Politechnika Śląska

PROJEKT INŻYNIERSKI

Renderowanie terenu metodą ray marching

Aleksander MICKIEWICZ
Nr albumu: 293830

Kierunek: Informatyka
Specjalność: Grafika komputerowa

PROWADZĄCY PRACĘ
Dr inż Michał Staniszewski
KATEDRA
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2022

Tytuł pracy

Renderowanie terenu metodą ray marchingRenderowanie terenu metodą ray marching

Streszczenie

(Streszczenie pracy – odpowiednie pole w systemie APD powinno zawierać kopię tego streszczenia.)

Słowa kluczowe

Grafika komputerowa, ray marching, renderowanie

Thesis title

Rendering terrain using ray marching methodRendering terrain using ray marching method

Abstract

(Thesis abstract – to be copied into an appropriate field during an electronic submission – in English.)

Key words

Computer graphics, ray marching, rendering

Spis treści

Rozdział 1

Wstęp

Przedmiotem niniejszej pracy jest przedstawienie metody renderowania metodą ray marching dla problemu renderowania trójwymiarowego terenu. Ray marching jest to technika generowania obrazów dla dwuwymiarowych oraz trójwymiarowych scen. Działanie metody ray marching, podobnie do metody ray tracing, polega na odnalezieniu przecięcia wiązek światła wraz z renderowaną sceną. Rozwiązania oparte o wiązki światła dość dobrze odwzorowują rzeczywistość dzięki czemu można w dość prosty sposób uzyskać fotorealistyczny obraz. Techniki tego typu świetnie radzą sobie z problemami takimi jak odbicia czy załamania światła, których efekt znacznie trudniej jest osiągnąć w rozwiązaniach opartych o rasteryzacje trójkątów. Największą wadą tego rodzaju renderowania jest ich złożoność obliczeniowa. W najprostszym przypadku z kamery wysyłana jest jedna wiązka odpowiadająca każdemu pikselowi, dla której trzeba obliczyć przecięcia z całą sceną. Czas działania znacząco pogarsza się wraz z implementacją cieniowania, odbić światła, przezroczystości, załamania światła, efektów wolumetrycznych i innych działań poprawiających jakość oraz realizm tworzonego obrazu. Uwzględniając ilość wykonywanych obliczeń, metody renderowania oparte o wiązki światła nie nadają się do renderowania obrazów w czasie rzeczywistym, przy użyciu dostępnego obecnie sprzętu. Rozwiązania tego typu idealnie nadają się jednak do zastosowań, w których nie ma takich ograniczeń czasowych jak np. obrazy czy filmy.

Technika ray tracing jest analitycznym rozwiązaniem problemu znalezienia przecięcia wiązki z powierzchnią. Oznacza to, że punkt przecięcia można uzyskać przez rozwiązanie wzoru matematycznego, opisującego to przecięcie. Pomimo istnienia analitycznych rozwiązań przecięć z niektórymi powierzchniami (np. z kulą, z trójkątem), brak jest rozwiązania pozwalającego na proste znalezienie przecięcia z dowolną powierzchnią.

Alternatywą dla metod analitycznych jest technika ray marching. Jest to numeryczna metoda estymacji przecięć wiązki światła z dowolną powierzchnią. Technika ta polega na sukcesywnym przesuwaniu wiązki światła o daną odległość do momentu przecięcia z powierzchnią. Rozwiązanie to jest dość wolne oraz niedokładne, jednak

istnieją różnego rodzaju optymalizacje pozwalające poprawić dokładność oraz szybkość algorytmu (takie jak: sphere tracing, bounding spheres, itp.)

Celem pracy jest utworzenie aplikacji renderującej proceduralnie generowany teren. Użytkownik posiada możliwość poruszania kamerą oraz zmiany parametrów wpływających na generacje oraz renderowanie terenu.

