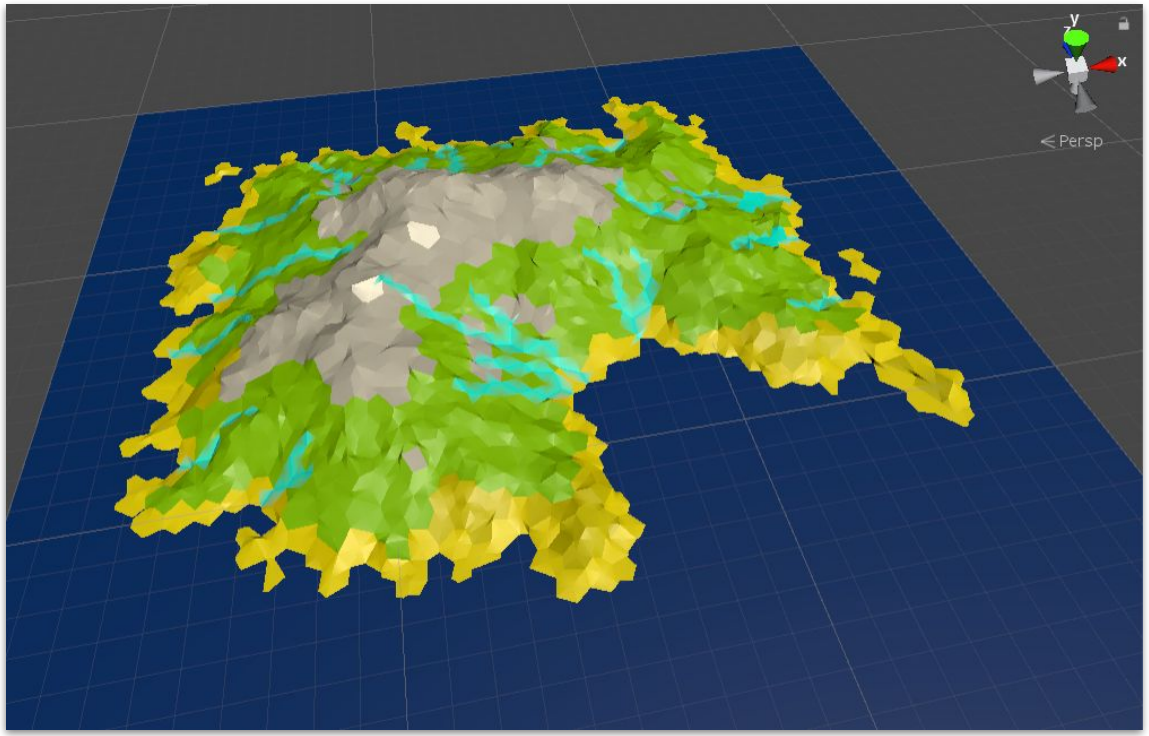
Proceduralne generowanie terenu

Grafika Komputerowa

Wstęp

Wstęp

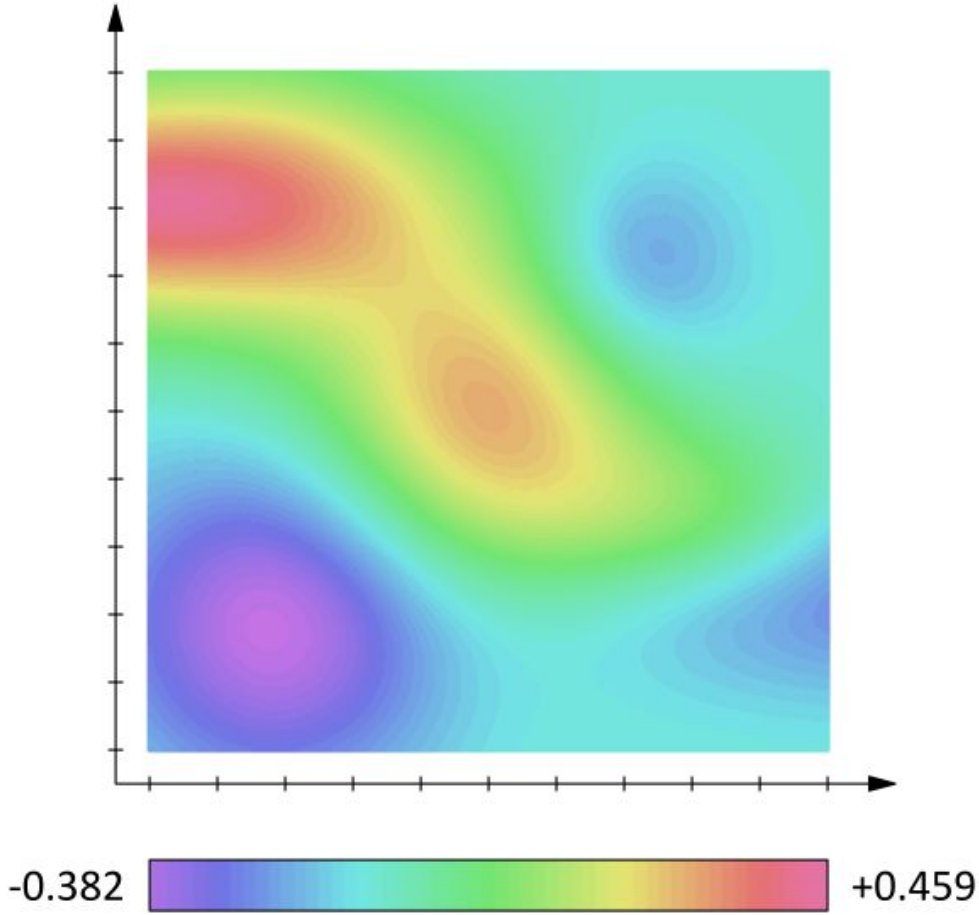
- kiedyś tereny generowane przez CPU a GPU renderowało
- GPU lepiej się do tego nadaje
- Shadery dają większe możliwości niż CPU



