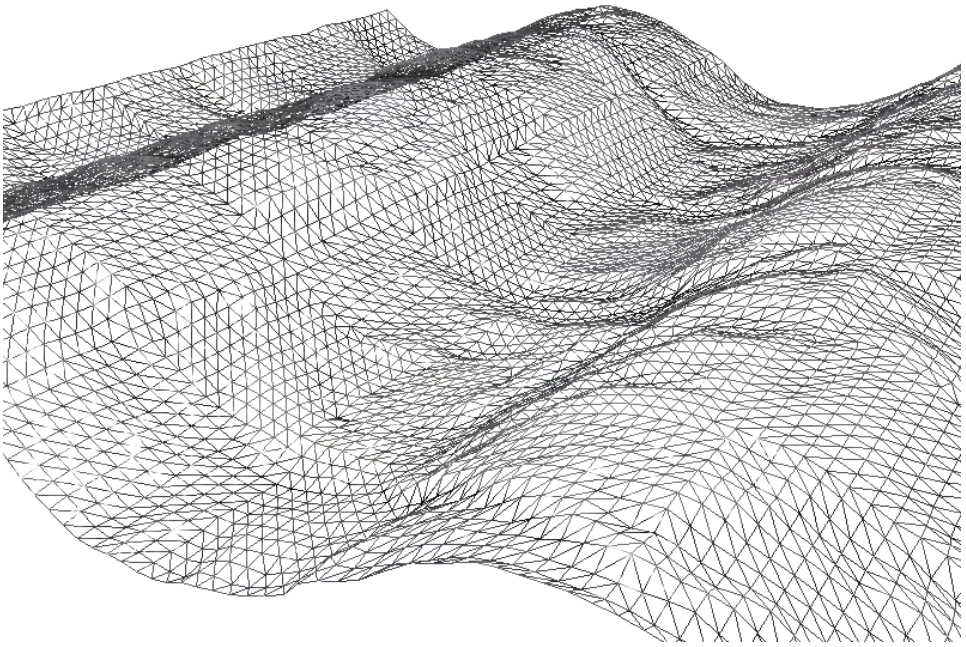
Projekt 4. Teselacja

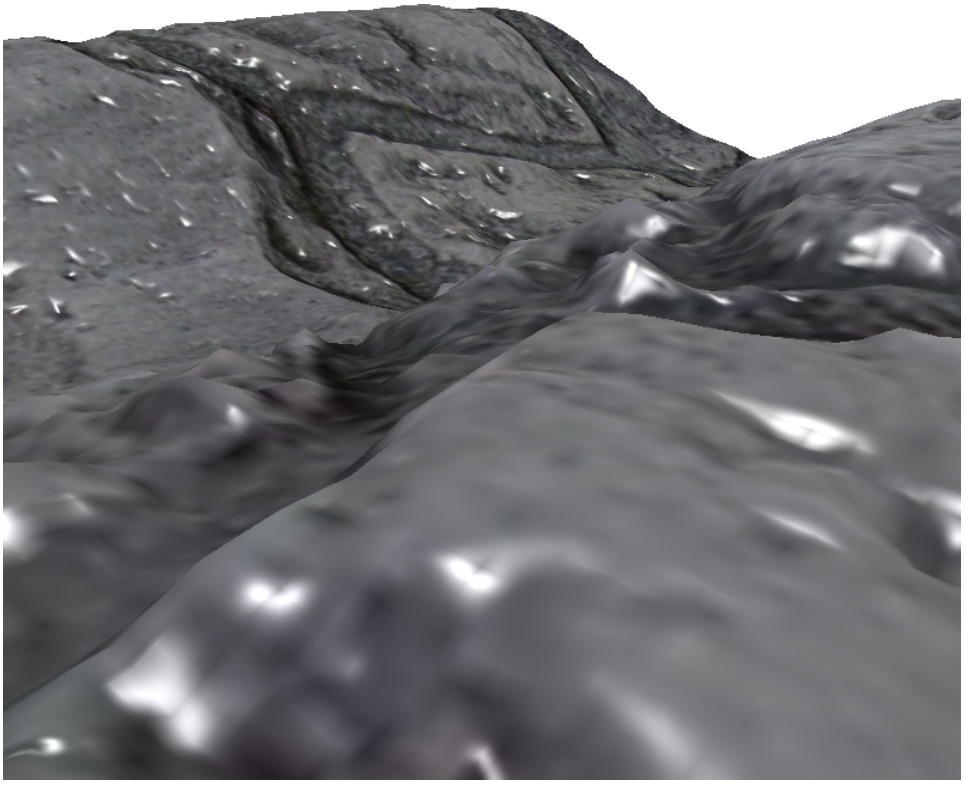
Paweł Aszklar
P.Aszklar@mini.pw.edu.pl

Warszawa, 29 maja 2014

1 Wstęp

1. Projekt jest indywidualny
2. Termin oddania projektu to 12. czerwca
3. Rozwiązanie należy zaprezentować na laboratoriach bądź przesłać spakowany kod źródłowy razem pozostałymi. plikami niezbędnymi do uruchomienia programu na adres P.Aszklar@mini.pw.edu.pl.
