Zachodniopomorski
Uniwersytet
Technologiczny
w Szczecinie

http://bazyluk.net/dydaktyka

Bartosz Bazyluk

Teksturowanie

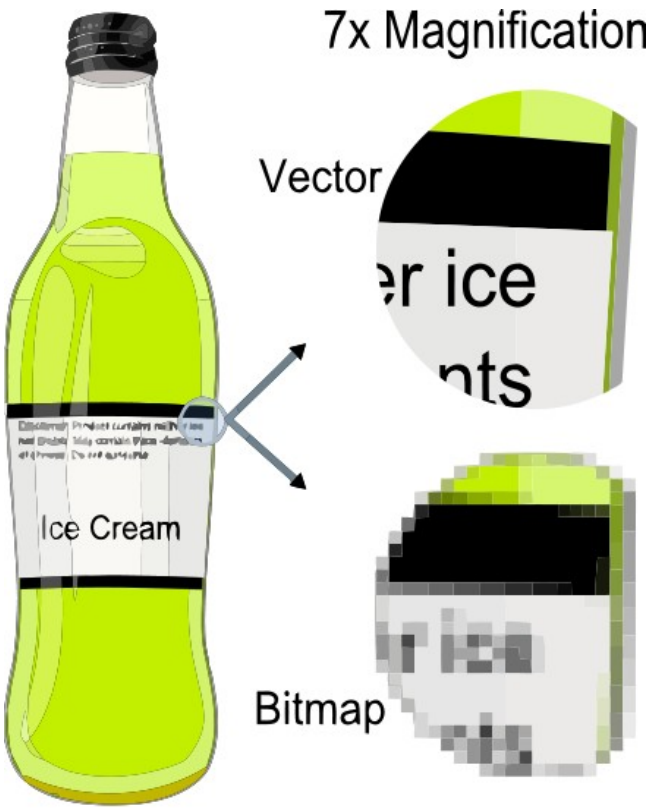
Mapy bitowe, pojęcie tekstury, potok, zastosowania.

Wydział
Informatyki

Grafika Komputerowa i Wizualizacja, Informatyka S1, II Rok

SPOSOBY REPREZENTACJI OBRAZU

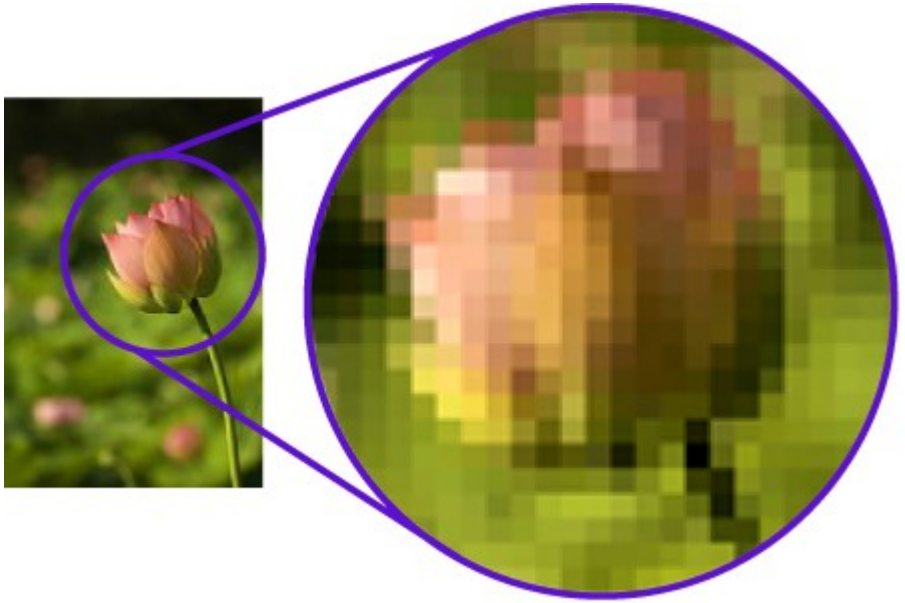
- Grafika **rastrowa**
  - Obraz jako **macierz** wartości pikseli
- Grafika **wektorowa**
  - Obraz jako **zbiór figur** geometrycznych



Źródło obrazu:
http://www.wikipedia.org

GRAFIKA RASTROWA

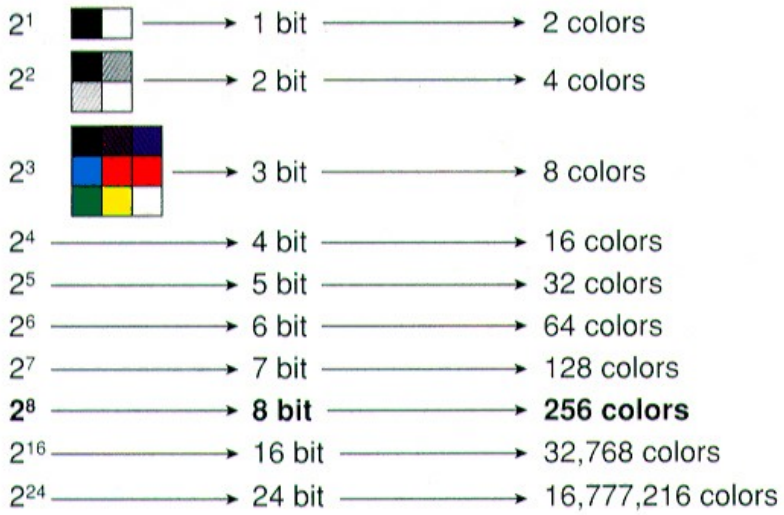
- Obraz to **macierz pikseli** – tzw. *mapa bitowa*, popularnie *bitmapa* (od ang. *bitmap*)
- Bitmapy są *najczęściej* prostokątne
- Piksel jest **najmniejszym fizycznym elementem obrazu**, który możemy kontrolować i adresować



Źródło obrazu:
http://www.vector-conversions.com

GRAFIKA RASTROWA

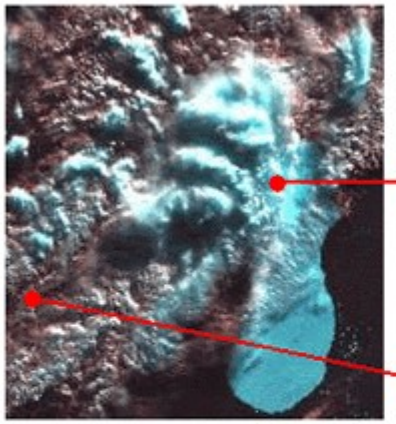
- Bitmapę charakteryzują:
  - **szerokość**, **wysokość** [px]
    - np. 1680×1050px
  - **liczba kanałów**
    - np. RGB (*Red/Green/Blue*), czyli 3 kanały
    - np. RGBA (RGB+Alpha, czyli przezroczystość)
  - **rozmiar piksela** [bpp]
    - np. 24bpp w przypadku RGB = 8 bitów na kanał
      - oznacza to, że każdy z kanałów może przyjąć 2^8 różnych wartości



Źródło obrazu:
http://www.foothill.edu

GRAFIKA RASTROWA

- Bitmapa może posiadać **kolory indeksowane**
  - czyli np. zamiast konkretnych wartości RGB pikseli, zapisane są **identyfikatory kolorów** (oszczędność pamięci)
  - powiązanie identyfikatorów z konkretnymi kolorami (*paleta*) może być dołączone do bitmapy jako słownik
  - identyfikatory mogą też dotyczyć np. domyślnej palety systemowej



8-bit raster image

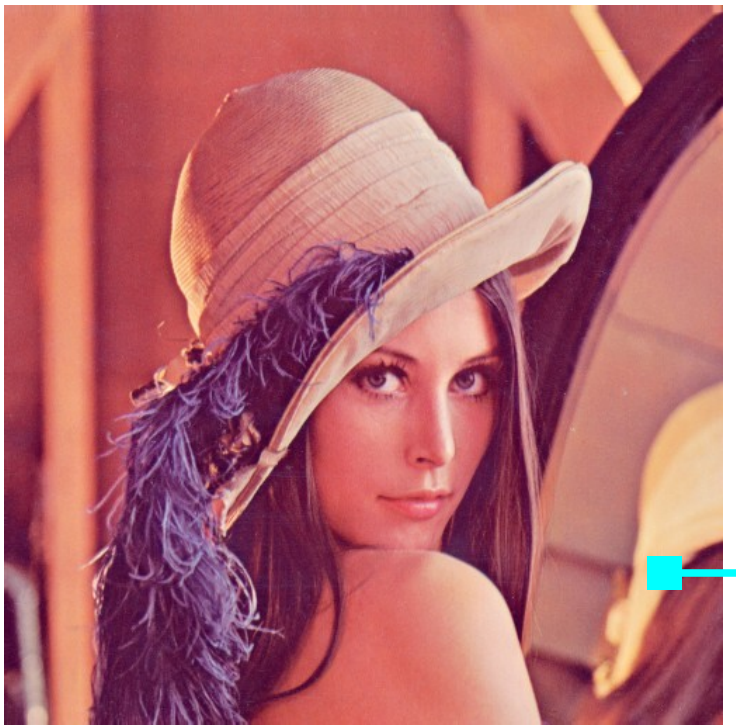
	RED	GREEN	BLUE	RGB
0	10	9	15	
1	28	130	124	
2	152	131	131	
3	109	65	69	
4	152	200	203	
5	64	196	203	
6	34	68	77	
7	96	135	141	
8	209	196	199	
9	36	100	118	
10	97	166	168	
11	61	25	30	
12	206	152	155	
13	153	227	233	
14	113	98	101	
15	153	166	168	
16	209	229	227	
17	90	227	236	
18	80	67	71	

Color look-up table (palette)

Źródło obrazu:
http://www.hdfgroup.org

GRAFIKA RASTROWA

- Ilość **pamięci** potrzebna do zapisania mapy bitowej:
  - szerokość × wysokość × rozmiar piksela
  - Przykład:



$(512 \times 512 \times 24) / 8 = 786.432B =$ **768kB**

ponieważ 1B = 8b

GRAFIKA RASTROWA

- Zapisując bitmapy w pamięci masowej w celu przechowania, często stosuje się **kompresję**
- Algorytmy kompresji można podzielić na dwie podstawowe grupy:
  - Kompresja **bezstratna**
    - Możliwe jest całkowite **odtworzenie pierwowzoru**
  - Kompresja **stratna**
    - W wyniku kompresji dochodzi do **utraty jakości**, np. szczegółów lub koloru
    - Powstają zniekształcenia, tzw. *artefakty*

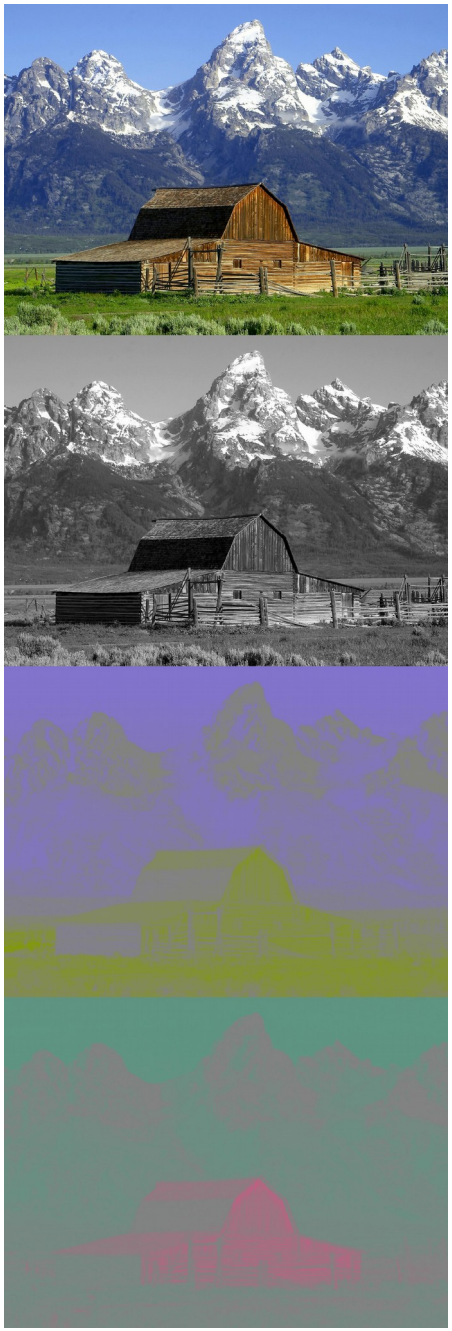
Źródło obrazu:
http://cscie12.dce.harvard.edu

GRAFIKA RASTROWA

- Popularne **formaty zapisu bitmap** i **kodeki**:
  - **BMP**
    - duża dowolność formatu danych
    - może wykorzystywać kompresję bezstratną *RLE*
  - **JPEG**
    - kompresja **stratna**
    - brak obsługi przezroczystości
    - 24bpp
  - **PNG**
    - kompresja **bezstratna**
  - **GIF**
    - kompresja **bezstratna**
    - zazwyczaj max. 8bpp
    - możliwość zapisu sekwencji obrazów jako animacja
  - **TIFF**
    - duża dowolność formatu danych i sposobów kompresji

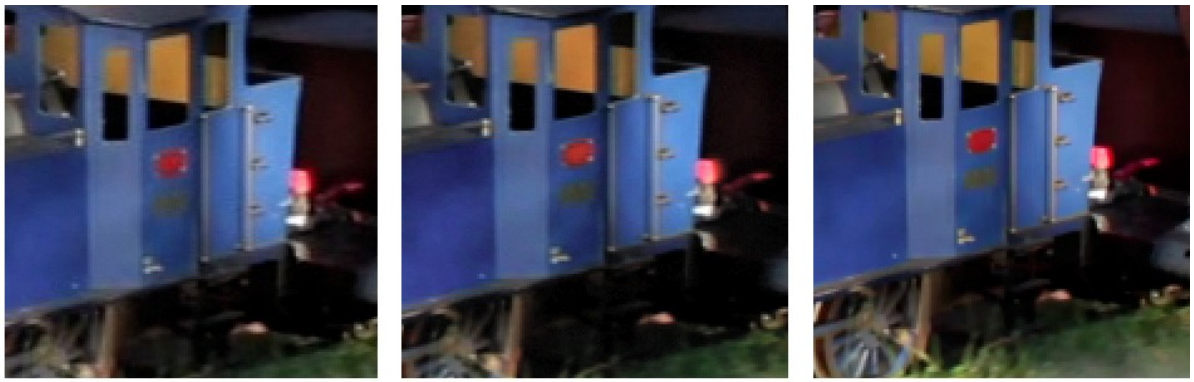
- **Kompresja *JPEG***
  - Standard *JPEG* opisuje działanie **kodeka**, a nie **format pliku**
    - **Kodek** jest algorytmem zamiany obrazu w strumień bajtów oraz zamiany strumienia bajtów w obraz
  - Najpopularniejsze **formaty**:
    - *JPEG/Exif*
    - *JPEG/JFIF*

- **Kompresja *JPEG* – krok 1/4**
  - Przejście z przestrzeni **RGB** do **Y'C_BC_R**
    - **Y'** – luma
      - Poziom jasności (nieliniowy!)
    - **C_B**, **C_R** – chroma
      - **C_B** – różnica koloru niebieskiego i Y'
      - **C_R** – różnica koloru czerwonego i Y'
  - Dzięki temu możliwe jest odrębne traktowanie lumy, a co za tym idzie kontrastu, od informacji o kolorze
    - Ludzki aparat widzenia jest bardziej czuły na zmiany w kontraście, niż w kolorze



Źródło obrazu:
http://wikipedia.org

- **Kompresja *JPEG* – krok 2/4**
  - **Redukcja rozdzielczości** informacji o **kolorze** (ang. *chroma subsampling*)
    - Skoro człowiek trudniej zauważa różnicę w zmianach koloru niż kontrastu, to *po co marnować pamięć* na przechowywanie informacji o kolorze w pełnej rozdzielczości?
    - 4:4:4 – bez redukcji
    - 4:2:2 – 2-krotna redukcja w poziomie
    - 4:2:0 – 2-krotna redukcja w pionie i poziomie
    - Wyjaśnienie zapisu: http://en.wikipedia.org/wiki/Chroma_subsampling#Sampling_systems_and_ratios
  - Krok **opcjonalny**



4:2:0 4:2:2 4:4:4

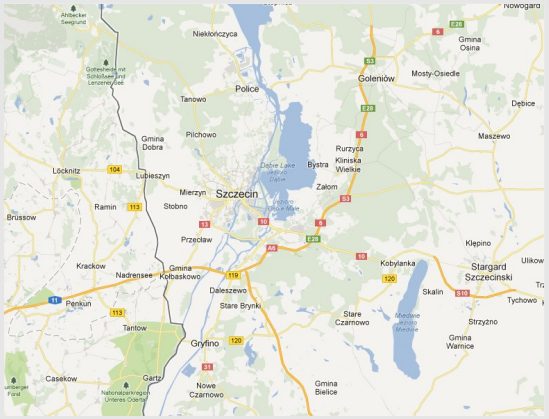
Źródło obrazu:
http://wikipedia.org

Charakterystyczne, **blokowe artefakty** są wynikiem osobnego przetwarzania bloków 8x8 pikseli.

