Programowanie w API Graficznych

Laboratorium

DirectX 12 – ćwiczenie 5

1 Cele ćwiczenia

W tym ćwiczeniu zapoznasz się z kolejnym istotnym etapem potoku renderującego – teselacją. Etap ten pozwala wygenerować dodatkową geometrię dla rysowanych brył, a co za tym idzie, uczynić je bardziej szczegółowymi. Etap teselacji został wprowadzony w DirectX 11, jest on podzielony na cztery podetapy – powłoki (ang. hull), stały etap powłoki (ang. constant), teselacja i i domeny (ang. domain), trzy z nich wymaga odzielnego programu cieniującego:

- stały program cieniujący powłoki (constant shader)
- program cieniujący punkty kontrolne powłoki (hull shader)
- program cieniujący dziedziny (domain shader)

Etap teselacji jest wykonywany automatycznie, możesz go jednak konfigurować za pomocą atrybutów hlsl. Etap ten może wydawać się początkowo bardzo podobny do etapu cieniowania geometrii, po wykonaniu zadania zastanów się na różnicami pomiędzy nimi oraz efektami jakie można uzyskać stosując je.

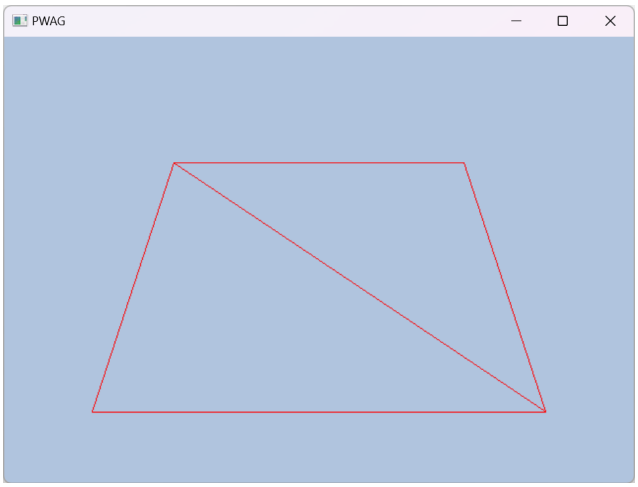
2 Zadania

Ćwiczenie zostało podzielone na dwa etapy. W pierwszym etapie zapoznasz się ze wspomnianymi we wstępie programami cieniującymi, a w drugim napiszesz program mapujący przemieszczenia.

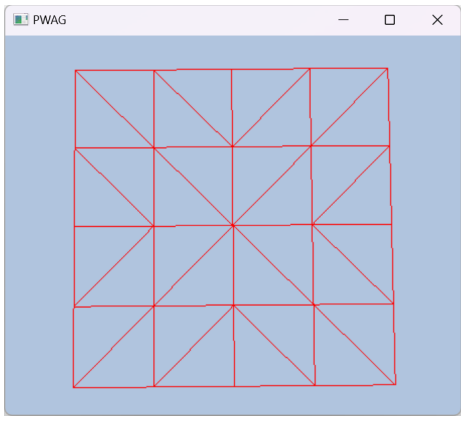
2.1 Teselacja

Zaraz po uruchomieniu aplikacji powinieneś zobaczyć czerwony kwadrat. W poprzednich zadaniach w buforze wierzchołków znajdowały się współrzędne wierzchołków trójkątów, w tym zadaniu w buforze znajdują się tylko cztery wierzchołki kwadratu. Otwórz funkcję `RenderWidget::LoadGeometry()`, podobnie jak w poprzednich zadaniach odpowiada ona za wygenerowanie współrzędnych rysowanego obiektu, zwróć szczególną uwagę na rozmieszczenie wierzchołków, jest to kluczowe dla zrozumienia tego zadania. Typem geometrii z jakim masz teraz do czynienia jest `D3D11_PRIMITIVE_TOPOLOGY_4_CONTROL_POINT_PATCHLIST`, transformacja współrzędnych na współrzędne trójkąta nastąpi na etapie teselacji.

Zmień tryb renderowania z pełnego (solid) na siatkę (wireframe), tak jak na obrazku poniżej, powinieneś zobaczyć siatkę renderowanego kwadratu. Wykonywane na etapie teselacji programy cieniujące zostały już za ciebie napisane. Otwórz plik shaders.fx i dokładnie zapoznaj się z tymi programami (ConstantHS, HS_Main i DS_Main), ważne jest abyś potrafił powiedzieć za co odpowiada każda linia kodu.