4. Projekt można pisać w C# lub C++ korzystając z biblioteki DirectX lub OpenGL i języka shaderów HLSL, GLSL lub Cg.
5. Zadanie składa się z kilku części. Nie trzeba realizować wszystkich.





2 Opis zadania

2.1 Część I [1 pkt]

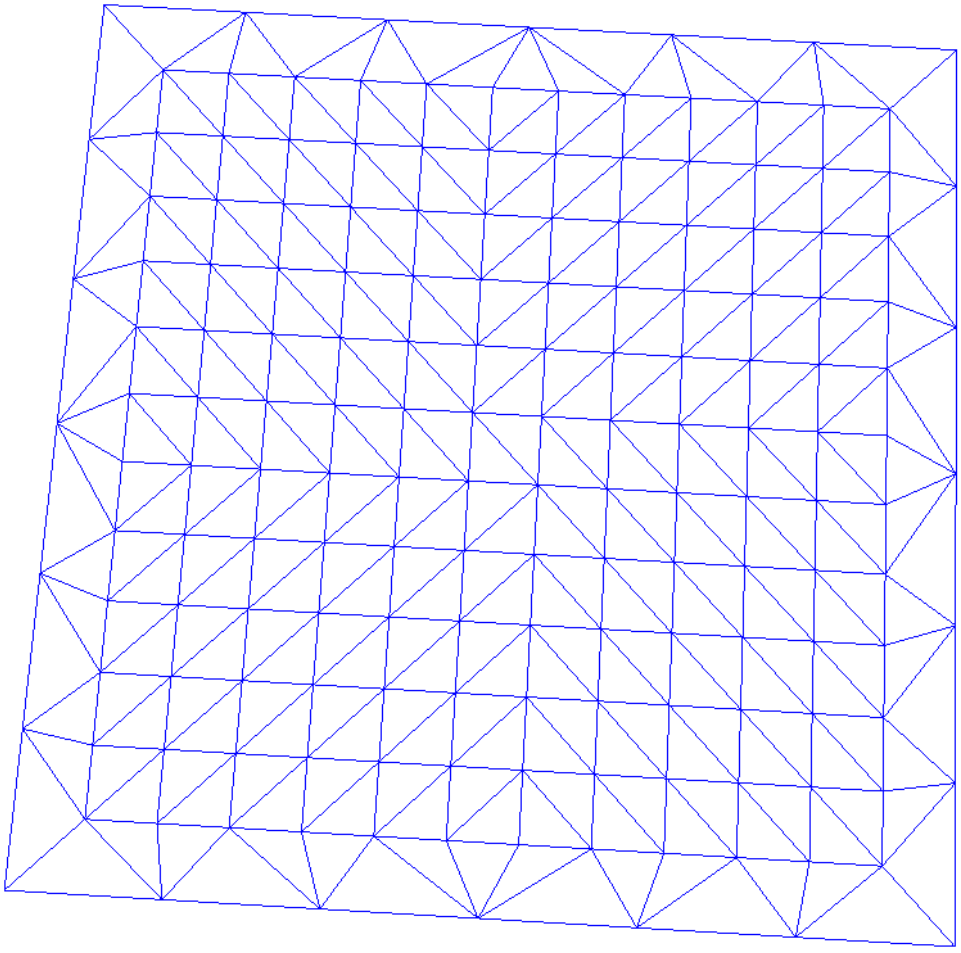
a) Teselacja powierzchni czworoboku. Program powinien wyświetlać czworobok opisany za pomocą 4 wierzchołków (stanowiących jego *punkty kontrolne*).