- **Kompresja *JPEG* – krok 3/4**
  - Obraz dzielony jest na **bloki 8x8 pikseli**
    - Stąd bierze się często widoczny "blokowy" artefakt w obrazach JPEG
  - *Każdy* z kanałów $Y'C_BC_R$ w *każdym* z bloków jest osobno poddawany działaniu **dyskretnej transformacji kosinusowej** (ang. *discrete cosine transform, DCT*)
    - Przejście do dziedziny częstotliwości
    - Przed zastosowaniem *DCT* wartości [0;255] przesuwane są do zakresu [-128;127]
    - Tymczasowo każdy komponent zajmuje więcej niż 8 bitów
  - Wartości po *DCT* poddawane są **kwantyzacji**
    - Jedyny (poza subsamplingiem) **stratny** element kodeka, tego etapu dotyczy ustawienie **stopnia kompresji**
    - Pozwala pominąć zmiany **wysokiej częstotliwości**
- **Kompresja *JPEG* – krok 4/4**
  - Dodatkowa, **bezstratna kompresja Huffmana**
  - Może być poprzedzona bezstratnym **kodowaniem entropii**

GRAFIKA RASTROWA

- Różne metody kompresji nadają się do **różnego rodzaju obrazów**
- Należy zawsze wziąć pod uwagę charakter kompresowanego obrazu by uzyskać **optymalny stosunek jakości do rozmiaru pliku**
- Większość metod kompresji pozwala **dobrać jej parametry**
- Przykłady:



Brak kompresji: **1386kB**
PNG: **96kB**, jakość *idealna*
JPG: **99kB**, jakość *b.zła*
JPG: **285kB**, jakość *db.*

Brak kompresji: **1875kB**
PNG: **1268kB**, jakość *idealna*
JPG: **480kB**, jakość *b.db.*
JPG: **246kB**, jakość *akceptowalna*

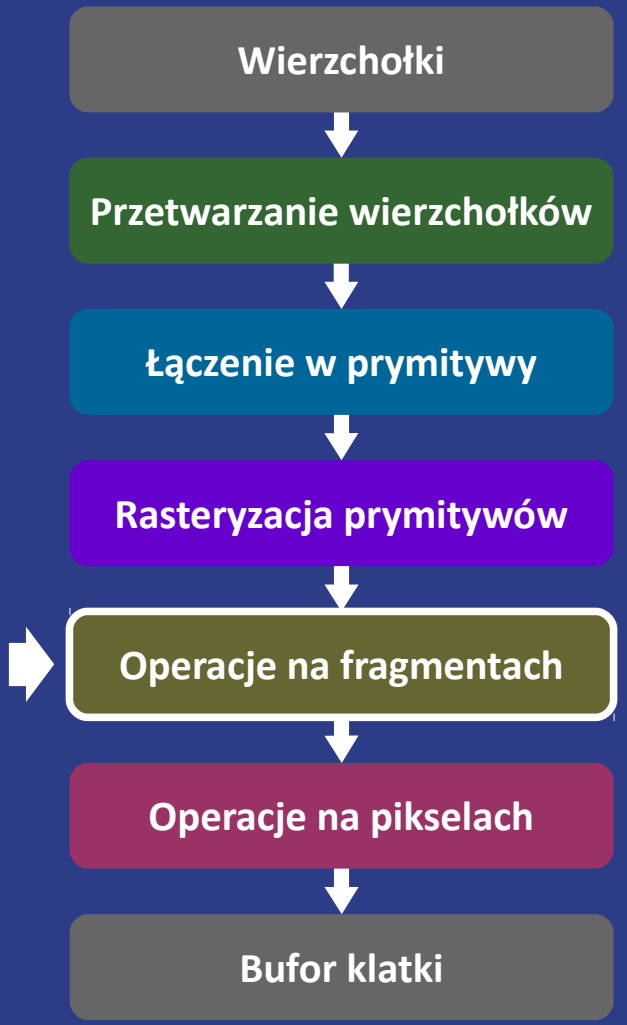
RENDEROWANIE DETALI

Problem:

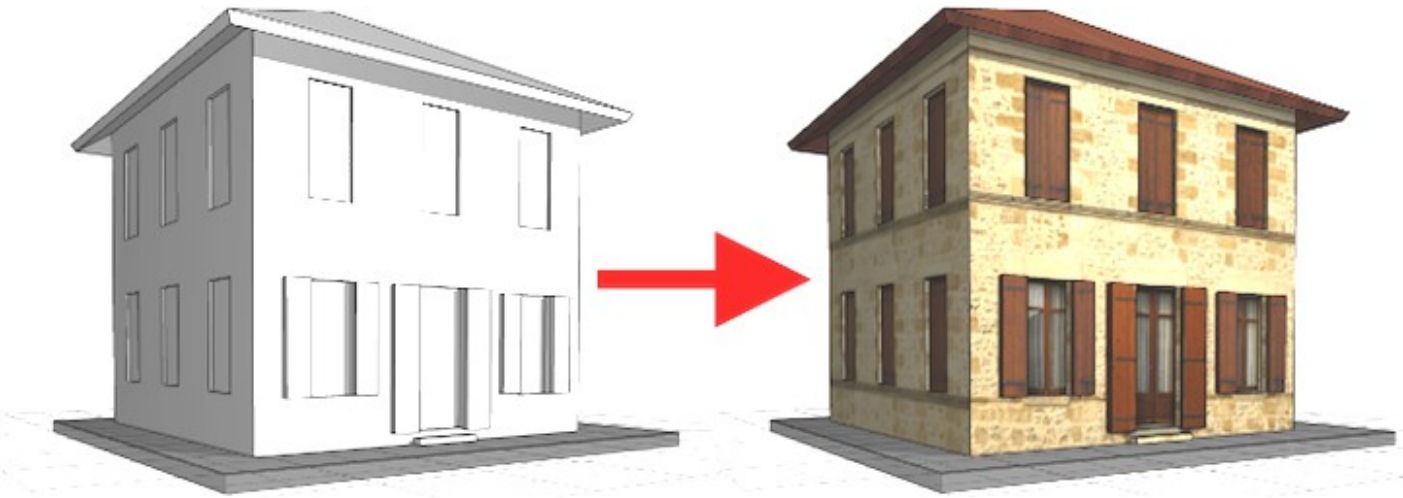
W jaki sposób korzystając z dotychczas omówionych metod opisu cech materiału wyrenderować powierzchnię lub element o bardzo dużej **szczegółowości**?

Źródło obrazu:
http://www.texturex.com

TEKSTUROWANIE



- Technika **teksturowania** pozwala na **modyfikację** cech powierzchni bryły **pomiędzy jej wierzchołkami**
  - Teksturowanie jest z założenia **operacją wykonywaną na fragmentach**
    - Możliwe jest wykorzystanie tekstur także **na innych etapach potoku renderowania**, np. podczas przetwarzania wierzchołków
  - Najczęściej teksturowanie kojarzy się z **modulacją koloru na podstawie koloru tekstury**, jednak jest to tylko **jedno z wielu zastosowań** tej techniki



Oświetlona bryła **Modulacja koloru z użyciem tekstury**

Źródło obrazu:
http://helpcenter.graphisoft.com/

TEKSTUROWANIE

- Na **teksturę** składa się **jeden lub więcej obrazów**
  - Jest więc ona swojego rodzaju **kontenerem na obrazy**, posiadającym pewne szczególne cechy
  - **Nie należy utożsamiać tekstury z obrazem**
- Teksturę określają parametry:
  - **Typ** tekstury
    - Określa ułożenie obrazów w obrębie tekstury
  - **Rozmiar** tekstury
    - Rozdzielczość obrazów
  - **Format** obrazu
    - Ułożenie danych w obrębie każdego obrazu

Źródło obrazu:
http://filterforge.com/

TEKSTUROWANIE

Najczęściej wykorzystywany
i zarazem **najbardziej uniwersalny**
typ tekstury w OpenGL
to *GL_TEXTURE_2D*.

Źródło obrazu:
http://filterforge.com/

- **Typy tekstur**
  - **Wymiarowość**:
    - **1D** – obrazy posiadają tylko szerokość
    - **2D** – obrazy posiadają szerokość i wysokość
    - **3D** – obrazy posiadają szerokość, wysokość i głębokość
  - **Zorganizowanie** obrazów:
    - Np. **tablica** (ang. *array*), **mapa sześcienna** (ang. *cube map*)
  - Dostępne w **OpenGL** typy:
    - *GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D*
    - *GL_TEXTURE_RECTANGLE*
    - *GL_TEXTURE_BUFFER*
    - *GL_TEXTURE_CUBE_MAP*
    - *GL_TEXTURE_1D_ARRAY, GL_TEXTURE_2D_ARRAY*
    - *GL_TEXTURE_CUBE_MAP_ARRAY*
    - *GL_TEXTURE_2D_MULTISAMPLE,*
      GL_TEXTURE_2D_MULTISAMPLE_ARRAY

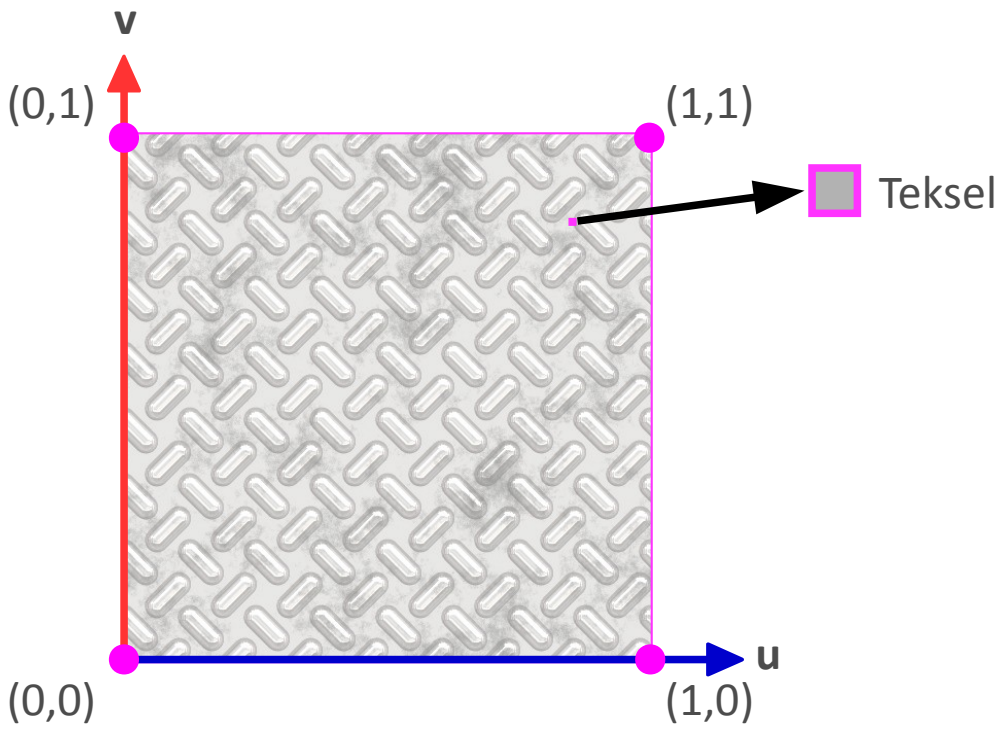
TEKSTUROWANIE

Pojedynczy element macierzy używanej jako tekstury nazywa się **tekselem**.

Dla tekstur 1D jest zwyczajowo jedna **współrzędna *U***, dla tekstur 3D są **współrzędne *UVW***.

W **DirectX** współrzędna **pionowa jest odwrócona**: początek przestrzeni tekstury znajduje się w jej **lewym górnym narożniku**.

- W uogólnieniu **tekstura** jest **macierzą wartości** *(niekoniecznie koloru!)*
- Wykorzystuje się tekstury **jedno- (1D), dwu- (2D) i trójwymiarowe (3D)**
  - Najpowszechniejsze są **tekstury 2D**
  - Niezależnie od liczby wierszy i kolumn macierzy, dwuwymiarową teksturę opisuje się za pomocą znormalizowanych **współrzędnych UV**:



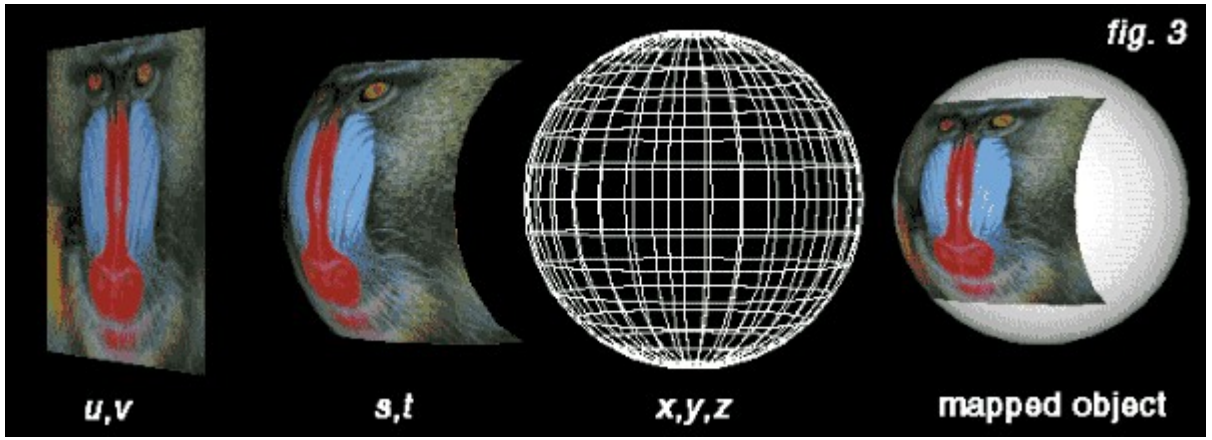
Źródło obrazu:
http://filterforge.com/

TEKSTUROWANIE

Zdecydowanie częściej słyszy się o współrzędnych *UV*, niż o współrzędnych *ST*.

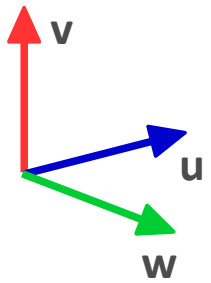
W środowisku ludzi zajmujących się grafiką komputerową zdania odnośnie słuszności jednej bądź drugiej konwencji i ich znaczeń są bardzo podzielone.

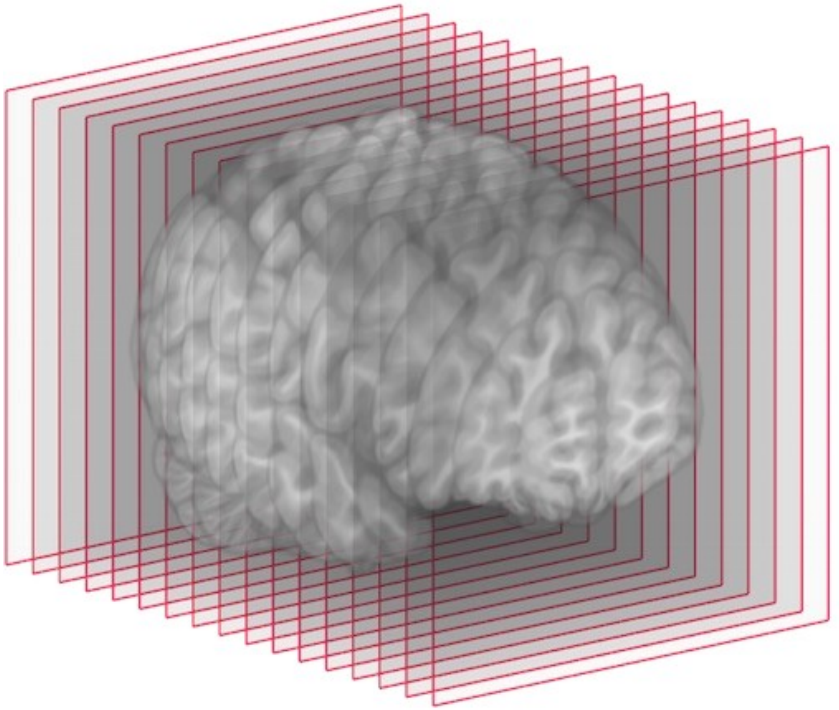
- **Adresowanie tekstury** odbywa się z użyciem współrzędnych w jej przestrzeni, tj. *dwuwymiarowych* jeśli obraz jest *dwuwymiarowy*
  - Współrzędne w przestrzeni tekstury nazywa się zwyczajowo *U*, ***UV***, *UVW* zależnie od wymiarowości
  - Współrzędne powierzchni bryły na którą nakłada się teksturę zwyczajowo nazywa się *S*, ***ST***, *STR* zależnie od wymiarowości
  - W *OpenGL* przyjęto tę drugą konwencję jako dominującą aby odróżnić współrzędne *UV* tekstury od współrzędnych ewaluatora, które też nazywane są *UV*



Źródło obrazu:
Cindy Reed. *Texture Mapping in VRML*.

