Podstawy Grafiki Komputerowej

Wykład 12: . . . ?

Andrzej Łukaszewski

Pracownia Grafiki Komputerowej
(LIGHT — Laboratory of Imaging and GrapHic Techniques)

Instytut Informatyki, Uniwersytet Wrocławski

7 stycznia 2020

Teksturowanie:

Modelowanie lokalnych zmian własności powierzchni:

1. kolor
2. własności odbicia
3. wektor normalny $\rightarrow$ bump mapping
4. zaburzenia kształtu $\rightarrow$ displacement mapping

Skomplikowana geometria czy prostsza geometria $+$ tekstura

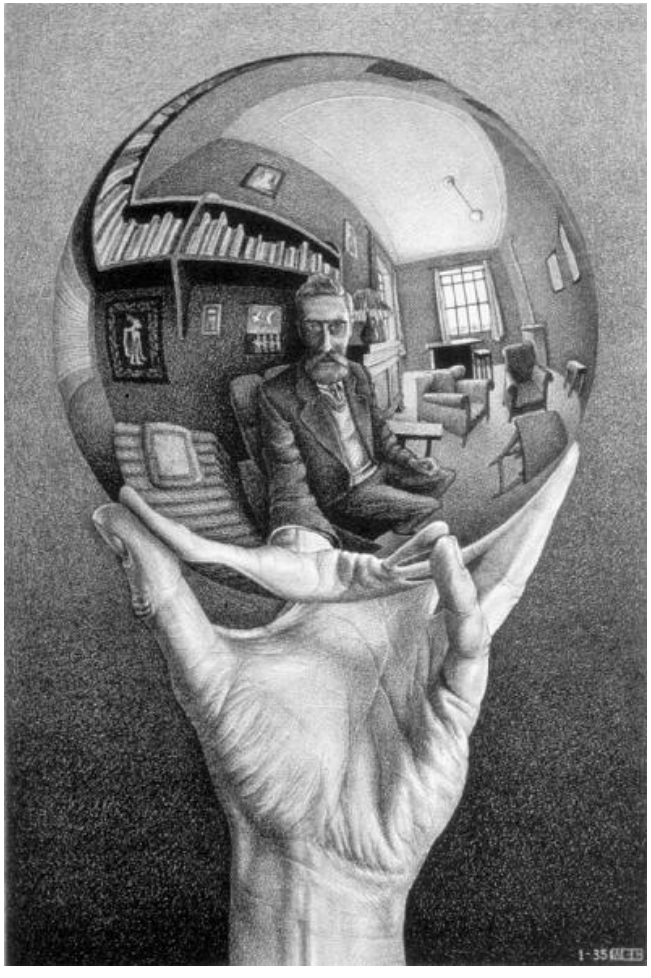
Źródło: wikipedia, BenTheWikiMen

Własności:

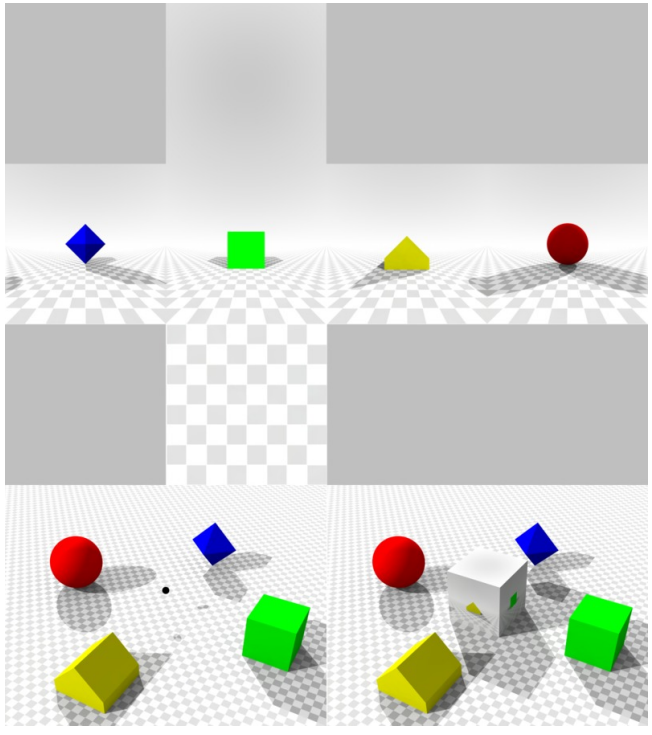
1. wymiar tekstury : 2D, 3D,..
2. parametryzacja: odwzorowanie między punktem obiektu, a punktami w przestrzeni tekstury
3. czy tekstura jest proceduralna czy też jest zaglądaniem do tablicy/bitmapy