b) Należy wykorzystać dwa elementy programowalnego potoku renderowania do podziału powierzchni czworoboku na siatkę trójkątów. Te dwa nowe shadery to *shader powłoko* (*Hull Shader*) oraz *shader dziedziny* (*Domain Shader*). — W opisie zadania używana jest nomenklatura z biblioteki DirectX. W bibliotece OpenGL użycie teselacji przebiega w bardzo podobny sposób, jednak poszczególne etapy potoku renderowania znane są po innymi nazwami. *Hull Shader* z DirectX nazwany został *Tessellation Control Shader* natomiast *Domain Shader* jako *Tessellation Evaluation Shader*.

c) Podział powinien być sterowany za pomocą dwóch współczynników przekazywanych do shaderów przez program: współczynnika teselacji krawędzi oraz współczynnika teselacji wnętrza czworoboku. Program

powinien umożliwiać zmianę wartości tych współczynników w trakcie działania, za pomocą klawiatury.

d) Siatka powinna być wyświetlana ze stałym kolorem w trybie krawędziowym (rysowane powinny być tylko krawędzie trójkątów, bez ich wypełniania). Wyłączyć należy też obcinanie ścian tylnych.



2.2 Część II [+2 pkt]

a) Teselacja płatka Béziera. Tym razem generowana siatka opisana powinna być przez 16 punktów, będących punktami kontrolnymi prostokątnego płatka Béziera stopnia (3, 3)

b) Punkty wygenerowanej siatki leżeć powinny na powierzchni płatka. Do wyznaczenia ich współrzędnych można skorzystać bezpośrednio ze

wzoru:

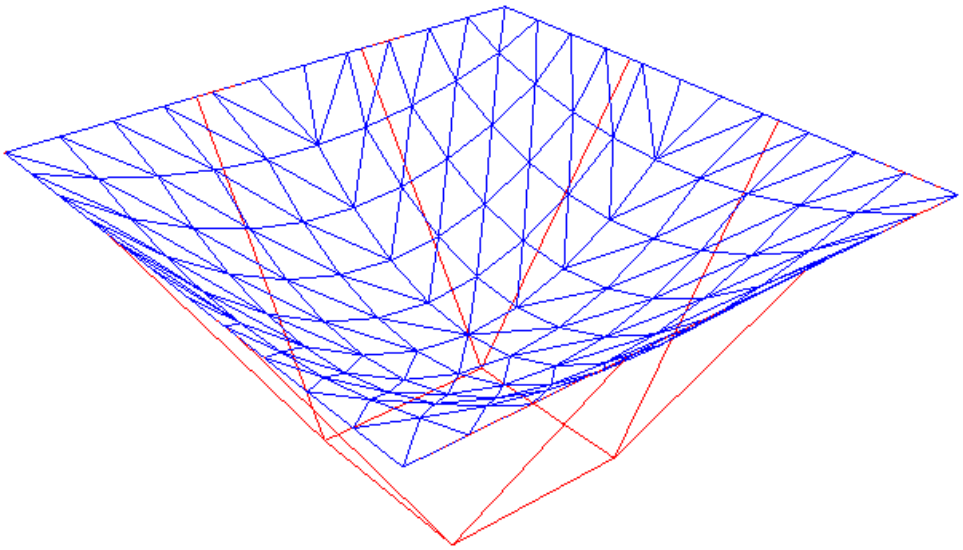
$$p(u,\, v) = \sum_{i=0}^{3} \left(\sum_{j=0}^{3} p_{ij} B_j^3(v) \right) B_i^3(u)$$

