MECHANIKA
CZASOPISMO TECHNICZNE
TECHNICAL TRANSACTIONS
MECHANICS
WYDAWNICTWO POLITECHNIKI KRAKOWSKIEJ
4-M/2011
ZESZYT 7
ROK 108
ISSUE 7
YEAR 108

MIROSŁAW GŁOWACKI*, ADAM CZUBERNAT**

WIZUALIZACJA TRÓJWYMIAROWYCH PÓL SKALARNYCH Z WYKORZYSTANIEM TECHNIK RENDERINGU OBJĘTOŚCIOWEGO

VISUALISATION OF THREE-DIMENSIONAL SCALAR FIELDS USING VOLUME RENDERING TECHNOLOGY

Streszczenie

W artykule przedstawiono wyniki badań związanych z opracowaniem i implementacją algorytmu renderingu objętościowego, który umożliwia uzyskanie obrazów trójwymiarowych pól skalarnych w czasie rzeczywistym. Zbudowany system komputerowy wyposażony został w interaktywny interfejs użytkownika oraz narzędzia pozwalające na swobodną manipulację wyświetlanymi danymi. Aplikacja realizuje główne założenia dzięki użyciu akceleracji sprzętowej oferowanej przez programowalne układy graficzne. Obraz generowany jest z wykorzystaniem masywnej równoległości, co pozwala na jego wyświetlanie w czasie rzeczywistym oraz stosowanie technik kosztownych obliczeniowo. Zastosowane metody i ich optymalizacje umożliwiają intuicyjną oraz efektywną analizę danych wolumetrycznych.

Słowa kluczowe: *wizualizacja, rendering objętościowy*

Abstract

The paper deals with development and implementation of volume rendering algorithm that allows the creation of images of three-dimensional scalar fields in real time. The developed computer system is equipped with an interactive user interface and tools allowing for the free manipulation of displayed data. The application executes main objectives through the use of hardware acceleration provided by programmable graphics chips. Image is generated using the massive parallelism, which allows to display it in real time and the use of computationally expensive techniques. The methods used and their optimizations enable an intuitive and efficient analysis of volumetric data.

Keywords: *visualization, volume rendering*

* Prof. dr hab. inż. Mirosław Głowacki, Katedra Informatyki Stosowanej i Modelowania, Wydział Inżynierii Metali i Informatyki Przemysłowej, Akademia Górniczo-Hutnicza w Krakowie, Uniwersytet Humanistyczno-Przyrodniczy w Kielcach.

** Inż. Adam Czubernat, Wydział Inżynierii Metali i Informatyki Przemysłowej, Akademia Górniczo-Hutnicza w Krakowie.

1. Wstęp

Wizualizacja trójwymiarowych pól wektorowych lub skalarnych wiąże się z przetwarzaniem danych, często zbyt skomplikowanych i obszernych, aby mogły być wyświetlane z użyciem standardowych technik generowania obrazu. Dane te uzyskiwane są dziś w bardzo szerokim spektrum działań inżynieryjno-naukowych, obejmujących tomografię komputerową, tomografię mikroskopową, graficzne efekty projektowania inżynierskiego czy symulacje numeryczne. Aby umożliwić ich zobrazowanie, powstała odrębna dziedzina wizualizacji nazywana „renderingiem objętościowym". Ma ona na celu uzyskanie kompletnego, precyzyjnego i efektywnego wyświetlania danych wolumetrycznych, pozwalającego na dokładną analizę rozpatrywanego pola skalarnego. Wykorzystanie najnowszych osiągnięć w dziedzinie grafiki komputerowej pozwala na wizualizację nie tylko powierzchni obiektów, ale ich wnętrza – przez wykorzystanie efektów przezroczystości i półprzezroczystości.

