POLITECHNIKAWROCŁAWSKA

WYDZIAŁELEKTRONIKI

KIERUNEK: INFORMATYKA (INF)

SPECJALNOSC: INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH (INS)

PRACA DYPLOMOWA

INŻYNIERSKA

Symulator graficzny procesu fizycznego w środowisku DirectX

Graphic simulator of a physical process in DirectX

AUTOR:

Wojciech Sobczak

PROWADZACY PRACĘ:

doc. dr inż. Jacek Jarnicki, ZSKiD

OCENA PRACY:

WROCŁAW, 2016

**Spis treści**

[Spis rysunków 4](#_Toc466844123)

[Spis tabel 5](#_Toc466844124)

[Spis listingów 6](#_Toc466844125)

[Skróty 7](#_Toc466844126)

[1. Wstęp 8](#_Toc466844127)

[1.1. Wprowadzenie 8](#_Toc466844128)

[1.2. Cel i zakres pracy 8](#_Toc466844129)

[2. Problem 9](#_Toc466844130)

[2.1. Sformułowanie problemu 9](#_Toc466844131)

[2.2. Przegląd dostępnych rozwiązań 9](#_Toc466844132)

[2.2.1. Unity 9](#_Toc466844133)

[2.2.2. Unreal Engine 10](#_Toc466844134)

[2.2.3. Blender 10](#_Toc466844135)

[2.2.4. Autodesk 3DSMax 11](#_Toc466844136)

[3. Aplikacja 13](#_Toc466844137)

[3.1. Założenia 13](#_Toc466844138)

[3.2. Przyjęte technologie 13](#_Toc466844139)

[3.2.1. DirectX 13](#_Toc466844140)

[3.2.2. Bullet Physics 15](#_Toc466844141)

[3.2.3. DirectXTK 16](#_Toc466844142)

[3.3. Opis działania 16](#_Toc466844143)

[3.4. Opis użytkowania 16](#_Toc466844144)

[4. Testy 17](#_Toc466844145)

[4.1. Plan testów 17](#_Toc466844146)

[4.2. Wyniki 17](#_Toc466844147)

[4.3. Wnioski 17](#_Toc466844148)

[5. Literatura 18](#_Toc466844149)

# Spis rysunków

[Rys. 1. Przykład podpisu rysunku 4](#_Toc465685478)

# Spis tabel

[Tab. 1. Przykład podpisu tabeli 4](#_Toc465685652)

# Spis listingów

[Listing. 1. Początkowe żądanie HTTP 12](#_Toc466828340)

# Skróty

**GPL** (ang. *General Public License*)

**API** (ang. *Application Programming Interface*)

**SIMD** (ang. *Single Instruction, Multiple Data*)

1. Wstęp

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

* 1. Wprowadzenie

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

* 1. Cel i zakres pracy

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

1. Problem
   1. Sformułowanie problemu

Właściwie to nie miałem z tym problemu.

* 1. Przegląd dostępnych rozwiązań

W przypadku tego problemu, ciężko jest znaleźć rozwiązanie okalające tak wąski zakres funkcjonalności. Dzieje się tak dlatego, że programy implementujące różnorakie biblioteki symulujące fizykę, są zazwyczaj programami pełniącymi funkcję edytorów scen. Edytory te są rozbudowane, zarówno pod względem implementowanych procesów fizycznych, jak i znacznie bardziej skomplikowanych, niż w tej pracy, efektów graficznych opartych o niezwykle skomplikowany aparat matematyczny. Przykładem takich efektów mogą być systemy cząsteczkowe (particles systems), które np. pozwalają w sposób relatywnie realistyczny symulować zachowanie wody. Przykłady więc będą opisywać programy, których zakres pracy jest znacznie bardziej rozbudowany niż opisywanej tu aplikacji, nie mniej, pozwalający na zrealizowanie dokładnie tych samych procesów.

* + 1. Unity

Produkt „Unity Game Engine”, firmy „Unity Technologies”, jest to środowisko pozwalające na w pełni programowalne budowanie trójwymiarowych oraz dwuwymiarowych scen, z pomocą graficznego edytora. Docelowo produkt reklamowany jest jako narzędzie do tworzenia gier komputerowych. Jest to narzędzie, którego wytworzony zestaw scen, może zostać odtworzony na 22 środowiskach uruchomieniowych do których należą np: Windows, OSX, Linux w dystrybucji firmy Canonical, Ubuntu, PS3, PS4, Xbox360, XONE, oraz wiodące platformy mobilne takie jak Android, IOS oraz BlackBerry.