, gdzie

$$B_i^3(t) = \binom{3}{i} t^i (1-t)^{3-i}$$

c) Zdefiniuj przynajmniej dwa różne zestawy punktów kontrolnych, pomiędzy którymi można przełączać w trakcie działania programu.

d) Poza wygenerowaną siatką, program powinien mieć też możliwość wyświetlania siatki wielościanu kontrolnego danego płatka

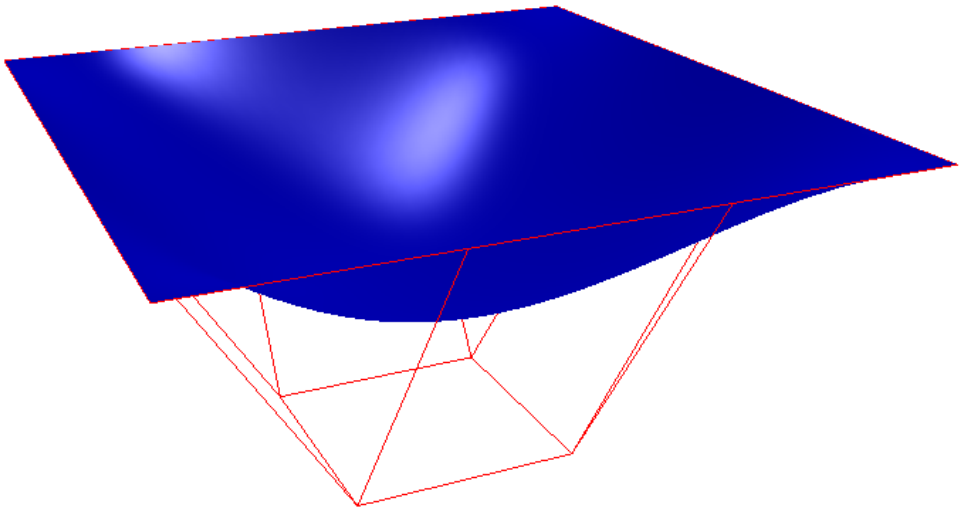


2.3 Część III [+2pkt]

a) Płatek powinien być wyświetlany tak jak w poprzedniej części, ale dodatkowo w każdym punkcie wygenerowanej siatki należy wyznaczyć wektor normalny do powierzchni danego płatka.

b) Trójkąty siatki powinny być rysowane wypełnione i pocieniowane zgodnie z modelem Phonga. Do sceny należy dodać pojedyncze punktowe źródło światła koloru białego.

c) Program powinien pozwalać na przełączanie pomiędzy rysowaniem pocieniowanej powierzchni oraz widokiem krawędziowym.



2.4 Część IV [+2.5 pkt]

a) Tym razem program powinien wyświetlać powierzchnię złożoną z siatki 4×4 płatków Béziera połączonych z ciągłością co najmniej C^0.