Funkcja gęstości

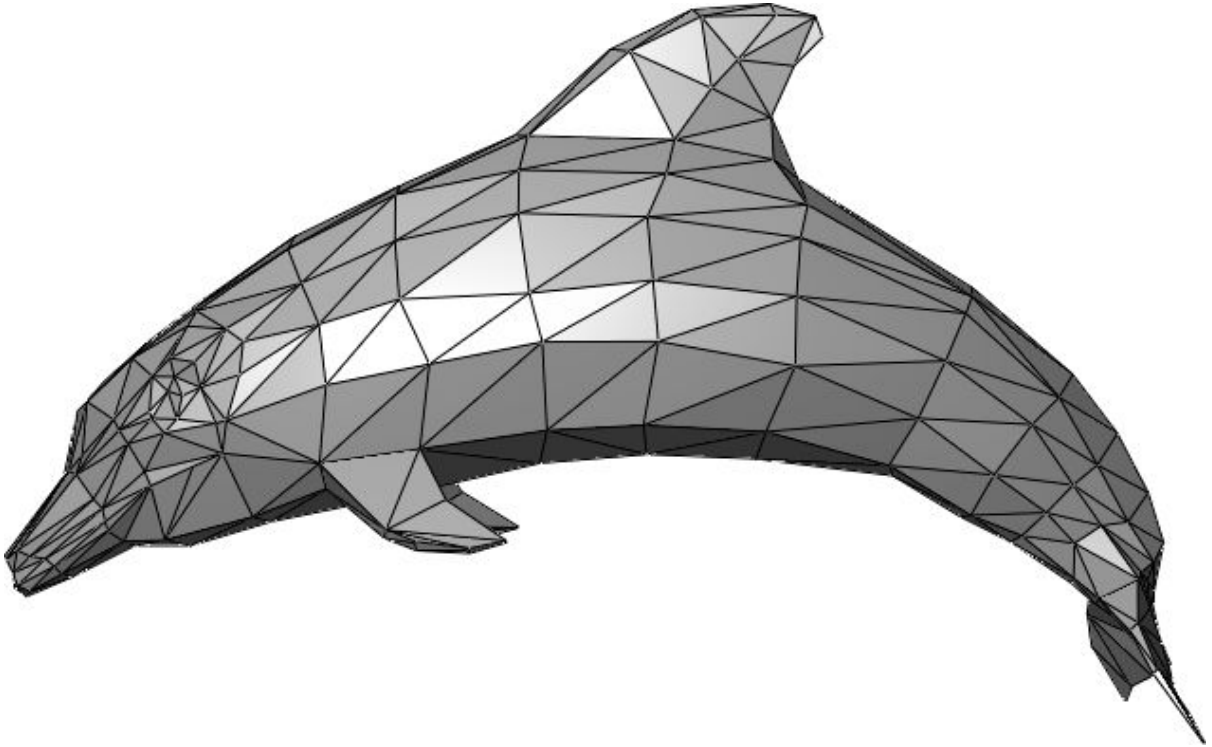
Funkcja gęstości

- wejściem jest pole skalarne
- dla każdego punktu w przestrzeni przypisana jest wartość skalarna (float)



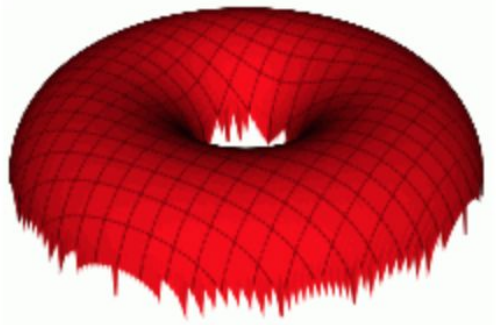
Funkcja gęstości

- całkowicie określa powierzchnię terenu
- wartości pozytywne to punkt wewnątrz terenu
- wartości negatywne to punkt poza terenem
- wartość 0 to powierzchnia



