STRESZCZENIE

Cieniowanie odroczone jest techniką grafiki 3D czasu rzeczywistego popularną wśród twórców gier komputerowych pozwalającą na obsługę wielu światłen na scenie bez znaczącego spadku wydajności.

W niniejszej pracy zaprojektowano i zaimplementowano silnik renderujący używając języka C i biblioteki graficznej Vulkan. Opisano elementy silnika renderującego oraz nisko i wysokopoziomowe techniki graficzne używane w nowoczesnych grach 3D z naciskiem na renderowanie odroczone. Opisano architekturę silnika i szczegóły implementacji. Wyrenderowano przykładową scenę i zbadano wydajność użytych technik graficznych.

Słowa kluczowe: silnik renderujący, renderowanie odroczone, renderowanie bez dowiązań, renderowanie pośrednie, Vulkan, glTF.

Dziedzina nauki i techniki według OECD: 1.2 Nauki o komputerach i informatyka.

ABSTRACT

// TODO

SPIS TREŚCI

Streszczenie . . . 1
Abstract . . . 2
Spis treści . . . 3
Wykaz ważniejszych oznaczeń i skrótów . . . 4
1. Wstęp . . . 5
    1.1. Cel pracy . . . 5
    1.2. Zakres pracy . . . 6
    1.3. Struktura pracy . . . 6
2. Wprowadzenie do dziedziny . . . 7
    2.1. GPU . . . 7
    2.2. Potok graficzny . . . 7
    2.3. Potok zasobów . . . 7
    2.4. Vulkan . . . 7
    2.5. Mapowanie tekstur . . . 8
3. Technologie, algorytmy i narzędzia . . . 9
    3.1. Użyte narzędzia . . . 9
        3.1.1. Proces budowania . . . 9
        3.1.2. Biblioteki zewnętrzne . . . 9
4. Projekt systemu . . . 10
5. Badania . . . 11
6. Podsumowanie . . . 12
Wykaz literatury . . . 13
Spis rysunków . . . 14
Spis tablic . . . 15

WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

Skróty:

- API - Application Programming Interface, interfejs programistyczny aplikacji;
- CPU - Central Processing Unit, procesor;
- GPU - Graphics Processing Unit, procesor graficzny.

1. WSTĘP

Podręcznik [1] definiuje grafikę komputerową jako dziedzinę interdyscyplinarna zajmującą się komunikacją wizualną za pomocą wyświetlacza komputera i jego urządzeń wejścia-wyjścia.

Rozwój teoretyczny grafiki komputerowej jest zdominowany przez doroczną konferencję SIGGRAPH [2], podczas której prezentacje i dyskusje akademickie są przeplecione z targami branżowymi. Rozwój grafiki komputerowy jest w znacznej mierze napędzany wymaganiami stawianymi przez przemysł rozrywkowy. Przykładem jest dobrze znana seria kursów przeznaczona dla twórców gier komputerowych obejmująca najnowsze prace i postępy w technikach renderowania czasu rzeczywistego używanych w silnikach graficznych rozwijanych przez producentów gier komputerowych [3].

Renderowanie to proces konwersji pewnych prymitywów na obraz przeznaczony do wyświetlania na ekranie. Wyświetlany obraz jest nazywany klatką (ang. frame).

Renderowanie czasu rzeczywistego nakłada ograniczenie czasowe dotyczące liczby klatek na sekundę (ang. frames per second, FPS), która musi być na tyle wysoka, by dawać iluzję ciągłości ruchu. Przyjmuje się, że ograniczenie to jest spełniane poprzez zapewnienie wyświetlanie minimum 30 klatek na sekundę (renderowanie trwa krócej niż $\frac{1}{30}$ sekundy). Eliminuje to kosztowne obliczeniowo techniki renderowania dające fotorealisteczne rezultaty takie jak śledzenie promieni (ang. ray tracking) i wymaga od programistów zastosowania technik aproksymacji mniej lub bardziej luźno opertych na prawach fizyki oraz użycia bibliotek graficznych wspierających akceleracją sprzętową takich jak Vulkan lub OpenGL.

Silnik renderujący, zwany też silnikiem graficznym to element aplikacji odpowiadający za renderowanie czasu rzeczywistego. Zapewnia on wysokopoziomową warstwę abstrakcji pozwalającą użytkownikowi na operowanie używając takich konceptów jak sceny, obiekty, materiały lub światła oraz ukrywają niskopoziomowe detale użytych bibliotek i technik graficznych.

Zaprojektowanie i zaimplementowanie silnika graficznego jest złożonym procesem wymagającym znajomości szerokiego wachlarza technik graficznych z całego możliwego spektrum poziomów abstrakcji, dlatego też coraz więcej twórców gier komputerowych decyduje się na licencjonowanie i użycie gotowego silnika graficznego zamiast powolnej i mozolnej pracy nad własnymi rozwiązaniami.

Jeśli jednak celem inżyniera jest poszerzenie osobistego zrozumienia grafiki komputerwej, to warto podjąć próbę stworzenia własnego silnika graficznego.

1.1. Cel pracy

Celem pracy jest zaprojektowanie i zaimplementowanie silnika graficznego, którego potok graficzny używa techniki cieniowania odroczonego.