W przedmiotowym zakresie pracy zostanie przedstawiona metoda proceduralnego generowania terenów, która może być wykorzystywana w obszarach związanych z tworzeniem filmów, gier komputerowych, animacji itp. Metoda ta zostanie zaimplementowana na karcie graficznej, w języku programowania GLSL (*OpenGL Shading Language*) jako jednostka cieniująca fragmentów.

W dalszej części pracy zostaną ujęte wymagania funkcjonalne, niefunkcjonalne oraz wskazane zostaną narzędzia użyte podczas tworzenia programu. Opisane zostaną metody instalacji programu na różnych systemach operacyjnych, jego sposób obsługi oraz zostanie przedstawiony przykład działania programu dla osiągnięcia planowanych efektów.

W pracy zostaną przedstawione graficzne wyniki zastosowania różnych algorytmów oraz ich parametrów. W dalszej części zostanie poddany analizie sposób działania wykorzystanych algorytmów.

Rozdział 2

Analiza tematu

W niniejszym opracowaniu zostaną przedstawione zagadnienia związane z renderowaniem scen z wykorzystaniem metody ray marching oraz techniką tworzenia proceduralnie generowanego terenu.

Rozwiązania tego typu mogą być wykorzystywane w obszarach związanych z tworzeniem filmów, gier komputerowych, animacji itp. Ta metoda renderowania na dzień dzisiejszy nie jest jeszcze zbyt powszechna, a przynajmniej nie w takim stopniu jak metody oparte o rasteryzację trójkątów czy ray tracing.

Biorąc pod uwagę sposób działania technik renderowania opartych o wiązki światła, przedstawienie generowanego terenu wykorzystując siatkę trójkątów nie jest wydajnym rozwiązaniem. Wiąże się to z ilością obliczeń koniecznych do wykrycia kolizji z każdym trójkątem siatki dla każdej wiązki światła. Znacznie lepszym i wydajniejszym rozwiązaniem jest przedstawienie terenu jako (dwu argumentowej) funkcji matematycznej.

Do kształtowania terenu w pierwszej kolejności zostanie wykorzystana funkcja zwracająca pseudolosową wartość dla danego dwuwymiarowego punktu. W tym celu można użyć z wielu różnych metod, przykładowo wykorzystując dwuwymiarową teksturę lub funkcję matematyczną której wyniki przypominają losowe wartości. Aby uzyskać ciągłą funkcję, wartość każdego punktu jest interpolowana na podstawie czterech najbliższych punktów, których współrzędne są wartościami całkowitymi.

Aby uzyskać porządany efekt zaimplementowana została interpolacja biliniowa opisana wzorem 2.1, dla wybranych punktów a, b, c i d. Jako parametry tej funkcji x i y należą do przedziału $x, y \in [0, 1]$. Przykład tego typu (może jakoś inaczej) interpolacji

przedstawia rysunek 2.1a.

$$\begin{aligned} f(x,y) = a \\ &+ (b-a) * x \\ &+ (c-a) * y \\ &+ (a-b-c+d) * x * y \end{aligned} \tag{2.1}$$

Można zauważyć, że efektem tej interpolacji są bardzo widoczne granice między grupami punktów, co przedstawia rysunek **??**.

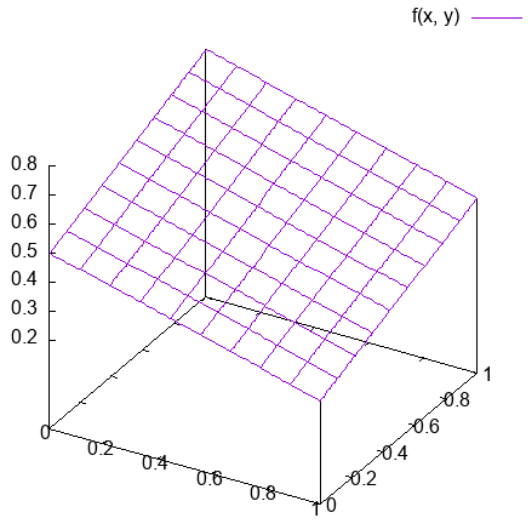
Rozwiązaniem tego problemu jest zastosowanie funkcji *smoothstep* pozwalającej na uzyskanie wygładzonej interpolacji. W tym celu zmodyfikowano wzór 2.1, zastępując parametry x i y wykorzystując funkcję *smoothstep* n-tego stopnia przedstawionej jako $S_n(p)$. Ostateczną postać funkcji przedstawia wzór 2.2. Obliczając gradient tej funkcji, można zauważyć, że wartości pochodnej dla każdego z czterech granicznych punktów jest równa 0, dzięki czemu przejścia między grupami punktów nie są widoczne.