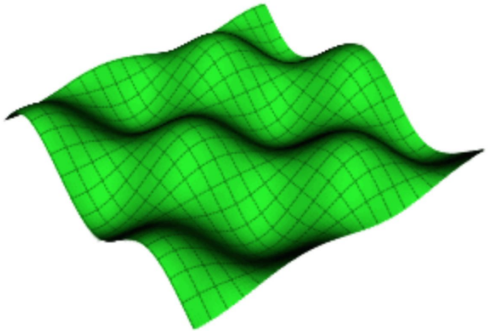
Torus

(0.4^2-(0.6-(x^2+y^2)^0.5)^2)^0.5



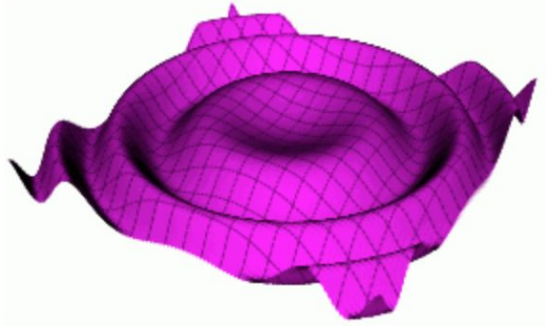
Bumps

sin(5x)*cos(5y)/5



Ripple

sin(10(x^2+y^2))/10



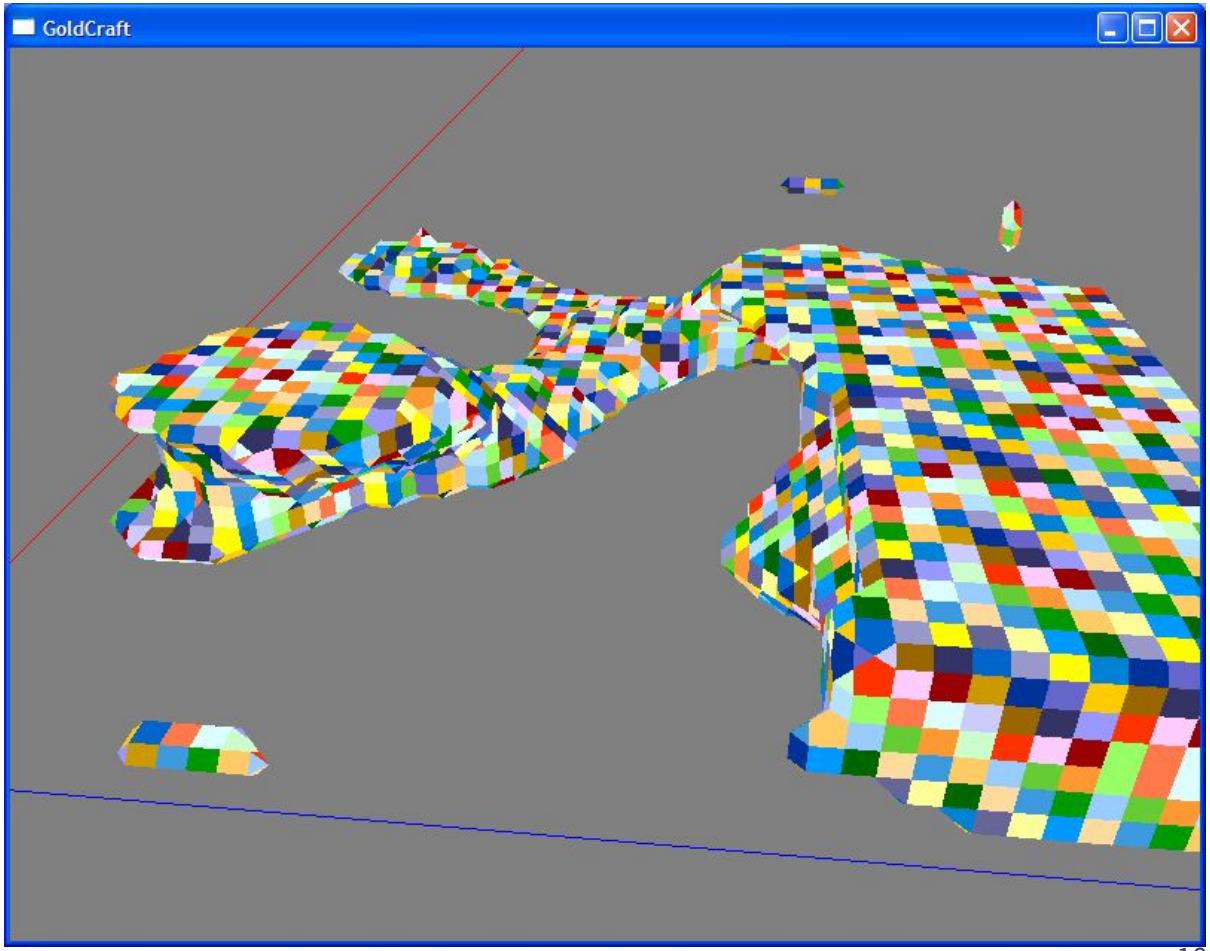
Przykładowe funkcje

Algorytm maszerujących sześcianów

Idea algorytmu

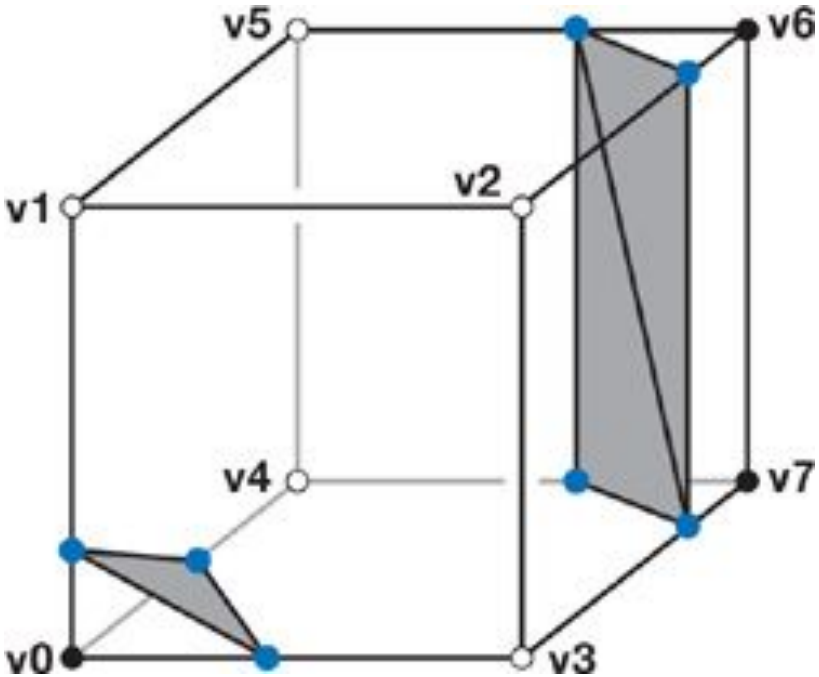
Idea algorytmu

- przestrzeń dzielona na sześciany (voxele)
- w ramach voxeli tworzymy poligony



Idea algorytmu

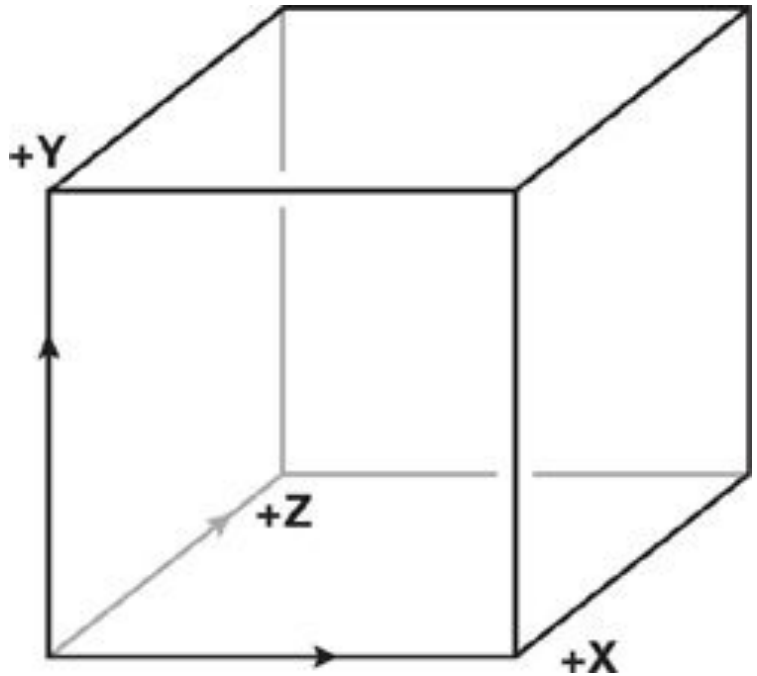
- wejściem jest wartość funkcji gęstości dla każdego wierzchołka voxela
- wyjściem 0 do 5 poligonów