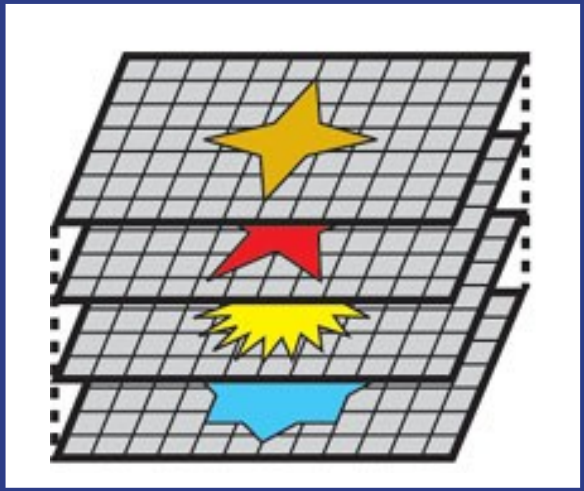
- Tekstura **trójwymiarowa** (3D)
  - Jest złożona ze zbioru **plastrów** (**warstw**, ang. *slices*), z których **każdy stanowi macierz 2D** o tych samych wymiarach
  - Dzięki interpolacji pomiędzy warstwami możliwe jest opisanie ciągłego, **wolumetrycznego zbioru danych**
  - Wykorzystuje się np. do **opisu danych medycznych** pozyskanych metodami takimi jak tomografia





Źródło obrazu:
http://github.com/neurolabusc/vx

TEKSTUROWANIE



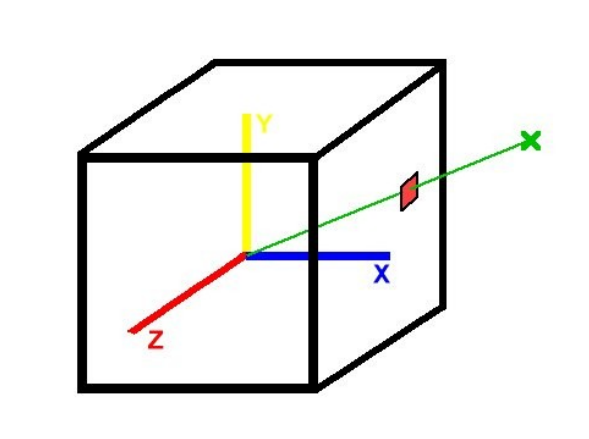
Podobnie jak w przypadku tekstury 3D, adresowanie odbywa się za pomocą **trzech współrzędnych**.

Trzecia współrzędna jest **identyfikatorem warstwy** i jest w tym przypadku **liczbą całkowitą**.

Źródła obrazów:
NVIDIA
http://www.paulsprojects.net/

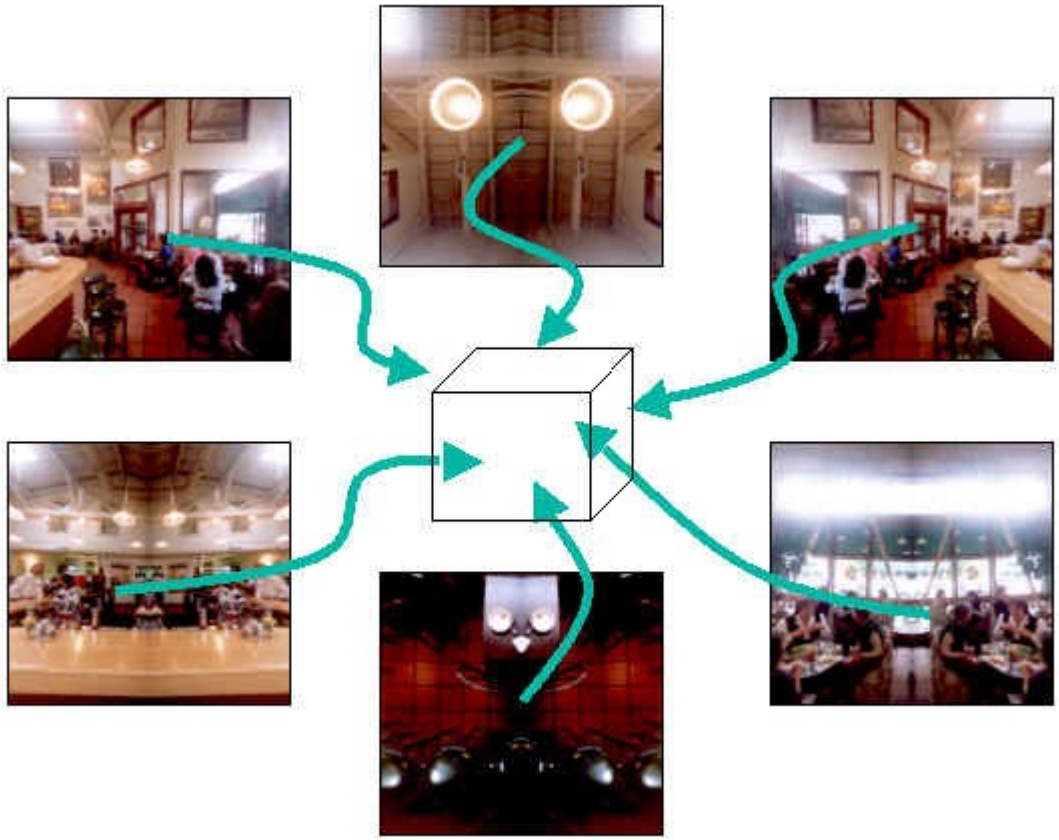
- Tekstura będąca **tablicą** (ang. *array texture*)
  - Podobnie jak w przypadku tekstury 3D, **tekstura składa się z plastrów** (ang. *slices*), ale tutaj każdy z nich jest **niezależnym obrazem – nie ma interpolacji** pomiędzy nimi
    - Obrazy muszą być identycznych wymiarów
  - **Przełączanie się** pomiędzy teksturami jest **kosztowne**
    - Tablica pozwala na **jednoczesne załadowanie kilku obrazów** jako jednej tekstury – jest to duża oszczędność czasu
  - Lepsza alternatywa dla **atlasów tekstur**
    - Nie występuje problem ***bleedingu*** (mieszania wartości przy krawędziach) podczas filtracji
    - Możliwy łatwy ***wrapping***
    - Rozwiązanie problemów z **mipmappingiem**
  - **Nieużyteczna** w nieprogramowalnym potoku renderowania

TEKSTUROWANIE



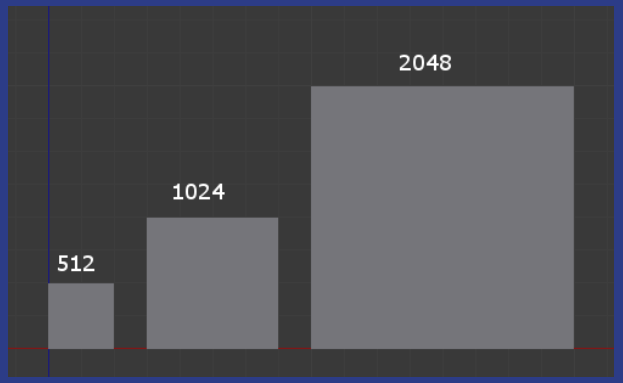
Podanie trójwymiarowego **wektora kierunku** prowadzi do znalezienia odpowiadającego mu teksela w sześciennej mapie.

- **Mapa sześcienna** (ang. *cube map*)
  - Struktura **sześciu obrazów** stanowiących ściany sześcianu
  - Możliwość **adresowania** za pomocą trójwymiarowego **kierunku obserwacji**
  - Przydatna np. do zbudowania ***skyboxa*** lub do techniki **mapowania środowiska**



Źródła obrazów:
NVIDIA
http://www.paulsprojects.net/

TEKSTUROWANIE



Wymiar tekstury będący każdą kolejną **potęgą liczby 2** oznacza **cztery razy więcej tekseli**.

Szybszy odczyt danych z tekstury o wymiarach będących potęgami liczby 2 sprawia, że **czasem warto poświęcić pamięć i zwiększyć jej rozmiar**.

Źródło obrazu:
StackExchange, Gamedev, psal

- Liczba wierszy i kolumn obrazu w teksturze to jej **rozdzielczość**
  - Dwuwymiarowe tekstury są często **kwadratowe**, choć **nie jest to konieczne** (*OpenGL 1.1+*)
  - Najczęściej tekstury mają wymiary będące **potęgami liczby 2**, choć nie jest to konieczne (*OpenGL 1.4+*)
    - Np. *128×128*, *512×512*, *1024×512*, *2048×256*...
    - Pozwala to na korzystanie z **uproszczonych obliczeń** podczas np. mnożenia, dzielenia współrzędnych
    - Dzięki temu przetwarzanie tekstur o takich wymiarach **może być szybsze**
  - Można oczekiwać, że współczesne karty graficzne są w stanie obsłużyć tekstury **nawet o rozdzielczości *16384×16384***, choć w praktyce rzadko kiedy używa się większych niż *4096×4096*
    - **Najczęściej spotykane** rozmiary są w zakresie **od *128×128* do *2048×2048***
    - Możliwości zależą od ilości dostępnej **pamięci karty graficznej** jak i sprzętowych rozwiązań dotyczących jej adresowania

The Witcher 3 Texture Quality Comparison

Ultra

High

Medium

Low

Źródło obrazu:
http://gamersnexus.net

TEKSTUROWANIE

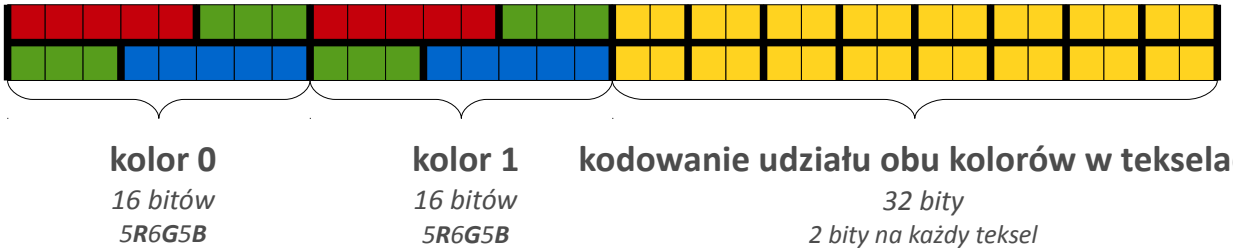
- **Format tekstury** określa sposób zorganizowania danych w obrębie obrazu
  - **Komponenty** (R, G, B, A, depth, stencil, ...)
  - **Precyzja** (liczba bitów na kanał)
  - **Typ danych** (signed/unsigned, integer, float, ...)
  - **Przykładowe formaty** oferowane przez *OpenGL*:
    - *GL_RGB8*
    - *GL_R3_G3_B2*
    - *GL_RGB10_A2*
    - *GL_SRGB8_ALPHA8*
    - *GL_DEPTH_COMPONENT24*
    - itd...
- Możliwa jest wspierana sprzętowo **kompresja danych** przechowywanych w teksturze
  - Metody charakterystyczne dla producentów kart graficznych
  - Uniwersalna **metoda *S3*** (*S3TC*), nazywana czasem *DXT*

TEKSTUROWANIE

Użycie kompresji *S3TC* w *OpenGL* nie jest objęte standardem, wymaga posłużenia się uniwersalnym **rozszerzeniem *EXT_texture_compression_s3tc***.

Możliwe jest przekazanie do *OpenGL* tekstury w postaci nieskompresowanej i zlecenie sterownikowi kompresji, jak i przekazanie postaci już skompresowanej. W drugim przypadku operacja jest **znacznie szybsza**.

- **Metoda kompresji tekstur S3** (*S3TC*), nazywana czasem metodą *DXT*
  - Kompresja **stratna**
  - **Stopień kompresji** (ang. *compression ratio*) jest **stały**
  - Operuje na **blokach** *4×4*
  - Pozwala na **wybiórczą dekompresję** tekstury
  - Odmiana *DXT1*:
    - Tylko **RGB**, **bez przezroczystości**
    - Stopień kompresji **6:1**
    - Blok **4×4 tekseli** – zawsze **64 bity** zamiast 384 bitów:



    - Cztery możliwe wartości kodowania udziału decydują o tym, **jak wyliczany jest kolor konkretnego teksela** na podstawie **obu określonych kolorów**

TEKSTUROWANIE

Użycie kompresji *S3TC* w *OpenGL* nie jest objęte standardem, wymaga posłużenia się uniwersalnym **rozszerzeniem *EXT_texture_compression_s3tc***.

Możliwe jest przekazanie do *OpenGL* tekstury w postaci nieskompresowanej i zlecenie sterownikowi kompresji, jak i przekazanie postaci już skompresowanej. W drugim przypadku operacja jest **znacznie szybsza**.

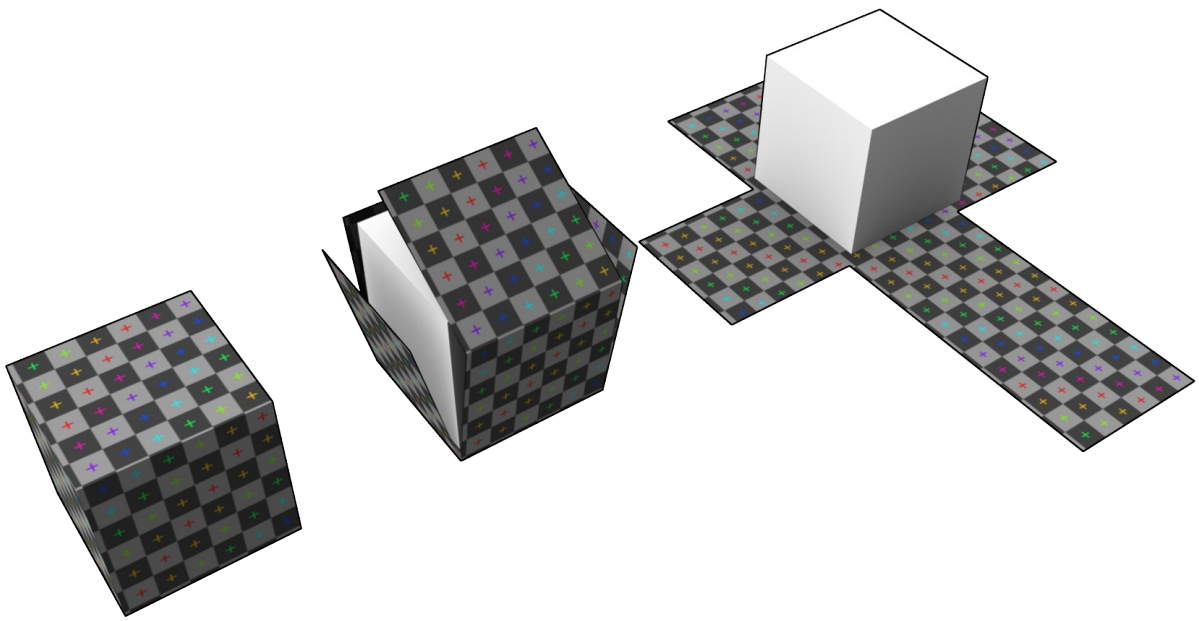
- **Metoda kompresji tekstur S3** (*S3TC*), nazywana czasem metodą *DXT*
  - Kompresja **stratna**
  - **Stopień kompresji** (ang. *compression ratio*) jest **stały**
  - Operuje na **blokach** *4×4*
  - Pozwala na **wybiórczą dekompresję** tekstury
  - Pozostałe odmiany *DXT*:

FOURCC	DX 10 Name	Description	Alpha premultiplied?	Compression ratio	Texture Type
DXT1	BC1	1-bit Alpha / Opaque	N/A	6:1(for 24 bit source image)	Simple non-alpha
DXT2	BC2	Explicit alpha	Yes	4:1	Sharp alpha
DXT3	BC2	Explicit alpha	No	4:1	Sharp alpha
DXT4	BC3	Interpolated alpha	Yes	4:1	Gradient alpha
DXT5	BC3	Interpolated alpha	No	4:1	Gradient alpha

Źródło obrazu:
Wikipedia

TEKSTUROWANIE

- **Mapowanie tekstury** jest to odwzorowanie powierzchni obiektu w przestrzeń tekstury
  - Jeśli posłużymy się **analogią** do owijania pudełka kolorowym papierem, to mapowanie określa **która część arkusza papieru trafi na którą stronę pudełka**