$$\begin{aligned} f(x,y) = a \\ &+ (b-a) * S_n(x) \\ &+ (c-a) * S_n(y) \\ &+ (a-b-c+d) * S_n(x) * S_n(y) \end{aligned} \tag{2.2}$$

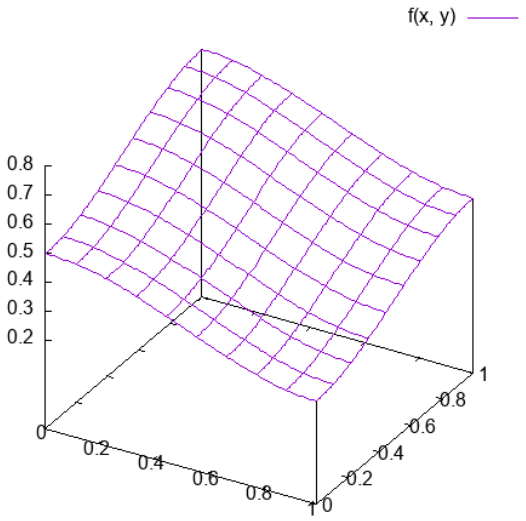
W programie wykorzystana została funkcja *smoothstep* pierwszego stopnia, której postać jest widoczna na wzorze 2.3.

$$S_1(x) = 3x^2 - 2x^3 \tag{2.3}$$

Wynik zastosowania tego wzoru ukazuje rysunek 2.1b.
2.1a 2.1



(a) interpolacja biliniowa



(b) interpolacja wykorzystujące funkcję *smoothstep* pierwszego stopnia

Rysunek 2.1: Porównanie wyników obu metod interpolacji między czterema punktami

Rozdział 3

Wymagania i narzędzia

3.1 Wymagania funkcjonalne

Przy realizacji projektu powinny zostać spełnione następujące wymagania funkcjonalne:

- Poruszanie kamerą - użytkownik będzie posiadał możliwość zmiany położenia kamery oraz jej orientacji. Ponadto, będzie istnieć opcja kontrolowania kąta pola widzenia kamery.

- Kontrola nad rozdzielczośćią obrazu - program pozwoli na konfigurację parametrów związanych z rozdzielczością renderowanego obrazu oraz jego współczynnikiem proporcji.

- Użytkownik będzie miał wpływ na ustawienia parametrów związanych z kształtowaniem terenu - w trakcie użytkowania programu dostępna będzie kontrola nad większością parametrów związanych z tworzeniem terenu.

- Kontrolowanie parametrów związanych z renderowaniem terenu - projekt ten umożliwia dostosowanie ustawień mających wpływ na dokładność oraz jakość renderowania obrazu.

- Interfejs graficzny - aplikacja wyposażona zostanie w graficzny interfejs użytkownika pozwalający na wykorzystanie dostępnej funkcjonalności programu

- Wyświetlenie średniego czasu renderowania sceny dla danych ustawień oraz ilość klatek na sekundę - interfejs aplikacji będzie na bieżąco przedstawiał informacje o obliczonym średnim czasie renderowania pojedynczej klatki oraz ilości wywietlanych klatek na sekundę.