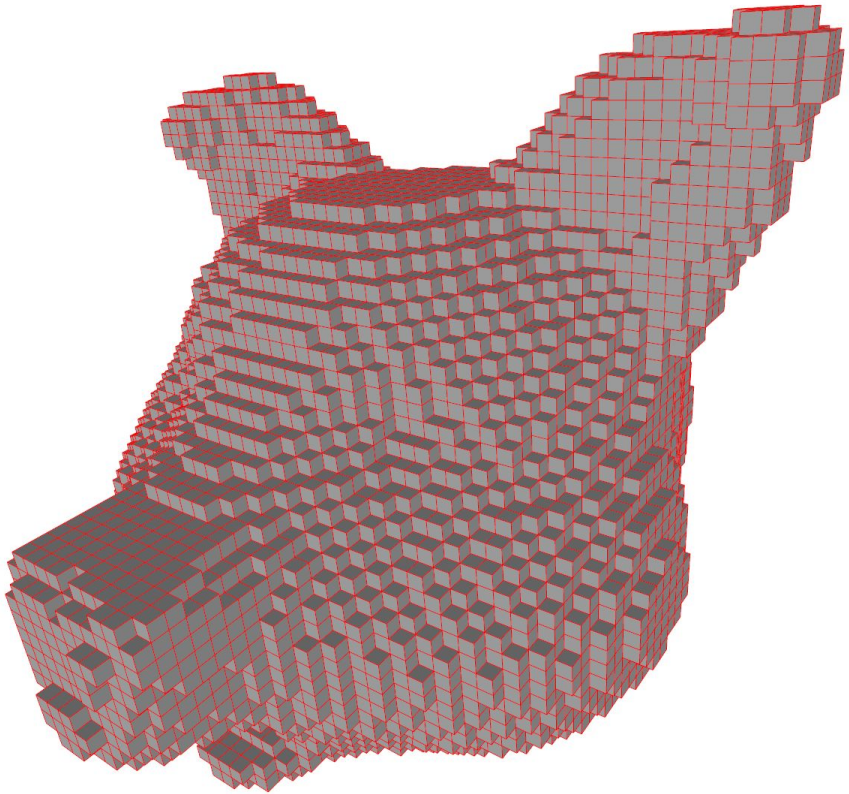
Idea algorytmu

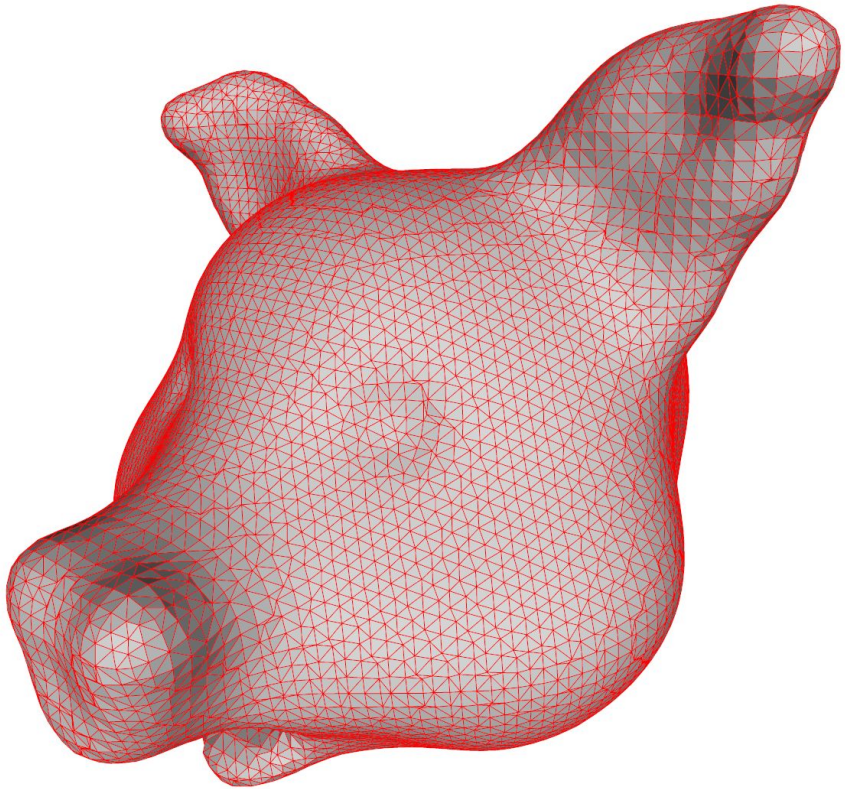
Jeśli wszystkie 8 wierzchołków ma:

- ten sam znak to nie generujemy poligonów
- inny znak to generujemy od 1 do 5 poligonów