Teksturowanie: mapy środowiska

Environment map



Źródło: M.C.Escher, 1935, wikipedia



Źródło: wikipedia, SharkD

Cube maps/mapy sześcienne

Skyboxes

Teksturowanie: 2D parametryzacja

Dla każdego wierzchołka/punktu powierzchni jeśli mamy dane współrzędne tekstury (u, v) to pobieramy wartość np. kolor z danej funkcji/bitmapy $T(u, v)$.

Często potrzebna też **funkcja odwrotna** która dla punktu 3D potrafi obliczyć współrzędne tekstury np. Ray Tracing.

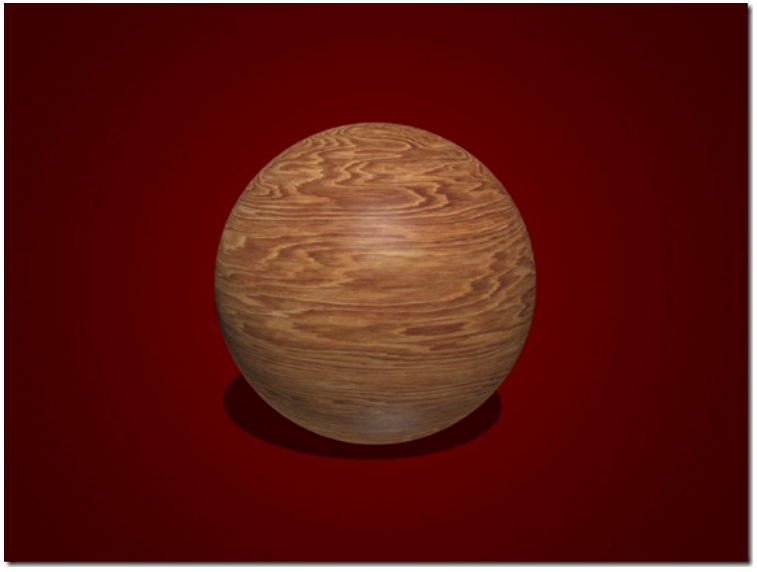
Problem z parametryzacja: jak zapewnić spójną parametryzację dla zbioru trójkątów, sześcianu, kuli,...

Dla kuli możemy zaproponować parametryzacje:

1. $(x, y, z) \rightarrow (u, v) = (x, y)$
2. współrzędne biegunowe, rzut na walec
3. inne rzuty na płaszczyznę

W żadnej nie unikniemy osobliwości. Pozostaje problem jaka jest naturalna parametryzacja, jak np. nałożyć teksturę drewna.

Teksturowanie: 2D czy 3D



Źródło: M.Hoffmann, http://www.tipsquirrel.com/working-with-3d-materials-in-photoshop-cs5/

3D:

Naturalność: kula z drewna ma kolory które biorą się z słoi 3D.