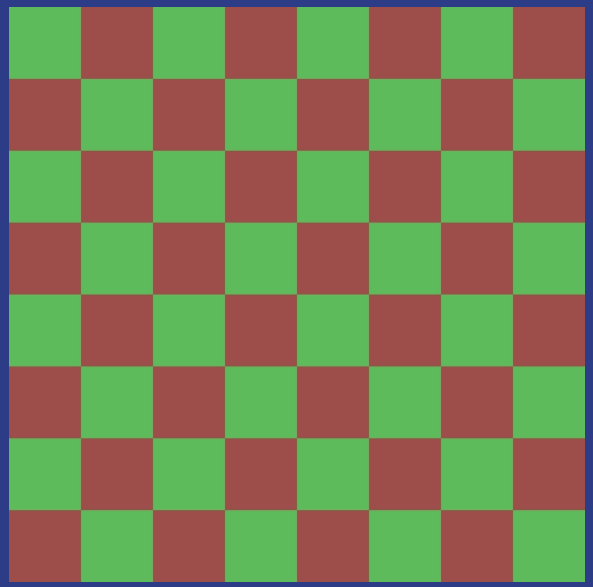
Źródło obrazu:
Wikipedia

TEKSTUROWANIE

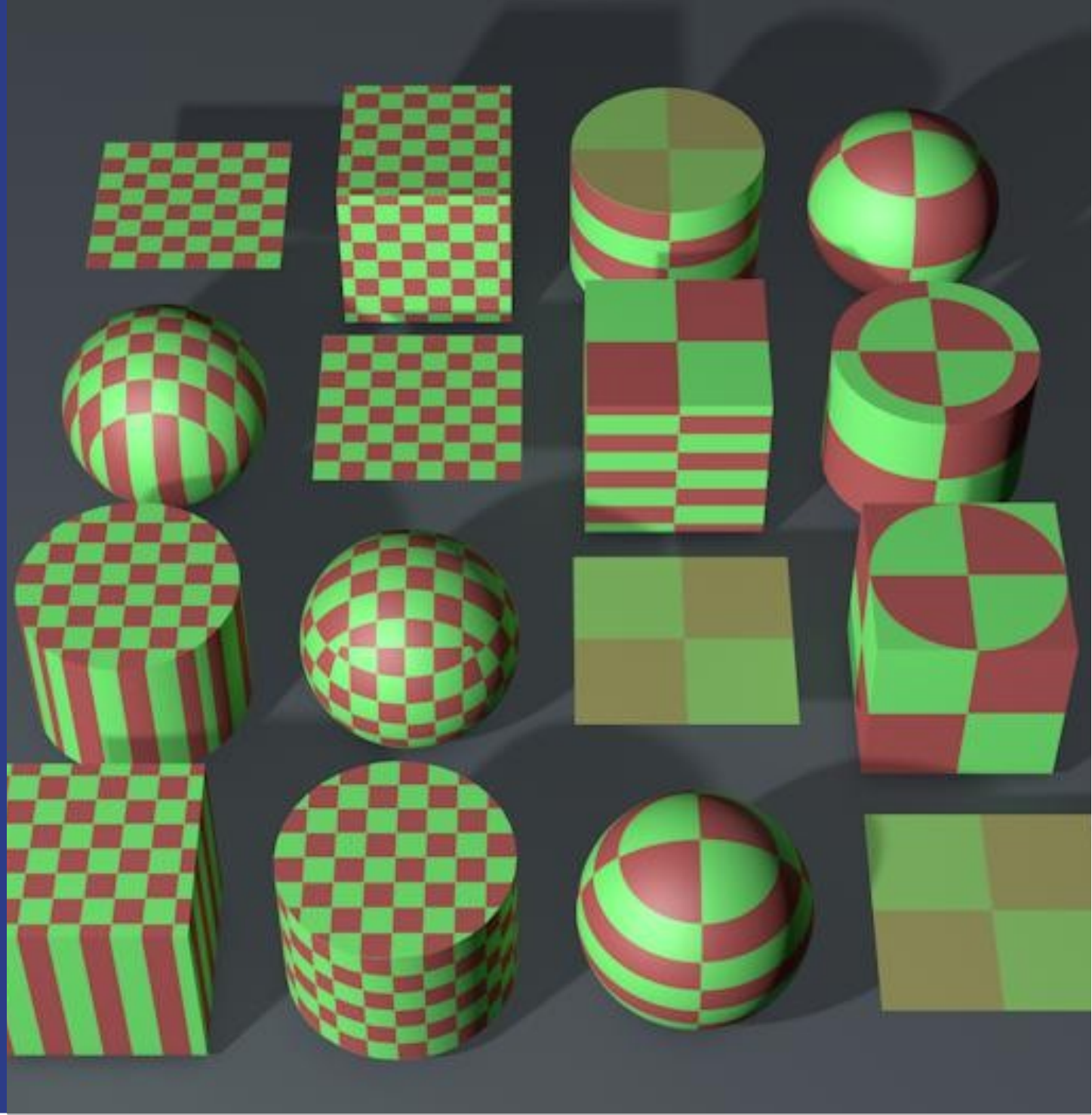
Podstawowe, automatyczne **techniki mapowania tekstury** (kolejno od lewej):

Płaskie (ang. *flat*)
Sześcienne (ang. *cube*)
Cylindryczne (ang. *cylindrical*)
Sferyczne (ang. *spherical*)

Użyta tekstura:

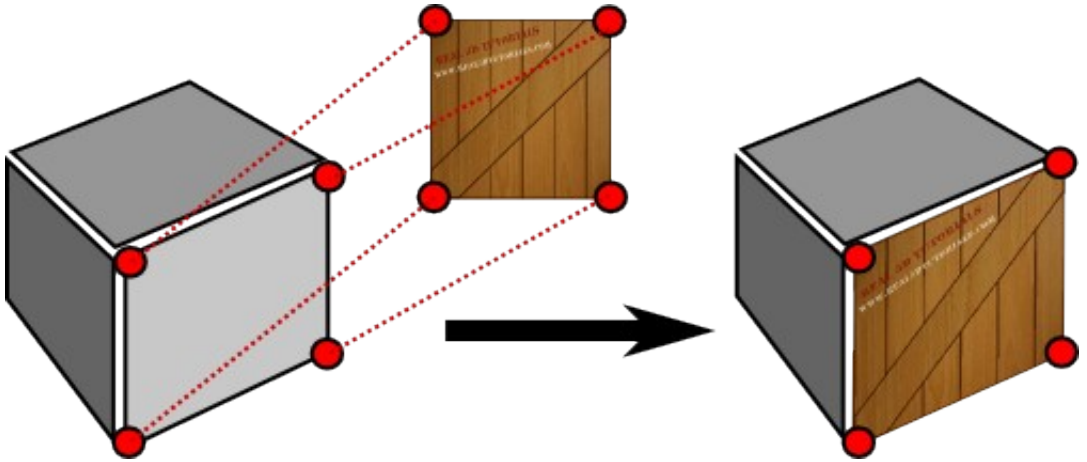


Źródło obrazu:
http://www.rafaelservantez.com/



TEKSTUROWANIE

- **Mapowanie UV** jest techniką która zakłada, że definicji geometrii bryły towarzyszy informacja o mapowaniu jej powierzchni w przestrzeń tekstury
  - Najczęściej *współrzędne tekstury* definiowane są **w wierzchołkach bryły**
  - **Każdemu wierzchołkowi** przypisane jest **położenie *UV*** w przestrzeni tekstury

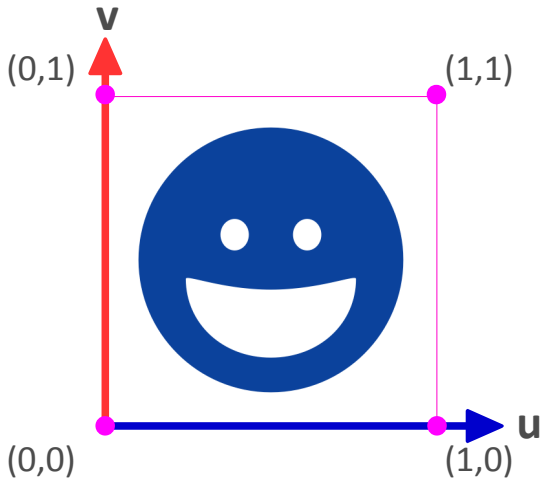


Źródło obrazu:
http://real3dtutorials.com

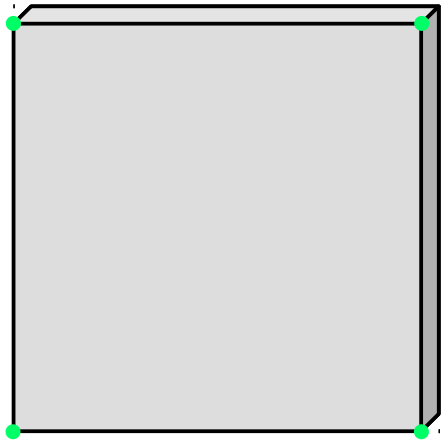
TEKSTUROWANIE

- Definiowanie współrzędnych tekstury polega na **przyporządkowaniu współrzędnych *UV* wierzchołkom** naszej teksturowanej bryły

Tekstura:



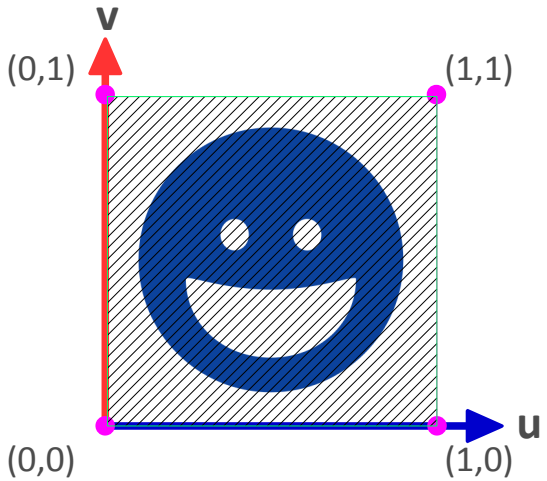
Teksturowana bryła:



TEKSTUROWANIE

- Definiowanie współrzędnych tekstury polega na **przyporządkowaniu współrzędnych *UV* wierzchołkom** naszej teksturowanej bryły

Tekstura:



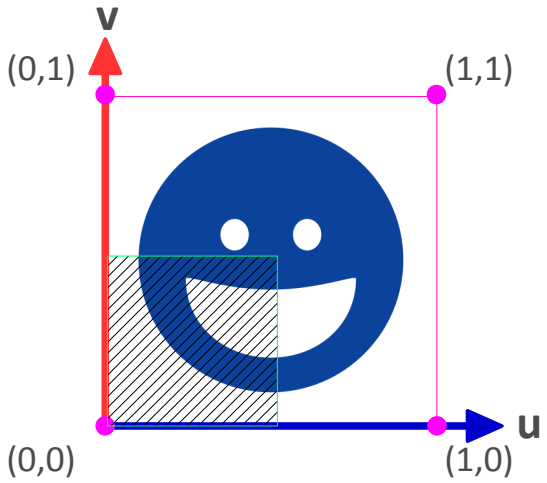
Teksturowana bryła:

W wyniku takiego mapowania zwyczajnie nakładamy **całą teksturę** na powierzchnię przedniej ściany bryły.

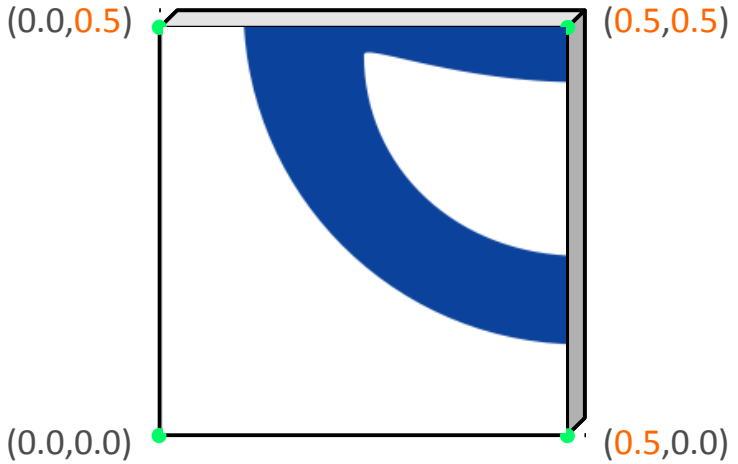
TEKSTUROWANIE

- Definiowanie współrzędnych tekstury polega na **przyporządkowaniu współrzędnych *UV* wierzchołkom** naszej teksturowanej bryły

Tekstura:



Teksturowana bryła:

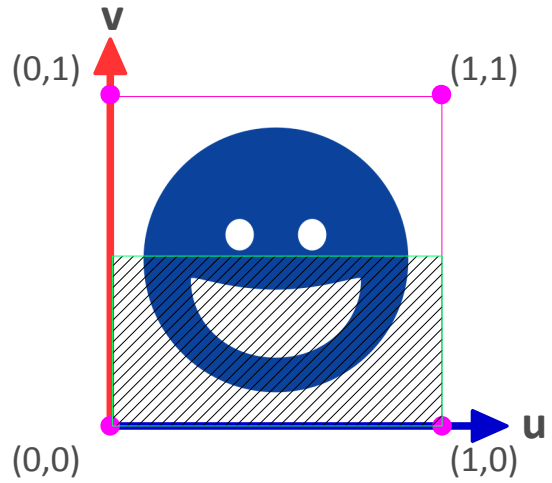


Można mapować jedynie **wycinek tekstury** na daną ścianę bryły.

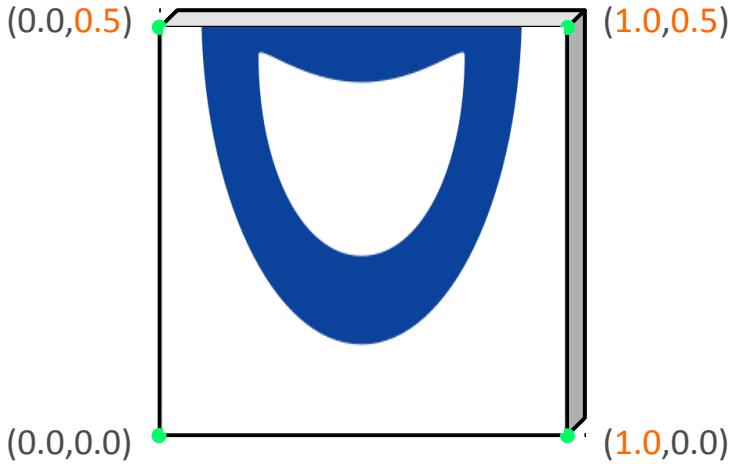
TEKSTUROWANIE

- Definiowanie współrzędnych tekstury polega na **przyporządkowaniu współrzędnych *UV* wierzchołkom** naszej teksturowanej bryły

Tekstura:



Teksturowana bryła:

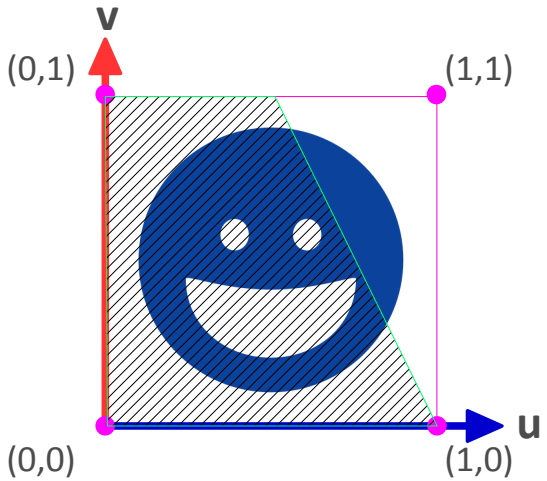


Proporcje wycinka tekstury wcale nie muszą odpowiadać proporcjom długości boków teksturowanej ściany.

TEKSTUROWANIE

- Definiowanie współrzędnych tekstury polega na **przyporządkowaniu współrzędnych *UV* wierzchołkom** naszej teksturowanej bryły

Tekstura:



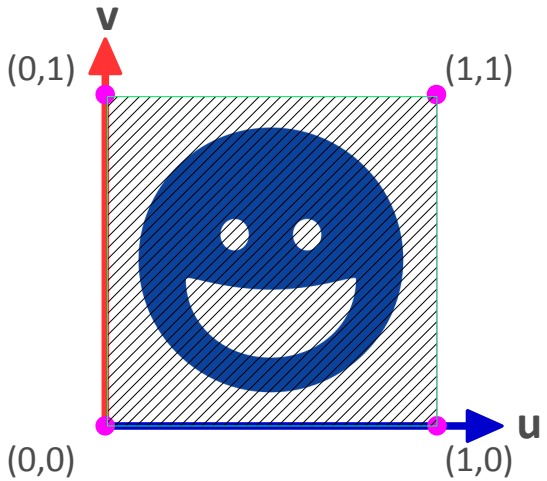
Kształt również nie musi odpowiadać kształtowi ściany.

Teksturowana bryła:

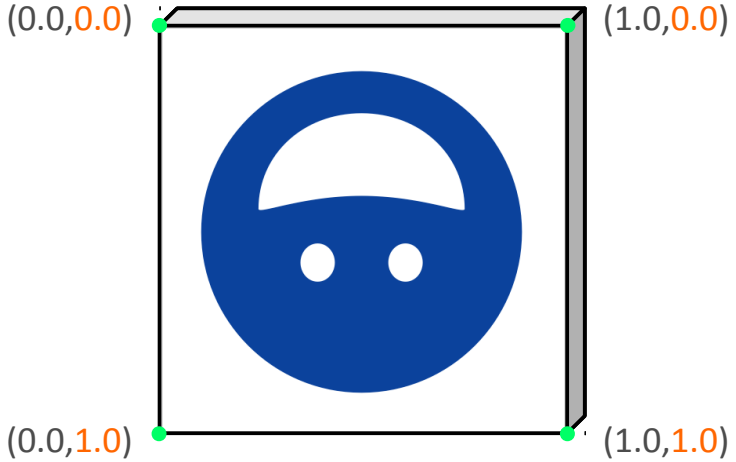
TEKSTUROWANIE

- Definiowanie współrzędnych tekstury polega na **przyporządkowaniu współrzędnych *UV* wierzchołkom** naszej teksturowanej bryły

Tekstura:



Teksturowana bryła:



Łatwo można też uzyskać **odbicie lustrzane** nakładanej tekstury.

TEKSTUROWANIE

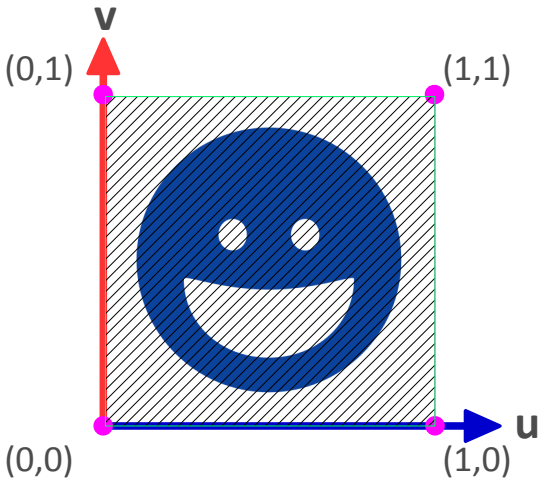
Współrzędne mogą wykraczać poza zakres od 0.0 do 1.0.
Wówczas **rezultat będzie zależał od ustawienia danej tekstury**.

