego budowy użyto dwóch trójkątów, które razem tworzą całą powierzchnię figury. Tekstura jest równomiernie rozciągnięta pomiędzy wierzchołkami, zapewniając poprawne odwzorowanie obrazu. Rysowanie rozpoczyna funkcja glBegin(GL_TRIANGLES), która pozwala definiować trójkąty za pomocą kolejnych wierzchołków. Każdemu z wierzchołków przypisywane są współrzędne tekstury za pomocą funkcji glTexCoord2f, co umożliwia mapowanie obrazu na powierzchnię kwadratu. Po zdefiniowaniu wszystkich wierzchołków funkcja glEnd() kończy proces rysowania. Aby poprawnie obsłużyć teksturę i zoptymalizować rysowanie obiektu, włączono tryb teksturowania za pomocą glEnable(GL_TEXTURE_2D). Dodatkowo aktywowano eliminację niewidocznych ścian poleceniem glEnable(GL_CULL_FACE), a funkcja glCullFace(GL_BACK) skonfigurowała odrzucanie tylnych ścian obiektu. Dzięki temu mechanizmowi możliwe jest bardziej efektywne renderowanie kwadratu z teksturą, co pozwala na lepszą optymalizację pracy GPU.



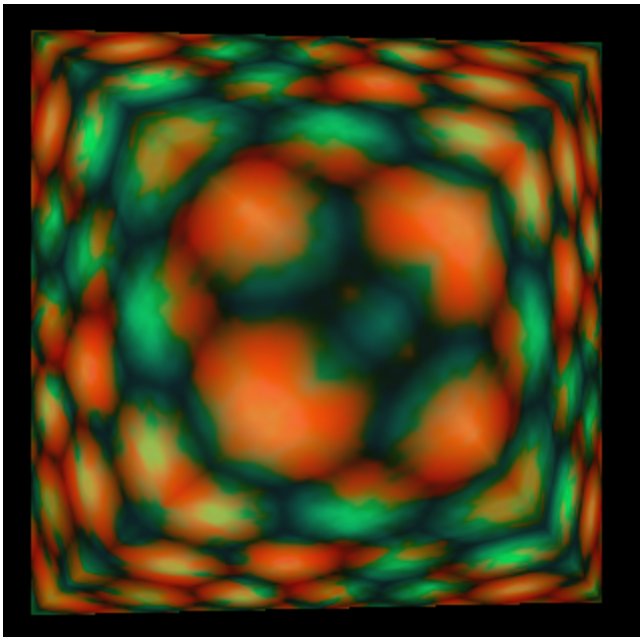
Rys 1 Wyświetlenie otekstorowanego kwadratu



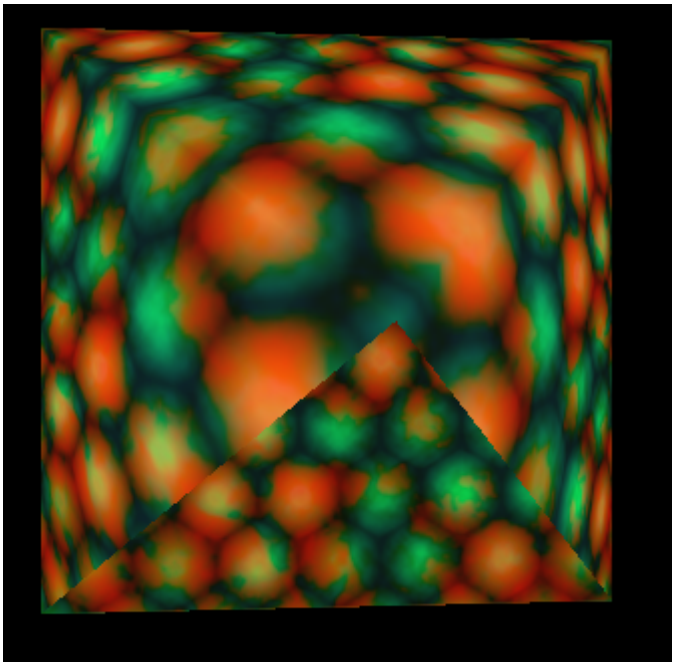
Rys 2 Wyświetlenie otekstorowanego kwadratu po jego obróceniu

