Dydaktyka Informatyki 14(2019)
ISSN 2083-3156; e-ISSN 2543-9847
DOI: 10.15584/di.2019.14.14
http://www.di.univ.rzeszow.pl

Artur HERMANOWICZ[1], **Agnieszka MOLGA**[2]

[1] *ORCID: 0000-0003-4401-7421, Dr, Uniwersytet Technologiczno-Humanistyczny w Radomiu, Wydział Informatyki i Matematyki, Katedra Informatyki, ul. Malczewskiego 22a, 26-600 Radom, e-mail: artur.hermanowicz@uthrad.pl*
[2] *ORCID: 0000-0002-0857-5111, Dr, Uniwersytet Technologiczno-Humanistyczny w Radomiu, Wydział Informatyki i Matematyki, Katedra Informatyki, ul. Malczewskiego 22a, 26-600 Radom, e-mail: agnieszka19216@wp.pl*

ZASTOSOWANIE PROGRAMÓW CIENIUJĄCYCH DO MODELOWANIA ODBICIA ROZPROSZONEGO

APPLICATION OF SHADER PROGRAMS FOR DIFFUSE REFLECTION MODELING

Słowa kluczowe: programy cieniujące, odbicie rozproszone, OpenGL, grafika 3D.
Keywords: shading programs, diffuse reflection, OpenGL, 3D graphics.

Streszczenie

W pracy przedstawiono wybrane problemy dotyczące generowania grafiki trójwymiarowej. Ukazany został problem modelowania oświetlenia, a w szczególności odbicia rozproszonego. Zademonstrowano możliwości, jakie daje wykorzystanie programów cieniujących w tym zakresie.

Abstract

The paper presents selected problems related to the generation of three-dimensional graphics. The problem of modeling lighting, in particular diffuse reflection, has been presented. Demonstrations of the possibilities offered by the use of shading programs in this area have been demonstrated.

Wstęp

Nieustanny rozwój sprzętu komputerowego, który można obserwować praktycznie z dnia na dzień, stwarza możliwości, które jeszcze niedawno mogły leżeć tylko w strefie marzeń. Jedną z dziedzin, która szczególnie mocno związana jest z mocą obliczeniową komputerów jest grafika komputerowa. Zwłaszcza ta, która dotyczy generowania grafiki trójwymiarowej.

Procesory kart graficznych rozwinęły się od układów niewiele więcej potrafiących niż wyświetlanie obrazu na ekranie monitora do maszyn często przewyższających mocą główny procesor komputera. Wraz z rozwojem układów graficznych zaistniała potrzeba wydajnego ujednoliconego sposobu ich obsługi i standaryzacji. W ten sposób narodziła się biblioteka graficzna OpenGL.

Rozwój układów graficznych w stronę dużej liczby małych wyspecjalizowanych układów działających równolegle tzw. shaderów[1], pociągnął za sobą potrzebę ich wygodnego i wydajnego programowania. W ten sposób narodził się język GLSL (ang. *OpenGL Shading Language*). Początkowo możliwość wykorzystywania programów cieniujących była dodatkiem do biblioteki, ale od wersji OpenGL 2.0 w 2004 roku stała się standardową częścią.

W 2004 roku było to rozwiązanie przyszłościowe (nie każdego było stać na kartę graficzną z takimi możliwościami), obecnie ciężko byłoby jednak znaleźć sprzęt niespełniający tych wymagań.

Jednym z najbardziej podstawowych elementów generowania realistycznej grafiki trójwymiarowej jest model oświetlenia. Idea polega na tym, że do obliczeń wykorzystywane są w jak największym stopniu znane prawa fizyki. W praktyce stosowane modele oświetlenia są znacznie uproszczone w stosunku do modelu fizycznego i zawierają wiele założeń. Im model oświetlenia będzie bardziej zbieżny z prawami fizyki, tym osiągnięty efekt również będzie bardziej realistyczny. Ceną za to będą dłużej trwające i bardziej skomplikowane obliczenia.

Przedstawiony w niniejszym artykule materiał powinien, zdaniem autorów, pomóc wzbogacić proces dydaktyczny w zakresie programowania i grafiki komputerowej. Zastosowanie powinien znaleźć w procesie kształcenia uczniów szkół ponadgimnazjalnych oraz studiów pierwszego stopnia związanych z dyscypliną informatyka.

Modele oświetlenia

Jednym z najważniejszych elementów, który umożliwia zwiększenie realizmu generowanej grafiki trójwymiarowej jest model oświetlenia[2], który będzie jak najbardziej zbieżny z prawami fizyki, w tym wypadku z optyką.