Na przykład możliwe jest **powtarzanie tekstury**:
GL_REPEAT,
GL_MIRRORED_REPEAT

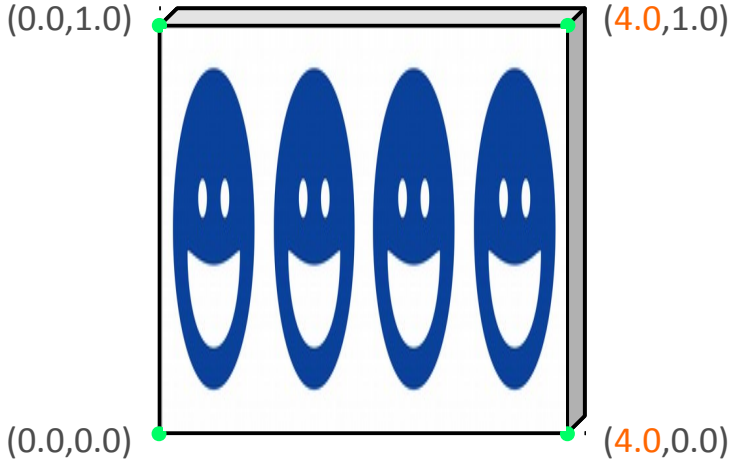
Źródło obrazu:
http://open.gl/

- Definiowanie współrzędnych tekstury polega na **przyporządkowaniu współrzędnych *UV* wierzchołkom** naszej teksturowanej bryły

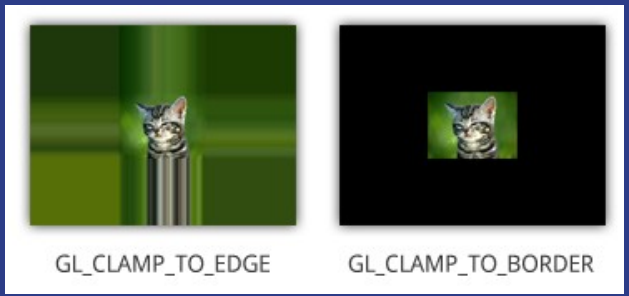
Tekstura:



Teksturowana bryła:



TEKSTUROWANIE



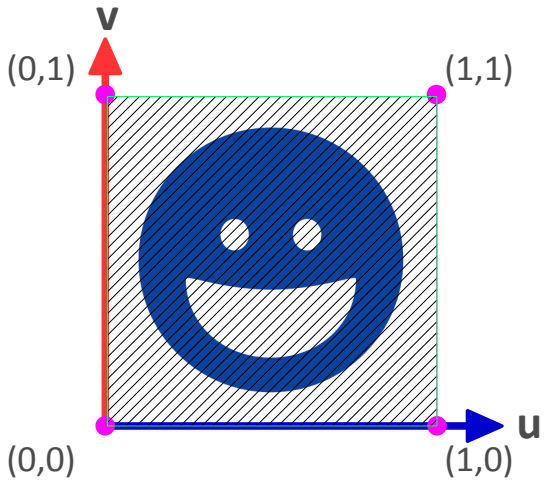
Na przykład możliwe jest też **przycinanie tekstury**: *GL_CLAMP_TO_BORDER*, *GL_CLAMP_TO_EDGE*

Wówczas ostatni wiersz/kolumna **są powtarzane** w nieskończoność.

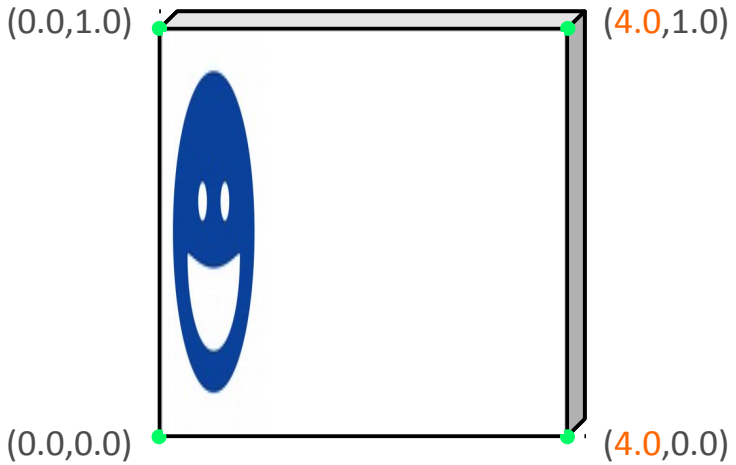
Źródło obrazu:
http://open.gl/

- Definiowanie współrzędnych tekstury polega na **przyporządkowaniu współrzędnych *UV* wierzchołkom** naszej teksturowanej bryły

Tekstura:

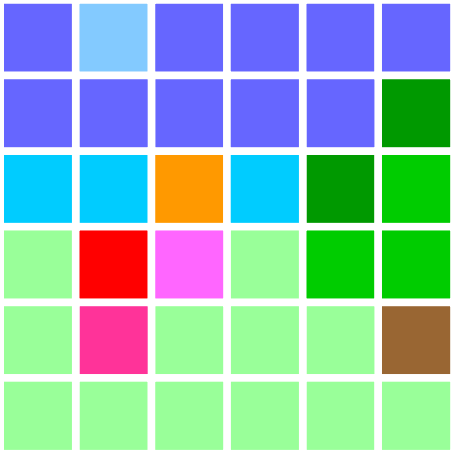


Teksturowana bryła:



TEKSTUROWANIE

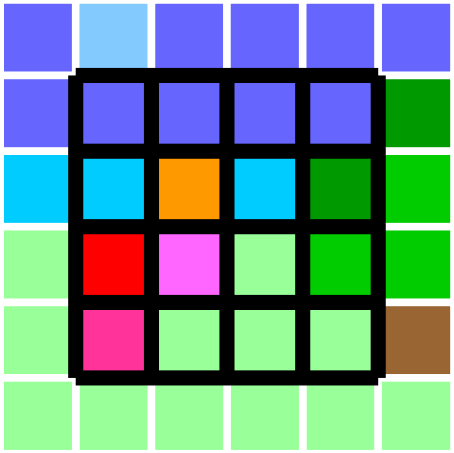
- Zadanie: **zwrócić wartość na podstawie tekstury** dla zadanych współrzędnych, wynikających z danego fragmentu teksturowanej powierzchni.



Nasza tekstura ma określoną rozdzielczość, mamy załadowany obraz.

TEKSTUROWANIE

- Zadanie: **zwrócić wartość na podstawie tekstury** dla zadanych współrzędnych, wynikających z danego fragmentu teksturowanej powierzchni.

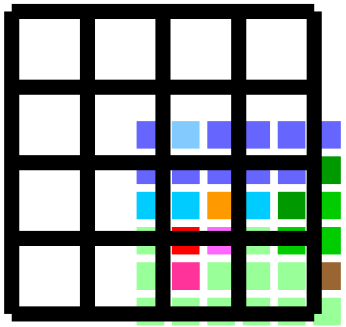


Mapujemy tę teksturę na powierzchnię ściany naszej bryły (czarna siatka).

Wyznaczenie wartości poszczególnych fragmentów nie stanowi problemu – po prostu pobierane są wartości poszczególnych tekseli.

TEKSTUROWANIE

- Zadanie: **zwrócić wartość na podstawie tekstury** dla zadanych współrzędnych, wynikających z danego fragmentu teksturowanej powierzchni.



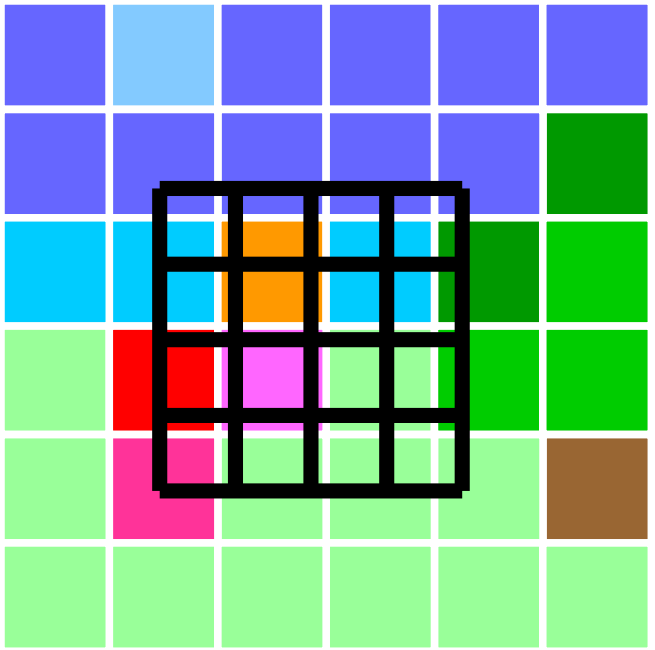
Mapujemy tę teksturę na powierzchnię ściany naszej bryły (czarna siatka).

Co jeśli z mapowania lub projekcji wynika, że pikselowi bufora klatki wcale **nie odpowiada dokładnie jeden teksel**?

Operację wyliczenia wartości na podstawie jednego lub więcej tekseli w wyniku nieregularnego jej mapowania nazywa się **filtrowaniem tekstury**.

TEKSTUROWANIE

- Zadanie: **zwrócić wartość na podstawie tekstury** dla zadanych współrzędnych, wynikających z danego fragmentu teksturowanej powierzchni.

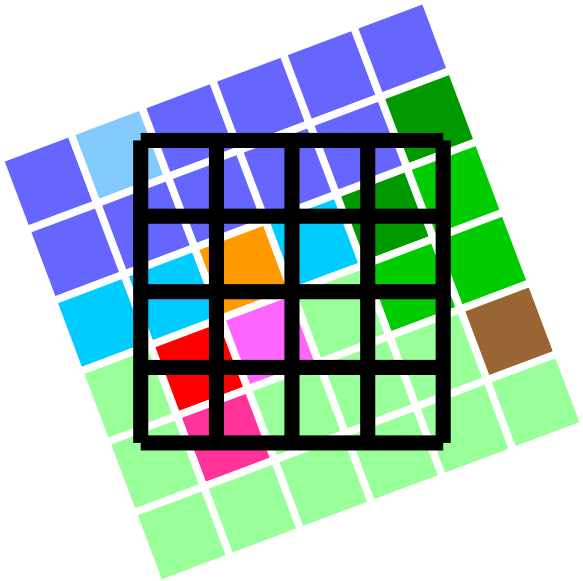


Mapujemy tę teksturę na powierzchnię ściany naszej bryły (czarna siatka).

Sytuacja taka ma miejsce także gdy teksele są podczas projekcji na bufor klatki „większe” niż piksele.

TEKSTUROWANIE

- Zadanie: **zwrócić wartość na podstawie tekstury** dla zadanych współrzędnych, wynikających z danego fragmentu teksturowanej powierzchni.

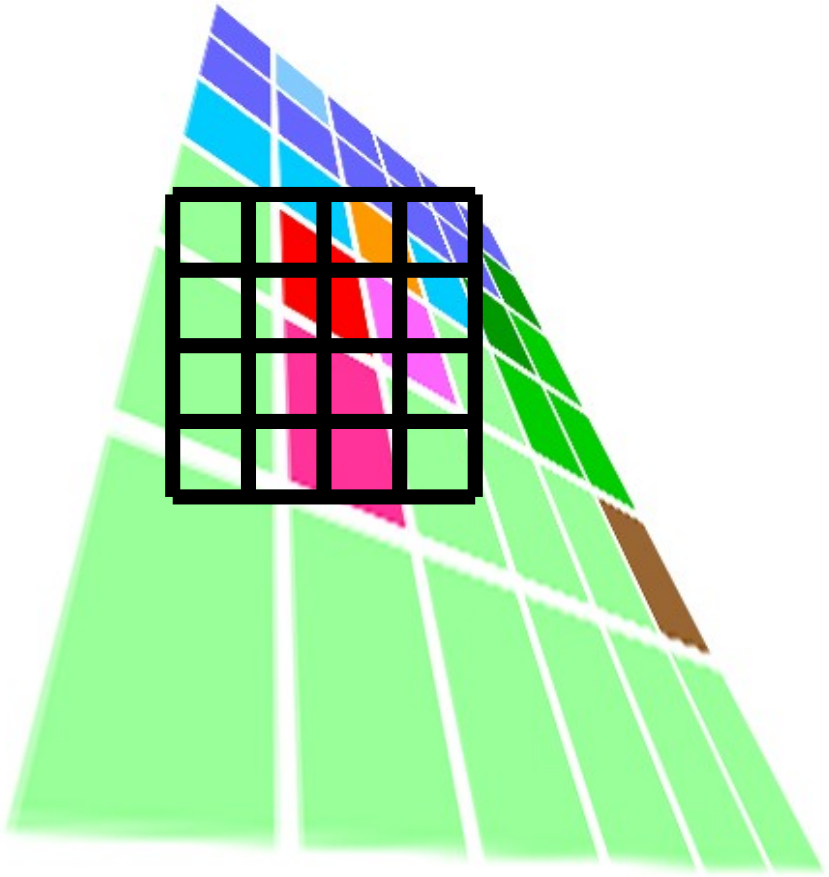


Mapujemy tę teksturę na powierzchnię ściany naszej bryły (czarna siatka).

Także wtedy, gdy tekstura jest obrócona względem docelowego rastra.

TEKSTUROWANIE

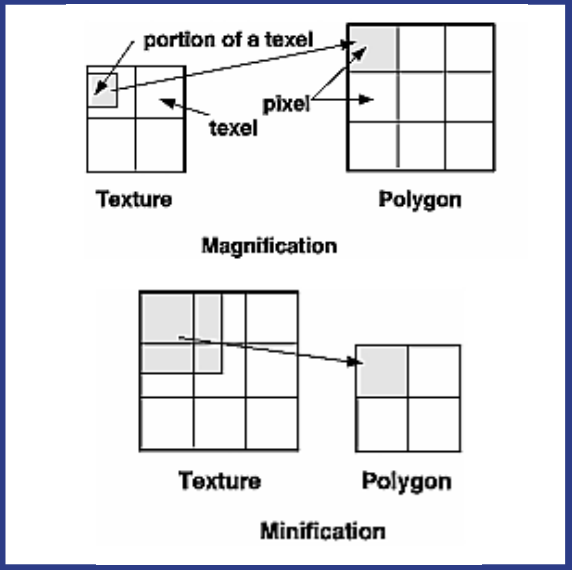
- Zadanie: **zwrócić wartość na podstawie tekstury** dla zadanych współrzędnych, wynikających z danego fragmentu teksturowanej powierzchni.



Mapujemy tę teksturę na powierzchnię ściany naszej bryły (czarna siatka).

Lub gdy uświadomimy sobie, że nasz świat ma przecież trzy wymiary i perspektywę...

TEKSTUROWANIE



Metoda **najmniej kosztowna**.

Nadaje się zarówno do **powiększania**, jak i **pomniejszania**.

Rezultaty **zachowują wyraźne krawędzie**, ale często spotyka się artefakty w postaci **aliasingu**.

Źródło obrazu:
beyond3D
id software
http://felixgers.de

- **Filtrowanie tekstury** jest metodą, która służy do uzyskania **wartości fragmentu** w wyniku mapowania tekstury **na podstawie wartości odpowiadających temu fragmentowi tekseli**
  - **Metoda najbliższego sąsiada** (ang. *nearest neighbor*)
    - Bierze pod uwagę tylko jeden teksel, który najlepiej odpowiada przetrwarzanemu fragmentowi

TEKSTUROWANIE

- **Filtrowanie tekstury** jest metodą, która służy do uzyskania **wartości fragmentu** w wyniku mapowania tekstury **na podstawie wartości odpowiadających temu fragmentowi tekseli**
  - **Metoda filtracji liniowej** (ang. *(bi)linear filtering*)
    - **Interpolacja liniowa** czterech najbliższych tekseli

Metoda pobiera zawsze cztery wartości teksela zamiast tylko jednej, przez co jest **wolniejsza**.

Nadaje się zarówno do **powiększania**, jak i **pomniejszania** – choć w tym drugim przypadku pojawiają się **artefakty** podobne do metody najbliższego sąsiada gdy tekstura jest znacznie pomniejszona.

Rezultaty **rozmywają krawędzie**.

Źródło obrazu:
beyond3D
id software

TEKSTUROWANIE

Podczas tworzenia, każdy teksel następnego poziomu mipmapy to **średnia arytmetyczna odpowiadających mu czterech tekseli** poprzedniego poziomu.

Aby przechować pełną piramidę mipmap, nigdy nie potrzeba więcej niż **o 1/3 więcej miejsca w pamięci** niż na przechowanie samej oryginalnej tekstury.