Silnik został napisana jako biblioteka programistyczna języka C używającej skryptów Python do automatycznej generacji kodu podczas procesu budowania. Renderowanie grafiki 3D jest obsługiwane przez Vulkan API. Zasoby uzywane podczas renderowania są wczytywane z bazy zasobów, która jest wyjściem potoku zasobów działającego podczas procesu budowania. Działanie biblioteki jest sterowane plikiem konfiguracyjnym dostarczonym przez użytkownika i jest demonstrowane przy użyciu pliku wykonywalnego renderujacego przykładową scenę używając cieniowania odroczonego.

Celem autora było zapoznanie się z teorią stojącą za elementami składającymi się na silnik graficzny i praktyczne zademonstrowanie zdobytej wiedzy.

1.2. Zakres pracy

Niniejsza praca ma charakter przeglądowy. Nowoczesne silniki graficzne składają się z wielu elementów, z których każdy może być niezależnie rozwijany do dowolnie wysokiego poziomu skomplikowania, dlatego trudno je wszystkie dokładnie i wyczerpująco opisać w ramach jednej pracy.

Zakres pracy obejmuje:

- opis algorytmów i technik graficznych używanych w nowoczesnych silnikach graficznych, ze szczególnym naciskiem na Vulkan API i renderowanie odroczone,
- omówienie architektury i implementacji projektu,
- demonstrację użycia silnika graficznego do wyrenderowania przykładowej sceny,
- analizę wydajności silnika graficznego.

Stworzony silnik nie może konkurować z silnikami graficznymi profesjonalnie rozwijanymi przez duże drużyny z myślą o zastosowaniu w grach komputerowych (takimi jak otwarty Godot [4] czy komercyjny Unreal Engine [5]). Jest on jednak przystosowany do względnie łatwej, szybkiej i elastycznej modyfikacji potoku graficznego oraz wspiera mechanizmy zgłaszania informacji debugowania oferowane przez Vulkan API, co pozwala na szybki cykl prototypowania i debugowania podczas zapoznawania się z technikami graficznymi.

1.3. Struktura pracy

// TODO

2. WPROWADZENIE DO DZIEDZINY

W tej sekcji przybliżono serię pojęć oraz technik związanych z grafiką komputerową, których zrozumienie jest wymagane przed rozpoczęciem implementacji silnika graficznego.

2.1. Podstawowe pojęcia

// TODO

2.2. Potok graficzny

// TODO

2.3. Potok zasobów

// TODO

2.4. Vulkan

// TODO

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Obrazek kawusi to rys. 2.1 [6].

Rysunek 2.1: Smacznej kawuni

Nunc venenatis in felis lobortis vehicula.

2.5. Mapowanie tekstur

// TODO

3. TECHNOLOGIE, ALGORYTMY I NARZĘDZIA

3.1. Użyte narzędzia

Silnik została napisany jako biblioteka w języku C w standardzie C11. Budowanie biblioteki ze źródeł wymaga generacji dodatkowego kodu przy pomocy skryptów w języku Python w wersji 3.9.7.

Silnik został opracowany na maszynie z systemem Linux i wspiera systemy Linux i Windows. Projekt został w całości napisany przy użyciu środowiska programistycznego CLion w wersji 2021.2.3.

Podczas pracy stosowano rozproszony system kontroli wersji git. Repozytorium jest utrzymywane na serwisie GitHub.

Pliki *.clang-tidy* i *.clang-format* znajdujące się w strukturze plików projektu pozwalają na automatyczne formatowanie kodu źródłowego zgodnie ze uprzednio zdefiniowanym standardem kodowania.

Proces budowania projektu jest zautomatyzowany przy użyciu narzędzia CMake, które w przypadku języków C i C++ jest praktycznie standardem podczas rozwoju wieloplatformowych projektów.

3.1.1. Proces budowania

// TODO

3.1.2. Biblioteki zewnętrzne

// TODO

4. PROJEKT SYSTEMU

5. BADANIA

6. PODSUMOWANIE

// TODO

WYKAZ LITERATURY

[1] J. F. Hughes i in., *Computer Graphics: Principles and Practice*, 3 wyd. Upper Saddle River, NJ: Addison-Wesley, 2013.

[2] *Konferencja SIGGRAPH*, 2022. adr.: `https://www.siggraph.org/`.

[3] *Advances in Real-Time Rendering in 3D Graphics and Games*, 2022. adr.: `https://advances.realtimerendering.com/`.

[4] Społeczność Godot Engine, *Godon Engine*, 2022. adr.: `https://godotengine.org`.

[5] Epic Games, *Unreal Engine*, wer. 5, 2022. adr.: `https://www.unrealengine.com`.

[6] Piotr Piwowarczyk, "Microsoft DirectX 12 Feature Level 12_2 - nowa wersja funkcji API `https://www.purepc.pl/microsoft-directx-12-feature-level-12-2-nowa-wersja-funkcji-api`", (**urldate** 2020-11-02).

SPIS RYSUNKÓW

2.1 Smacznej kawuni 7

SPIS TABLIC