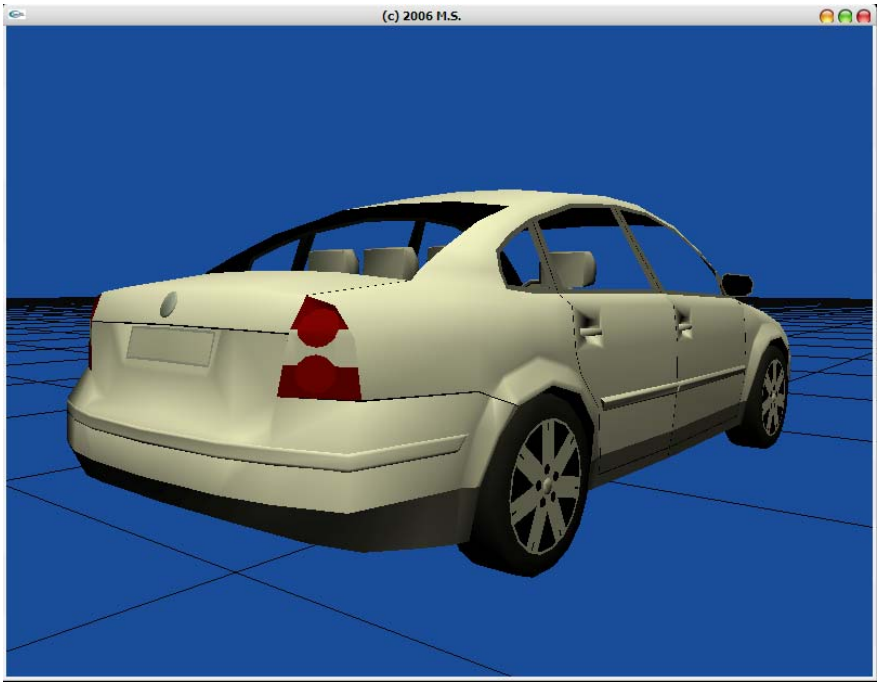
Wrocław, 20.06.2006 r.



Programowanie obiektowe

Projekt programistyczny

Mateusz Styrczula

- # Biblioteki i programy potrzebne przy kompilacji:
  - ## środowisko Microsoft Windows:
    - Microsoft C++ Optimizing Compiler and Linker (replaced by Visual C++ 2005 Express Edition):
      `http://msdn.microsoft.com/visualc/vctoolkit2003/`
      `http://msdn.microsoft.com/vstudio/express/visualC/default.aspx`

    - GNU make (MinGW version):
      `http://www.mingw.org/`
      `http://prdownloads.sf.net/mingw/mingw32-make-3.80.0-3.tar.gz?download`

    - GLUT for Win32:
      `http://www.xmission.com/~nate/glut.html`
      `http://www.xmission.com/~nate/glut/glut-3.7.6-bin.zip`

    - Platform SDK for Microsoft Windows:
      `http://www.microsoft.com/msdownload/platformsdk/sdkupdate/`

    - Materials and Geometry Format parser (MGF parser):
      `http://radsite.lbl.gov/mgf/`

  - ## środowisko Linux:
    - GNU Compiler Collection:
      `http://gcc.gnu.org/`

    - GNU make:
      `http://www.gnu.org/software/make/`

    - GLUT:
      `http://www.opengl.org/resources/libraries/glut.html`
      `http://www.mesa3d.org/`
      `http://freeglut.sourceforge.net/`

    - Materials and Geometry Format parser (MGF parser):
      `http://radsite.lbl.gov/mgf/`

- # Kompilacja programów:

Do katalogu ze źródłami programów dołączony jest plik `Makefile`. Jego możliwe cele to:

- **`lin`** - kompiluje i linkuje używając **`gcc`** (przewidziany do użycia w Linuksie)
- **`win`** - kompiluje i linkuje używając **`cl`** (przewidziany do użycia w środowisku Windows)
- **`console_win`** - kompiluje i linkuje używając **`cl`**, pozostawia dostęp do konsoli
- **`clean`** – usuwa wszystkie wyprodukowane binaria

Konsola w Windows:

Zgodnie z dokumentacją Microsoft, kompilator/linker **`cl.exe`** domyślnie linkuje program tak, by obsługiwał on konsolę. Aby to „obejść" (nie definiując w pliku źródłowym funkcji `WinMain` – zgodnie ze specyfikacją Win32api), należy użyć opcji linkera:

```
/link /SUBSYSTEM:WINDOWS /entry:mainCRTStartup
```

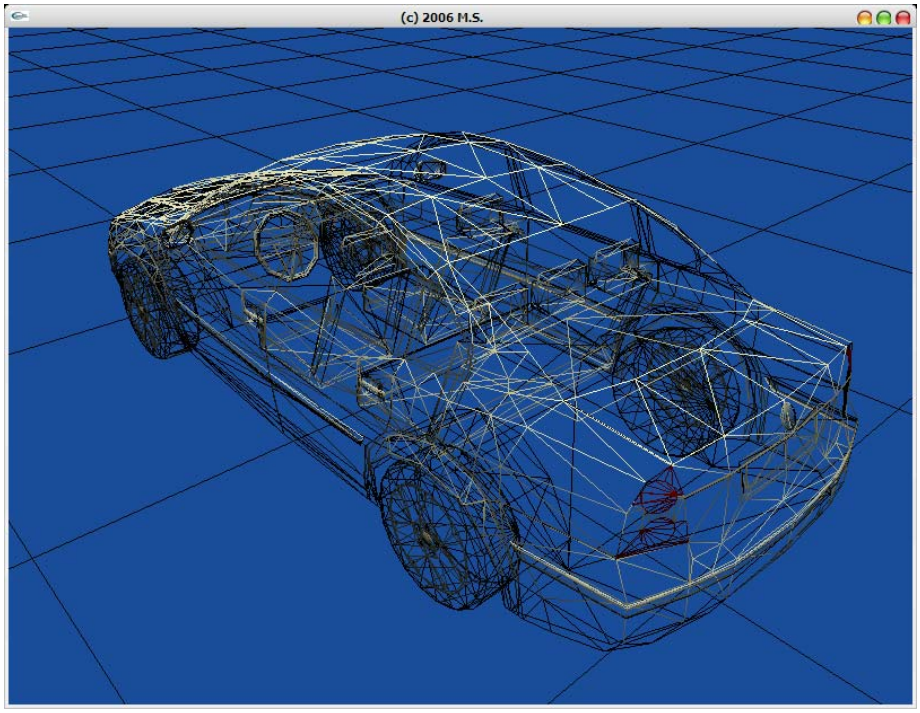
W tak zlinkowanym programie wszelkie operacje używające standardowego wejścia/wyjścia nie będą przynosiły żadnych rezultatów.

Optymalizacja kodu wynikowego:

Plik `Makefile` domyślnie nie zawiera wpisów dotyczących optymalizacji kodu wynikowego pod konkretną architekturę procesorów. Służą temu zmienne `MSARCHSET` oraz `GNUARCHSET`. Polecenia które należy wpisać znajdują się odpowiednio w dokumentacji kompilatora firmy Microsoft lub kompilatora GCC.

- # Opis programów:

Program **raycharles** jest (w zamierzeniu) przenośnym, wielosystemowym, wsadowym RayTracer'em, napisanym w języku C++. Aktualnie zaimplementowana funkcjonalność pozwala na oglądanie w czasie rzeczywistym scen w formacie MGF, generowanie statystyk ich dotyczących oraz przeprowadzenie procesu RayCasting'u. Integralnym elementem programu jest biblioteka **math3Dlib** zawierająca w pełni obiektową implementację operacji na kątach, macierzach (rozmiaru 4x4), punktach w przestrzeni trójwymiarowej, kwaternionach oraz wektorach trójwymiarowych. Cały projekt docelowo ma być platformą pozwalającą na testowanie/implementowanie szeregu różnych algorytmów związanych z szeroko pojętym zagadnieniem fotorealistycznego renderingu komputerowego.