Celem przeprowadzonych prac było opracowanie oraz implementacja algorytmu renderingu objętościowego, który umożliwiłby uzyskanie obrazu trójwymiarowych pól skalarnych w czasie rzeczywistym. Zakres prac obejmował także stworzenie interaktywnego interfejsu użytkownika oraz narzędzi pozwalających na swobodną manipulację wyświetlanymi danymi. Stworzona aplikacja realizuje główne założenia projektowe dzięki użyciu akceleracji sprzętowej, oferowanej przez programowalne układy graficzne. Uzyskany w ten sposób obraz generowany jest z wykorzystaniem masywnej równoległości, co pozwoliło na wyświetlanie w czasie rzeczywistym oraz zastosowanie w algorytmie technik kosztownych obliczeniowo, m.in. iluminacji. Zastosowane metody i ich optymalizacje umożliwiają intuicyjną oraz efektywną analizę danych wolumetrycznych, co sprawia, że aplikacja w pełni realizuje cele renderingu objętościowego.

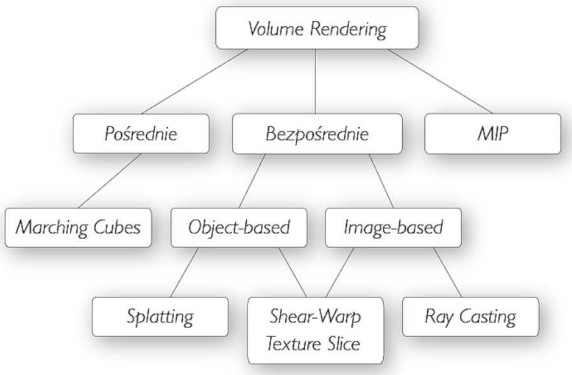
2. Rendering objętościowy

Rendering objętościowy jest to proces tworzenia dwuwymiarowej projekcji trójwymiarowych danych. Standardowe metody wyświetlania grafiki opierają się na założeniu, że większość generowanych obiektów składa się z nieprzeźroczystej powłoki, w ten sposób upraszczając złożoność problemu wyświetlania. Takie rozwiązanie nazywane jest „zorientowanym powierzchniowo" – elementami wykorzystywanymi w renderingu są punkty, linie i wielokąty. Pozwalają one na przedstawienie powierzchni, z których stworzone są wyświetlane obiekty. Jest to technika bardzo efektywna, stosowana w rozrywce interaktywnej, a także w wizualizacjach naukowych, co spowodowało, że w dzisiejszych czasach każde procesory graficzne implementują elementy tej techniki jak transformację, rasteryzację i oświetlenie na poziomie sprzętowym. Charakter powierzchniowy grafiki opartej na siatce wielokątów sprawia jednak, że przedstawienie dzięki tej metodzie obiektów wielowarstwowych lub posiadających bardzo złożoną geometrię, staje się obliczeniowo trudnym zadaniem, nawet dla dedykowanych układów graficznych. Ten aspekt, jak i fakt, że dane wolumetryczne nie są przystosowane do bezpośredniego wyświetlania powierzchniowego, uniemożliwia wykorzystanie tej techniki wprost w celu wizualizacji trójwymiarowych pól skalarnych. Rendering objętościowy, przyjmując za podstawę operowanie na ogromnej ilości danych, pozwala przedstawiać wnętrza skomplikowanych

obiektów o zmiennej przeźroczystości, umożliwiając wykonywanie wizualizacji niedostępnych lub obliczeniowo zbyt kosztownych dla innych metod.

Jest wiele metod pozwalających na wyświetlanie danych wolumetrycznych. Należy przy tym wyróżnić pośredni (IVR) oraz bezpośredni (DVR) rendering objętościowy. Pierwszy realizuje wyświetlanie przez konstruowanie izopowierzchni z danego pola skalarnego. Następnie używając tradycyjnych metod renderowania grafiki, opartej na siatce wielokątów, prezentuje powierzchnię utworzoną z voxeli o zadanej wartości. Metody DVR wizualizują trójwymiarowe dane wprost, bez kroków pośrednich jak reprezentowanie geometrii. Opierają się one na optycznym modelu, który opisuje zachowanie światła przechodzącego przez objętość. Techniki bezpośrednie dzielą się na obiektowe (*object-based*) oraz oparte na obrazie wynikowym (*image-based*). Obiektowe opierają się na samym obiekcie – wynikowy piksel zależy od iteracji przeprowadzonych na każdym voxelu. Pozostałe opierają się na zasadzie wykonywania obliczeń dla kolejnych pikseli ekranu, przez m.in. przeprowadzenie promienia przechodzącego przez każdy punkt ekranu i próbkowaniu voxeli leżących wzdłuż tego promienia. Pełny schemat przedstawionego podziału ilustruje rysunek 1. Skrót MIP oznacza tu metodę znaną jako Maximum Intensity Projection.

