Instrukcja

Uruchamianie: uruchomić aplikację w Visual Studio w trybie Release, może być konieczne wybranie Retarget Solution z menu kontekstowego otwierającego się po naciśnięciu prawym przyciskiem myszy na Solution w Solution Explorerze.

Lub z msbuild:

msbuild.exe CPUPipeline.sln -p:Configuration=Release

Aplikacja rozpoczyna się w widoku wolnym w którym użytkownik może się poruszać po scenie za pomocą WASD (do przodu / w lewo / do tyłu / w prawo) oraz Q i E (do góry / do dołu) oraz obracać kamerę za pomocą myszki.

W aplikacji po naciśnięciu określonych poniżej klawiszy w widoku wolnym otwierają się odpowiadające im ekrany, aby zamknąć dany ekran należy nacisnąć przycisk którym się go otworzyło.

Wszystkie poniżej opisane skróty klawiszowe widoku wolnego są opisane na ekranie pomocy dostępnym pod klawiszem H.

Sterowanie każdego z ekranów jest opisane na nim samym.

Skróty klawiszowe nieotwierające ekranu:

Wyjście - ESC

Renderowanie:

U – włączenie / wyłączenie wyświetlania tylko siatki figur (wireframe)

I – włączenie / wyłączenie buforowania głębi (depth buffering)

O – włączenie / wyłączenie poprawki pespektywy (perspective fix)

P – włączenie / wyłączenie obcinania ścian tylnych (backface culling)

Edytowanie:

R – zaznaczenie bryły na celowniku

X – usunięcie zaznaczonego obiektu (bryły/światła)

L – zaznaczenie najbliższego światła

K – stworzenie nowego świała 1 jednostkę przed kamerą

Kamera:

WASDQE – poruszanie się

Kółko myszy – zmiana pola widzenia (fov)

G – stworzenie nowej kamery 1 jednostkę przed aktualną

F – przejście do kolejnej kamery

J – usunięcie aktualnej kamery i przejście do kolejnej

,/. (przecinek/kropka) – zmniejszenie/zwiększenie przedniej płaszczyzny obcinania

SHIFT + ,/. – zmniejszenie/zwiększenie tylnej płaszczyzny obcinania

Zapisywanie i wczytywanie:

1-5 – wczytanie zapisu z odpowiadającego danej liczbie slotu

SHIFT + 1-5 – zapisanie do odpowiadającego danej liczbie slotu

Sloty znajdują się w podkatalogu data.

Nazwa pliku zawierającego slot to save{n} gdzie {n} to numer slotu

Dostępne ekrany:

H – pomoc widoku wolnego

C – tworzenie bryły – wybierz bryłę do stworzenia za pomocą 1-4

V – edycja zaznaczonej bryły/światła

-Jeśli jest zaznaczona bryła otwiera się ekran edycji bryły, użytkownik może w nim edytować własności siatki, materiału oraz przekształcenia bryły poprzez wpisywanie nowych wartości w ich pola.

Poruszanie się po polach za pomocą TAB lub SHIFT+TAB,

Zatwierdzenie i aktualizacja – ENTER

Załadowanie tekstury – L

Załadowanie mapy normalnej – N

Powrót do koloru stałego, kiedy obiekt używa tekstury – M

-Jeśli jest zaznaczone światło otwiera się ekran edycji światła, użytkownik może w nim edytować położenie, kolory oraz siłę światła, sterowanie jak w ekranie edycji bryły

TAB – lista brył, świateł i kamer na scenie (przewijalna za pomocą kółka myszy)

Lokalizacja implementacji wymagań:

Edycja sceny

- Dodawanie, edycja i usuwanie ze sceny figur geometrycznych złożonych z siatki trójkątów:

Dodawanie – CreateObjectScreen tworzony przez Editor po naciśnięciu C (showCreateScreen()), po wybraniu bryły tworzy ją z domyślnymi parametrami za pomocą odpowiedniej implementacji VirtualMeshGenerator i zwraca edytorowi, który dodaje ją do sceny

Edycja – EditObjectScreen tworzony przez Editor po naciśnięciu V gdy zaznaczona jest figura (showEditObjectScreen()), pozwala na edycję parametrów bryły

Usuwanie – Editor po naciśnięciu X usuwa obiekt ze sceny (deleteSelectedObject())

- Edycja wybranych figur poprzez zmianę ich parametrów

EditObjectScreen czyta nowe parametry figury podane przez użytkownika i przekazuje je implementacji VirtualMeshGenerator związanej z Mesh edytowanego obiektu, która tworzy nową siatkę dla figury

Prostopadłościan – CuboidMeshGenerator

Kula – SphereMeshGenerator

Walec – CyllinderMeshGenerator

Stożek – ConeMeshGenerator

- Dodawanie i usuwanie świateł punktowych

Dodawanie – Editor (addLight())

Usuwanie – Editor (deleteSelectedLight())

∗ Edycja parametrów światła (kolor, intensywność, zasięg (attenuation)). – EditLightScreen tworzony przez Editor po naciśnięciu V gdy zaznaczone jest światło (showEditLightScreen()), pozwala na edycję parametrów światła.

