GKOM - Projekty

Wymagania projektu

W ramach projektu należy stworzyć program, który będzie realizował opisane w temacie funkcje. Projekt jest zadaniem zespołowym, gdzie każdy zespół składa się z 3 osób.

Głównym językiem programowania może być język Python lub C++. Do realizacji funkcji graficznych należy wykorzystać bibliotekę OpenGL wraz z językiem shaderów GLSL.

Za projekt można uzyskać maksymalnie $x \times 15p.$, gdzie x to liczba osób w zespole. Każdy z członków zespołu może dostać maksymalnie 15 punktów.

Ocenie w ramach projektu podlegają:

1. Działanie programu - realizacja funkcji (9 p.)
2. Efekty wizualne - prezentacja działania programu oraz kroku algorytmu w przyjemnie wizualny sposób (przygotowanie modeli, scenerii itd.) (2 p.)
3. Jakość kodu (3 p.)
4. Prezentacja wykonana na ostatnim wykładzie (1 p.)

Projekt musi być pokazany odpowiedniemu prowadzącemu przed terminem ostatniego wykładu. Dodatkowo, brak prezentacji na ostatnim wykładzie skutkuje niezaliczeniem projektu.

Terminy

Projekt oddawany jest w 2 etapach:

1. prezentacja wstępnego programu np. implementacja okna, cieniowania i wczytywania obiektów

2. prezentacja ostatecznej wersji programu.

Konieczne jest zaprezentowanie projektu odpowiedniemu prowadzącemu 2 razy - brak pierwszej prezentacji uniemożliwia oddanie projektu.

Zadanie	**Ostateczny termin**
Deklaracja zespołów projektowych	31.10.2025
Przydział projektów	02.11.2025
Oddanie pierwszego etapu projektu	01.12.2025
Oddanie ostatecznej wersji programu	23.01.2026

Tabela 1: Terminy projektowe

1 Subdivision

Prowadzący: dr inż. Łukasz Dąbała

W ramach projektu należy stworzyć program, który będzie umożliwiał wizualizację podziału obiektu:

1. wsparcie dla różnego typu plików wejściowych z modelem 3D: obj
2. cieniowanie Phonga
3. interpretację właściwości materiału: diffuse, specular
4. wsparcie dla tekstur typu diffuse, specular
5. wsparcie dla świateł punktowych - wystarczy pojedyncze światło
6. wsparcie dla algorytmów: Loop'a oraz Catmull-Clark służących do podziału siatki
7. możliwość wizualizacji iteracyjnej algorytmów wymienionych w poprzednim punkcie
8. kamerę perspektywiczną - możliwość poruszania się po scenie oraz obrotu

2 Motion blur

Prowadzący: dr inż. Łukasz Dąbała

W ramach projektu należy stworzyć prostą grę wyścigową, w której dodany zostanie motion-blur

1. wsparcie dla różnego typu plików wejściowych z modelem 3D: obj. Interesuje nas wczytanie trasy, modelu samochodu oraz jego wnętrza.
2. cieniowanie Phonga
3. interpretację właściwości materiału: diffuse, specular
4. wsparcie dla tekstur typu diffuse, specular
5. wsparcie dla świateł punktowych - wystarczy pojedyncze światło
6. kamerę perspektywiczną
   (a) przeczepioną do wnętrza samochodu (możliwość kierowania z pierwszej osoby)
   (b) przyczepioną nad tyłem samochodu (możliwość kierowania z trzeciej osoby)
7. efekt motion-blur
   (a) obiekty powinny rozmazywać się wraz z ruchem
   (b) należy uwzględnić odległość od obiektów - obiekty bliższe rozmazują się bardziej niż dalsze

3 Generator drzew

Prowadzący: dr inż. Łukasz Dąbała

W ramach projektu należy stworzyć program, który będzie umożliwiał tworzenie roślin z wykorzystaniem L-systemów.

W tym celu należy zaimplementować:

1. wsparcie dla podstawowego typu plików wejściowych z modelem 3D: obj - ewentualna baza dla rośliny i podłoże
2. cieniowanie Phonga
3. interpretację właściwości materiału: diffuse
4. kamerę perspektywiczną - możliwość poruszania się po scenie oraz obrotu
5. możliwość generacji roślin:
   (a) możliwość wyboru 3 różnych predefiniowanych roślin
   (b) możliwość podania swoich zasad generacji
   (c) rozsiewanie modeli w zadanym obszarze - należy pamiętać, o transformacjach, żeby roślinność nie wyglądała zbyt sztucznie
6. zmienny kolor rośliny w zależności od jej wysokości