Do najpopularniejszych należą trzy metody renderowania. Pierwszą z nich jest technika „maszerujących sześcianów" (ang. *marching cubes*). Polega na ekstrakcji siatki wielokątów reprezentujących izopowierzchnię trójwymiarowego pola skalarnego. Jest to technika pośredniej wizualizacji wolumetrycznej, wyróżniająca się wśród innych metod ograniczoną możliwością reprezentowania obiektów półprzezroczystych, jednak, mimo iż właściwe wyświetlane wykonywane jest za pomocą standardowej grafiki bazującej na wielokątach, to istnieje możliwość przedstawienia trójwymiarowych danych w sposób właściwy dla renderingu objętościowego.



Rys. 1. Schematyczny podział metod renderingu objętościowego
Fig. 1. Schematic distribution of volume rendering methods

Slice rendering jest kolejną z metod. Można tu wymienić techniki, takie jak: *volume splatting*, *shear-warp*, *texture slice*, itp. Opiera się ona na renderowaniu płatów i bazuje na zdolności współczesnych kart graficznych do efektywnego rasteryzowania i teksturowania geometrii, dlatego też technikę tę nazywa się często *texture-based volume ren-*

dering. Techniki płatowe bazujące na GPU dzieli się na dwie kategorie – metody korzystające z tekstur 2D oraz na te, które korzystają ze sprzętowej obsługi tekstur 3D [2]. Pierwsze przechowują dane wolumetryczne jako stos dwuwymiarowych płatów, drugie zaś jako pojedynczy blok danych objętości. Podział ten wywodzi się z historii układów graficznych – pierwsza technika powstała, gdy nie udostępniały one użycia trójwymiarowych tekstur. Dziś większość z dostępnych kart oferuje ich obsługę. Algorytm w najprostszej formie zakłada stworzenie dużej liczby dwuwymiarowych półprzezroczystych płatów (w postaci czworokątów), wyodrębnionych z trójwymiarowego pola skalarnego, a następnie wyświetlanie ich z zachowaniem kolejności wynikającej z odległości od obserwatora. Równanie renderingu aproksymowane jest przez udostępniany przez układy graficzne *alpha blending*. Inne aspekty, takie jak wyświetlanie geometrii oraz teksturowanie, również akcelerowane są sprzętowo.

Ostatnia z trzech wymienionych technik – *volume ray casting*, nazywana także *volume ray marching*, jest metodą umożliwiającą wizualizację trójwymiarowych danych wprost, bez używania dodatkowej geometrii, co klasyfikuje ją jako bezpośrednią technikę renderowania objętościowego (DVR). Dostarcza ona rezultatów o najwyższej jakości. Spowodowane jest to tym, że wywodzi się bezpośrednio z dyskretyzacji równania renderingu, a co za tym idzie odzwierciedla prawdziwy mechanizm transportu światła w obiektach [1, 6]. Metoda jest oparta na obrazie wynikowym – wizualizacja tworzona jest przez przeprowadzenie promieni dla każdego punktu ekranu, a dzięki technikom adaptacyjnym, możliwe jest uniknięcie przetwarzania voxeli niebiorących udziału w obrazie końcowym. *Volume ray casting*, tak jak inne algorytmy generujące grafikę na podstawie przebiegających promieni, ma charakter pozwalający na łatwą budowę opartych na niej algorytmów równoległego przetwarzania danych, co w wypadku implementacji metody z wykorzystaniem programowalnych układów graficznych pozwala na uzyskanie bardzo dużej efektywności. Opracowany system graficzny bazuje na tej właśnie metodzie wizualizacji.