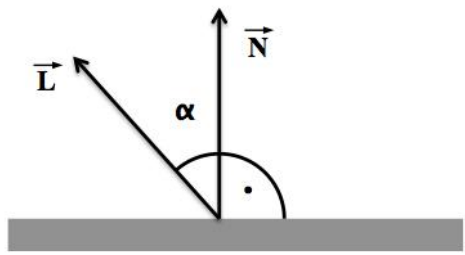
Spośród wielu warto wymienić przynajmniej trzy elementy: światło otoczenia, odbicie rozproszone, odbicie zwierciadlane. Światło otoczenia w pewien

[1] V.S. Gordon, J. Clevenger, *Computer Graphics Programming in OpenGL with Java*, Mercury Learning and Information 2017.

[2] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, R.L. Philips, *Wprowadzenie do grafiki komputerowej*, WNT, Warszawa 1995.

sposób rekompensuje brak wiedzy w opisie sceny i umożliwia zwiększenie realizmu. Natężenie oświetlenia nie zależy od żadnego źródło światła, a umożliwia uwidocznienie obiektów, które znajdują się w cieniu. Odbicia zwierciadlane umożliwiają symulację efektu, który można zaobserwować oświetlając błyszczące powierzchnie, jak np. karoseria samochodu w słoneczny dzień.

Odbicie rozproszone (rys. 1) to próba symulacji oświetlenia przez punktowe źródło światła. Natężenie oświetlenia w tym modelu jest proporcjonalne do iloczynu natężenia źródła światła, współczynnika odbicia materiału, z którego wykonany jest obiekt oraz cosinusa kąta pomiędzy wektorem normalnym do powierzchni w danym punkcie i wektorem do źródła światła.



Rys. 1. Schemat odbicia rozproszonego

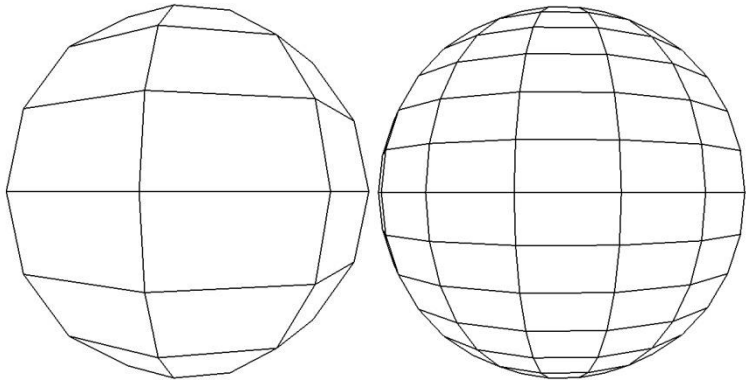
Źródło: opracowanie własne.

Ze względów obliczeniowych obydwa wektory powinny być znormalizowane. Umożliwia to zastąpienie obliczenia cosinusa kąta między wektorami poprzez iloczyn skalarny tych wektorów.

Generowanie grafiki trójwymiarowej

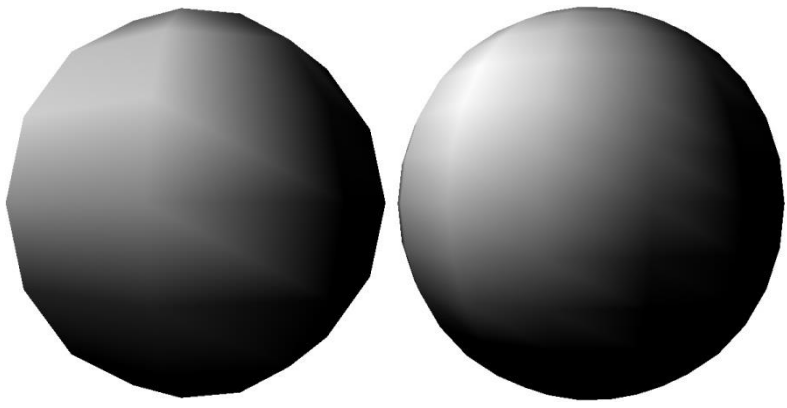
Obiekty tworzone na potrzeby generowania grafiki trójwymiarowej zazwyczaj składają się z wielu prostych elementów (zwykle trójkątów). Bryły dzielone są przekrojami (rys. 2) w celu uzyskania zbioru wierzchołków i ścianek zapewniających odpowiednią jakość odwzorowania.

Model cieniowania Gorauda polega na tym, że natężenie oświetlenia obliczane jest jedynie dla wierzchołków. Dla pozostałych punktów kolor jest interpolowany, co czyni jakość grafiki (rys. 3) bardzo zależną od liczby przekrojów, co w sposób oczywisty redukuje wydajność. W modelu cieniowania Phonga interpolowane są natomiast wektory normalne, a natężenie oświetlenia obliczane jest dla każdego punktu. Niestety, zastosowanie tego modelu w grafice generowanej w czasie rzeczywistym nie było praktycznie możliwe aż do momentu wprowadzenia programów cieniujących.