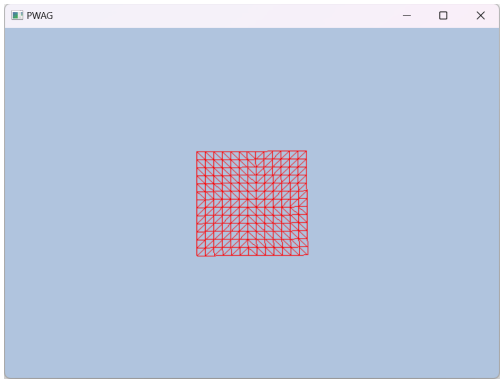
W stałym programie cieniowania powłoki zmodyfikuj współczynniki teselacji na 4. Powinieneś uzyskać następujący obraz:



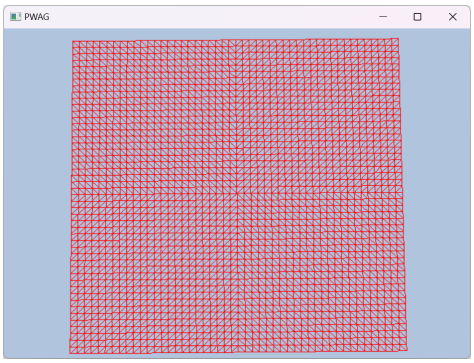
Sprawdź jak na wynik teselacji wpływają inne wartości tego współczynnika. Zachęcam cię abyś spróbował także zmodyfikować pozostałe programy i zobaczył jak poszczególne komendy wpływają na wygenerowaną geometrię.

2.2 Dynamiczna teselacja

Współczynniki teselacji odpowiadają za ilość wygenerowanej geometrii. Częstą optymalizacją stosowaną w grach komputerowych jest ograniczenie ilości generowanej geometrii w zależności od odległości do kamery, im obiekt jest dalej tym mniej szczegółowym go wyświetlamy. Zaimplementuj podobny efekt w programie cieniującym, położenie kamery znajduje się w strukturze `gCameraPosition`, a generowany obiekt znajduje się w pozycji (0, 0, 0). Powinieneś uzyskać efekt jak na obrazkach umieszczonych poniżej.

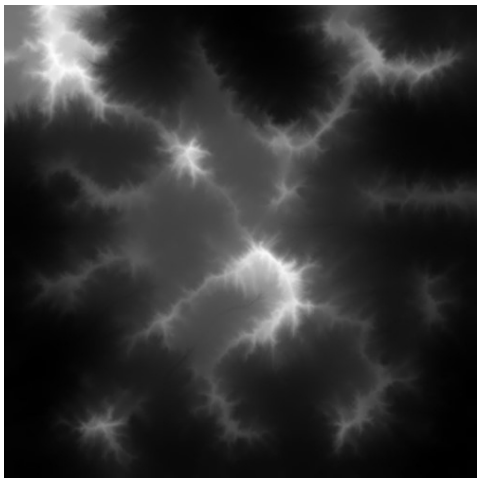


Obiekt umieszczony blisko kamery zbudowany jest z większej ilości trójkątów niż umieszczony dalej.



2.3 Współrzędne UV

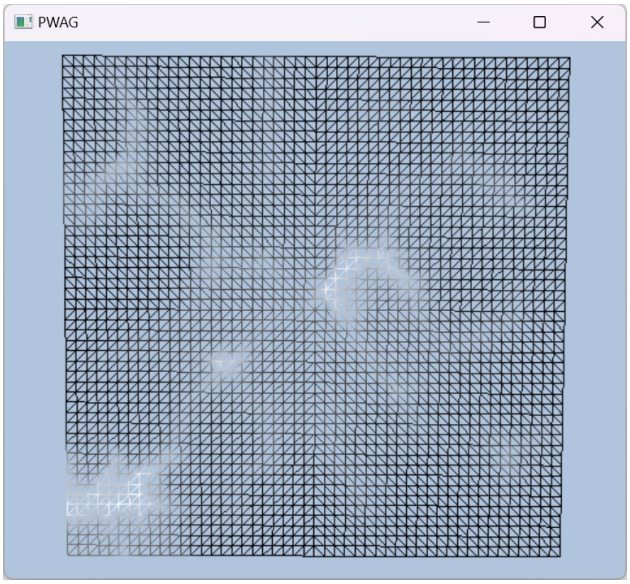
Do tej pory na etapie teselacji generowałeś jedynie współrzędne 3D rysowanego modelu, w tym zadaniu wygenerujesz współrzędne uv tekstury i nałożysz teksturę na generowany model. Kod odpowiedzialny za załadowanie zaprezentowanej poniżej tekstury został już za ciebie napisany (obiekt `gTexture1`), pozostaje ci tylko wczytać ją w programie cieniującym piksele.