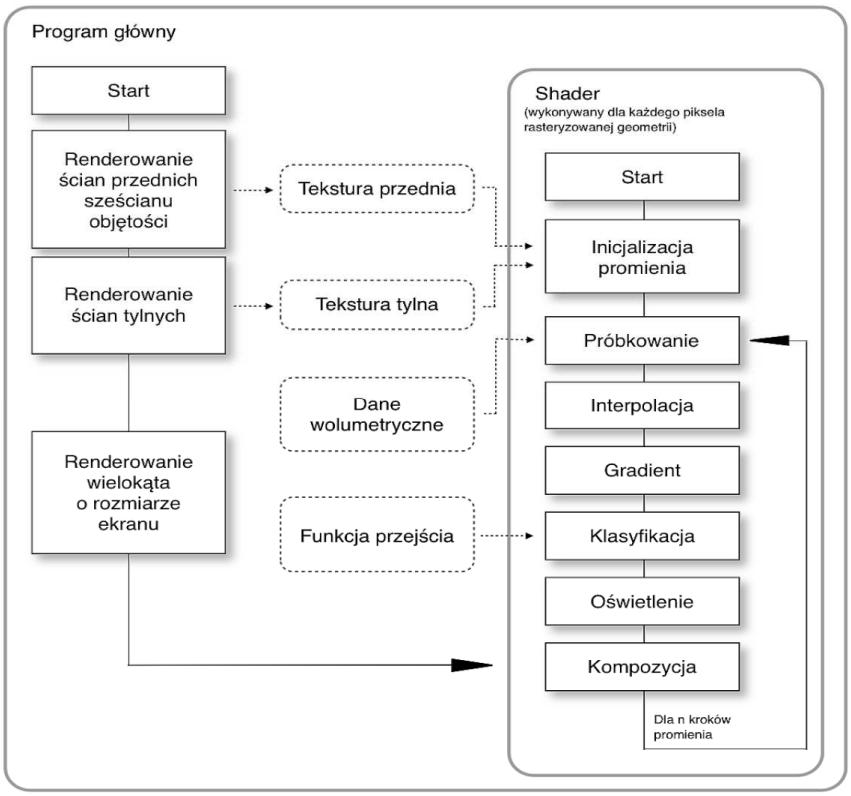
3. Zastosowany algorytm

Opracowany algorytm aplikacji implementuje wszystkie etapy metody *volume ray casting*. Aplikacja wykorzystuje akcelerację sprzętową uzyskiwaną przez stosowanie technologii OpenGL i GLSL, a sposób przetwarzania potoku graficznego przedstawiono na rysunku 2.

Program obsługuje dane wolumetryczne w postaci surowych plików z rozszerzeniem „raw”. Pliki takie nie posiadają nagłówków zawierających informacje o rozmiarach czy liczbie bitów przyporządkowanych poszczególnym voxelom. Zaprojektowany interfejs graficzny przed załadowaniem pliku do tekstury 3D wymaga zatem od użytkownika podania tych danych. Ze względu na zastosowanie i zaplanowaną prostotę działania, aplikacja operuje jedynie na danych 8-bitowych.

W celach optymalizacji oraz uproszczenia algorytmu, kierunek przebiegu oraz intersekcja promienia obliczane są przez wykorzystanie funkcji rasteryzujących oferowanych przez kartę graficzną. Zaimplementowana technika opiera się na założeniu, że promień przechodzi przez prostopadłościenną objętość, najpierw w punktach leżących na ścianach zwróconych przodem do obserwatora, a następnie opuszcza ją, przecinając ściany zwrócone tyłem [3].