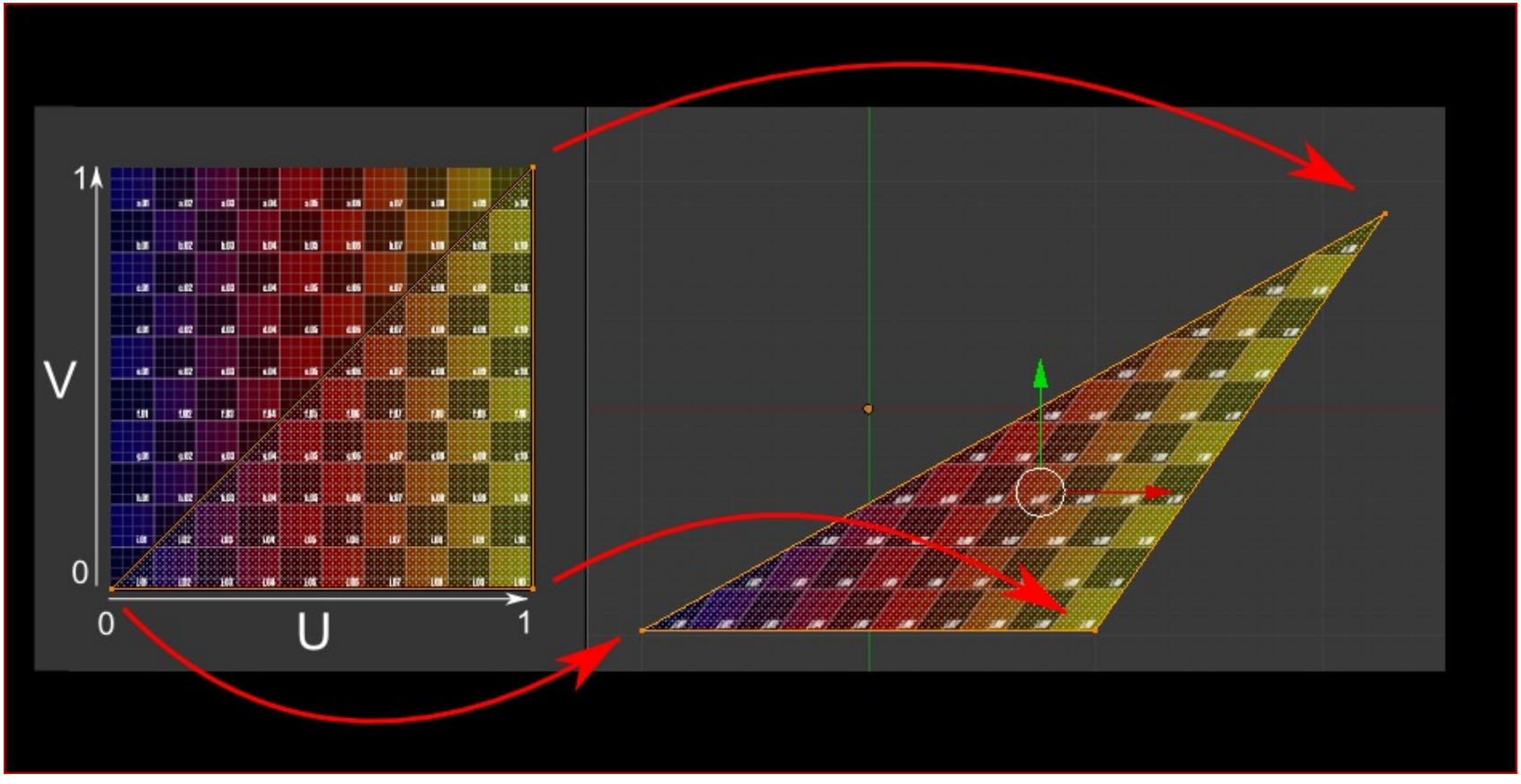
Tablice: ilość pamięci, pozyskiwanie danych wolumetr.

Proceduralnie: ...

Teksturowanie: 2D w OpenGL

OpenGL Tutorial:

http://www.opengl-tutorial.org/beginners-tutorials/tutorial-5-a-textured-cube



Texture Unit, Texture Binding

Teksturowanie: proceduralne 3D

Źródło: Ken Perlin, 1985

Reprezentacja proceduralna tekstury w postaci funkcji $T(x, y, z)$:

- jest ekstremalnie mała
- nie ma określonej rozdzielczości lecz dowolną dokładność
- ma nieograniczoną przestrzeń parametrów
- funkcję można sparametryzować umożliwiając zmiany

Literatura: D.S.Ebert, F.K.Musgrave et al. – "Texturing and Modeling, Third Edition: A Procedural Approach" Morgan Kaufmann, 2003.