3.2 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne mają na celu zapewnienie:

- Łatwej obsługi programu
- Wsparcia dla wielu systemów operacyjnych - w tym Windows oraz Linux
- Wysokiej wydajności, poprzez wykorzystanie potoku programowalnego karty graficznej
- Weryfikacji ustawień podanych przez użytkownika

3.3 Przypadki użycia

W programie nie znalazły zastosowania diagramy przypadków użycia.

3.4 Wykorzystane narzędzia

Do opracowania projektu wykorzystane zostaną narzędzia takie jak:

- gdb - Narzędzie debugowanie programu
- valgrind - wykrywanie wycieków pamięci oraz profilowanie programu
- shadertoy.com - strona internetowa umożliwiająca tworzenie programów w jednostce cieniującej fragmentów. Narzędzie zostało wykorzystane do prototypowania i testowania różnych algorytmów i metod renderowania.

Rozdział 4

Specyfikacja zewnętrzna

4.1 Wymagania sprzętowe i programowe

Program wykorzystuje potok programowalny bilbioteki OpenGL oraz jednostki cieniujące napisane w języku GLSL w wersji 3.30, oznacza to, że system użytkownika musi posiadać kartę graficzną wspierającą OpenGL conajmniej w wersji 3.3. Dodatkowo program wykorzystuje następujące biblioteki, których kod źródłowy znajduje się wewnątrz projetu:

- GLAD
- GLFW
- GLM
- Dear ImGui

Projekt został skompilowany i przetestowany wykorzystując kompilator g++ w wersji 12.2.0 na systemie Linux. Kompilacja progamu odbywa się przy pomocy narzędzia CMake.

4.2 Sposób instalacji

4.2.1 System Linux

Przykład kompilacji programu został wykonany przy wykorzystaniu systemu Ubuntu 22.04. Pierwszym krokiem jest instalacja narzędzi takich jak program CMake, kompilator języka C i C++, itp. oraz instalacja wymaganych bibliotek. Do tego celu wykorzystane są następujące polecenia

```
1 sudo apt install cmake
2 sudo apt install build-essential
3 sudo apt install xorg-dev
```

```
4  sudo apt install git
```

Następnym krokiem jest pobranie repozytorium git projektu. Istotnym jest by pobrać repozytorium rekurencyjnie, gdyż dołączone biblioteki zostały dodane do repozytorium jako submoduły. Można tego dokonać poleceniem

```
1  git clone https://github.com/Rei-sen/raymarch-terrain --
       recursive
```

Alternatywnie, jeżeli repozytorium nie zostało pobrane rekurencyjnie można zastosować polecenia

```
1  git submodule init
```

a następnie

```
1  git submodule update
```

Ostatnim krokiem jest wygenerowanie plików reguł Makefile a następnie kompilacja projektu. Generacji plików reguł Makefile można dokonać wywołując następujące polecenie w głównym katalogu projektu

```
1  cmake -B build -S .
```

W celu kompilacji należy wykorzystać poniższe polecenie

```
1  cmake --build build
```

Końcowy plik wykonywalny o nazwie raymarch-terrain znajduje się w katalogu build.