Aby wykorzystać tę zależność zakłada się, że współrzędne danych skalarnych zawierają się w przedziale [0, 1]. Utworzona zostaje tablica wierzchołków odpowiadająca jednostkowemu sześcianowi objętości, zawierająca składowe kolorów wierzchołków, dobranych tak, aby odpowiadały bezpośrednio ich pozycjom, tj. {r, g, b} = {x, y, z}. Utworzoną geometrię skaluje się przez macierz widoku, aby uzyskać z jednostkowego sześcianu proporcjonalny prostopadłościan objętości o najdłuższym wymiarze równym 1. Następnie uzyskana geometria jest wyświetlana w dwóch etapach renderowania: ścian przednich oraz tylnych. Pomiędzy operacjami przełączony zostaje bufor ramki, w celu zachowania wyników. Rezultatem procesu są dwie tekstury posiadające informacje o współrzędnych punktów przecięcia promieni, przechodzących przez każdy punkt ekranu z prostopadłościanem objętości. Wartości zachowane są w składowych kolorów tekseli, które używane są następnie w programach shaderów do określenia pozycji oraz kierunku promienia. Dzięki przedstawionej technice oraz zastosowaniu sprzętowej rasteryzacji i wydajnych FBO (Frame Buffer Objects), możliwe jest mało kosztowne obliczenie wartości potrzebnych do inicjalizacji promieni.



Rys. 2. Schemat blokowy procesu renderingu
Fig. 2. The flowchart of rendering process

Właściwy rendering następuje po zmianie typu projekcji na ortogonalną, która ma celu utworzenie czworokąta dostosowanego do szerokości ekranu. Następnie do potoku

renderowania przyłączony zostaje program cieniujący, powodując dla każdego punktu obrazu wynikowego wykonanie instrukcji zawartych w shaderze.

Program cieniujący realizuje bezpośrednio metodę śledzenia promieni. Na podstawie danych zawartych w teksturach ścian tylnych i przednich obliczane są dane niezbędne do utworzenia promienia, a następnie wykonywana jest pętla, w której promień jest propagowany przez objętość. Dla każdej iteracji pierwszym krokiem jest próbkowanie tekstury danych wolumetrycznych. Dzięki stosowaniu samplerów 3D możliwa jest zoptymalizowana interpolacja trzyliniowa bez potrzeby przeprowadzania kosztownych obliczeń. W kolejnym etapie następuje obliczenie gradientu metodą różnic centralnych, co pozwala na określenie m.in. wektora normalnego do powierzchni w tym punkcie. Następnie aby umożliwić analizę danych na podstawie funkcji przejścia, przypisuje się danym objętościowym, opisującym właściwości, takie jak gęstość, prędkość czy temperatura, właściwości optyczne, jak kolor i przezroczystość. Etap ten nazywany jest „klasyfikacją". Kolejnym krokiem renderingu jest obliczenie natężenia światła w danym punkcie na podstawie modelu Phonga, uwzględniając obliczony wcześniej gradient. W celach optymalizacji obliczeń wektorów, wykorzystywane są funkcje matematyczne GLSL (OpenGL Shading Language). Ostatnim, najważniejszym krokiem iteracji, w pętli propagującej promień w wizualizowanej objętości jest kompozycja. Jest ona dyskretną formą równania renderingu opisującego model emisyjno-absorpcyjny. Polega na odpowiednim mieszaniu koloru voxela ze składanym kolorem wynikowego piksela, w zależności od ich przezroczystości.

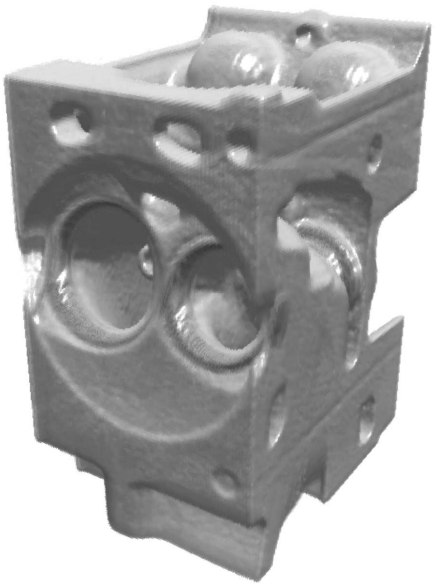
Implementacja wykorzystuje także dodatkowe funkcjonalności, których celem jest kończenie lub pomijanie niektórych iteracji. W celach optymalizacji zaimplementowano podstawową technikę wczesnej terminacji promienia.

