<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:
26.11.2024</td><td rowspan="2">Temat ćwiczenia laboratoryjnego:
Oświetlanie scen</td><td>Ocena:</td></tr>
<tr><td>Nr ćwiczenia:
5</td><td>Prowadzący:
dr inż. arch. Tomasz Zamojski</td></tr>
</table>

1. Wstęp

Oświetlenie w aplikacjach 3D jest jednym z kluczowych elementów tworzenia realistycznych wizualizacji. Składa się ono z szeregu złożonych zjawisk, które mają na celu odwzorowanie interakcji światła z powierzchniami obiektów. Ważnym aspektem jest odwzorowanie kolorów obiektów oraz uwzględnienie ich parametrów materiałowych. Dzięki temu możliwe jest tworzenie powierzchni o różnorodnych właściwościach wizualnych, takich jak półprzeźroczystość czy efekty załamania światła. Kluczowe znaczenie mają również techniki cieniowania i przesłaniania obiektów, które zwiększają realizm sceny 3D. Efekty wizualne, takie jak efekty cząstkowe, efekty wolumetryczne, paralaksa, rozmycie oraz rozproszenie światła, są istotne w zaawansowanych aplikacjach graficznych. Oświetlenie w grafice komputerowej można podzielić na dwa podstawowe rodzaje: oświetlenie lokalne i oświetlenie globalne. Modele oświetlenia lokalnego skupiają się na interakcji światła z pojedynczym obiektem, podczas gdy oświetlenie globalne uwzględnia wzajemne oddziaływanie światła między wieloma obiektami w scenie. Jednym z najczęściej stosowanych modeli oświetlenia w grafice komputerowej jest model Phonga. Model ten bazuje na trzech komponentach światła: świetle kierunkowym, świetle rozproszonym oraz świetle otoczenia. Światło kierunkowe generuje refleksy odbite zgodnie z prawem Snella, co pozwala na realistyczne odwzorowanie błyszczących powierzchni. Światło rozproszone uwzględnia wpływ bezpośredniego oświetlenia na powierzchnię obiektu, podczas gdy światło otoczenia odpowiada za jednorodne oświetlenie obiektu, niezależnie od jego orientacji w przestrzeni. Model Phonga dobrze oddaje wygląd obiektów wykonanych z tworzyw sztucznych, jednak jego zastosowanie może być trudniejsze w przypadku materiałów metalicznych czy szklanych. Formuła modelu Phonga uwzględnia transformacje obiektów, które wpływają na wektory normalne, co jest istotne dla poprawnego wyznaczania efektów świetlnych. Model ten nie uwzględnia jednak cieniowania, a jedynie określa kolor powierzchni w danym punkcie. Dobór odpowiednich parametrów modelu, takich jak współczynniki odbicia, może być trudny, szczególnie w przypadku bardziej skomplikowanych materiałów. Oprócz modelu Phonga istnieją inne zaawansowane modele oświetlenia, takie jak model Cook-Torrance'a oraz model Ashikhmina-Shirley'a. Współczesne podejście do renderowania realistycznego, znane jako Physically Based Rendering (PBR), korzysta z fizycznych właściwości materiałów, aby precyzyjnie odwzorować interakcje światła. Ważnym elementem tych technik jest poprawne wyznaczanie wektorów normalnych, które mają kluczowe znaczenie w obliczeniach świetlnych. Dzięki temu możliwe jest uzyskanie jeszcze bardziej realistycznych efektów wizualnych w aplikacjach 3D.

2. Cel ćwiczenia

- Zapoznać się i zrozumieć zawiłości zagadnienia oświetlania scen.
- Zgłębić zasadę działania przykładowego modelu oświetlenia – Phonga.
- Nauczyć się jak programowo obsługiwać źródła światła w OpenGL.
- Poznać sposób definiowania wektorów normalnych dla własnych modeli.

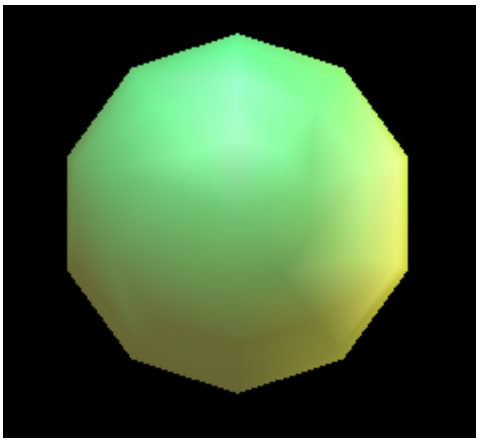
3. Wykonane zadania

Udało się zrealizować wszystkie zadania.