Generowanie poligonów

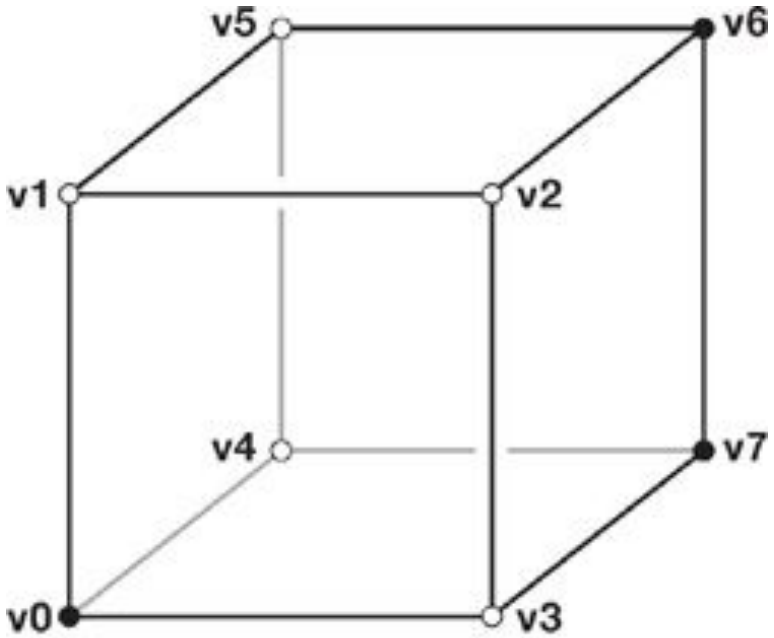




Teren przed i po poligonizacji

Generowanie poligonów

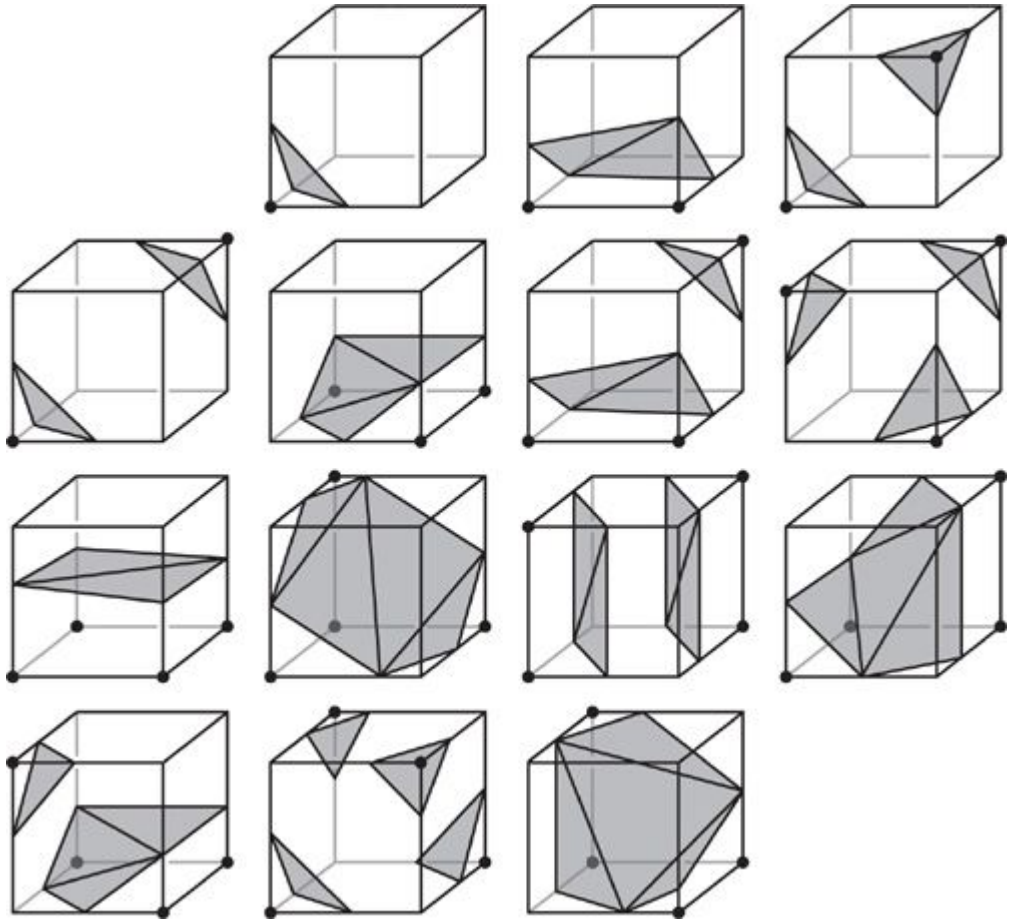
- mapujemy wartości f. gęstości dla wierzchołków na 1 (pozytywne) i 0 (negatywne)
- konkatenujemy uzyskane wartości w jedną 8-bitową liczbę binarną (bitwise OR)