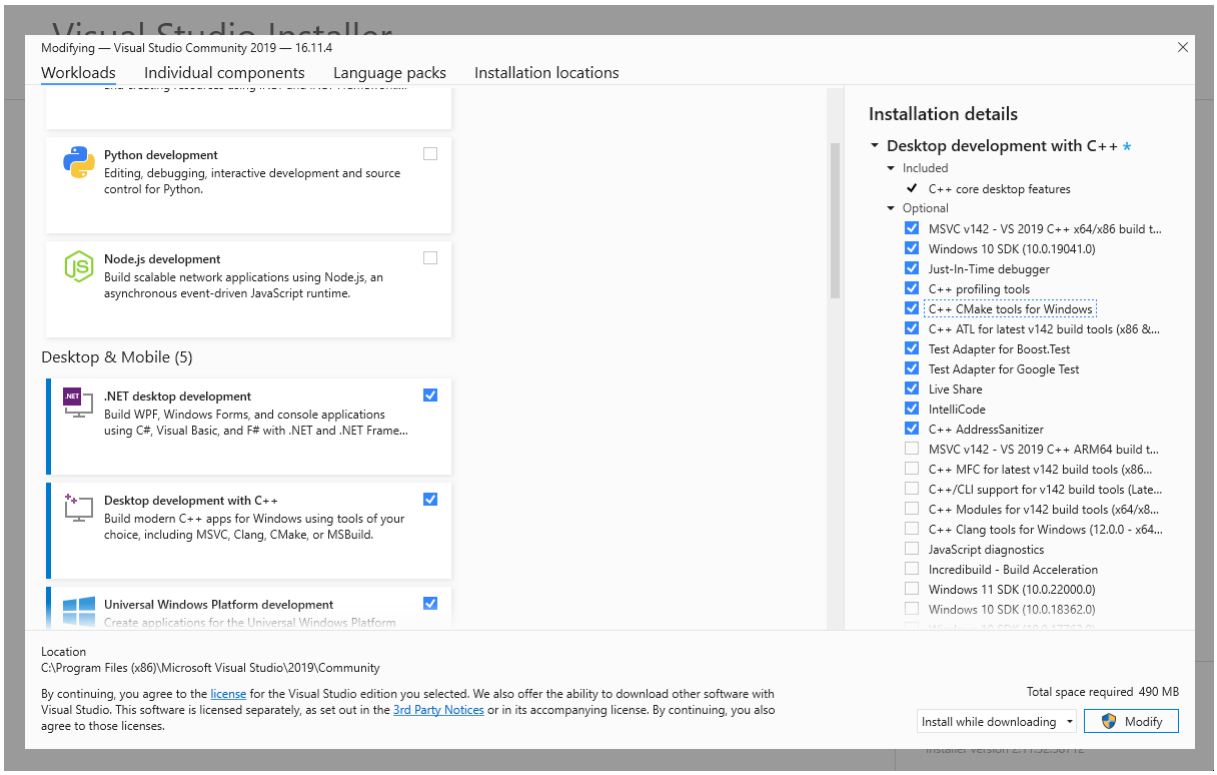
4.2.2 System Windows

Pierwszym krokiem instalacji programu na systemie Windows jest pobranie projektu z repozytorium github, które można znaleźć pod adresem https://github.com/Rei-sen/raymarch-terrain. Podobnie jak przy instalacji programu na systemie Linux, repozytorium należy pobrać rekurencyjnie, z wszystkimi submodułami. Dalszą część instalacji można dokonać w środowisku Visual Studio lub wykorzystując narzędzie CMake. Oba sposoby zostaną przedstawione poniżej.

Instalacja przy wykorzystystaniu środowiska Visual Studio

Przed przystąpieniem do otwarcia projektu należy się upewnić, że wsparcie systemu budowy CMake jest zainstalowane w środowisku Visual Studio. Można tego dokonać w instalatorze Visual Studio, jak na rysunku 4.1.

Po zweryfikowaniu instalacji wsparcia systemu CMake należy uruchomić środowisko Visual Studio, a następnie wybrać opcję *Open local folder*. Kolejnym krokiem jest



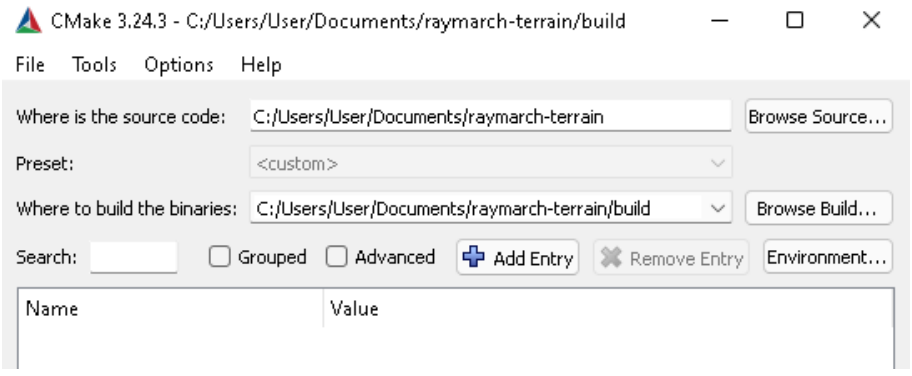
Rysunek 4.1: Instalacja wsparcia systemu CMake w instalatorze środowiska Visual Studio.