Nim to jednak zrobisz, będziesz musiał wygenerować współrzędne uv. Na ogół odpowiada za to program cieniujący domeny, możesz to zrobić w sposób analogiczny jak generowane są współrzędne 3D:

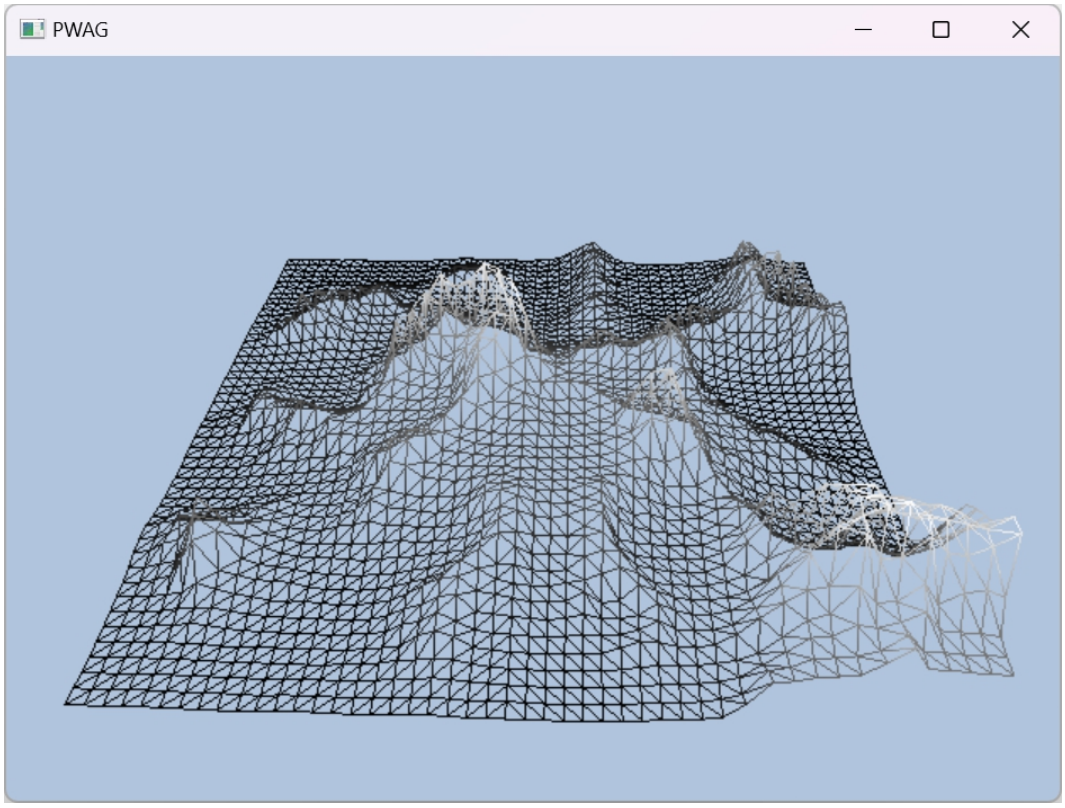
```
//Texture coordinate

float2 uv_v1 = lerp(quad[0].uv, quad[1].uv, uv.x);
float2 uv_v2 = lerp(quad[3].uv, quad[2].uv, uv.x);
float2 p_uv = lerp(uv_v1, uv_v2, uv.y);
```