- otrzymaną wartość interpretujemy jako liczbę dziesiętną (przypadek)

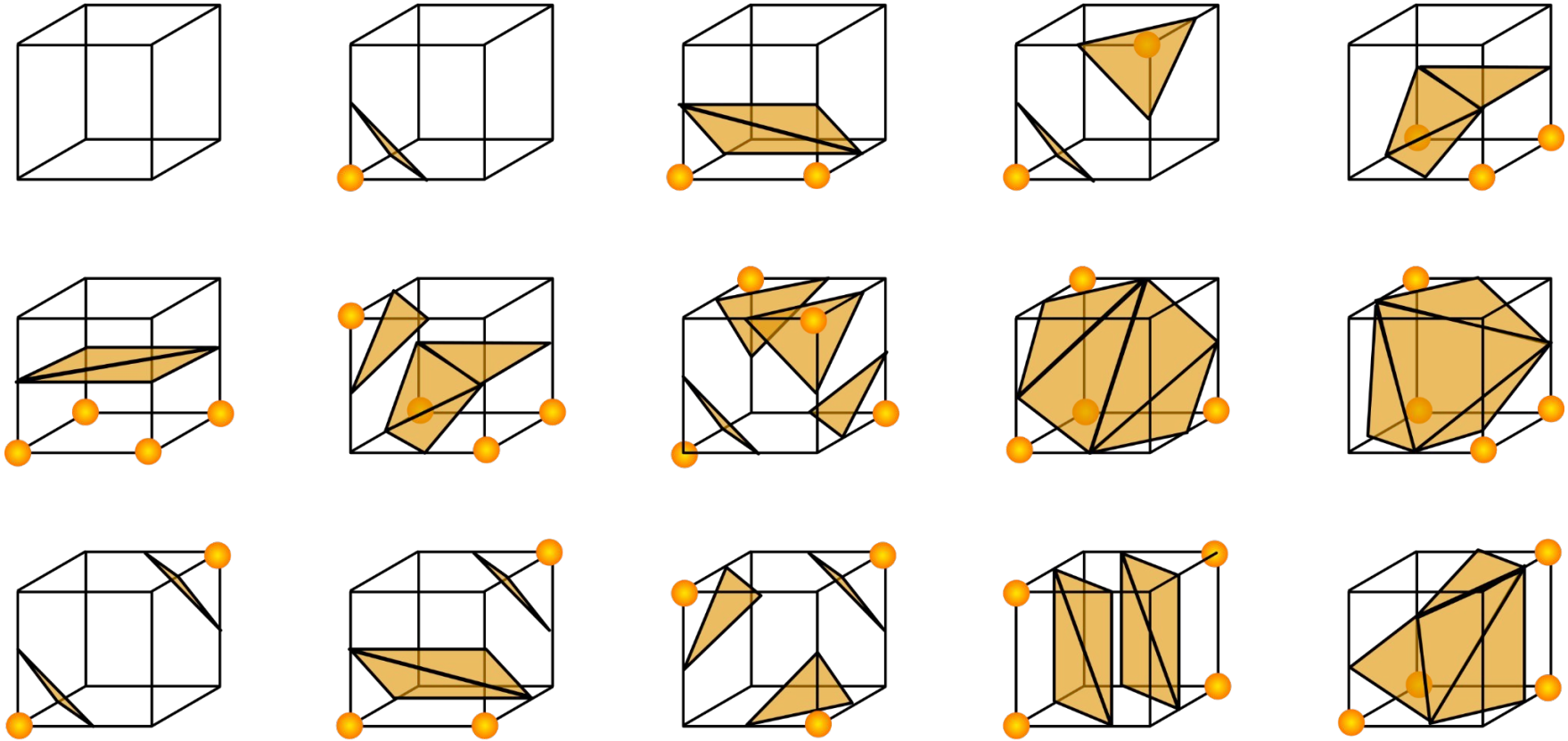


Case = v7|v6|v5|v4|v3|v2|v1|v0
= 11000001
= 193

Generowanie poligonów

- jeśli przypadek to 0 lub 255 to nie generujemy poligonów
- przypadki służą nam do indeksowania wzorów poligonów

