Grafika 3D – charakterystyka zajęć

W ciągu całego semestru do zrealizowania są 3 etapy projektu z grafiki komputerowej. Do zdobycia jest w sumie 60 punktów z laboratorium oraz 40 punktów z kolokwium na wykładzie.

Etap	Termin Oddania	Max Punktacja
1	31 października 2013	20 punktów
2	12 grudnia 2013	20* punktów
3	25 stycznia 2013	20 punktów

* - w drugim etapie jest możliwość przekroczenia 20 punktów na zasadzie przeniesienia nadmiarowych punktów do etapu nr 3.

Wystarczy pojawić się na trzech zajęciach w celu oddania projektów, jednak w każdym terminie będzie możliwość skorzystania z konsultacji w sali 304. Nie trzeba uprzedzać wcześniej o chęci przyjścia na konsultacje, ale wysłanie wcześniej swojego problemu lub pytania mailem gwarantuje szybsze uzyskanie odpowiedzi.

Projekty, które nie zostaną oddane w terminie, uzyskują 0 punktów, natomiast mogą być oddane w ramach kolejnego etapu. Nie ma jednak możliwości przekroczenia limitu 20 punktów za dany etap.

Do zaliczenia zadania wymagane jest **zaprezentowanie programu na zajęciach** oraz **wysłanie mailem kodu i wszystkich zasobów niezbędnych do prawidłowego działania** (np. siatek modeli, tekstur, plików shaderów).
Proszę dołączyć także krótką instrukcję z opisem sterowania klawiszami/myszką – może być w pliku tekstowym lub bezpośrednio w treści maila.

Jeśli ktoś zdecyduje aby dołączyć również plik *.exe, to sugeruję zmienić rozszerzenie i umieścić całą paczkę w archiwum z hasłem np. `gk3d`. Serwer alpha często odrzuca załączniki z plikami binarnymi.

Zadania jak również pytania dotyczące projektów należy wysłać na adres:
p.aszklar@mini.pw.edu.pl

W czym piszemy?

Do wykonywania zadań można użyć następujących API:

- OpenGL (C, C++)
- Direct3D (C++)
- XNA (C#)
- Managed DirectX (C#, VB. NET)

Uwaga: W XNA jak i nowszych wersjach OpenGL (od wersji core OpenGL 3.1) oraz Direct3D (od wersji 10) zrezygnowano z tzw. stałego potoku renderowania (fixed-function pipeline). Jedyną możliwością jest używanie shaderów.

Krótka charakterystyka

OpenGL

- z założenia międzyplatformowy (poza kilkoma wyjątkami rzeczywiście taki jest)
- jedyne API programowania grafiki 3D, którego natywnie można używać na systemach z rodzin Linux oraz UNIX (np. Mac OS-X)
- nieobiektowy:
    - wywołujemy globalnie dostępne funkcje
    - co najwyżej dostajemy identyfikatory (w postaci liczb całkowitych) pewnych elementów (np. tekstur). Ewentualne opakowanie tych elementów w klasy należy wyłącznie do programisty
- maszyna stanowa – OpenGL zarządza wewnętrznym stanem wyświetlania, którego poszczególne elementy modyfikujemy przy pomocy około 250 funkcji dostępnych w podstawowej implementacji;
  Dodatkowe funkcje dostępne są w postaci tak zwanych rozszerzeń.
- wspierany przez środowiska open-source;
  Rozwojem OpenGL, w przeciwieństwie do Direct3D, nie zajmuje się pojedyncza korporacja, a konsorcjum Khronos Group oraz rada ARB (Architecture Review Board), w której skład obecnie wchodzi 9 firm;
- można używać zarówno tradycyjnego, stałego potoku renderowania (fixed-function pipeline – tylko w starszych wersjach) jak i programów cieniowania wierzchołków i pikseli (shaderów);
  OpenGL posiada swój własny język wysokiego poziomu do pisania shaderów: GLSL (OpenGL Shading Language);
- stosowany głównie:
    - w CAD i wizualizacjach naukowych
    - w telefonach komórkowych (iPhone, Android, Symbian)
    - w konsolach (PSP, Playstation3)

Direct3D (podzbiór DirectX odpowiedzialny za grafikę 3D)

- opracowany przez Microsoft i dostępny wyłącznie na systemach Windows (nie uwzględniając uciekania się do emulacji lub maszyn wirtualnych);
- w dużym stopniu obiektowy (aczkolwiek nie do końca - jak to zwykle w programowaniu bliskim sprzętowi bywa);
- stosunkowo trudny do opanowania na samym początku (np. w przeciwieństwie do OpenGL wymaga napisania dużo kodu do inicjalizacji), jednak po opanowaniu podstaw staje się całkiem poręczny;
- zawiera użyteczne moduły pomocnicze np. DXUT wspomagający podstawowe operacje (tworzenie okna, urządzenia, pętli wyświetlania, obsługa zdarzeń urządzenia) lub biblioteka matematyczna (zawiera operacje na wektorach, macierzach, kwaternionach, płaszczyznach)
- można używać zarówno tradycyjnego potoku renderowania (fixed-function pipeline – tylko w starszych wersjach) jak i programów cieniowania wierzchołków i pikseli (shaderów).
  Direct3D posiada swój własny język wysokiego poziomu do pisania shaderów: HLSL (High Level Shading Language).
- stosowany głównie w grach na PC i Xbox 360

XNA

- Nie rozwijane już API wprowadzone przez Microsoft jako następca zarządzanego DirectX (Managed DirectX);
- dostępne na Windows, Xbox 360 oraz Windows Phone, lecz istnieje wersja dla innych systemów rozwijana w ramach projektu Mono;
- bazuje na Direct3D;
- obiektowe;
- wymaga darmowego XNA Game Studio (nakładka na Visual Studio);

Managed DirectX

- API wycofane i oznaczone przez Microsoft jako „deprecated”;
- następcą Managed DirectX jest XNA;
- MDX to lekko okrojony port DirectX do środowiska .NET. Pojawiły się wersje przygotowane dla .NET Framework 1.1 oraz 2.0, jednak w każdej nowszej wersji .NET framework można używać starszego MDX;
- obiektowe - intefejsy DirectX istnieją tutaj w postaci obiektów C#;
- rzadko spotykany w komercyjnych zastosowaniach, ale wady MDX nie ograniczają w żaden sposób możliwości wykonania zadań na laboratorium.