Rozwiązanie jest niezwykle popularne wśród początkujących twórców gier ze względu na prostotę tworzenia scen, jednak zdarzają się produkcję typu „AAA” (gry wysokobudżetowe), których architektura opiera się o właśnie to rozwiązanie. Przykładem takiego produktu może być „Torment: Tides of Numenera” lub „Pillars of Eternity”.

Większość etapu budowania polega na przeciąganiu myszą z menu edytora, do menu opisywanych obiektów, jako właściwości, oraz nadawaniu parametrów tym właściwościom, takich jak w przypadku obiektu fizycznego masa, środek masy itd.

Możliwość programowania interakcji oraz samych scen została zaimplementowana poprzez umożliwienie użytkownikom używania 3 języków programowania. Najpopularniejszym wyborem, a zarazem najmniej hermetycznym w obrębie środowiska Unity, jest C#, jednak istnieje możliwość wyboru innych języków takich jak „Boo”, który składnią przypomina język „Python” oraz autorski produkt twórców silnika, „Unity Script”, który jest bardzo podobny do języka rozwiązań webowych „JavaScript”.

* + 1. Unreal Engine

„Unreal Engine” jest to produkt firmy „Epic Games”, twórców takich gier jak „Unreal Tournament” czy „Gears of War”, który podobnie jak Unity, jest pełnoprawnym silnikiem do tworzenia gier oraz różnorakich scen trój i dwuwymiarowych. Znaczącą jednak różnicą jest fakt, że silnik ten, dużo częściej pada wyborem twórców wysokobudżetowych gier oraz symulacji. Dzieje się tak dlatego, że silnik ten pozwala na budowanie aplikacji w języku C++ co daje nam znacznie większe możliwości kontroli nad programem niż wysokopoziomowe języki silnika Unity. Drugim powodem częstszego wybierania tego rozwiązania może być fakt, że Epic Games silnik ten tworzy od roku 1998, co daje 8 lat doświadczenia więcej, niż firmie Unity Technologies.

Tworzenie scen oraz definiowanie obiektów w tym silniku to również przeciąganie właściwości z menu to menu obiektu, także pod względem ogólnym te dwa rozwiązania nie różnią się znacząco.

Unreal Engine to również rozwiązanie multiplatformowe, pozwalające wydać aplikację na różne systemy po prostu definiując profile ustawień kompilacji.

Najpopularniejszymi przykładami pokazującymi możliwości tego rozwiązania mogą być „Batman: Arkham Knight”, czy też polska gra „Hatred”.

* + 1. Blender

Blender jest to program do modelowania obiektów 3D oraz tworzenia animacji. Jest to rozwiązanie typu OpenSource na licencji GPL, którego głównym programistą jest, i był od samego początku powstawania programu, Ton Roosendaal. Program ten oprócz możliwości modelowania oferuje wachlarz innych możliwości przez co jest on niezwykle popularny na rynku.

Ze wspomnianych wcześniej możliwości wymienić trzeba, że program posiada własny silnik renderujący, który umożliwia szeroką ingerencję użytkownika w proces rysowania obiektu na ekranie. Do dyspozycji użytkownika oddano np. możliwość tworzenia własnych filtrów oraz shaderów, możliwość użycia rendererów innych niż domyślny Blender Internal, takich jak YafaRay czy LuxRender oraz możliwość definiowania własnych dając do dyspozycji możliwość pisania skryptów w języku Python.

Blender wykorzystuje silnik fizyczny Bullet Physics, użyty również w implementacji pracy inżynierskiej. Pozwala on na budowanie dokładnych symulacji fizycznych z wykorzystaniem modeli tworzonych wewnątrz programu co przekłada się na wygodę użytkowania, gdyż raz wymodelowane przez nas obiekty posiadają zintegrowaną fizykę, której wystarczy tylko nadać odpowiednie właściwości liczbowe.

Rozwiązanie to, ze względu na wymienione wyżej funkcje, pozwala również na tworzenie efektów specjalnych, filmów i animacji, czego przykładem może być produkcja „Big Buck Bunny” oraz „Big Miracle”.

* + 1. Autodesk 3DSMax

3DS Max jest w pełni zamkniętym i komercyjnym produktem firmy Autodesk. Jest on, podobnie jak Blender, programem do tworzenia zaawansowanych modeli 3D oraz tworzenia animacji. Jako że nie jest to rozwiązanie tworzone przez społeczność, posiada znacznie większe i bardziej stałe zaplecze twórców oraz wsparcie, dlatego też jest częściej wybierane w komercyjnych rozwiązaniach.

Jeżeli chodzi o użytkowanie, Blender i 3DSMax są bardzo zbliżone w sensie ogólnym. Definicja obiektów to przeciąganie obiektów z menu, modelowanie siatek odbywa się poprzez przesuwanie poszczególnych linii, itd. Dodano doń również obsługę języka MAXScript, jednak nie umożliwia on ingerencji w proces budowania obrazu, a tylko dodaje możliwość definiowania własnych wtyczek do programu, czy też automatyzowania niektórych procesów twórczych.

