

**SPOŁECZNA AKADEMIA NAUK
W ŁODZI**

**WYDZIAŁ STUDIÓW MIĘDZYNARODOWYCH
I INFORMATYKI**

KIERUNEK INFORMATYKA

KAMIL CECHERZ

Nr albumu: 82739

**CHARAKTERYSTYKA BIBLIOTEKI
OPENGL**

ŁÓDŹ 2017

WPROWADZENIE

OpenGL to niskopoziomowa biblioteka, zestaw funkcji, API umożliwiające tworzenie zaawansowanej grafiki komputerowej w sposób pozwalający programiście/projektantowi na uniezależnienie się od specyfikacji sprzętu. OpenGL należy wyobrażać sobie jako fasadę dla użytkownika, interfejs zwiększający poziom abstrakcji pomiędzy procesem projektowania a programowaniem operacji niskopoziomowych. Dostajemy „[...] 250 podstawowych wywołań, umożliwiających budowanie złożonych trójwymiarowych scen z podstawowych figur geometrycznych” [1]. API zapewnia prostą obsługę operacji graficznych nie odbierając zarazem programiście dostępu do zaawansowanych funkcji, dzięki czemu stało się ono narzędziem wszechstronnym, powszechnie stosowanym zarówno do produkcji gier jak prostych aplikacji typu wygaszacz ekranu.

Opisane narzędzie służy, mówiąc wprost, do rysowania. Komputer nie posiada umiejętności potrzegania i interpretowania rzeczywistości, ten oczywisty fakt jest kluczowy do zrozumienia pojęć stosujących u podstaw grafiki komputerowej. Geometria i prosta arytmetyka pozwalają na modelowanie, tworzenie rzeczywistości wirtualnej; wychodząc od elementarnych operacji dochodzimy do złożonych kształtów. Pewne uniwersalne algorytmy pozwalają na nam przeliczanie położenia zbiorów punktów w trzech wymiarach. Okazuje się, że zrozumienie tego jak komputer interpretuje liczbowo takie obiekty jak: punkt, prosta, trójkąt otwiera ogrom możliwości.

„Podstawowa jednostka renderingu w OpenGL nosi nazwę prymitywu (ang. primitive). OpenGL obsługuje wiele rodzajów prymitywów, ale trzy najbardziej podstawowe to: punkt, linia i trójkąt. W zasadzie wszystko, co widać na ekranie, jest połączeniem (bardzo sprytnie pokolorowanych) punktów, linii i trójkątów. Aplikacje najczęściej dzielą złożone powierzchnie na bardzo dużą liczbę trójkątów i wysyłają je do zrenderowania za pomocą akceleracji sprzętowej w procesie zwanym rasteryzacją.” [2]

Wszystko to niezmiernie ważne, w gruncie rzeczy chodzi o tworzenie kodu ale bez logicznych umiejętności kreowania rzeczywistości przy pomocy obiektów programowanie nie wystarczy. Kod musi nauczyć się „widzieć”.

INSTALACJA

Aby skorzystać z OpenGL musimy odpowiednio połączyć projekt programistyczny z bibliotekami. Te z kolei trzeba umieścić w odpowiednim miejscu by miał do nich dostęp kompilator. Musimy wiedzieć co tak naprawdę robi komputer. Znajomość narzędzi odpowiedzialnych za przygotowanie programu oraz środowiska, a także systemu operacyjnego, w którym pracujemy mogą okazać nieodzowne.

Przedstawię szybki sposób na przetestowanie funkcjonalności OpenGL, zaznaczam jednak, że chcąc zarządzać dużym projektem wedle własnych preferencji warto zajrzeć do dokumentacji dostępnej na: <https://www.opengl.org/>. Pokażę jak posługując się kompilatorem z poziom wiersza poleceń wygenerować plik wykonalny dla systemu Windows. Zaprezentowany sposób nie musi zadziałać pod każdą wersją systemu (pomijam omawianie architektury systemu).

Do uruchomienia programu w C/C++ potrzebny jest rzecz jasna kompilator. Do prezentacji przykładu wykorzystuję MinGW (Minimalist GNU for Windows). Instalacja programu rozszerza pulę poleceń dostępnych z konsoli o m.in. `gcc`, `g++` (narzędzia do kompilacji kodu napisanego w językach C/C++). Narzędzia do prawidłowej pracy potrzebują konfiguracji, mającej na celu lepszą integrację z systemem [3]

By MinGW miał zapewnioną obsługę OpenGL potrzebne jest dodanie plików GLUT (The OpenGL Utility Toolkit).

„OpenGL jest niezależny od dowolnego systemu okienkowego. W rezultacie nie zawiera żadnych funkcji do otwierania okien ani interakcji z użytkownikiem. Każdy system okienkowy obsługujący OpenGL ma swoją własną bibliotekę, która obsługuje wsparcie.