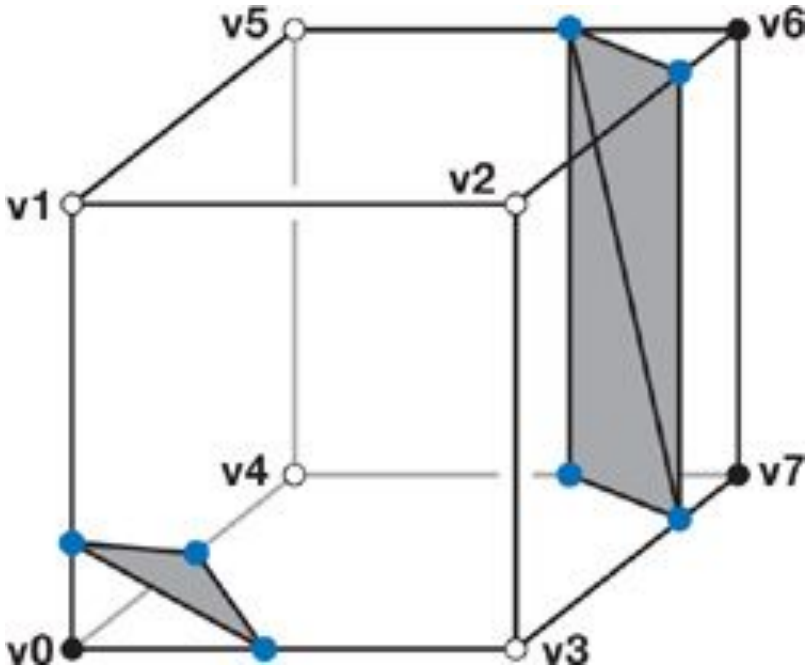


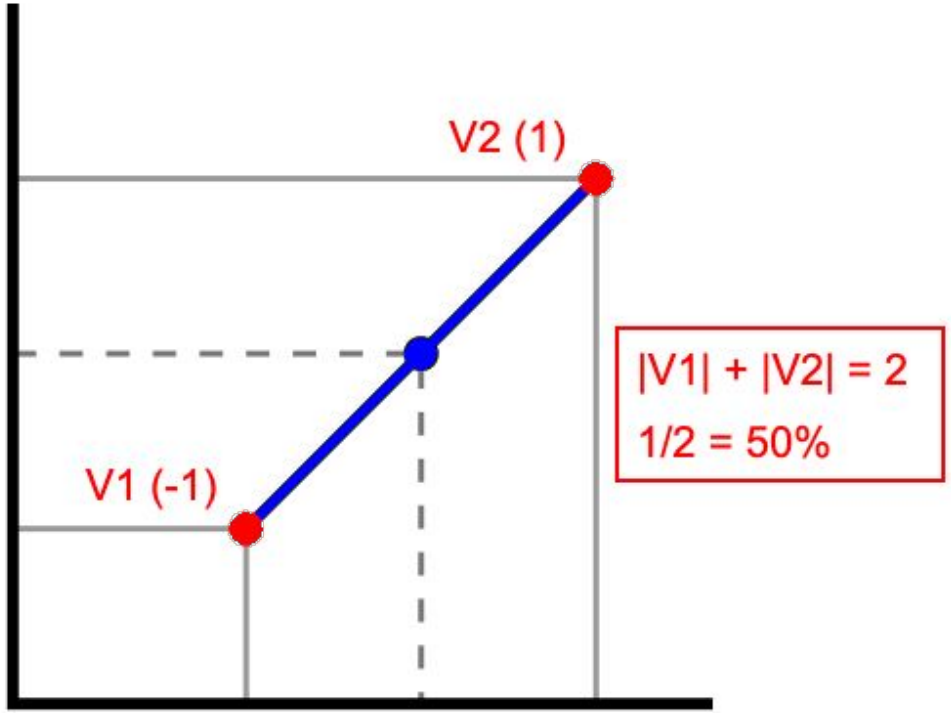


15 kanonicznych przypadków

Generowanie poligonów

- łączymy trzy punkty na 12 krawędziach voxela
- aby uzyskać pozycję punktu na krawędzi interpolujemy liniowo dwie wartości na wierzchołkach
- pozycja dla której wartość jest najbliższa 0 jest wybierana