4. Prezentacja i omówienie funkcjonalności

4.1. Wprowadzenie drugiego źródła światła

W kodzie dodano fragment definiujący właściwości (kolor, pozycję i tłumienie) drugiego źródła światła w scenie OpenGL.

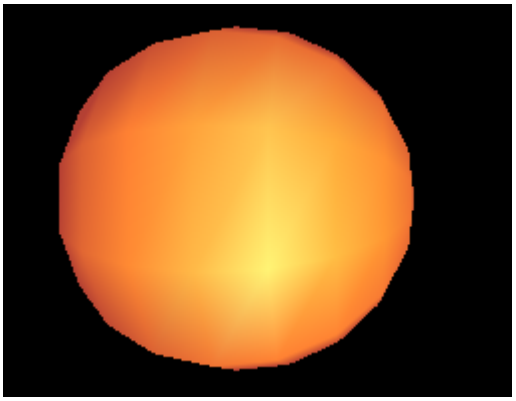


Rys 1 Dodatkowe źródło światła

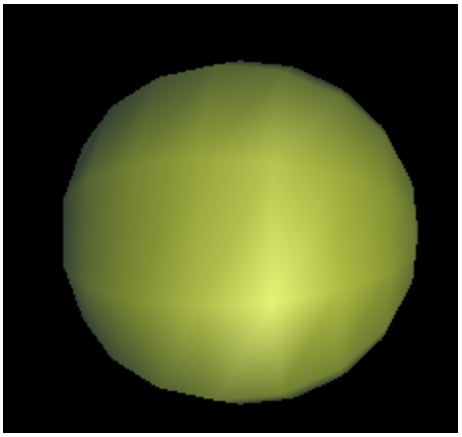
4.2. Dynamiczna zmiana składowych koloru światła.

Celem tego zadania było wprowadzenie możliwości dynamicznej zmiany składowych koloru światła przy użyciu klawiatury. Klawisze Q, W i E służą do wyboru odpowiedniej składowej światła: odpowiednio ambient (otaczające), diffuse (rozproszone) lub specular (odbite). Wybrana składowa jest przechowywana w zmiennej current_component, a jej zmiana jest potwierdzana komunikatem tekstowym w konsoli. Klawisze strzałek UP i DOWN pozwalają zwiększać lub zmniejszać wartość aktualnie wybranej składowej. Gdy naciśnięty zostanie klawisz UP, wywoływana jest funkcja modify_light, która zwiększa wartość wybranej składowej o stałą wartość 0.1. Analogicznie, naciśnięcie klawisza DOWN powoduje zmniejszenie wartości składowej o 0.1. Po każdej zmianie wywoływana jest funkcja update_light, która wprowadza nowe wartości do

ustawień światła w scenie. Funkcja keyboard_key_callback odpowiada za obsługę zdarzeń związanych z klawiaturą. Rozpoznaje naciśnięcie klawiszy Q, W, i E, zmieniając aktywną składową światła. Rejestruje naciśnięcie i zwolnienie klawiszy UP i DOWN, ustawiając odpowiednie flagi (key_up_pressed, key_down_pressed), co steruje modyfikacją wartości światła. Dzięki temu mechanizmowi użytkownik może w prosty sposób kontrolować różne składowe światła w scenie graficznej, dostosowując jego wygląd w czasie rzeczywistym.