Rys. 2. Wizualizacja dwóch sfer z różną liczbą przekrojów: 8 na 8 (po lewej), 16 na 16 (po prawej)

Źródło: opracowanie własne.



Rys. 3. Wizualizacja oświetlenia dla sfer z różną liczbą przekrojów: 8 na 8 (po lewej), 16 na 16 (po prawej)

Źródło: opracowanie własne.

Zastosowanie programów cieniujących

Na potrzeby niniejszego opracowania powstała aplikacja napisana w języku Java[3] z wykorzystaniem biblioteki JOGL, która umożliwia obsługę wszystkich funkcji biblioteki OpenGL w sposób obiektowy. Stanowi to znakomite ułatwienie pracy z programami cieniującymi redukując znacząco nakład pracy konieczny na ich włączenie do programu i obsługę. Dzięki temu programista może skupić się na programowaniu samych shaderów w znacznie większym stopniu.

[3] C. Horstmann, G. Cornell, *JAVA. Podstawy*, wyd. 9, Helion, Gliwice 2014.

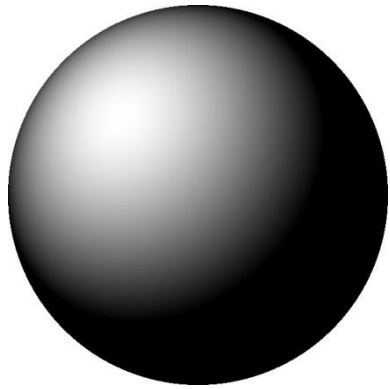
Dwa najważniejsze rodzaje programów cieniujących to Vertex Sharder i Fragment Shader. Zadaniem pierwszego z nich są różnego rodzaju operacje na wierzchołkach. Zadaniem drugiego – działania podejmowane na poziomie pikseli. Język GLSL ma składnię niezwykle podobną do języka C, co owocuje prostym i czytelnym kodem (listing 1). Podstawowe typy danych to wektory i macierze, choć dostępne są również podstawowe typy jak np. float.

```
#version 410

in vec3 pos;
out vec4 color;

structLight{
  vec4 color;
  vec3 position;
};

voidmain(void){
  vec4 white=vec4(1.0);
float r=1.0;
floatdist=length(pos.xy);
if(dist>=r){
color=white;
    return;
  }
Lightlight=Light(white,vec3(-1.0,1.0,2.0));
  vec3 N=pos.xyz;
N.z=sqrt(1.0-N.x*N.x-N.y*N.y);
  vec3 L=light.position-N;
  L=normalize(L);
float I=dot(L,N);
  I=clamp(I,0.0,1.0);
  vec4 mat=white;
color=mat*light.color*I;
}
```

Listing 1. Fragment Shader

Źródło: opracowanie własne.

Aby zademonstrować możliwości programów cieniujących, na potrzeby tego przykładu umieszczono całość implementacji algorytmu modelowania odbicia rozproszonego na powierzchni sfery w jednym programie. Efekt działania (rys. 4) został osiągnięty bez dzielenia sfery na mniejsze powierzchnie. W praktyce nie ma też żadnego modelu sfery. Wszystkie niezbędne obliczenia zostały wykonane przez program cieniujący (listing 1). W rzeczywistości obiekt widoczny na rys. jest płaskim kwadratem składającym się z zaledwie czterech wierzchołków.



Rys. 4. Symulacja oświetlenia rozproszonego na sferze z wykorzystaniem programów cieniujących

Źródło: opracowanie własne.

Zakończenie

Rzeczy, które do niedawna były tylko w sferze marzeń, jak na przykład tworzenie realistycznej grafiki trójwymiarowej w czasie rzeczywistym, są już możliwe do zrealizowania. Powstały narzędzia, które umożliwiają osiągnięcie tego celu w sposób wygodny i szybki. Niestety, widoczna jest również tendencja do tego, aby posługiwać się gotowymi programami użytkowymi zamiast pogłębiać wiedzę i kształtować umiejętności poprzez samodzielne tworzenie oprogramowania.

Zdaniem autorów niniejszego opracowania przedyskutowane w nim zagadnienia powinny stanowić uzupełnienie w procesie dydaktycznym w ramach zajęć dotyczących programowania grafiki komputerowej. Dodatkowym atutem jest możliwość, jak w tym przykładzie, łatwego powiązania zagadnień z zakresu fizyki z programowaniem.

Bibliografia

Foley J.D., van Dam A., Feiner S.K., Hughes J.F., Philips R.L., *Wprowadzenie do grafiki komputerowej*, WNT, Warszawa 1995.

Gordon V.S., Clevenger J., *Computer Graphics Programming in OpenGL with Java*, Mercury Learning and Information 2017.

Horstmann C., Cornell G., *JAVA. Podstawy*, wyd. 9, Helion, Gliwice 2014.