GLUT, OpenGL Utility Toolkit, to prosty system okienkowy, który został przeniesiony do kilku różnych systemów operacyjnych. Jest powszechnie używany podczas nauczania OpenGL.”

Na stronie **James Madison University** dostępny jest również poradnik i niezbędne pliki binarne.

„Skopiuj **glut.h** do katalogu MinGW \ include \ GL.

Skopiuj **glut32.lib** do swojego katalogu kompilacji (np. Do katalogu, z którego kompilujesz i do którego **glut32.lib**).

Skopiuj **glut32.dll** do tego samego katalogu, w którym zostanie utworzony plik wykonywalny.”[4]

Trzy pliki wystarczają by zacząć przygodę z grafiką. By jednak stworzyć projekt trzeba oswoić się z kompilacją z wiersza poleceń. Podczas korzystania z IDE takiego jak Visual Studio używamy makr, wszystko możemy wykonać przy pomocy jednego przycisku, jednak warto bliżej przyjrzeć się etapom powstawania programu, wykonując wszystko ręcznie.

PIERWSZY PROGRAM

Program graficzny można napisać w każdym edytorze tekstu, nadając plikowi rozszerzenie **.c** (w tym przykładzie kompilujemy program w **C**). Spodziewamy się, że efektem kompilacji poleceń będzie plik **.exe** wyświetlający nam okienko z wygenerowaną grafiką. Warto przyjrzeć się bliżej zestawowi poleceń wywołujących proces kompilacji z określonymi parametrami:

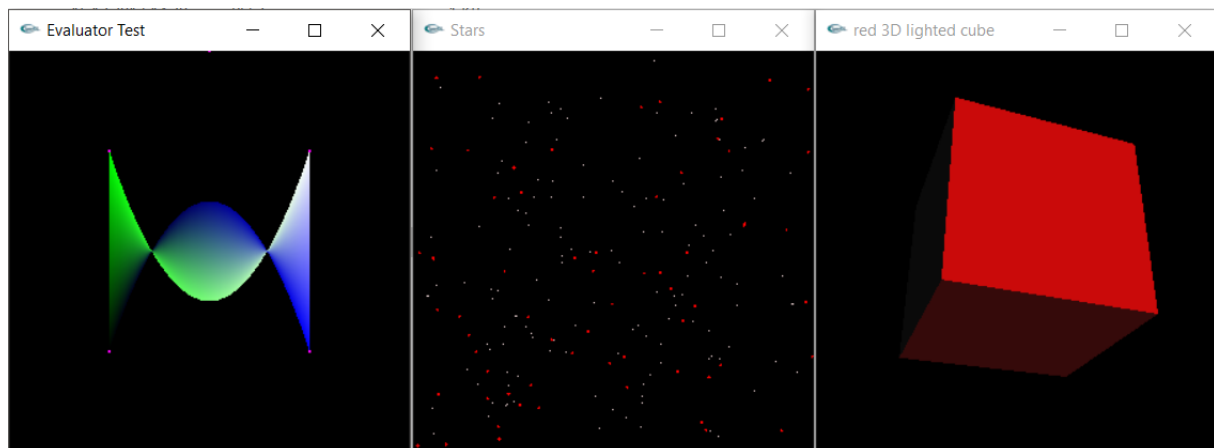
```
g++ -o test -Wall test.c -mwindows glut32.lib -lopengl32 -lglu32
```

Plikowi test.c (gdzie zawarty jest nasz zapisany program) przypisujemy plik wykonalny o tej samej nazwie poleceniem **-o**; **-Wall** wymusza wszystkie dostępne komunikaty; **-mwindows** generuje okno bez wyświetlenia konsoli; dalej mamy wywołanie bibliotek. Wywołanie jak kompilacja programu następuje bardzo szybko, co wprowało mnie w zdumienie.

Modyfikując plik któremu jako niezbędne określimy dyrektywy include:

```
#include <windows.h>
#include "GL/glut.h"
```

Możemy przetestować działanie wielu ciekawych programów, zawierających także elementy animowane. [5]



Ilustracja 1. Przykładowe programy uruchomione na moim komputerze. Zawartość okienek okazuje się być skalowalna, co dowodzi że nie mamy do czynienia ze statyczną grafiką.

KOD PROGRAMU

Trudno o program możliwie prosty. Okazuje się bowiem że nawet najprostsza grafika wymaga sporo poleceń. Łatwiej jest narysować niż opisać, a my tylko będziemy opisywać w sensie matematyczno-programistycznym.