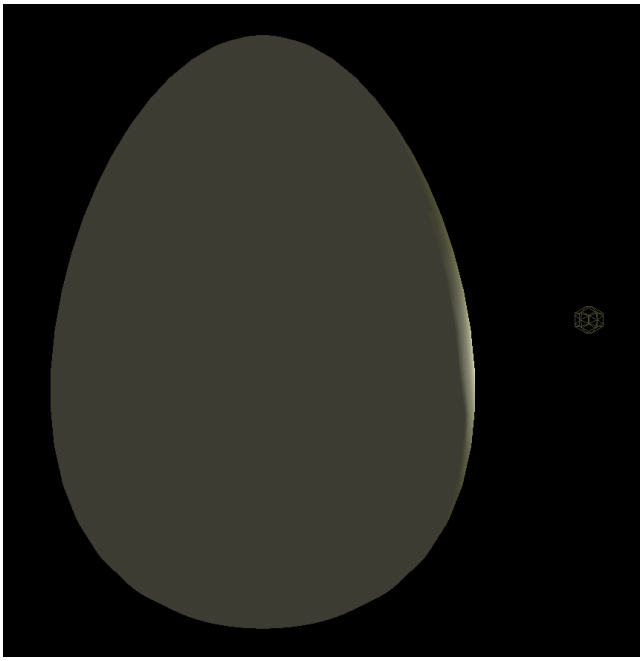
Rys 2 Maksymalny ambient



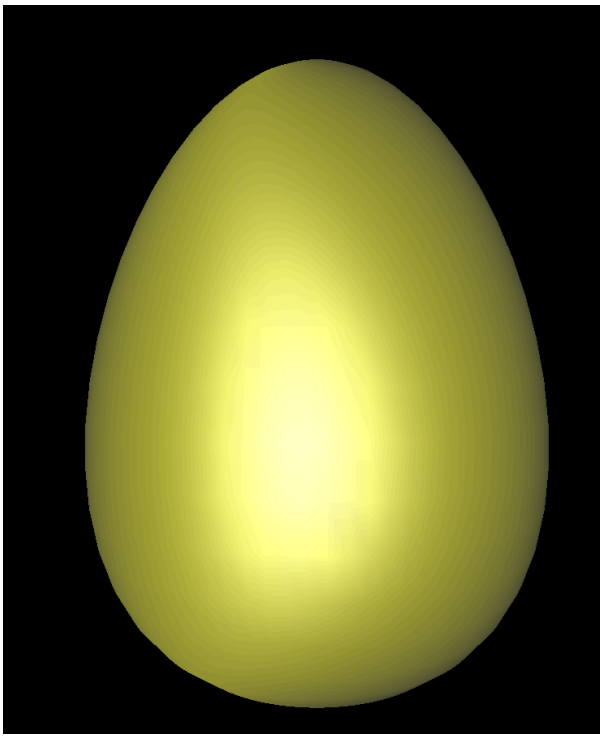
Rys 3 Minimalny ambient

4.3. Poruszanie źródłami światła i ich wizualizacja

Celem tego fragmentu kodu było umożliwienie dynamicznego poruszania źródłem światła w przestrzeni trójwymiarowej oraz wizualizacja jego aktualnej pozycji za pomocą sfery. Realizacja tego zadania wymagała odpowiedniego śledzenia interakcji użytkownika i przekształcania współrzędnych źródła światła w zależności od ruchów myszy. Najpierw zrealizowano mechanizm zmiany pozycji źródła światła w oparciu o przesunięcia myszy. Gdy użytkownik naciska lewy przycisk myszy, zmiana pozycji w osi pionowej (delta_y) powoduje zmianę kąta phi, który odpowiada za pozycję światła w osi Y. Z kolei naciskając prawy przycisk myszy, przesunięcie w osi poziomej (delta_x) modyfikuje kąt theta, który wpływa na pozycję światła w osiach X i Z. Dzięki tym zmianom kąty theta i phi kontrolują obrót źródła światła wokół modelu. Na podstawie zmodyfikowanych wartości theta i phi obliczono nowe współrzędne źródła światła w przestrzeni kartezjańskiej. Współrzędna X została wyznaczona jako iloczyn promienia R oraz kosinusów kątów theta i phi, współrzędna Y zależy od sinusa kąta phi, natomiast współrzędna Z wynika z kombinacji sinusa theta i kosinusa phi. Obliczone współrzędne pozwalają precyzyjnie ustalić położenie światła w przestrzeni 3D. Następnie, w celu wizualizacji pozycji źródła światła, użyto funkcji OpenGL. Na początku zapisano bieżący stan macierzy transformacji za pomocą funkcji glPushMatrix. Następnie przetransformowano układ współrzędnych w punkt odpowiadający pozycji źródła światła, korzystając z funkcji glTranslate. Aby zobrazować pozycję światła, stworzono obiekt typu sfera przy użyciu funkcji gluSphere. Sfera ta jest rysowana jako druciana konstrukcja, co zapewnia wizualne odróżnienie jej od pozostałych elementów sceny. Na koniec przywrócono poprzedni stan macierzy transformacji za pomocą glPopMatrix, co zapobiega niezamierzonym przekształceniom innych elementów sceny.