4. Efekty działania systemu

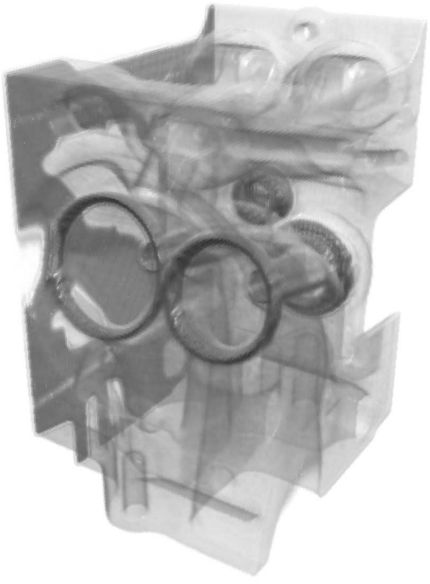
Jako przykład możliwości opracowanego systemu wizualizacji przedstawiono rendering bloku silnika, opisanego zaczerpniętym z Internetu zestawem szczegółowych danych numerycznych. Dane zawierają informacje o poszczególnych elementach składowych bloku, co pozwoliło na zastosowanie klasyfikacji i wykazanie jej kluczowego znaczenia dla wizualizacji wolumetrycznych. Pozwala ona na identyfikowanie i wyróżnienie interesujących elementów w przetwarzanych danych. Rysunki 3 i 4 przedstawiają wyniki manipulacji funkcją przejścia na przykładzie dwóch cylindrów bloku silnika.

Obraz przedstawiony na rys. 3 otrzymano po wybraniu opcji, która nie wprowadza przezroczystości obiektu, podczas gdy obraz przedstawiony na rysunku 4, uzyskany został po zastosowaniu półprzezroczystości ustalonej w funkcji przejścia. Pozwala to dostrzec umiejscowienie części wykonanych z metalu o większej gęstości. Na rysunkach barwnych poszczególne elementy przedstawiane są zazwyczaj różnymi kolorami.

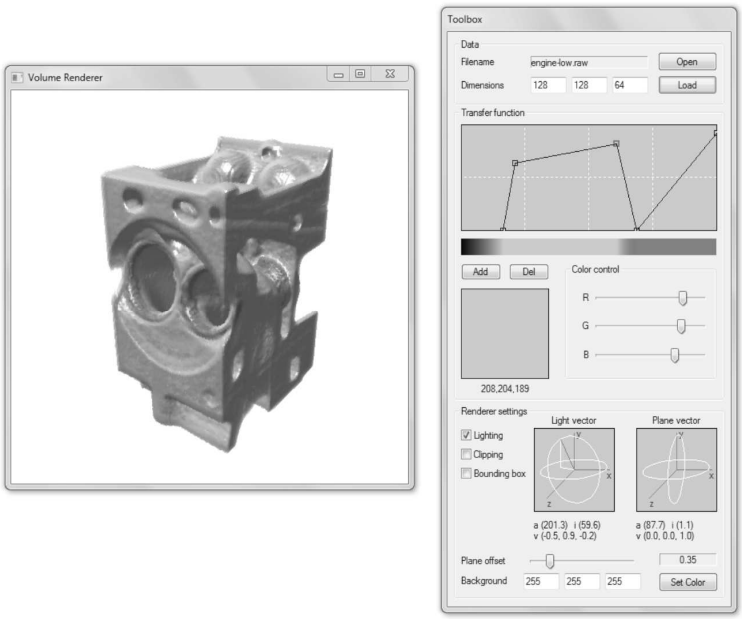
Na rysunku 5 przedstawiono zostały dwa główne okna aplikacji. Pierwsze z nich jest oknem wizualizacji, podczas gdy drugie służy do manipulacji otrzymywanym obrazem. Ponieważ rendering dokonywany jest z użyciem mocnego wsparcia sprzętowego, aplikacja charakteryzuje się wysokim komfortem obsługi. Wszelkie żądania dotyczące opcji wyświetlania realizowane są zgodnie z życzeniem użytkownika w czasie rzeczywistym.