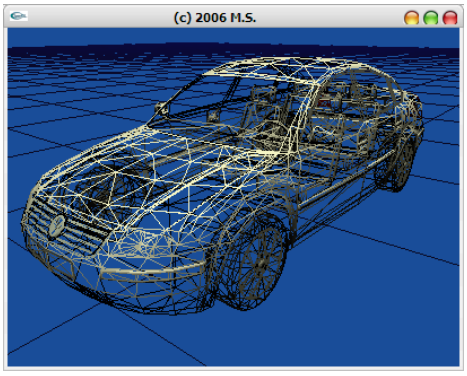


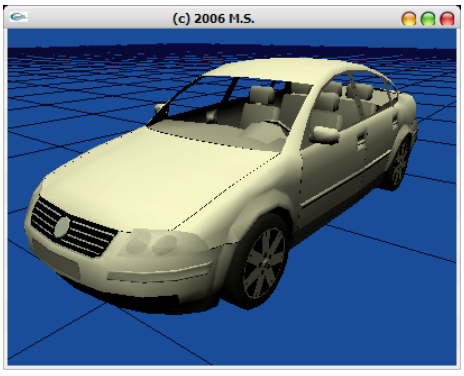
Obsługa programu **raycharles** sprowadza się do podania pliku `*.mgf` jako argument w linii poleceń. Jeśli plik `*.mgf` nie jest uszkodzony, oraz podsystem OpenGL działa prawidłowo na ekranie pojawi się okno podobne do pokazanego powyżej. Jest to widok z wirtualnej kamery na obiekt. „Oko" kamery porusza się zgodnie z ruchem myszy, natomiast „cel" kamery można przemieszczać przy pomocy przycisków:

- 'w' oraz 's' – wzdłuż osi z,
- 'a' oraz 'd' – wzdłuż osi x,
- '1' oraz '2' – wzdłuż osi y,

Odległość oka kamery od jej celu można regulować przyciskami '3' oraz '4'. Inne, możliwe do wykonania operacje to:

- 'q' – opuszczenie programu,
- 'z' – zmiana pomiędzy trybami wyświetlania „wireframe" i „solid":





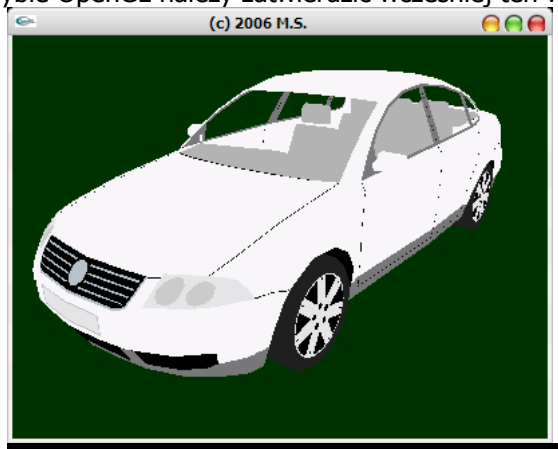
- 'f' – odwrócenie kierunku normalnych,
- 'p' – wypisanie na konsolę statystyk dotyczących załadowanej sceny:

```
      Triangles: 76647
      Materials: 19
Light Triangles: 3
```

- '`' – wypisanie na konsolę aktualnej pozycji oka i celu kamery oraz bieżącej rozdzielczości renderowanego okna:

```
eye ----------   x: 9.91942  y: 1.21869  z: 0.346394
center -------   x: 0  y: 0  z: 0
resolution --    w: 800  h: 600
```

- 'r' – przejście w tryb RayTracing'u (aby uzyskana pozycja kamery była zgodna z widokiem w trybie OpenGL należy zatwierdzić wcześniej ten widok kliknięciem lewym przyciskiem myszy):

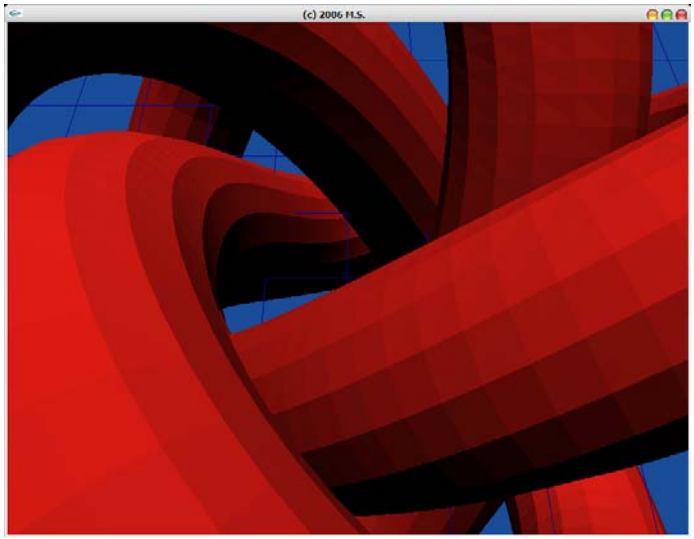



```
aspect:1.33333 fovy:0.785398
ray length: 1
```