- Dodawanie i usuwanie kamer

Dodawanie – Editor (createNewCamera())

Usuwanie – Editor (deleteCurrentCamera())

Przełączanie się pomiędzy istniejącymi kamerami – Editor (swapToNextCamera())

Obsługa kamery myszką – Editor (rotateCamera(), moveCamera())

Zmiana parametrów kamery – Editor (updateCameraClippingPlanesAndFov())

- Edycja figur na scenie (przesuwanie, skalowanie, obracanie)

Zmiania przez EditObjectScreen, nowa pozycja, rotacja (kąty Eulera) i skala przekazywane do Mesh figury, który tworzy odpowiednie macierze transformacji za pomocą TransformationMatrices

- Lista obiektów na scenie

ListScreen tworzony w Edytorze po naciśnięciu TAB

- Wczytywanie i zapisywanie sceny do pliku

Po naciśnięciu 1-5 (SHIFT + 1-5) tworzy się obiekt PersistentStorage z odpowiednią ścieżką, następnie wywoływana jest jego metoda load (save) która wczytuje (zapisuje) scenę z pliku binarnego.

Każdy obiekt który można zapisać lub wczytać jest typu SaveableObject, w celu jego wczytania lub zapisu wywołuje się jego metodę load/save (w przypadku klas abstrakcyjnych funkcję loadStatic). Poprzez wywołania kolejnych metod load/save czytana lub zapisywana jest hierarchia obiektów typu SaveableObject o korzeniu w Scenie.

SceneDataReader i SceneDataWriter pozwalają na odczyt/zapis prymitywów takich jak float, int, glm::vec3, czy tablica bajtów

Potok renderowania

- Rasteryzacja tylko przy pomocy funkcji rysowania pikseli – RenderThread przy wypełnianiu ScanLinii wywołuję metodę SetPixel FrameBuffera
- Wypełnianie algorytmem scan linii – SceneRenderer w ScanLineHorizontalBase przygotowuje parametry scan linii i przekazuje je do RenderThread za pomocą kolejki blokującej.
- Obcinanie niewidocznych części trójkątów – SceneRenderer w metodzie DrawClippedTriangle wywołuje metodę ClipTriangle TriangleClippera a następnie dzieli otrzymany wielokąt na trójkąty które wypełnia algorytmem scan linii.
- Algorytm obcinania ścian tylnych – SceneRenderer (drawObjectTriangles())
- Interpolacja atrybutów wierzchołków z korekcją perspektywy: SceneRenderer dla każdego trójkąta figury ustawia wartości interpolatorów każdego z parametrów w InitInterpolators, interpolacją każdego z parametrów zajmuje się oddzielny TriangleInterpolator, wszystkie Interpolatory znajdują się w strukturze Interpolators, zarząda nimi InterpolatorsManager, któremu przekazuje się pozycje wierzchołków i poprawkę perspektywy, on dla każdego punktu wylicza odpowiednie wagi barycentryczne z korekcją perspektywy i przekazuje je TriangleInterpolatorom
- Przełączanie się pomiędzy wypełnianiem trójkątów a rysowaniem samych krawędzi:

Wypełnianie: ScanLine w SceneRenderer

Rysowanie samych trójkątów: WireFrame w SceneRenderer

Przełączanie się: w Editor

- Algorytm buforowania głębi

SetPixel w FrameBuffer

Cieniowanie trójkątów

- Cieniowanie Phonga – InitInterpolators() w SceneRenderer, wartości w punktach uzyskiwane z *Interpolator.getValue()
- Model oświetlenia Phonga – odpowiedni kolor piksela ustalany w getPixelColor w RenderThread.

- Teksturowanie – Obraz zapisany w obiekcie typu Image, dostęp do obrazu za pomocą ImageSampler któremu przekazuje się odpowiednie współrzędne UV przy rasteryzacji
- Mapowanie normalnych – macierz TBN jako parametr wierzchołka, interpolowana, przekształcony wektor normalny uzyskany przez przemnożenie wartości z ImageSampler przez TBN

Inne

- Własne klasy macierzy i wektorów – glm::vec3, glm::mat4, macierze przekształceń w TransformationMatrices
- Skalowalne okno aplikacji – odpowiednie aspect ratio obliczane w UpdateCameraClippingPlanesAndFov() w Editor i przekazywane jako parametr do setPerspective głównej kamery
- Wyświetlanie liczby klatek na sekundę – main.cpp