4 Environment Rendering

Prowadzący: dr inż. Michał Chwesiuk

W ramach projektu należy stworzyć program, który będzie renderował teren przy wykorzystaniu mapy wysokości.
W tym celu należy zaimplementować następujące punkty:

1. Przygotowanie geometrii środowiska opartą o wczytaną mapę głębokości
   (a) Wybór pliku zawierającego mapę głębokości może być zaimplementowana w kodzie, występować jako parametr programu, bądź nazwa pliku wczytywana jest z pliku konfiguracyjnego
2. Geometria powinna zawierać macierz punktów N x M, gdzie M i N to ilość próbek z mapy głębokości.
   (a) Punkty powinny być oddalone od siebie we współrzędnych X i Z o stałą odległość, najlepiej definiowaną przez zmienną.
   (b) Współrzędna Y pozycji wierzchołka powinna być wczytywana z mapy głębokości.
   (c) Kolejność renderowania wierzchołków powinna być przekazana do GPU jako Element Buffer Object.
3. Cieniowanie Phong'a
4. Kontekstowe teksturowanie
   (a) Przekazanie do shaderów mininum trzech tekstur.
   (b) Pobranie koloru dla wszystkich tekstur w fragment shaderze.
   (c) Wybranie, bądź blending tekstur w zależności od parametrów danego wierzchołka (np. pozycja, wysokość, nachylenie powierzchni itd.)
5. Dodanie do sceny obiektu imitującą wodę
   (a) Geometria wody jako quad rozciągający się na całą scenę.
   (b) Quad powinien mieć naniesioną teksturę wody.
   (c) Opcjonalnie: Teksturę wody można wykorzystać jako normal mapę, zaaplikowaną we fragment shaderze wody.
   (d) Dodać poruszanie się fal wody np. jako dodanie UV do tekstury modyfikowanej z czasem zmiennej uniform.

5 Particle system editor

Prowadzący: dr inż. Michał Chwesiuk

W ramach projektu należy stworzyć program, który będzie renderował system cząsteczek o ustalonych parametrach.
W tym celu należy zaimplementować następujące punkty:

1. Implementację systemu cząsteczek o ustalonych parametrach, z danym stanem początkowym i aktualizacją po czasie
2. System cząsteczek powinien zawierać następujące parametry:
   (a) Częstotliwość emitowania nowych cząsteczek
   (b) Czas życia danych cząsteczek
   (c) Zanikanie cząsteczek po zakończeniu ich czasu życia
   (d) Wielkość cząsteczek
   (e) Tekstura cząsteczek
   (f) Kierunek ruchu modyfikowany z czasem, z opcjonalnym uwzględnieniem grawitacji
   (g) Prędkość cząsteczek
   (h) Opcjonalnie: oddziaływanie cząstek na siebie
   (i) Opcjonalnie: Przezroczystość cząstek wraz z sortowaniem po kanale alfa
3. Wczytywanie parametrów systemu cząsteczek z pliku konfiguracyjnego
4. Możliwość modyfikowania parametrów systemy cząsteczek
5. Wczytywanie prostej sceny celem wizualizacji cząsteczek na zadanym tle
   (a) Scena powinna zawierać minimum 2 obiekty
   (b) Siatka obiektów może być podstawowa, np. sześcian
   (c) Modele powinny być oświetlone modelem cieniowania Phong'a
   (d) Modele nie muszą posiadać kolizji z wygenerowanymi cząsteczkami
6. Kamerę perspektywiczną - możliwość poruszania się po scenie oraz obrotu

6 Geometry Shader

Prowadzący: dr inż. Michał Chwesiuk

W ramach projektu należy stworzyć program, który będzie renderował efekt wybuchu obiektu wykorzystując Geometry Shader.
W tym celu należy zaimplementować następujące punkty:

1. Tworzenie Geometry Shader, który dzieli obiekt na fragmenty (np. dzieląc trójkąty na mniejsze części).
2. Przesuwanie fragmentów w przestrzeni na podstawie wektora eksplozji.
3. Efekt powinien być dynamiczny i interaktywny, a jego parametry powinny być modyfikowalne w czasie rzeczywistym.
   (a) Możliwość pauzowania i wznawiania efektu.
   (b) Możliwość resetowania efektu i ustawiania obiektu do pozycji startowej.
   (c) Możliwość regulacji siły i kierunku eksplozji.
4. Wsparcie dla różnego typu plików wejściowych z modelem 3D: obj, fbx, etc.
5. (Opcjonalnie) Możliwość dodania losowego rozrzutu kawałków dla bardziej realistycznego efektu.
6. (Opcjonalnie) Zastosowanie instancingu do wydajnego generowania wielu odłamków.
7. Kamerę perspektywiczną - możliwość poruszania się po scenie oraz obrotu.