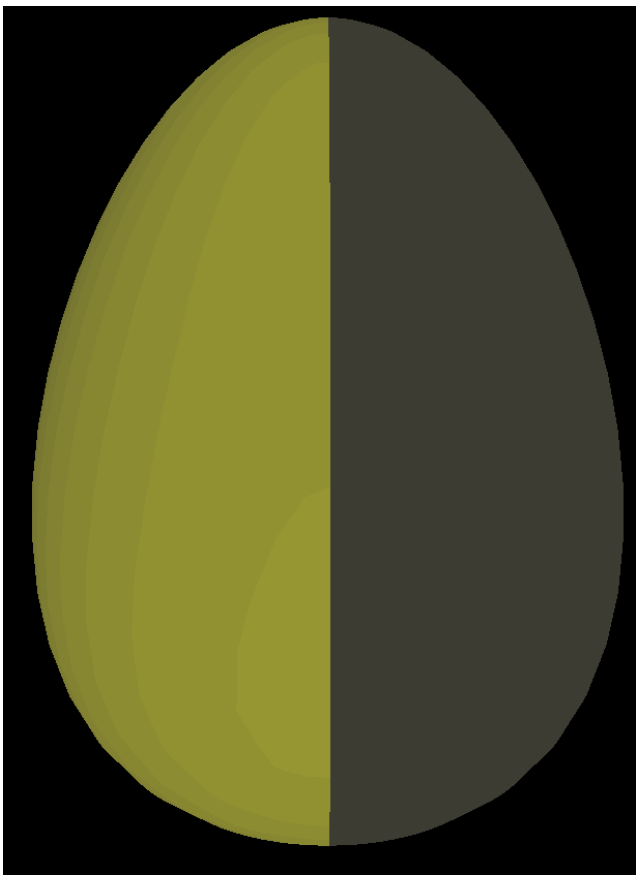
Rys 4 Obracanie kamerą wokół obiektu 1



Rys 5 Obracanie kamerą wokół obiektu 2

4.4. Dodanie wektorów normalnych do modelu jajka

Celem zadania było dodanie wektorów normalnych do modelu jajka, aby poprawnie wyświetlać oświetlenie na jego powierzchni. Wymagało to wcześniejszego stworzenia funkcji generującej wierzchołki modelu oraz obliczenia wektorów normalnych dla każdego wierzchołka.Najpierw wyznaczono współrzędne wierzchołków jajka w przestrzeni trójwymiarowej na podstawie parametrów w układzie parametrycznym. Współrzędne te zostały zapisane w strukturze danych, reprezentując pełny model jajka.Następnie obliczono wektory normalne dla każdego wierzchołka. W tym celu wyznaczono pochodne powierzchni względem parametrów, co pozwoliło określić orientację powierzchni w poszczególnych punktach. Na ich podstawie, wykorzystując iloczyn wektorowy, wyznaczono prostopadłe wektory normalne. Wektory te zostały znormalizowane, aby miały długość jednostkową, co zapewnia poprawne działanie w procesie oświetlenia. Wektory normalne zostały przypisane do odpowiednich wierzchołków modelu. W OpenGL zrealizowano to za pomocą funkcji, która ustawia wektor normalny tuż przed określeniem współrzędnych wierzchołka. Dzięki temu każdy wierzchołek ma poprawnie zdefiniowany wektor normalny, co pozwala na realistyczne obliczanie oświetlenia.