wybranie głównego folderu projektu. Po otwarciu projektu i ukończeniu generacji środowiska CMake, należy zmienić opcję *Select startup item* na raymarcher.exe. Po wykonaniu powyższych kroków, projekt można budować oraz kompilować jak zwykły projekt środowiska Visual Studio.

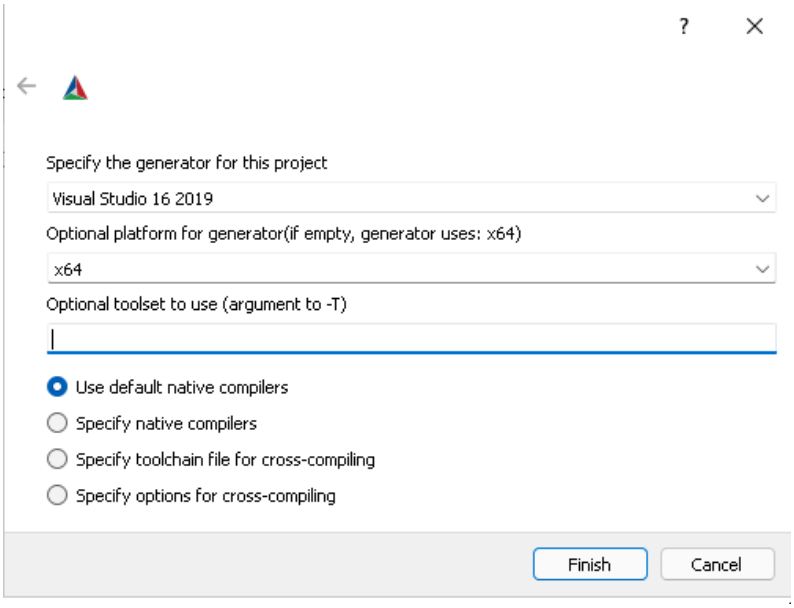
4.2.3 Instalacja wykorzystując narzędzie CMake

Pierwszym etapem jest instlacja narzędzia CMake, które można znaleźć pod adresem https://cmake.org.

Następnie należy uruchomić narzędzie CMake w trybue graficznym. Kolejnym etapem jest wybór głównego katalogu projektu oraz katalogu budowy, w którym



Rysunek 4.2: Wybór folderu projektu oraz folderu budowy narzędziu graficznym CMake.



Rysunek 4.3: Konfiguracja systemu budowy dla środowiska Visual Studio 16.

znajdą się końcowe pliki binarne. Jako główny katalog projektu należy wskazać katalog, w którym znajduje się plik CMakeLists.txt. Jako folder budowy został utworzony nowy folder o nazwie *build* w głownym folderze projektu. Przykład powyższych kroków został przedstawiony na rysunku 4.2. Kolejnym etapem jest konfiguracja systemu budowy, której można dokonać przyciskiem *Configure*, a następnie wybierając (odpowiednie, interesujące nas) docelowe środowisko programistyczne. Rysunek 4.3 przedstawia przykładową konfiguracje dla środowiska Visual Studio 16 dla systemów 64-bitowych.

Następnie należy wygenerować pliki budowy wykorzystując przycisk *Generate*. Ostatnim etapem jest otwarcie projektu poprzez pliki znajdujące się w folderze budowy lub poprzez przycisk *Open Project* w oknie programu CMake. W dalszej kolenjonści należy dokonać kompilacji programu korzystając z wcześniej wybranego środowiska programistycznego.