Przykładowy kod dla OpenGL – (c) Janusz Ganczarski

```
// funkcja generująca scenę 3D

void Display()
{
    glClearColor( 1.0, 1.0, 1.0, 1.0 ); // kolor tła - zawartość bufora koloru
    glClear( GL_COLOR_BUFFER_BIT ); // czyszczenie bufora koloru
    glColor3f( 1.0, 0.0, 0.0 ); // kolor kwadratu
    glBegin( GL_POLYGON ); // początek definicji wielokąta
    // kolejne wierzchołki wielokąta
    glVertex3f( 0.0, 0.0, 0.0 );
    glVertex3f( 0.0, 1.0, 0.0 );
    glVertex3f( 1.0, 1.0, 0.0 );
    glVertex3f( 1.0, 0.0, 0.0 );
    glEnd(); // koniec definicji prymitywu
    glFlush(); // skierowanie poleceń do wykonania
    glutSwapBuffers(); // zamiana buforów koloru
}

// zmiana wielkości okna
void Reshape( int width, int height )
{ Display(); // generowanie sceny 3D }

// stałe do obsługi menu podręcznego
enum { EXIT // wyjście };

// obsługa menu podręcznego
void Menu( int value )
{
    switch( value )
    {
        // wyjście
        case EXIT:
            exit( 0 );
    }
}
```

```

}

int main( int argc, char * argv[] )
{
    glutInit( & argc, argv );// inicjalizacja biblioteki GLUT
    glutInitDisplayMode( GLUT_DOUBLE | GLUT_RGB );// inicjalizacja bufora ramki
    glutInitWindowSize( 400, 400 );// rozmiary głównego okna programu
    glutCreateWindow( "Kwadrat 1" );// utworzenie głównego okna programu
    glutDisplayFunc( Display );// dołączenie funkcji generującej scenę 3D
    // dołączenie funkcji wywoływanej przy zmianie rozmiaru okna
    glutReshapeFunc( Reshape );
    glutCreateMenu( Menu );// utworzenie menu podręcznego
#ifdef WIN32// dodatnie pozycji do menu podręcznego
    glutAddMenuEntry( "Wyjście", EXIT );
#else
    glutAddMenuEntry( "Wyjscie", EXIT );
#endif
    // określenie przycisku myszki obsługującego menu podręczne
    glutAttachMenu( GLUT_RIGHT_BUTTON );
    // wprowadzenie programu do obsługi pętli komunikatów
    glutMainLoop();
    return 0;
}

```

Kod ze strony <http://cpp0x.pl/> [6] choć na pierwszy rzut wydaje się obszerny. Szybko zauważymy, że wiele jego elementów odpowiada za samo zachowanie sceny. Musimy od początku ją narysować określić rozmiary, a nawet powiedzieć w jaki sposób okno ma być zamykane. Nie jest to bez znaczenia, że kwadrat rysujemy z pomocą kilku poleceń zamiast jednego; w przypadku bardziej złożonych operacji na obiekcie nie utracimy pełnej kontroli nad jego zachowaniem (np. w przypadku zmiany położenia jednego z wierzchołków czworokąta, nie będzie potrzebna dodatkowa operacja, gdyż każdy punkt opisano osobno).

WNIOSKI

Przygotowanie pola do pracy z OpenGL może okazać trudne; konfiguracja wymaga wielu zagadnień związanych ze znajomością systemu i języka C/C++. W Internecie stosunkowo trudno wyszukać prosty przewodnik. Praca z grafiką wymaga solidnych opanowania solidnych podstaw programowania, a także dobrej znajomości matematyki.

OpenGL nie jest frameworkiem dla adeptów programowania, im poleciłbym raczej Processing (<https://processing.org/>). Narzędzie to ma profesjonalne zastosowanie, nadaje się do tworzenia silników gier komputerowych; w przypadku prostych wizualizacji i animacji efektywniej użyć prostszych narzędzi.

BIBLIOGRAFIA

1. Praca zbiorowa, *Wikipedia.pl: OpenGL*, <https://pl.wikipedia.org/wiki/OpenGL> stan na dzień 30.12.2017
2. G. Sellers, R.S Wright, N. Haemel, *OpenGL Księga Experta wydanie 7* (wersja internetowa), wyd. Helion, s. 41 - dostępna stronie: <https://docer.pl/doc/5c8xev> dzień 09.01.2018
3. *MinGW - Getting Started*, http://www.mingw.org/wiki/getting_started stan na dzień 09.01.2018

Strona opisuje m.in. poprawne ustawienie zmiennej środowiskowej PATH co znacznie ułatwia pracę.

4. James Medison University / JMU - Department of Computer Science (2014), *Installing GLUT for MinGW*, https://w3.cs.jmu.edu/bernstdh/Web/common/help/cpp_mingw-glut-setup.php stan na dzień 09.01.2018

Test został przetłumaczony na język polski.

5. *Examples – OpenGL*, https://www.opengl.org/archives/resources/code/samples/glut_examples/examples/examples.html, stan na 09.01.2018
6. Janusz Ganczarski (Power) <http://www.januszg.hg.pl>, e-mail JanuszG@enter.net.pl; *Pierwszy program - Rozdział 2. Pierwszy program w OpenGL: rysowanie sceny 3D (kolor tła, czyszczenie bufora koloru, kolor obiektu, definiowanie obiektu, zamiana buforów koloru), tworzenie okna renerderingu, obsługa menu podręcznego*; stan na dzień 09.01.2018