Teksturowanie: przykłady ceglany mur

Przykład cegieł w języku RenderMan'a (Ebert et al. - "Texturing and Modelling", str.40–41)

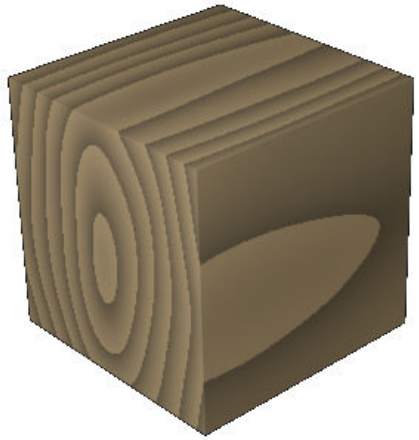
```
#define MWF (MORTARTHICKNESS*0.5/BMWIDTH)
#define MHF (MORTARTHICKNESS*0.5/BMHEIGHT)
surface brick(uniform float Ka = 1;   uniform float Kd = 1;
              uniform color Cbrick  = color (0.5, 0.15, 0.14);
              uniform color Cmortar = color (0.5, 0.5, 0.5);)     {
   color Ct;   point Nf;
   float ss, tt, sbrick, tbrick, w, h;
   float scoord = s;   float tcoord = t;
   Nf = normalize(faceforward(N, I));
   ss = scoord / BMWIDTH;   tt = tcoord / BMHEIGHT;

   if (mod(tt*0.5,1) > 0.5)
   ss += 0.5; /* shift alternate rows */
   sbrick = floor(ss); /* which brick? */
   tbrick = floor(tt); /* which brick? */
   ss -= sbrick;             tt -= tbrick;

   w = step(MWF,ss) - step(1-MWF,ss);
   th = step(MHF,tt) - step(1-MHF,tt);
   Ct = mix(Cmortar, Cbrick, w*h);
   /* diffuse reflection model */
   Oi = Os;
   Ci = Os * Ct * (Ka * ambient() + Kd * diffuse(Nf));
}
```

Teksturowanie: przykłady tekstury drewna

Kolor zależy od promienia: $r = \sqrt{x^2 + y^2}$
Jeśli:

$$\left(\frac{r}{w_d} - floor(\frac{r}{w_d}) \right) \leq 0.5$$

to zwracamy kolor jasnego słoja w p.p. ciemny kolor



Źródło: wikipedia, Falstaff

Warto zaburzyć regularność:

1. zastępując r przez: $r + c \cdot cos(k\alpha)$ gdzie $\alpha = arccos \frac{x}{\sqrt{x^2+y^2}}$
2. zmiany względem z zastępując r przez: $r + c \cdot cos(k\alpha + m \cdot z)$
3. zamiast regularnych funkcji trygonometrycznych lepiej użyć losowego szumu o ściśle określonych częstotliwościach

Przykład lepszego drewna: http://lodev.org/cgtutor/randomnoise.html#Wood

Teksturowanie: generowanie szumu

Biały szum (np. TV) jest zły bo zawiera wszelakie częstotliwości i sąsiednie punkty mają często zupełnie inne wartości.

Funkcja `Noise()` **własności wymagane:**

- powtarzalna funkcja pseudolosowa
- określony zakres wartości np. $[-1, 1]$
- ograniczone widmo częstotliwości z max.= 1
- nie zawiera oczywistych okresowości i regularnych wzorów
- stacjonarność (statystyczna niezmienniczość na translacje)
- anizotropowość (statystyczna niezmienniczość na rotacje)

Teksturowanie: szum kratowy

Szum kratowy: generujemy pewne losowe wartości na kracie, a pomiędzy nimi stosujemy jakąś interpolację/aproksymację.