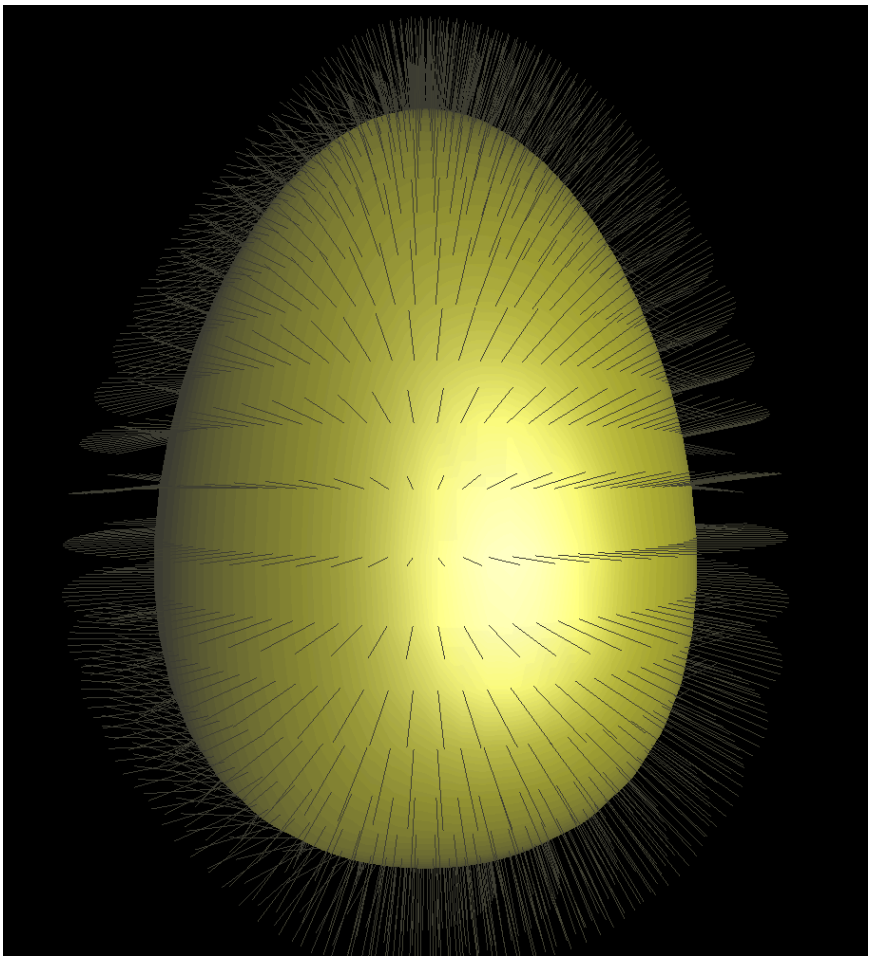


Rys 6 Jajko z wektorami normalnymi

4.5. Wyświetlenie wektorów normalnych i ich poprawa

Celem tego zadania było poprawienie wyświetlania wektorów normalnych na modelu jajka oraz naprawienie ich renderowania w drugiej połowie modelu, aby oświetlenie na całym obiekcie było poprawne. W tym celu wprowadzono dwie zmiany w kodzie, czyli odwrócenie normalnych na dolnej połowie modelu oraz dodanie funkcji umożliwiającej rysowanie tych normalnych w postaci linii. Pierwszym krokiem było zapewnienie poprawności normalnych w drugiej połowie modelu jajka. Model jajka jest symetryczny, a wektory normalne w różnych częściach obiektu muszą być skierowane w odpowiednich kierunkach, aby oświetlenie działało prawidłowo. W tym celu dodano warunek, który odwraca wektory normalne w drugiej połowie modelu, poprzez pomnożenie ich współrzędnych przez -1. Dzięki temu, normalne w dolnej części jajka zostały skierowane w przeciwnym kierunku, co pozwala na uzyskanie poprawnego efektu oświetlenia na całym modelu. Drugą zmianą było dodanie do funkcji odpowiedzialnej za rysowanie jajka fragmentu z wizualizowaniem wektorów normalnych. Funkcja ta obliczała punkty i normalne jajka, a następnie, wykorzystując odpowiednie funkcje OpenGL, rysowała model jajka w postaci

trójkątów. Dodatkowo, po narysowaniu jajka, zaimplementowano możliwość wyświetlania wektorów normalnych w postaci linii, jeśli użytkownik włączył tę opcję. W tym celu, dla każdego wierzchołka jajka, obliczano punkt końcowy wektora normalnego i rysowano linię od wierzchołka do tego punktu, co umożliwiało wizualizację kierunku normalnych na powierzchni jajka. Dzięki tym zmianom, model jajka jest teraz poprawnie oświetlany na całej powierzchni, a wektory normalne pomagają wizualizować, jak oświetlenie wpływa na różne części modelu.



Rys 7 Wyświetlenie wektorów

5. Wnioski

- Udało się zrealizować wszystkie zadania

6. Literatura

- Szymon Datko [GK] Lab6 :: Teksturowanie obiektów