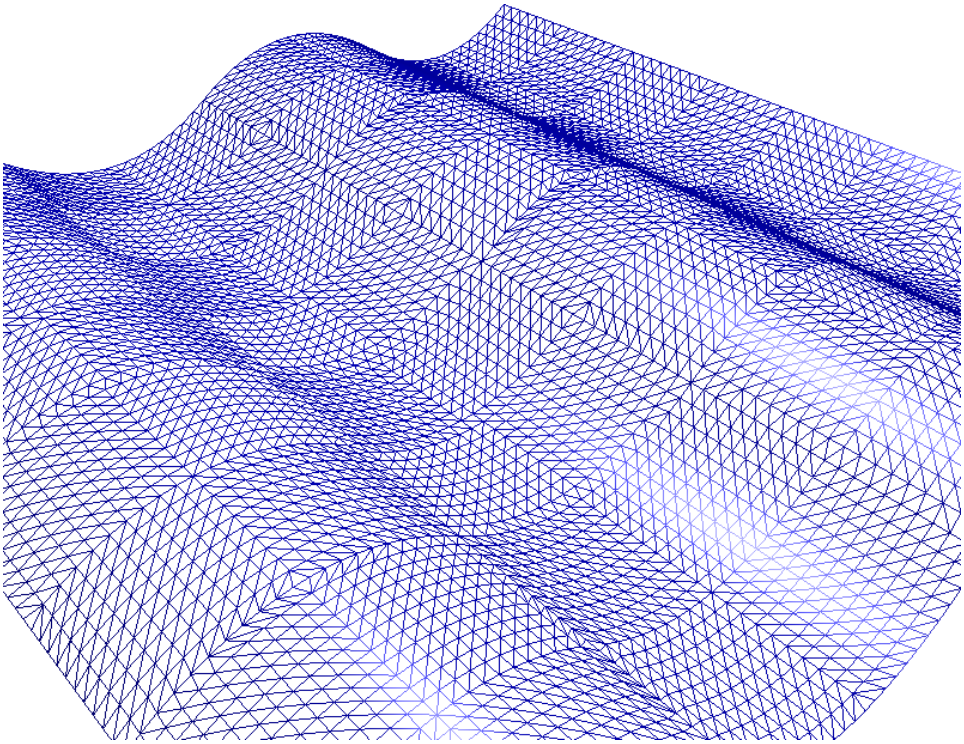
b) Zaimplementuj dynamicznie wyznaczany poziom szczegółowości (Level Of Detail) generowanej siatki w zależności od odległości płatka od kamery. Wartość współczynnika teselacji można wyznaczyć na podstawie wartości współrzędnej głębokości w układzie kamery (odległość od kamery) ze wzoru:

$$\text{factor}(z) = -16 * \log_{10}(z * 0.01)$$

Wzór ten zakłada, ze wartości współrzędnej głębokości należą do przedziału [0.01, 100].

c) Wygenerowana siatka nie powinna zawierać dziur. W tym celu należy zapewnić, że wspólne krawędzie sąsiadujących płatków ulegają podziałowi z tą samą wartością współczynnika teselacji.

d) Wartości współczynników teselacji, które do tej pory przekazywane były do shaderów powinny zostać użyte do modyfikacji wartości zwracanych przez funkcję factor, w efekcie globalnie zwiększając lub zmniejszając poziom szczegółowości sceny.



2.5 Część V [+2.5 pkt]

a) Dodaj do programu trzy tekstury: koloru, normalnych oraz przemieszczeń (wysokości), reprezentujących tą samą powierzchnię. Przykładowe tekstury do pobrania: http://www.mini.pw.edu.pl/~aszklarp/gk2/teselacja_tekstury.7z

b) Dla każdego wierzchołka wygenerowanej siatki należy pobrać wartość wysokości z tekstury przemieszczeń (*diplacement mapping*). Wartość ta będzie z przedziału [0, 1] i może wymagać przeskalowania.

c) Położenie wierzchołka należy przesunąć wzdłuż wektora normalnego w tym punkcie o wczytaną odległość.

d) Do pobierania wartości z tekstury w shaderze dziedziny (*Domain Shader*) należy użyć funkcji `SampleLevel`:

```
1 float h = dispMap.SampleLevel(sampler, texCoord, mipLevel).x;
```