Dokładnie tak samo jak w Blenderze, mamy tu możliwość tworzenia symulacji fizycznych, jednak w tym przypadku mamy do czynienia z symulacją wykorzystującą wewnętrzną implementacji fizyki firmy Autodesk. Umożliwia ona nam np. budowanie symulacji cząsteczkowych oraz tworzenie symulacji tkanin.

O przewadze tego rozwiązania nad jego darmowym konkurentem może stanowić portfolio jego filmów, gdyż użyto go do stworzenia efektów specjalnych w produkcjach takich jak „Mad Max: Fury Road”, „The Curious Case of Benjamin Button” czy „Black Hawk Down”.

1. Aplikacja
   1. Założenia

Stworzony symulator ma za zadanie ukazać trzy scenariusze prezentujące różne układy obiektów, które będzie można za pomocą sześcianów „wystrzeliwanych” przez użytkownika niszczyć, w sensie, zaburzać równowagę tych konstrukcji, doprowadzając je do upadku. Ukazane scenariusze to:

* **House of cards** - konstrukcja przypominającą domek z kart, skonstruowana z kostek o kształcie kostek domina
* **Towers** – wieże złożone z ułożonych na sobie sześciennych kostek, które ustawione są na okręgu wokół osi Y przestrzeni
* **Domino** – ułożone w spirale kostki, o kształcie kostek domina

Scenariusze te dostępne będą poprzez rozwijane menu na pasku głównym okna aplikacji. Dodatkowo aplikacja będzie umożliwiała swobodne poruszanie się użytkownika po przestrzeni udostępnionej przez aplikację, umożliwiając tym samym zmianę kierunku oraz kąta wystrzeliwanego obiektu.

Aplikacja ta działać będzie w środowisku Windows, począwszy od wersji 10. W folderze głównym aplikacji wymagać będzie obecności plików tekstur, które zostaną dostarczone wraz z plikiem wykonywalnym.

* 1. Przyjęte technologie
     1. DirectX

Główną technologią użytą w tym projekcie, będącą za razem częścią jego tematu, jest biblioteka DirectX.

DirectX jest stworzonym przez Microsoft, zestawem relatywnie wysokopoziomowych API, skonstruowanych z myślą o obsłudze multimediów takich jak grafika 3D, dźwięk i inne, ukierunkowany został jednak w stronę tworzenia gier wideo oraz symulacji. Został on w całości zaimplementowany w języku C++ budując wielopoziomową strukturę obiektową, tworzącą abstrakcję nad używanymi sprzętowymi komponentami.

Na cały pakiet składa się:

* **Direct3D (D3D)** – biblioteka obsługująca renderowanie obiektów 3D
* **DXGI** – biblioteka udostępniająca interfejsy programistyczne, tworzące logiczną warstwę pomiędzy sprzętęm potrzebnym do wyświetlania grafiki, a programistą. Udostępnia ona interfejsy dla monitorów (ekranów wyświetlania), kart graficznych, oraz umożliwia samodzielne zarządzania buforami klatek.
* **Direct2D** - biblioteka obsługująca renderowanie obiektów 2D
* **DirectWrite** - biblioteka obsługująca renderowanie czcionek
* **DirectCompute** - biblioteka umożliwiające prowadzenie zrównoleglonych obliczeń na karcie graficznej
* **DirectSound3D (DS3D)** – biblioteka umożliwiająca odtwarzanie dźwięku oraz jego manipulację
* **DirectX Media** – jest to zestaw bibliotek skupiający następujące narzędzia:
  + **DirectAnimation –** do tworzenia animacji 2D/3D w środowisku webowym
  + **DirectShow** – do odtwarzania multimediów oraz ich strumieniowania
  + **DirectX Transform –** do obsługi, edycji oraz animacji obrazów
* **DirectX Diagnostics (DxDiag)** – diagnostyczne narzędzie dla systemu Windows, pozwalające na tworzenie raportów na temat obecnych w systemie komponentów DirectX, takich jak urządzenia audio, wideo oraz urządzenia wejścia, takie jak kontrolery gier.
* **DirectX Media Objects** – jest to biblioteka podobna do zdeprecjonowanej już DirectShow. Służy do strumieniowego przesyłania i przetwarzania danych z wejścia do wyjścia. Zwykle są to interfejsy kodeków służące do przetwarzania danych multimedialnych
* **DirectSetup** – narzędzie stworzone do instalacji komponentów DirectX, oraz wykrywania ich wersji
* **XACT3** – stworzona z myślą o platformie XBOX wysokopoziomowa biblioteka do wieloplatformowej, w obrębie firmy Microsoft, obsługi dźwięku.
* **XAudio2** – biblioteka do obsługi dźwięku podobna do XACT3, jednak z tą różnicą, że udostępniony tu interfejs jest znacznie niżej poziomowy.