- **Mipmapy** (łac. *multum im parvo*) są wcześniej wyliczonymi sekwencjami pomniejszonych wersji tej samej oryginalnej tekstury
  - Każdy następny poziom mipmapy ma wymiary **dwukrotnie mniejsze** wzdłuż każdej krawędzi
  - Mipmapa zerowego poziomu to oryginalny obraz
  - Najmniejsza mipmapa to pojedynczy teksel

Źródło obrazu:
http://tomshardwareguide.com

TEKSTUROWANIE

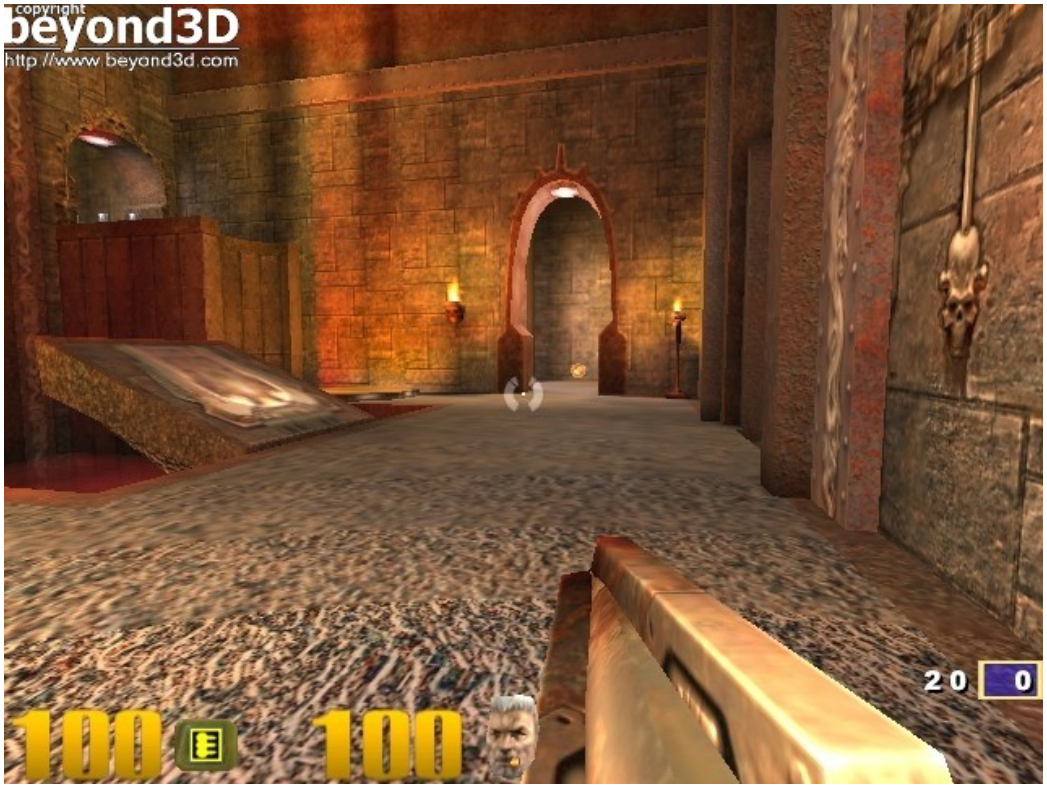
- **Filtrowanie tekstury** jest metodą, która służy do uzyskania **wartości fragmentu** w wyniku mapowania tekstury **na podstawie wartości odpowiadających temu fragmentowi tekseli**
  - **Najbliższa *mipmapa*** (ang. *nearest mipmap*)
    - Wartość pobierana jest **z poziomu mipmapy, który najlepiej odpowiada** wielkości przetwarzanego fragmentu

Metoda może wykorzystywać podejście najbliższego sąsiada lub filtrację liniową w obrębie poziomu *mipmapy*.

Wymaga wcześniejszego **wygenerowania *mipmap***.

Nadaje się **jedynie do pomniejszania** – *mipmapy* naturalnie nic nie dają gdy spodziewany obraz jest większy niż oryginał.

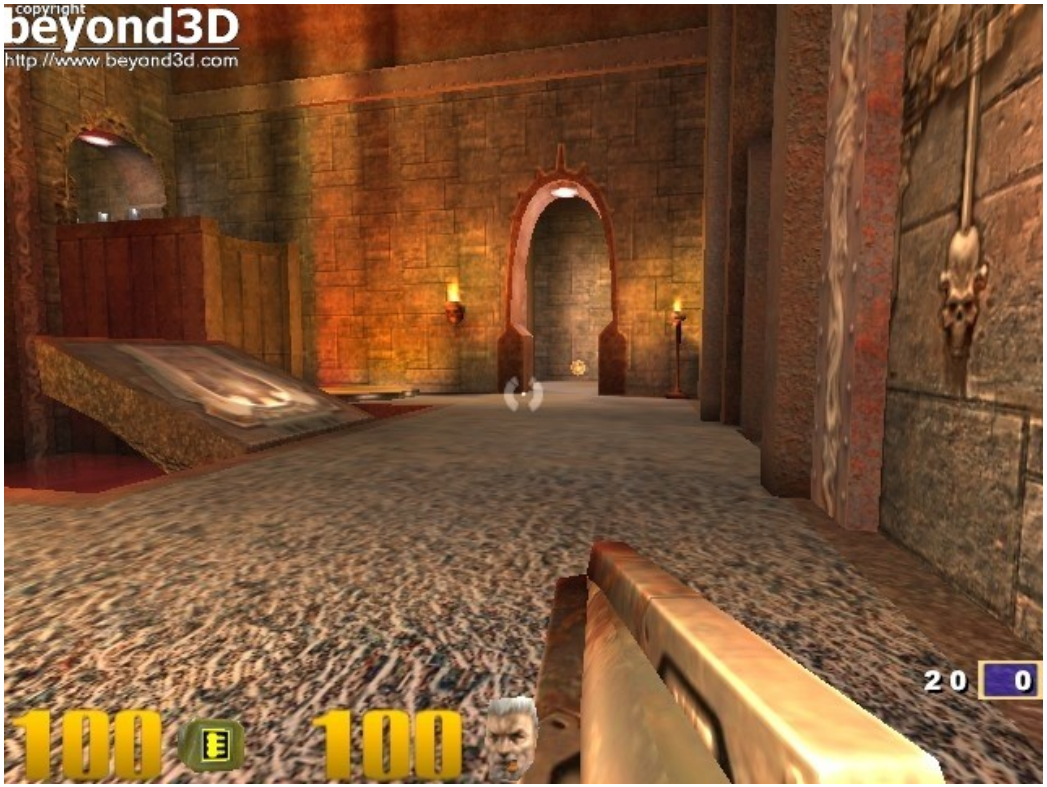
Mogą być **widoczne granice** poszczególnych poziomów *mipmap*.



Źródło obrazu:
beyond3D
id software

TEKSTUROWANIE

- **Filtrowanie tekstury** jest metodą, która służy do uzyskania **wartości fragmentu** w wyniku mapowania tekstury **na podstawie wartości odpowiadających temu fragmentowi tekseli**
  - **Filtracja trójliniowa** (ang. *trilinear filtering*)
    - Wartość jest obliczana jako **średnia ważona** pomiędzy najbliższymi mipmapami



Praktycznie zawsze pobierane są wartości z dwu sąsiadujących poziomów mipmap – co w przypadku filtracji liniowej w ich obrębie daje zawsze **8 odczytów z tekstury**. Jest to metoda kosztowna.

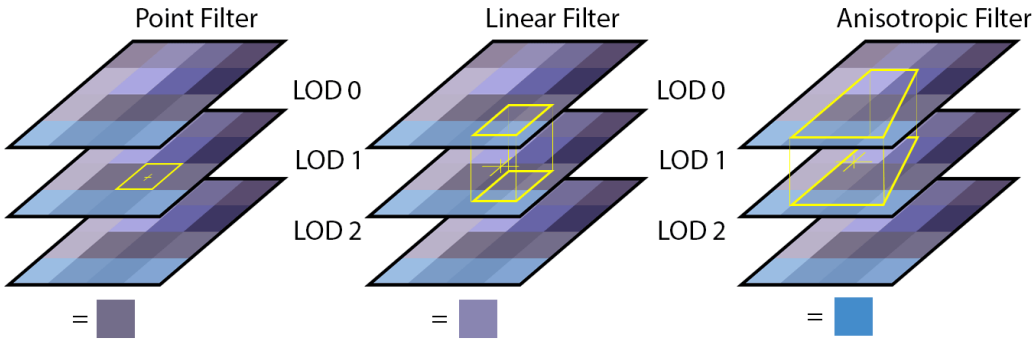
Wymaga wcześniejszego **wygenerowania *mipmap***.

Nadaje się **jedynie do pomniejszania** – *mipmapy* naturalnie nic nie dają gdy spodziewany obraz jest większy niż oryginał.

Granice pomiędzy poziomami *mipmap* zacierają się.

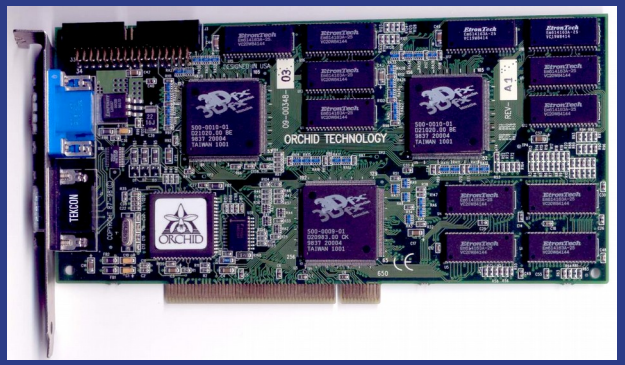
Źródło obrazu:
beyond3D
id software

- **Filtrowanie anizotropowe**
  - Technika **zmniejszająca uciążliwość rozmycia** wynikającego z dużej **deformacji tekseli** podczas oglądania tekstury pod małym kątem do jej powierzchni



Źródła obrazów:
NVIDIA
http://3dgep.com

TEKSTUROWANIE



3dfx Voodoo 2 (1998) był pierwszym *akceleratorem 3D*, który oferował *aż* **dwie jednostki teksturujące**.

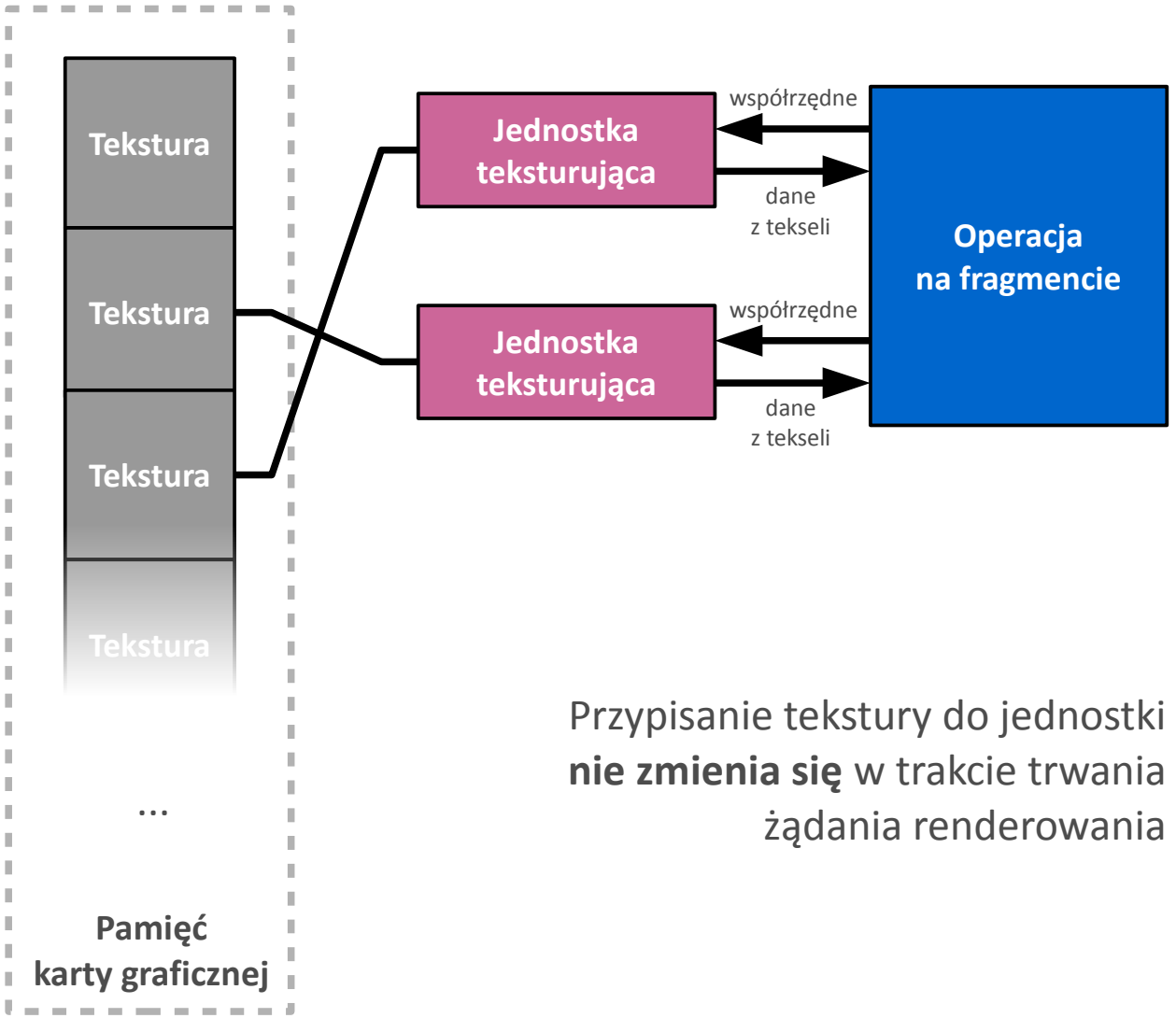
Źródło obrazu: Wikipedia

- **Jednostka teksturująca** (ang. *texture unit*)
  - Dawniej osobny układ logiczny karty graficznej (*TMU*) dedykowany operacjom na teksturze
  - Obecnie integralna część *GPU*, może być rozumiana jako ***lokacja*** do której **przypisujemy teksturę** celem jej użycia
    - Mówi się o niej jako o ***texture image unit (TIU)***
    - W danej chwili jednostka może mieć przypisaną (ang. *bound*) **tylko jedną teksturę**
    - Karta graficzna ma **ograniczoną liczbę jednostek**, np.:
      - 3dfx **Voodoo 2**: **2** jednostki
      - NVIDIA GeForce **7600 GT**: **12** jednostek
      - NVIDIA GeForce **750M**: **32** jednostki
      - NVIDIA GeForce **GTX 980 Ti**: **176** jednostek
  - Odpowiada za **adresowanie**, **dekompresję danych**, **dostęp** do nich i ich **filtrowanie**
    - Zadanie jednostki teksturującej: **dostarczyć wartość próbki** na podstawie danych tekstury, **dla zadanych współrzędnych**
    - Całą operację nazywa się często w jęz. ang. ***texture fetch***

TEKSTUROWANIE

To **ile jest dostępnych jednostek**, determinuje **z ilu tekstur jednocześnie** można korzystać przy operacji na danym fragmencie podczas jednego żądania renderowania

- **Tekstura** znajdująca się w pamięci karty graficznej musi zostać **przypisana** (ang. *bound*) **do jednostki teksturującej**, aby mogła zostać użyta



Przypisanie tekstury do jednostki **nie zmienia się** w trakcie trwania żądania renderowania

Źródło obrazu:
http://real3dtutorials.com

TEKSTUROWANIE

Jednostka teksturująca posiada **osobne miejsca przyłączenia dla każdego z typów tekstur**. Jednocześnie mogą być przypisane do niej różne tekstury pod warunkiem że każda jest innego typu. Jendak w danej chwili używana może być **tylko jedna z nich**.

- W OpenGL w danej chwili z punktu widzenia edycji danych i parametrów **aktywna jest tylko jedna jednostka teksturująca**
  - Analogicznie do aktywnej macierzy transformacji:

```
glActiveTexture(texture_unit); // np. GL_TEXTURE0 lub GL_TEXTURE6
```

  - **Podczas renderowania aktywne są wszystkie jednostki**, do których przypisane są tekstury
  - Dana jednostka teksturująca może mieć jednocześnie przypisane **po jednej teksturze dla każdego z typów**
    - Na przykład jednocześnie przypisana jest inna tekstura dla *GL_TEXTURE_1D* i inna dla typu *GL_TEXTURE_2D*.
    - W jednej chwili **tylko jedna z nich może być używana**, choć przypisanie pozostałych typów jest pamiętane
    - Wybór tego który typ tekstury, a więc de facto która z przypisanych do danej jednostki tekstur jest używana, odbywa się na podstawie typu *samplera* w programie cieniującym

TEKSTUROWANIE

- W *OpenGL* **mapowanie UV** odbywa się poprzez przypisanie wierzchołkowi jego współrzędnych tekstury
  - **Analogicznie do współrzędnych wektora normalnego** – współrzędne tekstury też są atrybutem wierzchołka
  - Służy do tego funkcja ***glTexCoord*()***

```
glTexCoord1f(s); // dla współrzędnych tekstury 1D
glTexCoord2f(s, t); // dla współrzędnych tekstury 2D
glTexCoord3f(s, t, r); // dla współrzędnych tekstury 3D
```

    - Są też dostępne odmiany dla **innych typów danych**
  - **Raz ustawione** współrzędne tekstury będą nadawane **wszystkim nowo tworzonym wierzchołkom**, dopóki nie zostaną one zmienione na inne
  - Możliwe jest oczywiście też korzystanie z ***VA*** i ***VBO***, wtedy miejsce w pamięci wskazujemy następująco:

```
glTexCoordPointer(num, type, stride, offset);
```

TEKSTUROWANIE

- **Stworzenie tekstury** w *OpenGL* polega na:
  - **Uzyskaniu jej identyfikatora**
    (w dokumentacji opisanego jako *texture name*)

```
GLuint id;
glGenTextures(1, &id); // Pierwszy parametr określa żądaną liczbę tekstur
```

  - **Podłączeniu tekstury** o zadanym identyfikatorze
    do jednostki cieniującej (ang. *binding*)

```
glBindTexture(GL_TEXTURE_2D, id);
```

    - W pierwszym parametrze określamy typ tekstury, do którego przypisujemy identyfikator
    - Od tego momentu wszystkie operacje na teksturach będą dotyczyły tej *zbindowanej* tekstury (dla aktywnej jednostki cieniującej)
  - **Wysłaniu danych** do pamięci karty graficznej

TEKSTUROWANIE

level – poziom *MIPmapy* (0, 1, 2...)
format – format danych w pamięci
w, ***h*** – liczba kolumn, wierszy
border – grubość obramowania
d_format – format danych wejściowych
d_type – typ danych wejściowych
d – wskaźnik na pierwszy element

- **Wysyłanie danych do pamięci karty graficznej** zawierających obraz będący częścią aktualnie wybranej tekstury:
  - Dla tekstury 2D:

```
glTexImage2D(GL_TEXTURE_2D, level, format, w, h, border, d_format, d_type, d);
```

    - Przekazanie *NULL* w miejscu ***d*** sprawia, że pamięć jest tylko **alokowana**

```
glTexSubImage2D(GL_TEXTURE_2D, level, xoffset, yoffset, w, h,
        d_format, d_type, d);
```

    - Pamięć **nie jest alokowana**, zmieniany jest tylko fragment danych tekstury
    - Format danych zostanie zachowany taki jak podczas alokacji

TEKSTUROWANIE

pname – stała określająca którego parametru wartość chcemy ustawić

Wartości związane z *MIPmapami* nie mają sensu dla powiększania tekstur.