Szum generowany przez wartości: Interpolacja przez wartości z punktów kratowych np. funkcje sklejane Catmull-Rom'a potrzebne 4 wartości w każdym kierunku, duży koszt (64 wartości). Widmo jest słabo ograniczone od dołu.

Szum generowany przez gradienty: w punktach kratowych ustalamy wartości $Noise = 0$, generujemy zaś losowe gradienty. Wartości pomiędzy obliczamy na podstawie wartości z rogów sześcianu (8 gradientów).
Zera w punktach kratowych ograniczają widmo częstotliwości.

Teksturowanie: szum Perlina

$$Noise(x,y,z) = \sum_{i=[x]}^{[x]+1} \sum_{j=[y]}^{[y]+1} \sum_{k=[z]}^{[z]+1} \Omega_{ijk}(x-i, y-j, z-k)$$

gdzie

$$\Omega_{ijk}(u,v,w) = \omega(u)\omega(v)\omega(w) < \Gamma_{ijk}, (u,v,w) >$$

gdzie

$$\omega(t) = 2|t|^3 - 3|t|^2 + 1 \; dla \; t \in [-1,1] \; i \; 0 \; poza$$

Generowanie wektorów gradientu. Perlin: wystarczy n=256 w jednej tablicy i używanie ich w sposób pseudolosowy.

$$\Gamma_{ijk} = G(R(i + R(j + R(k))))$$

G to tablica gradientów, P jest permutacją i $R(i) = P(i \mod n)$

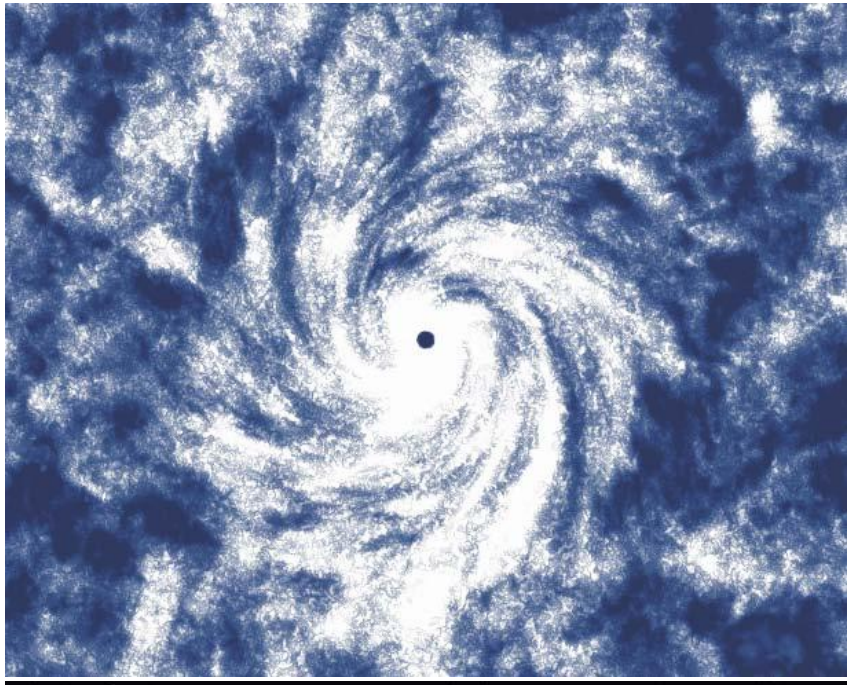
Jeśli mamy 2 kopie tablicy G to można pominąć operacje mod .