Z ogromu tych funkcjonalności tego pakietu, w tym projekcie został użyty tylko Direct3D w wersji 11. Biblioteka ta zawiera w sobie zdefiniowane nagłówki do biblioteki DirectXMath, które to są jedną z ważniejszych jej modułów.

DirectXMath jest główną i polecaną przez twórców tutoriali oraz dokumentacji Microsoftu, biblioteką do ułatwienia obliczeń. Jest to o tyle pożyteczna biblioteka, że jej typy zaprojektowane zostały pod kątem obsługi ich przez urządzenia wpierające SIMD, oczywiście jest to tylko dodatek, o którym zdecyduje środowisko kompilacji, całość natomiast kompatybilna jest z najbardziej podstawowym zestawem operacji w obrębie języka C++.

Udostępnia ona API do operacji na wektorach różnej długości (od 2 do 4), macierzy (3x3 oraz 4x4) oraz operacji zachodzących pomiędzy nimi, takich jak mnożenie, odwracanie, transformacja i inne. Dodatkowo jest ona dostosowana do obsługi operacji najczęściej używanych i potrzebnych w tworzeniu aplikacji 3D, np. rotacje wektorów, transformacje wektorów na podstawie kwaternionów czy też różnych macierzy (macierze obrotów, skalowania, translacji), łączenia tych macierzy w macierze transformacji, tworzenie macierzy widoków, perspektyw i wiele, wiele innych przydatnych funkcjonalności, których długo by wymieniać.

Jest to technologia na tyle rozwinięta i popularna, że wykorzystywana jest niemal w każdym rozwiązaniu dla technologii gier wideo. DirectX implementują wcześniej już wspomniane Unity oraz Unreal Engine, oraz każdy silnik chcący posiadać możliwość uruchomienia swojej aplikacji na systemie Windows czy XBOX z najbardziej optymalnym wsparciem sprzętowym.

* + 1. Bullet Physics

Bullet Physics jest to biblioteka przeznaczona do wykrywania kolizji, symulowania dynamiki brył sztywnych oraz ciał deformowalnych (miękkich) takich jak np. tkaniny czy sprężyny. Struktura tej aplikacji została stworzona w języku C++ wykorzystując w pełni model obiektowy, jednak niższe jej warstwy napisane są w języku C, celem zwiększenia wydajności przetwarzania danych.

Główne zalety tego silnika to:

* Wykrywanie kolizji brył sztywnych oraz ciał deformowalnych w trybie ciągłym lub dyskretnym
* Pełne wsparcie dla obiektów deformowalnych
* Udostępnione zestawy określonych „kształtów kolizji”, czyli obiektów geometrycznych na podstawie których wyznaczane są kolizje, oraz nań reakcje. A na nie składają się:
  + - Sfera
    - Prostopadłościan
    - Walec
    - Stożek
    - Otoczka wypukła, wykorzystująca algorytm Gilberta – Johnsona – Keerthiego
    - Otoczka niewypukła
    - Siatka trójkątów
  + Opcjonalny zestaw optymalizacji pod kątem technologii CUDA oraz OpenCL

Dodatkowym atutem dla potencjalnego użytkownika tej biblioteki może być dosyć obszerne forum projektu, na którym dyskutowane są problemy programistów podczas użytkowania tej biblioteki, gdzie często wypowiadają się jej kontrybutorzy. Znacząco ułatwia to użytkowanie podczas realnego tworzenia aplikacji, ponieważ ze względu na złożoność rozwiązania nie można poznać go dogłębnie z każdej strony w stosunkowo krótkim czasie.

Rozwiązanie to jest bardzo docenione w środowisku, o czym świadczy fakt, że jest to zintegrowany w takich programach jak Blender, Houdini, SoftImage czy Cinema 4D. Dodatkowo można wspomnieć, że jako rozwiązanie działające w czasie rzeczywistym również sprawdza się doskonale o czym świadczy fakt że zostało ono użyte w takich produkcjach jak Grand Theft Auto V, DIRT oraz Red Dead Redemption.

* + 1. DirectXTK
  1. Opis działania
  2. Opis użytkowania

1. Testy
   1. Plan testów
   2. Wyniki
   3. Wnioski
2. Literatura

[1] F. D. Luna. Introduction to 3D Game Programming with DirectX 11, Wydawnictwo HELION, 2014