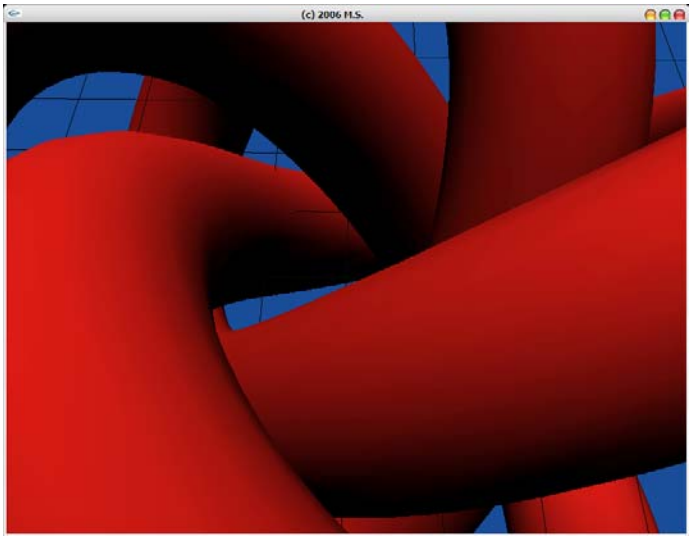
- 'r' – opuszczenie trybu RayTracing'u i powrót do wizualizacji czasu rzeczywistego,

W katalogu „bin" zostały umieszczone dwie wersje programu – `smooth` oraz `flat`. Różnią się one obsługą wektorów normalnych. Wersja `flat` ignoruje normalne występujące w plikach i oblicza je dla każdego trójkąta sceny jako iloczyn wektorowy dwóch wektorów leżących na krawędziach tego trójkąta, natomiast wersja `smooth` wykonuje takie obliczenia tylko dla trójkątów dla których w pliku nie zostały zdefiniowane normalne:

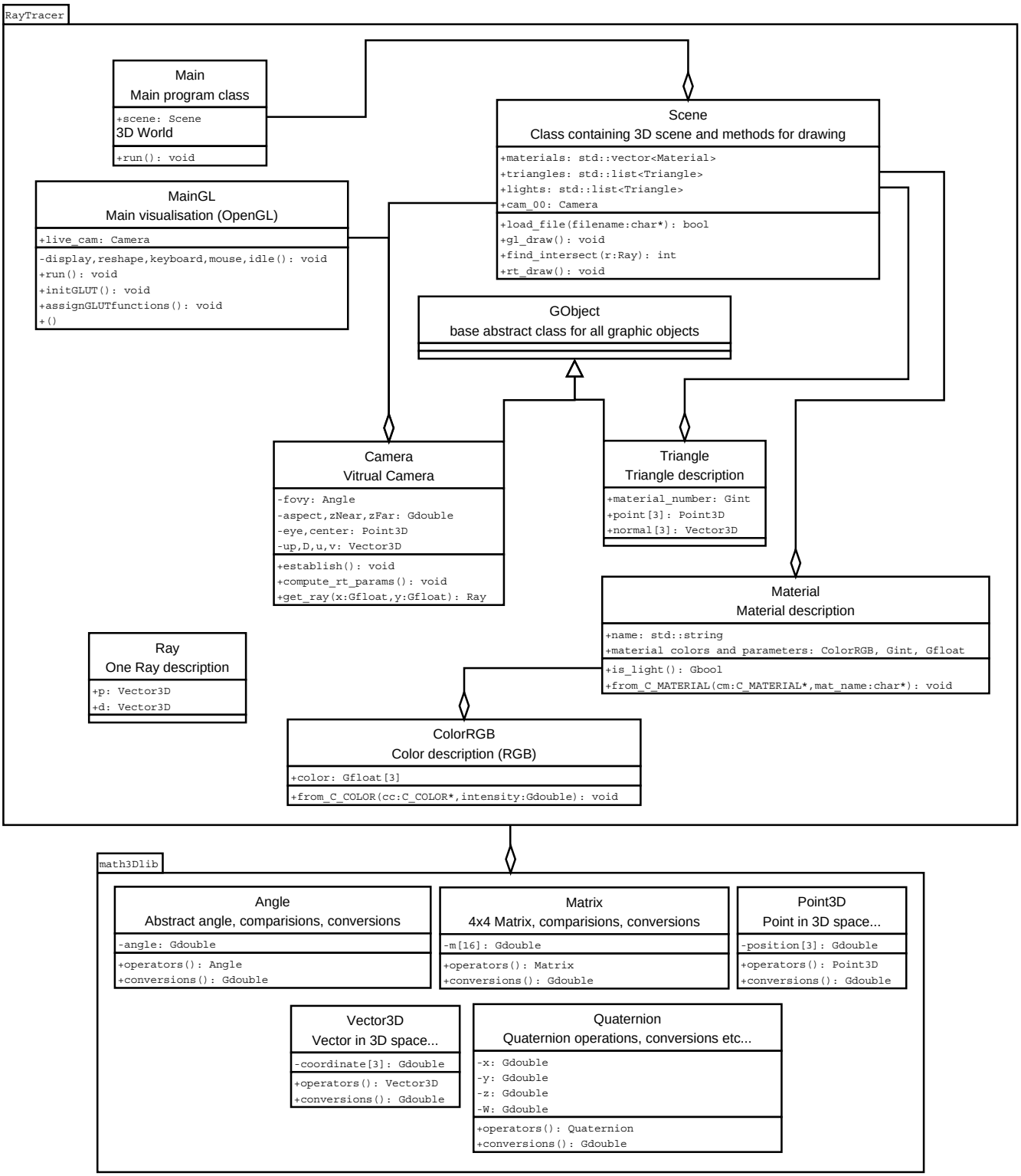
`flat`



`smooth`



- **Struktura wewnętrzna:**
  - **Diagram klas RayTracer'a i biblioteki math3Dlib:**



o **Główne cechy i funkcje klas:**

- klasa **`Main`** (pliki `main.h / main.cpp`):
  Główna klasa programu. Zawiera instancję klasy **`Scene`** oraz pola przechowujące ilość argumentów przekazanych do programu oraz tablicę tychże argumentów. Poza konstruktorem oraz destruktorem klasy, implementuje metodę `run` – wczytującą plik sceny oraz przekazującą sterowanie obiektowi **`MainGL`**.

- klasa **`MainGL`** (pliki `visGL/mainGL.h / visGL/mainGL.cpp`):
  Klasa odpowiedzialna za interakcję z użytkownikiem i obsługę podsystemu graficznego OpenGL. Implementuje m. in.:
  - inicjowanie kontekstu graficznego,
  - metody ustawiające parametry renderingu OpenGL i światło w scenie,
  - metody odświeżające kontekst graficzny (`*_display` oraz `*_reshape`),
  - metody obsługujące zdarzenia pochodzące od klawiatury i myszy,

  Klasa ta zawiera również instancję klasy **`Camera`** – w tym przypadku odpowiedzialnej za wyświetlanie obrazu w czasie rzeczywistym.

- klasa **`Scene`** (pliki `scene.h / scene.cpp`):
  Klasa zawierająca trójwymiarową scenę (przeczytaną z pliku MGF) oraz operacje rysowania w trybie GL oraz RT (RayTrace). Implementuje m. in.:
  - metodę `load_file()` ładującą wskazany plik `*.mgf`,
  - metodę `gl_draw()` rysującą scenę w trybie OpenGL,
  - metodę `rt_draw()` rysującą scenę w trybie RayTrace,
  - metodę `flip_normals()` odwracającą wektory normalne,
  - metody tworzące, zmieniające rozmiar oraz niszczące bufor bitmapy RGB,

  Klasa ta zawiera kontenery (w obecnej implementacji STL – do zamienienia przez bardziej odpowiednie i wyrafinowane struktury) przechowujące materiały, trójkąty i światła („świecące" trójkąty) oraz instancję klasy **`Camera`** – w tym przypadku odpowiedzialnej za wyświetlenie obrazu w trybie RT.