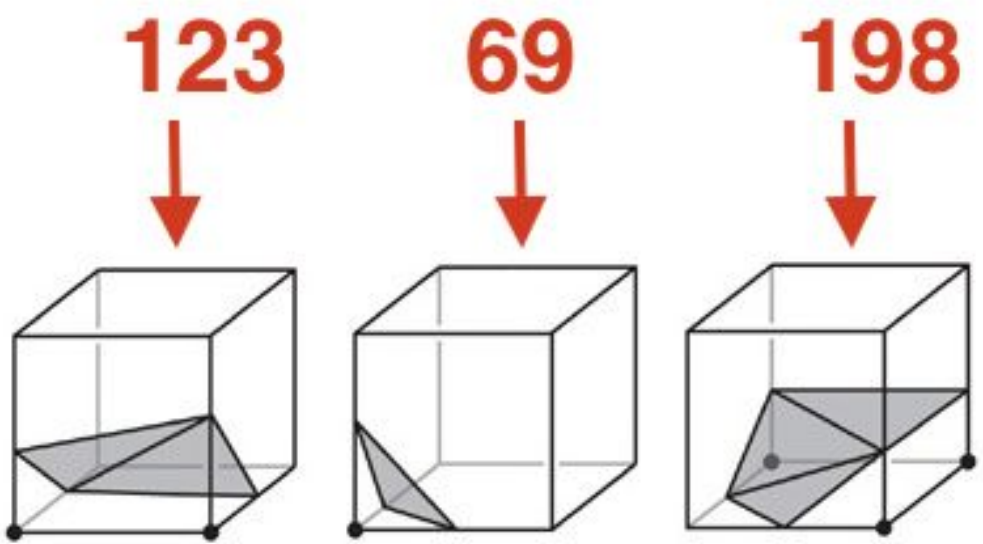




Interpolacja liniowa między dwoma punktami

Generowanie poligonów

- z lookup tables otrzymujemy informacje o położeniu wierzchołków poligonów
- pozycja wierzchołka poligonu na krawędzi voxela obliczana jest każdorazowo
- otrzymujemy listę trójkątów



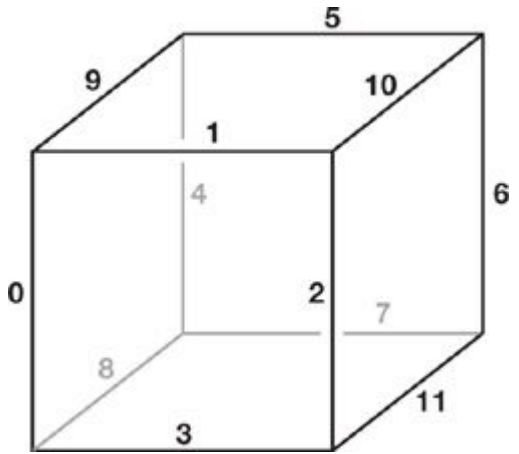
Lookup Tables

Lookup Tables

- używamy 2 lookup tables
- pierwsza tablica zwraca liczbę poligonów
- druga zwraca listę poligonów
- każdy poligon to 3 wartości int (numer krawędzi na której znajduje się wierzchołek)
- index to przypadek

```
int case_to_numpolys[256];
```

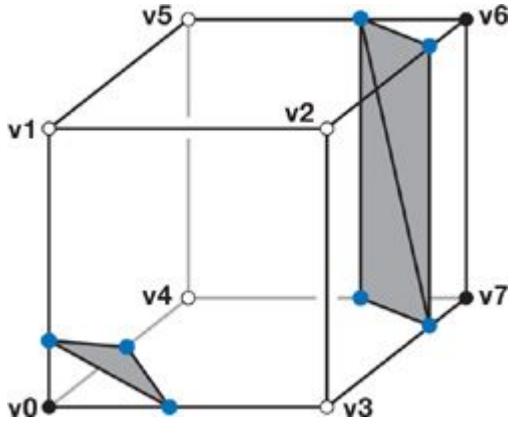
```
int3 edge_connect_list[256][5];
```



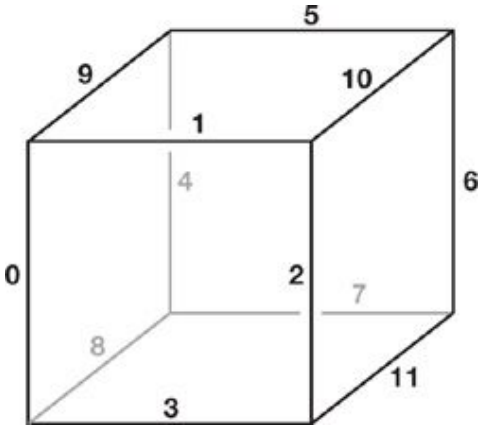
```
case_to_numpolys = [
  0: 0,
   ...
  193: 3,
   ...
]

edge_connect_list = [
  0: [
    [-1, -1, -1],
    [-1, -1, -1],
    [-1, -1, -1],
    [-1, -1, -1],
    [-1, -1, -1]
  ],
   ...
  193: [
    [0, 3, 8],
    [11, 7, 5],
    [11, 10, 5],
    [-1, -1, -1],
    [-1, -1, -1]
  ],
   ...
]
```

Struktura Lookup Tables



v7|v6|v5|v4|v3|v2|v1|v0 =
= 0b11000001 =
= 193



Koniec

Źródła

NVIDIA GPU Gems: Chapter 1. Generating Complex Procedural Terrains Using the GPU

Wikipedia: Marching Cubes

Gallery of Volume Graphics: Isosurface Polygonization