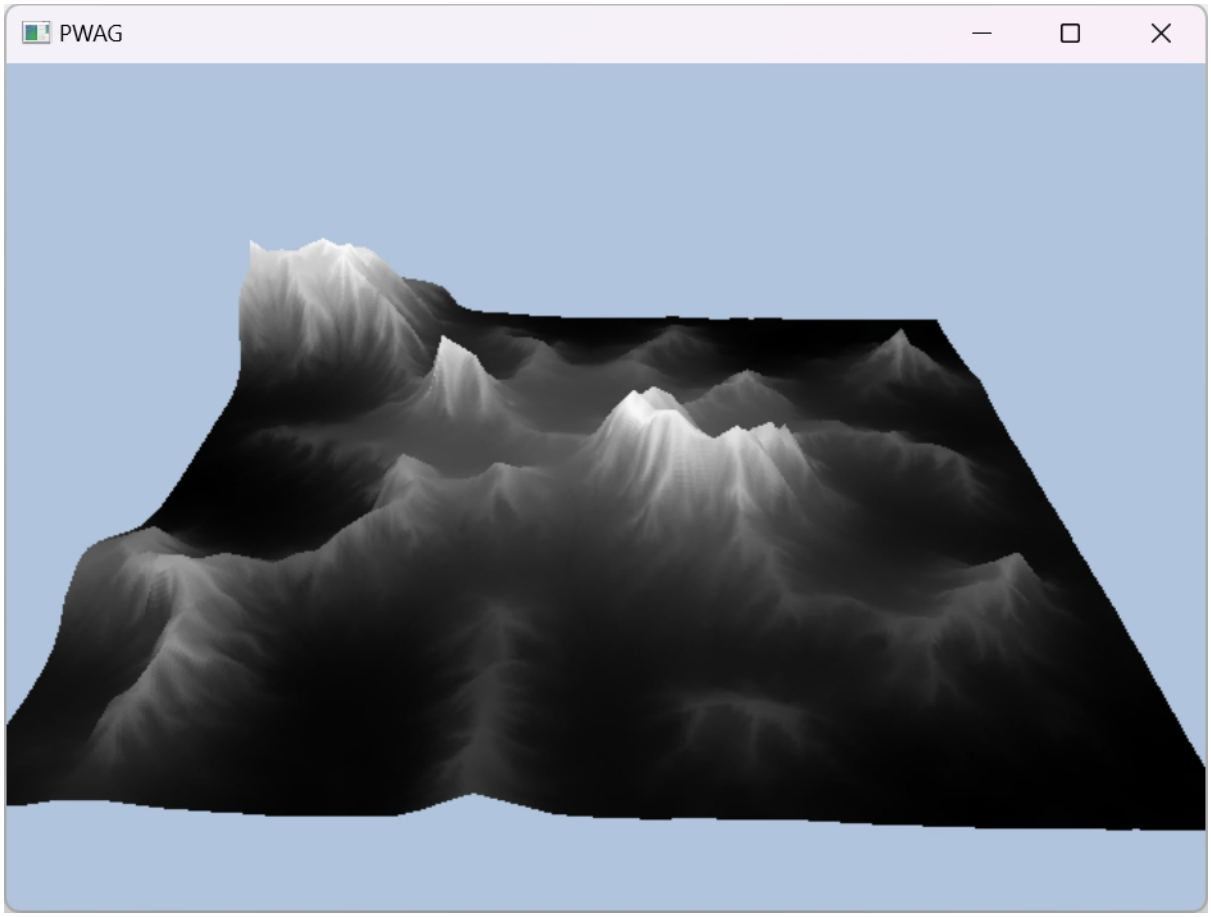
Jeżeli poprawnie wykonałeś to zadanie to powinieneś zobaczyć siatkę modelu z naniesioną na niego teksturą.



3 Mapowanie przemieszczeń (ang. Displacement mapping)

Wczytana przez ciebie w poprzednim zadaniu tekstura jest tak zwaną mapą wysokości lub mapą przemieszczeń. Tego typu tekstury nie zawierają informacji o kolorze rysowanej bryły lecz o odkształceniach w jej geometrii. Paleta tego typu tekstur jest ograniczona zazwyczaj do odcieni szarości, o stopniu modyfikacji decyduje kolor, czarny – brak a biały kolor to maksymalne przemieszczenie. Podobny efekt zaimplementujesz tym zadaniu, finalny efekt możesz zobaczyć na zamieszczonych poniżej obrazkach. Zmodyfikuj współrzędną Y (kierunek do góry) generowanych wierzchołków w zależności od koloru wierzchołka. Uwaga! Stosowana już przez ciebie wcześniej funkcja `Sample` działa jedynie w programie cieniującym piksele, w pozostałych programach do próbkowania tekstury musisz zastosować funkcję `SampleLevel`. Zmień tryb renderowania obiektu z siatki (wireframe) na pełny (solid)


PWAG


PWAG