Rys. 3. Powierzchniowy obraz przykładowego bloku silnika (źródło danych wejściowych: http://www.volvis.org/)
Fig. 3. Surface image of an example engine block (input data source: http://www.volvis.org/)



Rys. 4. Przezroczysty obraz przykładowego bloku silnika (źródło danych wejściowych: http://www.volvis.org/)
Fig. 4. Transparent image of an example engine block (input data source: http://www.volvis.org/)



Rys. 5. Widok podstawowych okien aplikacji
Fig. 5. The view of main application windows

5. Wnioski

Przedstawiona w artykule aplikacja wykorzystuje akcelerowaną sprzętowo metodę renderingu objętościowego, stanowiącego jedną z podstawowych metod wizualizacji wolumetrycznych. Przedstawiono metody generowania obrazu trójwymiarowych pól skalarnych, zasady ich działania oraz występujące ograniczenia.

Przeprowadzone studium pozwoliło na stworzenie kompletnej i efektywnej implementacji o wysokim stopniu interaktywności. Rozwiązanie oparto na programowalnych jednostkach graficznych. Oferowana przez układy masywna równoległość, pozwoliła na stworzenie sprawnego algorytmu i osiągnięcie wydajności, pozwalającej wyświetlać płynną animację na ogólnie dostępnym sprzęcie. Stworzony wyspecjalizowany interfejs graficzny użytkownika został zaprojektowany z myślą o efektywnej i nienastręczającej trudności manipulacji wizualizacjami danych wolumetrycznych. Pozwala on na bardzo elastyczne i precyzyjne dostosowanie wynikowego obrazu do potrzeb użytkownika.

Uzyskiwane wizualizacje charakteryzują się wysoką jakością. Program generuje obrazy realistycznie odwzorowujące rzeczywiste obiekty. Zaimplementowana aplikacja może służyć jako narzędzie do wizualizacji dowolnych danych wolumetrycznych. W związku z tym, że zastosowania wizualizacji trójwymiarowych pól skalarnych obejmują szerokie spektrum działań inżynieryjno-naukowych, program znajduje wiele zastosowań. Dzięki opracowanej metodzie dane utworzone przez metody rezonansu magnetycznego, mikrotomografów, generowane przez automaty komórkowe czy symulacje numeryczne mogą zostać wyświetlone i poddane szczegółowej analizie.

Zbudowany program ma także ograniczenia. Wczytywane dane limitowane są ze względu na liczbę bitów przypadających na voxel, jak również ze względu na ich rozmiar. Stabilność aplikacji jest nierozstrzygnięta, co jest spowodowane przetestowaniem aplikacji na niewielkiej liczbie konfiguracji sprzętowych. Przyszłe prace nad projektem wymagają dalszych optymalizacji, stworzenia nowej wersji interfejsu użytkownika opartego na zorientowanym obiektowo API, rozbudowa aplikacji o obsługę nowych formatów plików oferujących dane o większej precyzji, a także stworzenie modelu uwzględniającego pośrednie elementy iluminacji jak cienie i rozproszenie światła.

Praca wykonana w ramach działalności statutowej AGH – umowa nr: 11.11.110.011.

Literatura

[1] Engel K., *Real-time volume graphics*, a K Peters Ltd., Wellesley 2006.
[2] Ikits M., Kniss J., Lefohn A, Hansen C., *GPU Gems, Volume Rendering Techniques*, Addison-Wesley, Boston 2004.
[3] Kruger J., Westermann R., *Acceleration Techniques for GPU-Based Volume Rendering*, IEEE Visualization 2003.
[4] Rost R.J., *OpenGL Shading Language*, Addison-Wesley, Boston 2006.
[5] Ahn S.H., *OpenGL Vertex Buffer Object (VBO)*, 2005 (www.songho.ca/opengl/gl_vbo.html).
[6] Pawasauskas J., *Volume Visualization With Ray Casting*, 1997 (web.cs.wpi.edu/~matt/courses/cs563/talks/powwie/p1/ray-cast.htm).