Jej trzecie parametr pozwala określić poziom mipmap, która zostanie użyta przy wczytywaniu z tekstury. Wartość tą musimy podać ręcznie, gdyż tylko w shaderze pikseli może być ona wyznaczona automatycznie.

e) Poziom mipmap można wyznaczyć używając z następującego wzoru (korzystającego ze zdefiniowanej wcześniej funkcji factor):

$$\text{mipLevel}(z) = 6 - \log_2(\text{factor}(z))$$

Poziom ten musi zależeć wyłącznie od współrzędnej z danego punktu, aby wierzchołki w różnych płatkach o tych samych współrzędnych otrzymały takie samo przesunięcie (inaczej siatka będzie niespójna).

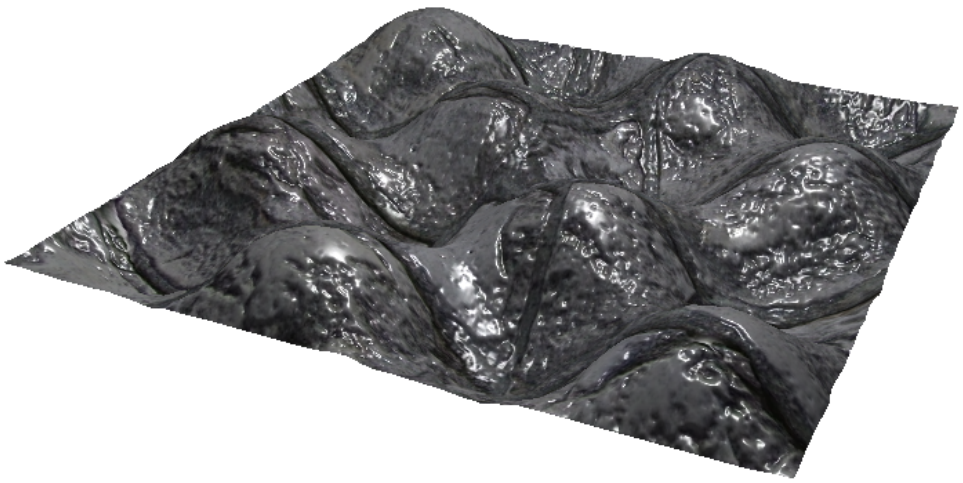
f) Współrzędne tekstury powinny być dobrane tak, aby wybrane tekstury rozciągnięte zostały bez zawijania na wszystkie 16 płatków.

g) Do shadera pikseli przekazać należy zarówno położenie, wektor normalny a także współrzędne tekstury, wektory styczny oraz binormalny.

h) Kolor powierzchni dla rysowanego piksela w shaderze pikseli pobrany powinien być z tekstury koloru, natomiast wektor normalny, z tekstury normalnych. W przykładowej teksturze wektory normalne zakodowane są w ten sam sposób, co w poprzednim projekcie. Zwrócić uwagę należy na fakt, że wektor normalny z tekstury określony jest w stycznym do powierzchni układzie lokalnym punktu (bazą tego układu są wektory styczny, binormalny i normalny otrzymane z wcześniejszych etapów potoku renderowania). Należy go przekształcić do układu globalnego (lub widoku) korzystając z poniższego wzoru:

$$texNorm.x * tangent + texNorm.y * binormal + texNorm.z * normal$$

Wektora otrzymanego w wyniku tej operacji należy użyć przy cieniowaniu powierzchni.



3 Program przykładowy

Przykładowy program dostępny do pobrania pod adresem http://www.mini.pw.edu.pl/~aszklarp/gk2/teselacja_start.7z pozwala zapoznać się z podstawami stosowania shaderów powłoki i dziedziny. Demonstruje on działanie potoku renderowania rozszerzonego o dwa dodatkowe shadery i stanowi dobry punkt startowy przy implementacji tego projektu.