4.3 Sposób obsługi

Po uruchomieniu program przedstawia prosty interfejs użytkownika, widoczny na rysunku 4.4. Cały obszar okna wykorzystywany jest do wyświetlania renderowanego terenu. Dodatkowo wewnątrz okna renderowany jest obszar zawierający pola pozwalające na zmianę parametrów związanych z renderowaniem oraz generacją terenu. Użytkownik ma możliwość poruszania kamerą korzystając z klawiszy W, S, A oraz D. Zmiana orientacji kamery jest możliwa poprzez poruszanie myszką gdy wciśnięty jest lewy przycisk myszki. Szczegółowe informacje dotyczące korzystania z interfejsu użytkownika można uzyskać naciskając przycisk *Show UI Help*. Program można zakończyć poprzez naciśnięcie przycisku Escape lub zamknięcie okna.

Rysunek 4.4: Interfejs użytkownika

4.4 Przykład działania

do dodania po implementacji reszty programu.

- wymagania sprzętowe i programowe
- sposób instalacji
- sposób aktywacji
- kategorie użytkowników
- sposób obsługi
- administracja systemem
- kwestie bezpieczeństwa
- przykład działania
- scenariusze korzystania z systemu (ilustrowane zrzutami z ekranu lub generowanymi dokumentami)

Rozdział 5

Specyfikacja wewnętrzna

Jeśli „Specyfikacja wewnętrzna”:

- przedstawienie idei
- architektura systemu
- opis struktur danych (i organizacji baz danych)
- komponenty, moduły, biblioteki, przegląd ważniejszych klas (jeśli występują)
- przegląd ważniejszych algorytmów (jeśli występują)
- szczegóły implementacji wybranych fragmentów, zastosowane wzorce projektowe
- diagramy UML

Krótka wstawka kodu w linii tekstu jest możliwa, np. `int a;` (biblioteka listings). Dłuższe fragmenty lepiej jest umieszczać jako rysunek, np. kod na rys 5.1, a naprawdę długie fragmenty – w załączniku.

```
1  class test : public basic
2  {
3      public:
4        test (int a);
5        friend std::ostream operator<<(std::ostream & s,
6                                       const test & t);
7      protected:
8        int _a;
9
10 };
```

Rysunek 5.1: Pseudokod w listings.

Rozdział 6

Weryfikacja i walidacja

- sposób testowania w ramach pracy (np. odniesienie do modelu V)
- organizacja eksperymentów
- przypadki testowe zakres testowania (pełny/niepełny)
- wykryte i usunięte błędy
- opcjonalnie wyniki badań eksperymentalnych

Tabela 6.1: Nagłówek tabeli jest nad tabelą.

	metoda						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Rozdział 7

Podsumowanie i wnioski

- uzyskane wyniki w świetle postawionych celów i zdefiniowanych wyżej wymagań
- kierunki ewentualnych danych prac (rozbudowa funkcjonalna …)
- problemy napotkane w trakcie pracy

Dodatki

Spis skrótów i symboli

DNA kwas deoksyrybonukleinowy (ang. *deoxyribonucleic acid*)

MVC model – widok – kontroler (ang. *model–view–controller*)

N liczebność zbioru danych

μ stopnień przyleżności do zbioru

$\mathbb{E}$ zbiór krawędzi grafu

$\mathcal{L}$ transformata Laplace'a

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść w to miejsce.

```
1 if (_nClusters < 1)
2    throw std::string ("unknown number of clusters");
3 if (_nIterations < 1 and _epsilon < 0)
4    throw std::string ("You should set a maximal number of
       iteration or minimal difference -- epsilon.");
5 if (_nIterations > 0 and _epsilon > 0)
6    throw std::string ("Both number of iterations and minimal
       epsilon set -- you should set either number of iterations
        or minimal epsilon.");
```

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

Spis tabel