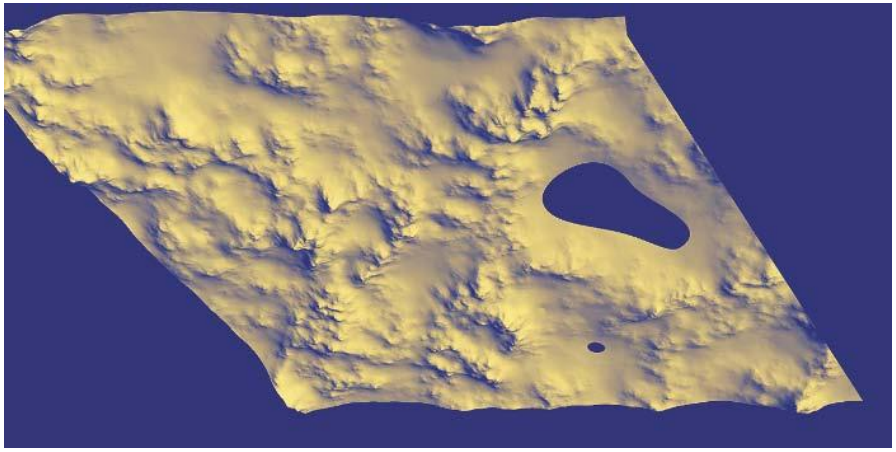
Teksturowanie: szum Perlina w GLSL

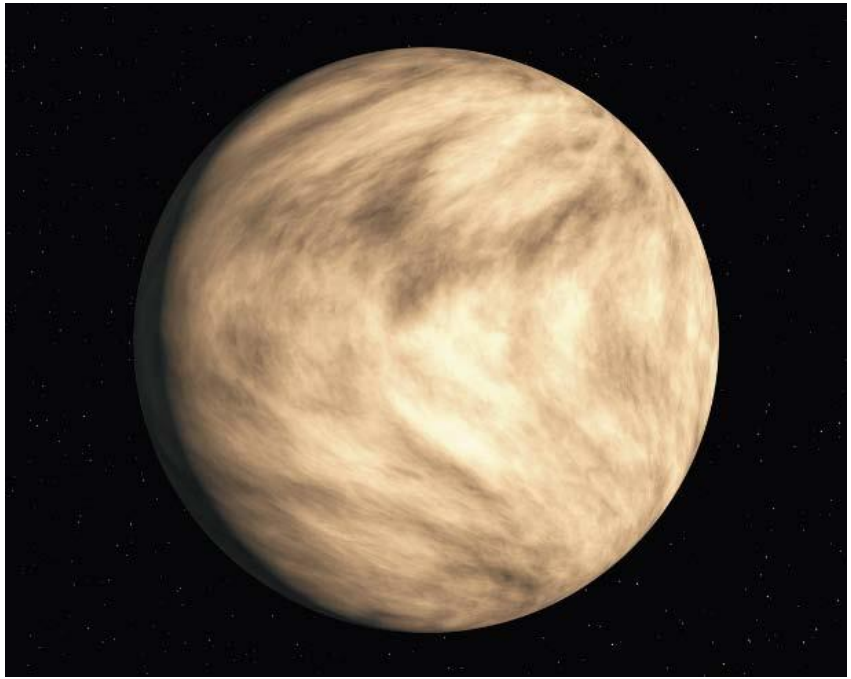
- 1985: Ken Perlin – “An Image Synthesizer” SIGGRAPH.
- 1997: Ken Perlin – Academy Award for Technical Achievement
- 2001: Ken Perlin – simplex noise patent
- Szum w shaderach z kodem: Ashima Arts, Stefan Gustavson: https://github.com/ashima/webgl-noise

Przykład z szumem na sferze

Lit.: Ebert et. al – “Texturing and Modelling”







Teksturowanie: suma fraktalna i turbulencja

Przykłady funkcji sumy fraktalnej i turbulencji

Suma fraktalna:

$$fractalsum(P) = \sum_{f=MIN,2MIN,4MIN,..MAX} \frac{1}{f} Noise(f \cdot P)$$

Turbulencja:

$$turbulence(P) = \sum_{f=MIN,2MIN,4MIN,..MAX} \frac{1}{f} |Noise(f \cdot P)|$$

(dobrze odpowiada teksturze marmuru, ostre nieciągłości pochodnej)