- klasa **`Camera`** (pliki `visGL/camera.h / visGL/camera.cpp`):
  Klasa umożliwiająca manipulację wirtualnymi kamerami. Przechowuje ich podstawowe parametry (takie jak pole widzenia, płaszczyzny obcinające, punkty „oka" i „celu", wektor wskazujący kierunek „góra", oraz wektory potrzebne przy procesie RayTracing'u). Implementuje m. in.:
  - szereg konstruktorów pozwalających na stworzenie kamery,
  - metody pozwalające na przesuwanie kamery w trójwymiarowej przestrzeni,
  - metody pozwalające na zmianę/odczyt wszystkich podstawowych parametrów,
  - metodę `establish()` ustawiającą parametry wirualnej kamery OpenGL,
  - metody ustawiające parametry potrzebne przy procesie RayTracing'u,
  - metodę get_ray() zwracającą całe spektrum promieni przechodzących przez scenę,

- klasa **`Triangle`** (pliki `geometry/triangle.h / geometry/triangle.cpp`):
  Klasa opisująca trójkąt w przestrzeni. Zawiera pola przechowujące punkty tworzące trójkąt, wektory normalne dla każdego z punktów oraz przypisany numer materiału.

- klasa **`Material`** (pliki `geometry/material.h / geometry/material.cpp`):
  Klasa zawierająca pola przechowujące nazwę materiału oraz kolory i natężenia poszczególnych parametrów opisujących materiał. Posiada zaimplementowaną metodę konwertującą odpowiednie wartości z opisu danego materiału w pliku MGF.

- klasa **`ColorRGB`** (pliki `geometry/color.h / geometry/color.cpp`):
  Zawiera pola przechowujące wartości Red, Green i Blue (Czerwony, Zielony i Niebieski). Implementuje m. in.:
  - metody pozwalające na konwersję kolorów z modelu XYZ (stosowanego w MGF),
  - metody pozwalające na odczyt/zmianę poszczególnych składowych,

Precyzyjne opisy wszystkich zaprezentowanych metod oraz klas biblioteki **`math3Dlib`** znajdują się w odpowiednich plikach nagłówkowych.

W programie została użyta oryginalna implementacja algorytmu Moller'a znajdującego przecięcie prostej i trójkąta (plik `rt/triangle_moller.cpp`).

Kod konwersji modelu kolorów XYZ -> RGB (plik `geometry/cvrgb.cpp`) pochodzi z biblioteki parsera MGF.

• Literatura:

- **„ANSI C"** – B.W. Kernighan, D. M. Ritche, WNT 2002,
- **„Thinking in C++"** (tom 1 i 2) – Bruce Eckel, Helion 2002,
- **„Wprowadzenie do Grafiki Komputerowej"** – Foley, van Dam, Keiner, Hughes, Phillips,
- **„OpenGL Programming Guide"** – http://www.rush3d.com/reference/opengl-redbook-1.1/
- **„OpenGL Reference Manual"** – http://www.rush3d.com/reference/opengl-bluebook-1.0/
- **„The OpenGL Utility Toolkit (GLUT) Programming Interface API Version 3"** – http://www.opengl.org/resources/libraries/glut/spec3/spec3.html
- **„Graphics Gems"** – A. S. Glassner, Academic Press, Boston 1990,
- **„Graphics Gems II"** – J. Arvo, Academic Press, Boston 1991,
- **„Graphics Gems III"** – D. Kirk, Academic Press, Boston 1992,
- **„Graphics Gems IV"** – P. S. Heckbert, Academic Press, Boston 1994,
- **„Graphics Gems V"** – A. W. Paeth, 1995,
- **„Global Illumination Compendium"** – P. Dutre, 2003,
- **„Heuristic Ray Shooting Algorithms"** – V. Havran Ph. D., 2000,