4.2 Otekstorowany ostrosłup

Celem tego zadania było utworzenie teksturowanej podstawy ostrosłupa oraz jego ścian bocznych z możliwością dynamicznego ukrywania ścian przy użyciu klawisza spacji. Tekstura jest nakładana zarówno na podstawę, jak i ściany ostrosłupa, co wzbogaca wizualnie scenę. Rysowanie podstawy ostrosłupa odbywa się przy użyciu dwóch trójkątów definiowanych za pomocą funkcji glBegin(GL_TRIANGLES). Każdy wierzchołek trójkąta jest przypisywany do odpowiednich współrzędnych tekstury za pomocą funkcji glTexCoord2f. Ściany trójkątne ostrosłupa są definiowane jako cztery trójkąty boczne, gdzie każdy jest rysowany przez wierzchołek szczytowy oraz dwa kolejne wierzchołki podstawy. Pętla iteruje przez listę ścian zdefiniowanych przez współrzędne wierzchołków v1 i v2. Dla każdej ściany, której indeks znajduje się w zmiennej faces, sprawdzane jest, czy powinna być widoczna, korzystając z tablicy logicznej show_face. Jeżeli wartość show_face[i] wynosi True, funkcja glBegin(GL_TRIANGLES) rysuje odpowiednią ścianę. Współrzędne tekstury ścian są dynamicznie ustawiane na podstawie indeksu ściany, co umożliwia ich odpowiednie mapowanie. Każda ściana jest zakończona funkcją glEnd(). Wprowadzono obsługę klawisza spacji, który pozwala na dynamiczne ukrywanie lub pokazywanie wybranej ściany ostrosłupa. Wciśnięcie spacji zmienia stan logiczny odpowiedniego elementu w tablicy show_face, co umożliwia kontrolowanie widoczności ścian ostrosłupa podczas działania programu. Dzięki temu mechanizmowi użytkownik może w interaktywny sposób manipulować wizualizacją ostrosłupa.



Rys 3 Wyświetlenie oteksturowanego ostrosłupa



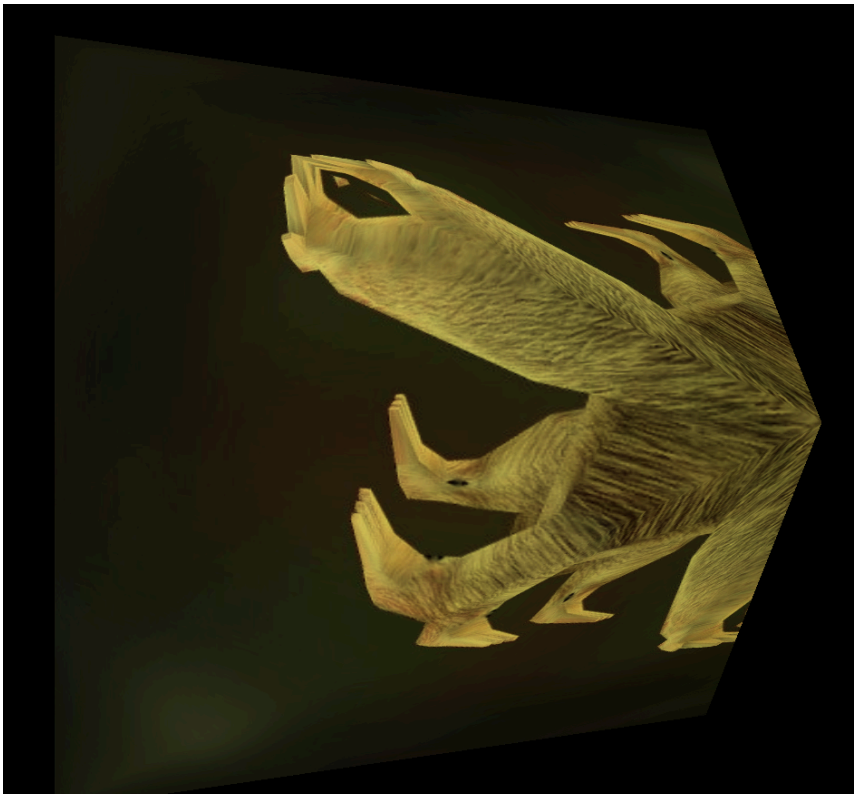
Rys 4 Wyświetlenie oteksturowanego ostrosłupa z ukrytą scianą

4.3 Tworzenie i zastosowanie własnej tekstury

Teksturę stworzyłem poprzez podążanie kroków z instrukcji oraz wykorzystanie programu graficznego gimp.