- **Ustawienie parametrów tekstury** w *OpenGL*
  - Funkcja do tego służąca:

```
glTexParameterf(GL_TEXTURE_2D, pname, value);
glTexParameteri(GL_TEXTURE_2D, pname, value);
glTexParameterfv(GL_TEXTURE_2D, pname, *value);
// ...
```

  - Wybrane parametry:
    - *GL_TEXTURE_WRAP_S, GL_TEXTURE_WRAP_T, GL_TEXTURE_WRAP_R*

      Określenie zachowania w momencie gdy **współrzędne tekstury wykraczają poza zakres 0..1**

      - *GL_CLAMP_TO_BORDER, GL_CLAMP_TO_EDGE, GL_MIRRORED_REPEAT, GL_MIRROR_CLAMP_TO_EDGE, GL_REPEAT*

    - *GL_TEXTURE_MIN_FILTER, GL_TEXTURE_MAG_FILTER*

      Techniki filtracji dla pomniejszania i powiększania tekstury

      - *GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_NEAREST, GL_LINEAR_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_LINEAR*

TEKSTUROWANIE

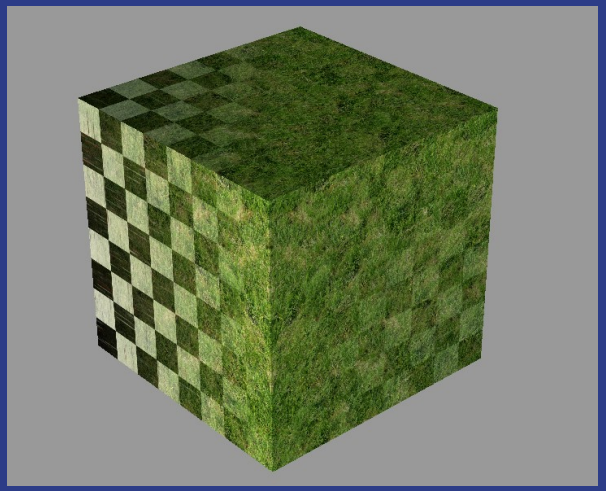
Problemy z atlasami tekstur:

Ograniczona rozdzielczość
Bleeding
Niemożliwy *wrapping*
Problemy z *MIPmapami*

- **Atlas tekstur**
  - Jest to podejście, w którym jeden obraz będący częścią tekstury zawiera niezależne informacje które zwyczajnie stanowiłyby kilka odrębnych obrazów
    - Idea podobna do *sprite sheet*
  - Dlaczego?
    - **Zmiana tekstury** podłączonej do danej jednostki jest **kosztowna**
    - Mając do narysowania **wiele elementów** z różnymi teksturami, konieczne byłoby **częste przełączanie tekstur**
    - Możliwość korzystania **z kilku obrazów** wyłącznie poprzez użycie **innych współrzędnych** w obrębie tej samej tekstury

Źródła obrazów:
EPIC Games
0fps.net

ZASTOSOWANIA TEKSTUROWANIA



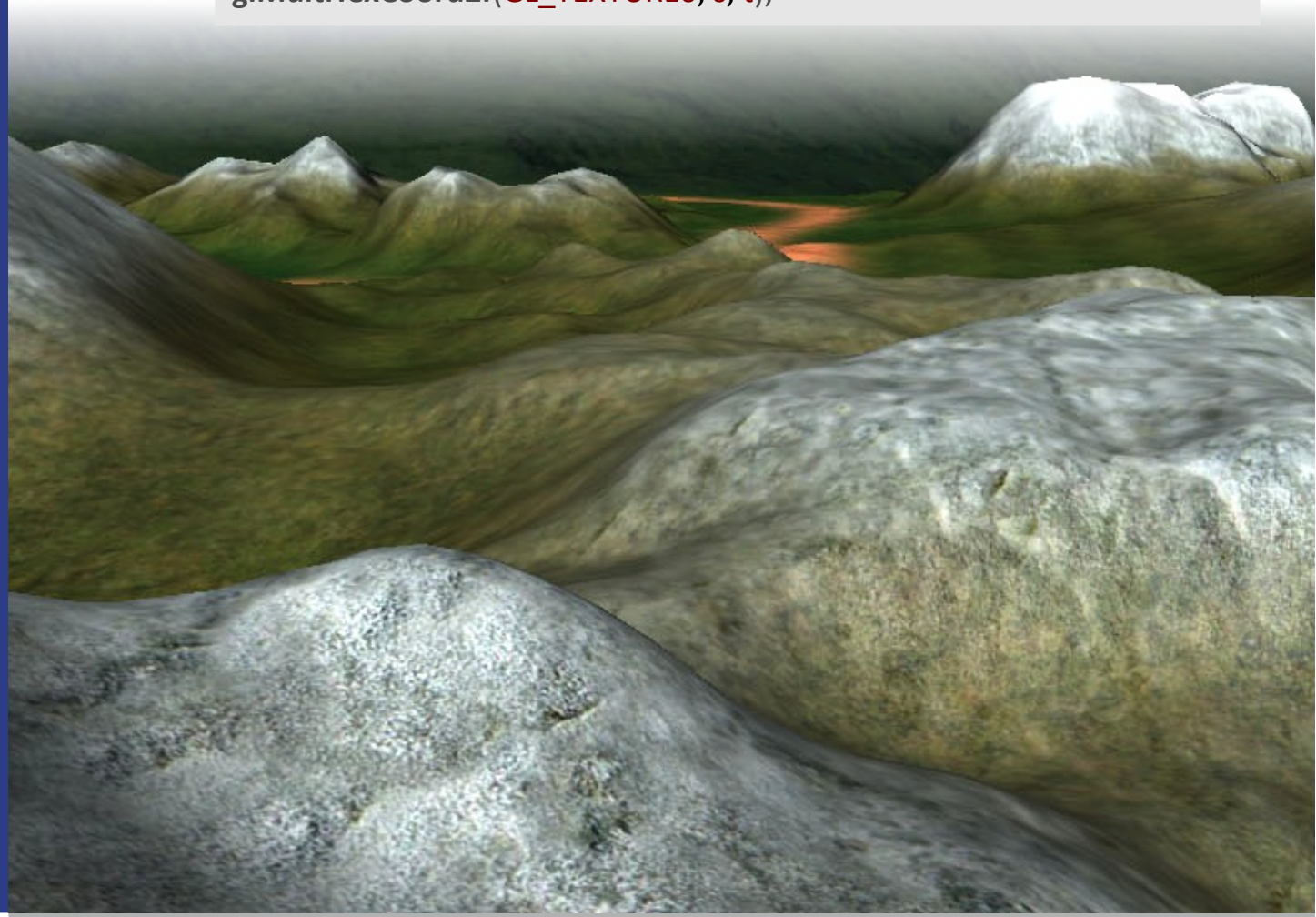
Wartości z różnych tekstur mogą być łączone by uzyskać jeden kolor, ale proces może być też bardziej złożony.

Jedna tekstura może służyć do modyfikacji koloru diffuse, inna do modyfikacji mnożnika specular, a jeszcze inna może być informacją o oświetleniu danego miejsca.

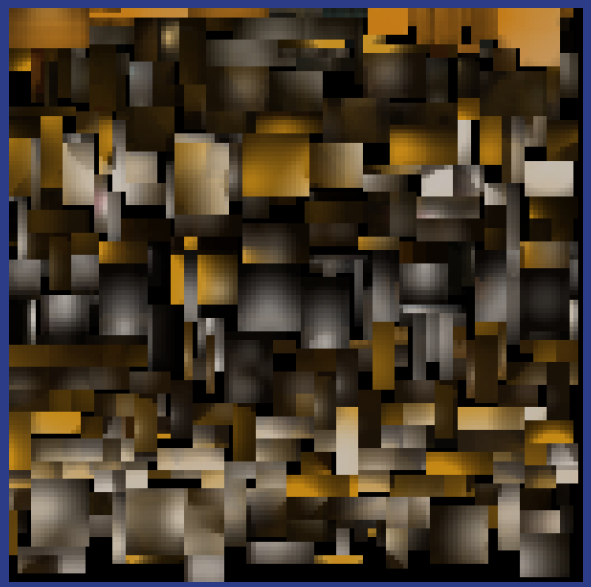
Źródła obrazów:
d3dcoder.net
http://felixfox.com

- **Multiteksturowanie** (ang. *multi-texturing*)
  - Wykorzystanie wartości pochodzących z **wielu tekstur** podczas **jednego żądania renderowania**
  - Można definiować różne współrzędne dla różnych tekstur

```
glMultiTexCoord2f(GL_TEXTURE0, s, t);
```



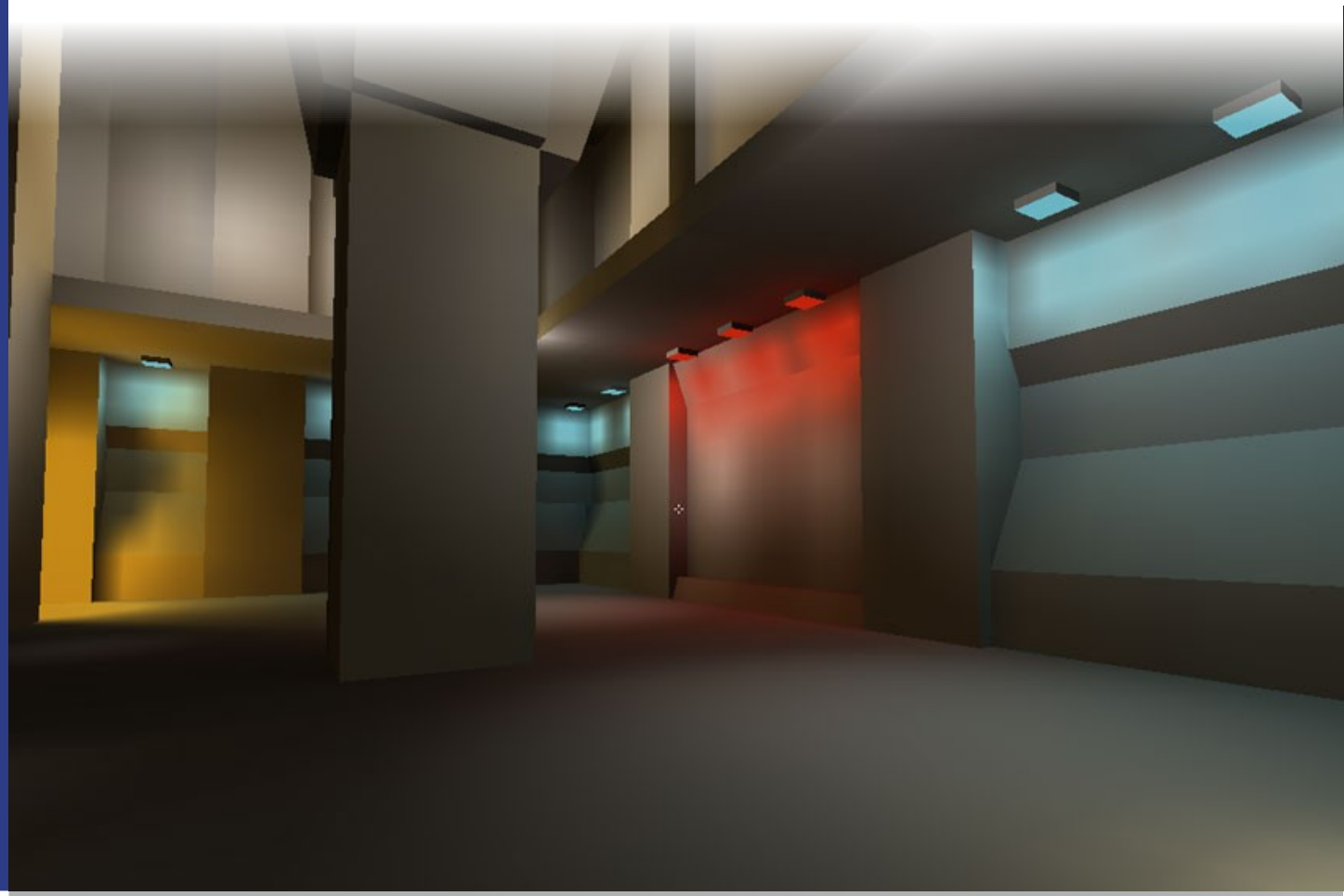
ZASTOSOWANIA TEKSTUROWANIA



Atlas map światła z gry *Quake 2*. Mapy są bardzo małe, ale zmiany oświetlenia nie mają dużej częstotliwości więc wystarcza to dla uzyskania zadowalającego efektu (po prawej)

Źródło obrazów:
id software
Fabien Sanglard

- **Mapowanie światła** (ang. *light mapping*)
  - Technika polegająca na "wypaleniu" *off-line* **mapy światła dla statycznych źródeł**, czyli takich które nie będą się zmieniały w trakcie cyklu życia sceny
  - Wartości z takiej mapy są później gotowymi **mnożnikami dla intensywności** poszczególnych **komponentów światła**

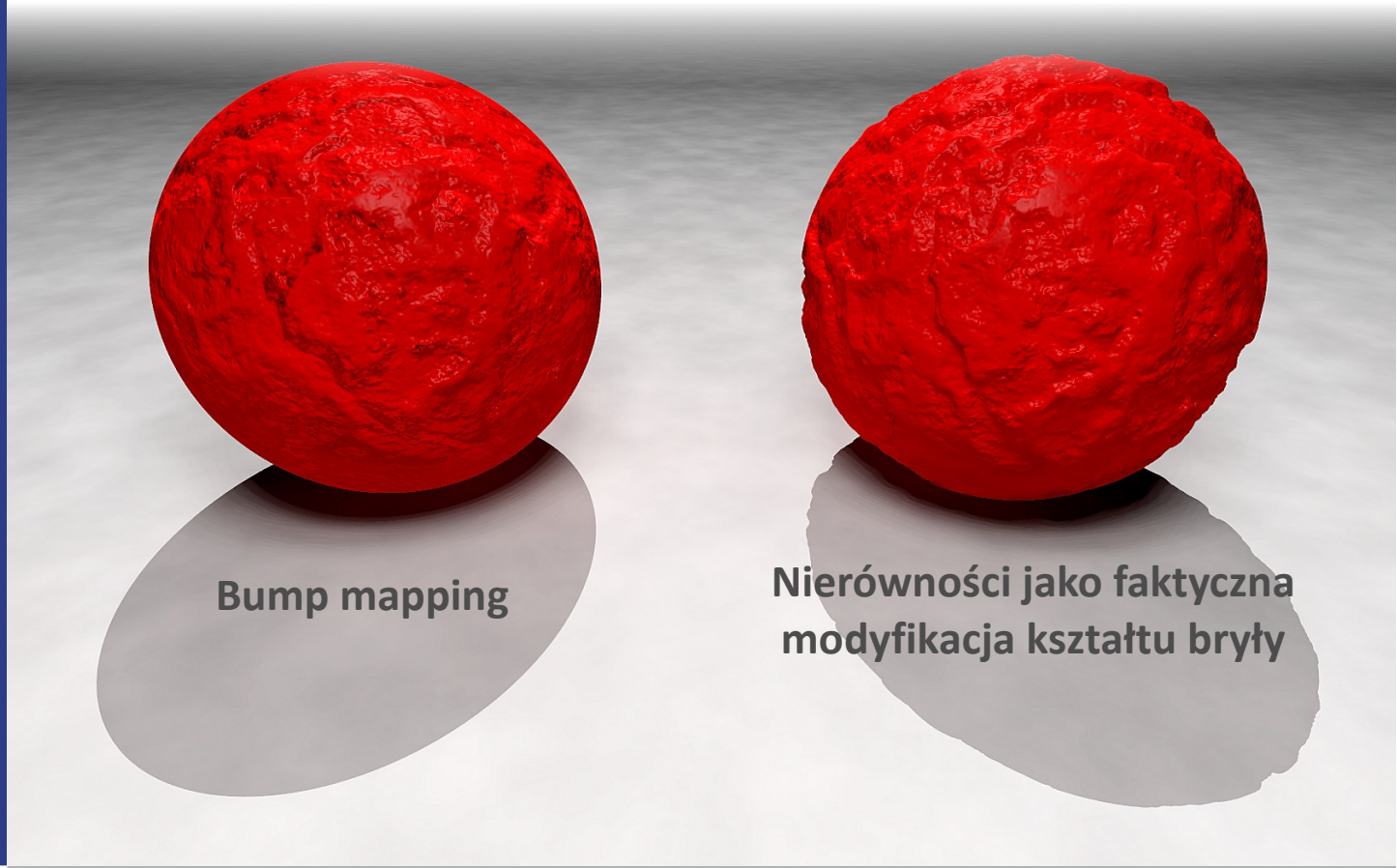


ZASTOSOWANIA TEKSTUROWANIA

Podstawowe techniki:

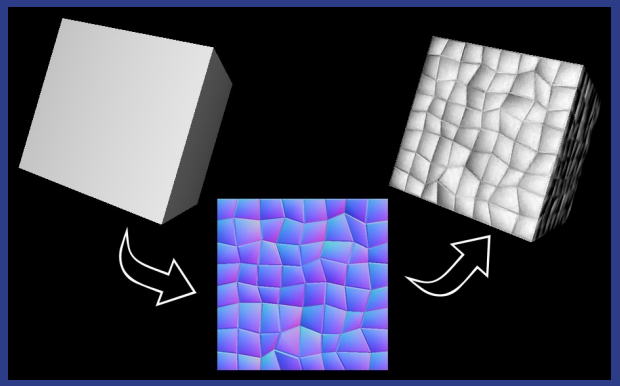
Normal mapping
Parallax mapping
Displacement mapping

- **Mapowanie nierówności** (ang. *bump mapping*)
  - **Symulacja nierówności** na powierzchni bryły **bez ingerencji** w jej oryginalny kształt
    - Wykorzystywane są **tekstury jako źródło informacji o nierównościach**, które mają być mapowane na powierzchnię
    - W większości technik *płaskość* powierzchni jest szczególnie widoczna przy krawędziach obiektu



Źródło obrazu:
Wikipedia

ZASTOSOWANIA TEKSTUROWANIA

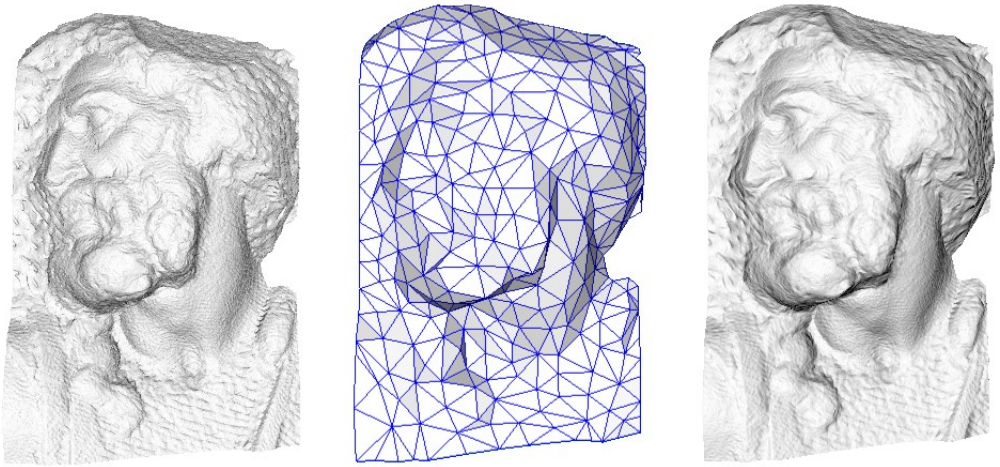


Współrzędne *XYZ* wektorów normalnych są zapisane jako **wartości *RGB*** w ich mapie.

Kiedy wizualizujemy mapę wektorów normalnych, często ma ona niebieskawe zabarwienie – jak na przykładzie wyżej.

Źródła obrazów:
Wikipedia
IndependentDeveloper.com

- **Mapowanie wektorów normalnych** (ang. *normal mapping*)
  - **Wektory normalne** dla fragmentów są wyznaczane na podstawie **wartości tekstury** – tzw. *mapy wektorów normalnych* (ang. *normal map*)
    - Mapa może zawierać wektory normalne zdefiniowane w:
      - **Przestrzeni obiektu lub świata** – wówczas nie jest potrzebna definicja wektorów normalnych w wierzchołkach
      - **Przestrzeni stycznej** do powierzchni bryły (ang. *tangent space*) – wówczas są one jedynie *modyfikacją* (*odchyleniem*) wektorów wyliczonych na podstawie zwyczajnej interpolacji pomiędzy wierzchołkami
  - Konieczne użycie cieniowania ***per-fragment***



original mesh
4M triangles

simplified mesh
500 triangles

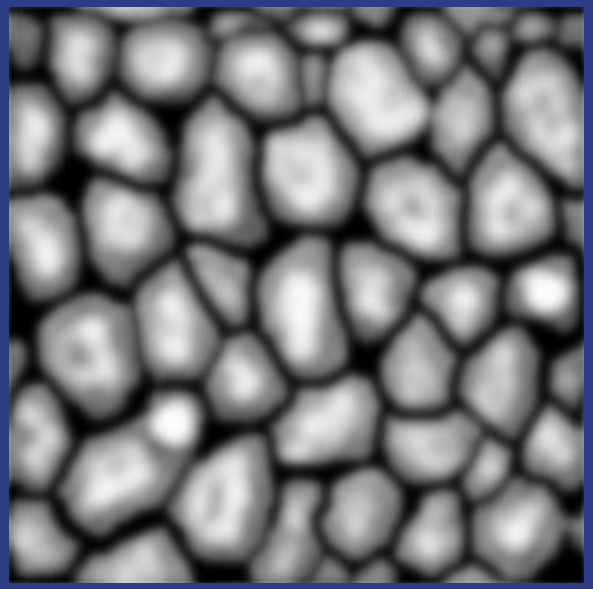
simplified mesh
and normal mapping
500 triangles

ZASTOSOWANIA TEKSTUROWANIA

- **Mapowanie wektorów normalnych** (ang. *normal mapping*)
  - Przykład - kolejno od lewej:
    - Cieniowanie **płaskie**
    - Cieniowanie **gładkie**
    - **Mapowanie wektorów normalnych** na podstawie ich mapy
    - Dodatkowo **modulacja koloru** na podstawie tekstury

Źródło obrazu:
CD Projekt RED

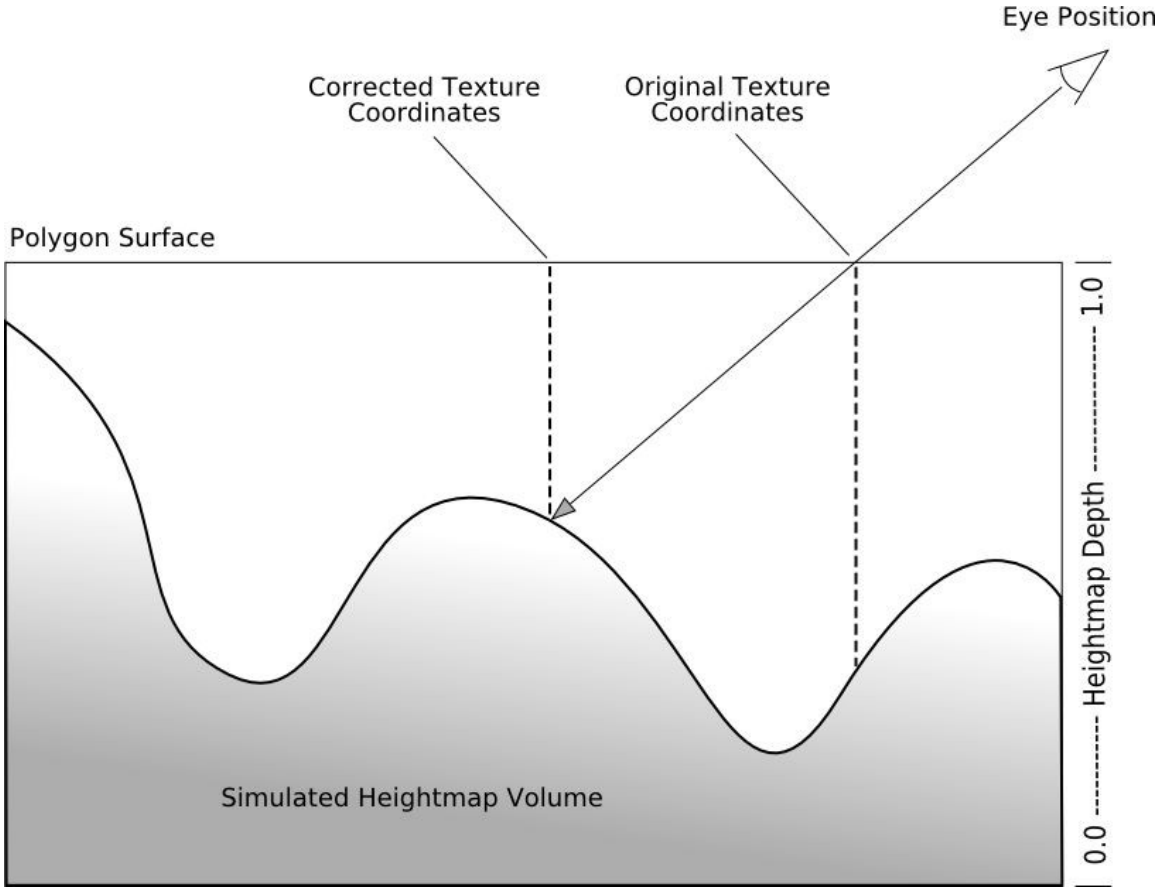
ZASTOSOWANIA TEKSTUROWANIA



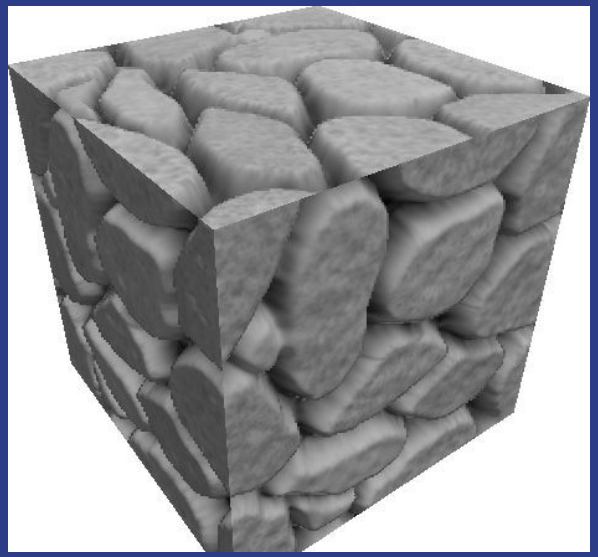
Lokalna mapa wysokości zawiera informacje o **głębokości każdego teksela** pod powierzchnią bryły.

Źródło obrazu:
GameDev.net, Jason Zink

- **Mapowanie paralaksy** (ang. *parallax mapping*)
  - Uwzględnienie **przesunięć tekseli** i ich wzajemnego **przysłaniania się** zależnie od położenia w jakim znajduje się obserwator
  - Opiera się na lokalnej **mapie wysokości** zawartej w teksturze



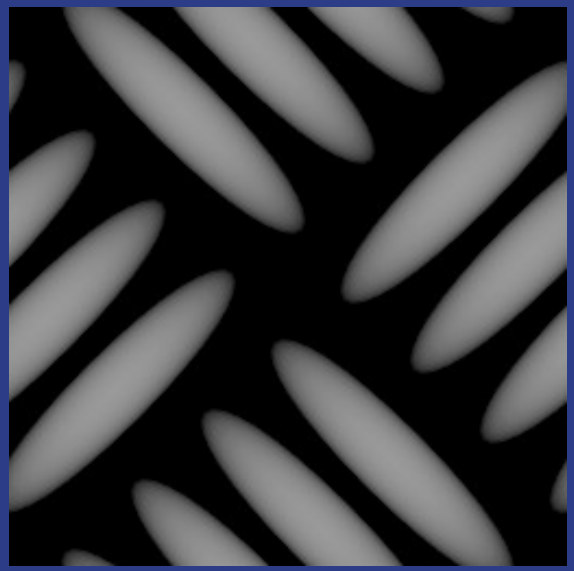
ZASTOSOWANIA TEKSTUROWANIA



Przy krawędziach oteksturowanej powierzchni wyraźnie widać, że pomimo paralaksy jest ona płaska.

Źródło obrazu:
AMD

ZASTOSOWANIA TEKSTUROWANIA

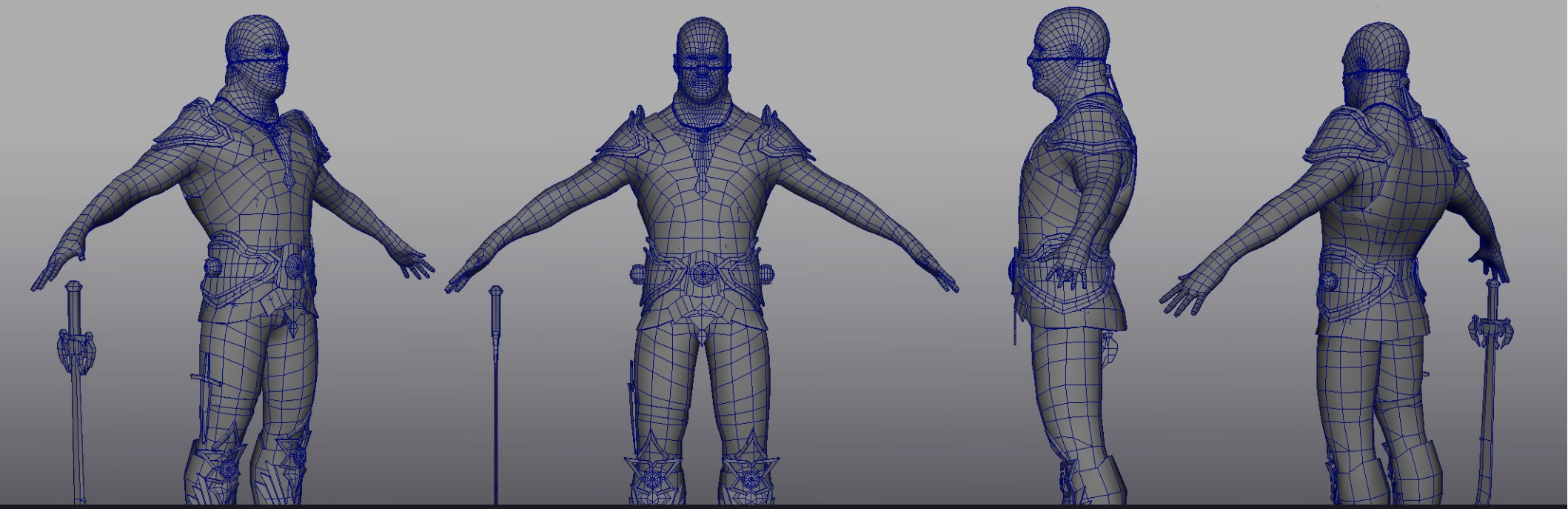


Wartości tekseli z zakresu od 0.0 do 1.0 są traktowane jako **mnożnik** komponentu *specular*. Zwykle wystarczy obraz jednokanałowy.

Źródło obrazu:
Masami Kuramoto

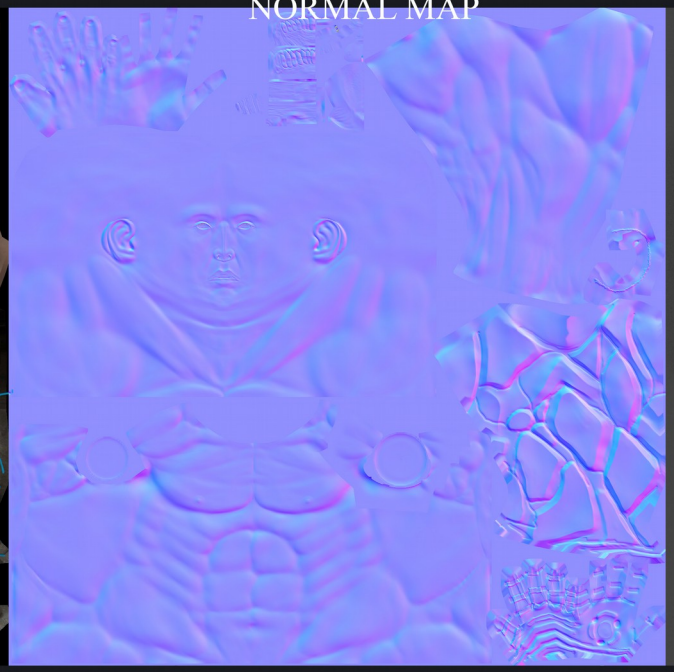
- **Mapowanie komponentu specular**
  (ang. *specular mapping*)
  - Pozwala modyfikować podatność powierzchni na powstanie odblasku na podstawie mapy wartości

Źródło obrazów:
Padalasuresh's Blog



COLOUR MAP

NORMAL MAP



SPECULAR MAP

BODY AND FACE TEXTURE MAPS

Zachodniopomorski
Uniwersytet
Technologiczny
w Szczecinie

http://bazyluk.net/dydaktyka

Bartosz Bazyluk

Teksturowanie

Mapy bitowe, pojęcie tekstury, potok, zastosowania.

Wydział
Informatyki

Grafika Komputerowa i Wizualizacja, Informatyka S1, II Rok