4.4 Wprowadzenie możliwości przełączania tekstur

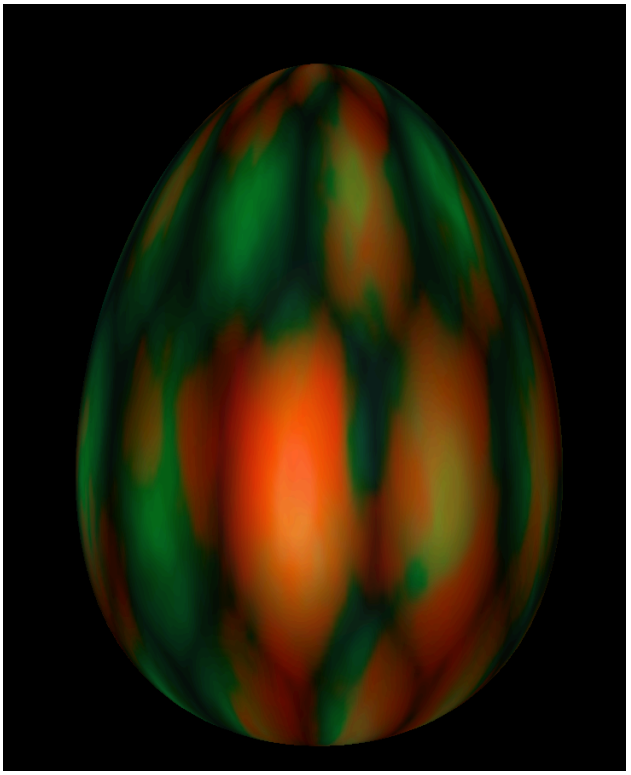
Celem tego zadania było wprowadzenie funkcji przełączania tekstur, które mogą być dynamicznie zmieniane podczas działania programu za pomocą naciśnięcia klawisza Enter. Na początku, w kodzie ładowane są dwie tekstury: "tekstura.tga" oraz "rat.tga". Funkcja load_texture odpowiedzialna jest za wczytanie tych plików i zwrócenie danych tekstury, które są następnie dodawane do listy textures.
W funkcji switch_texture() następuje zmiana aktualnie wyświetlanej tekstury. Zmienna current_texture przechowuje indeks obecnie używanej tekstury. Po naciśnięciu klawisza Enter, wartość current_texture jest inkrementowana o 1, a następnie wykonywana jest operacja reszty z dzielenia przez długość listy tekstur (% len(textures)), co pozwala na przełączanie się pomiędzy teksturami w sposób cykliczny. Dzięki temu, gdy osiągnięty zostanie koniec listy tekstur, powraca się do pierwszej. Po zmianie tekstury, z listy textures wyciągane są dane aktualnej tekstury (szerokość, wysokość i dane pikseli), które następnie są używane w funkcji glTexImage2D. Funkcja ta aktualizuje teksturę przypisaną do obiektu 3D, używając nowych danych, a tym samym zmienia wygląd obiektu w scenie. Cały proces przełączania tekstur odbywa się w odpowiedzi na zdarzenie naciśnięcia klawisza Enter, co jest obsługiwane w funkcji odpowiedzialnej za zdarzenia klawiatury za pomocą keyboard_key_callback.



Rys 5 Wyświetlenie oteksturowanego ostrosłupa z własną tekturą

4.5 Nałożenie tekstury na jajko

Celem tego kodu jest wygenerowanie i narysowanie modelu 3D jajka z możliwością nałożenia tekstury. Kod ten realizuje kilka kluczowych zadań, które składają się na finalny efekt wizualny. Na początku, definiowane są funkcje calculate_egg_points() i calculate_normals(). Pierwsza z nich, calculate_egg_points(), ma za zadanie wygenerować siatkę punktów, które po połączeniu utworzą kształt jajka. Punkty te są obliczane na podstawie parametrycznych równań, gdzie parametry u i v kontrolują położenie punktu na powierzchni jajka. Funkcja ta zwraca tablicę z wygenerowanymi punktami. Funkcja, calculate_normals(), oblicza wektory normalne dla każdego punktu na powierzchni jajka. Wektory normalne są niezbędne do prawidłowego oświetlenia obiektu w scenie 3D. Funkcja ta wykorzystuje pochodne cząstkowe do przybliżenia wektorów stycznych, a następnie oblicza iloczyn wektorowy tych stycznych, uzyskując wektor normalny. Wektor normalny jest następnie normalizowany, a funkcja zwraca tablicę z wektorami normalnymi dla każdego punktu. Funkcja draw_egg_with_texture() łączy wygenerowane punkty i wektory normalne w trójkąty, tworząc model 3D jajka. Dodatkowo, funkcja ta obsługuje teksturowanie, mapując teksturę na powierzchnię jajka. Współrzędne tekstury są przypisywane do każdego wierzchołka trójkąta, co umożliwia przyklejenie tekstury do modelu. Funkcja ta wykorzystuje paski trójkątów (GL_TRIANGLE_STRIP) do efektywnego rysowania powierzchni.



Rys 6 Jajko z domyślna teksturą

Rys 6 Jajko z moją teksturą

5. Wnioski

- Udało się zrealizować wszystkie zadania

6. Literatura

- Szymon Datko [GK] Lab6 :: Teksturowanie obiektów