

Wprowadzenie do grafiki komputerowej

Mapowanie tekstur (wersja Three.js/WebGL)

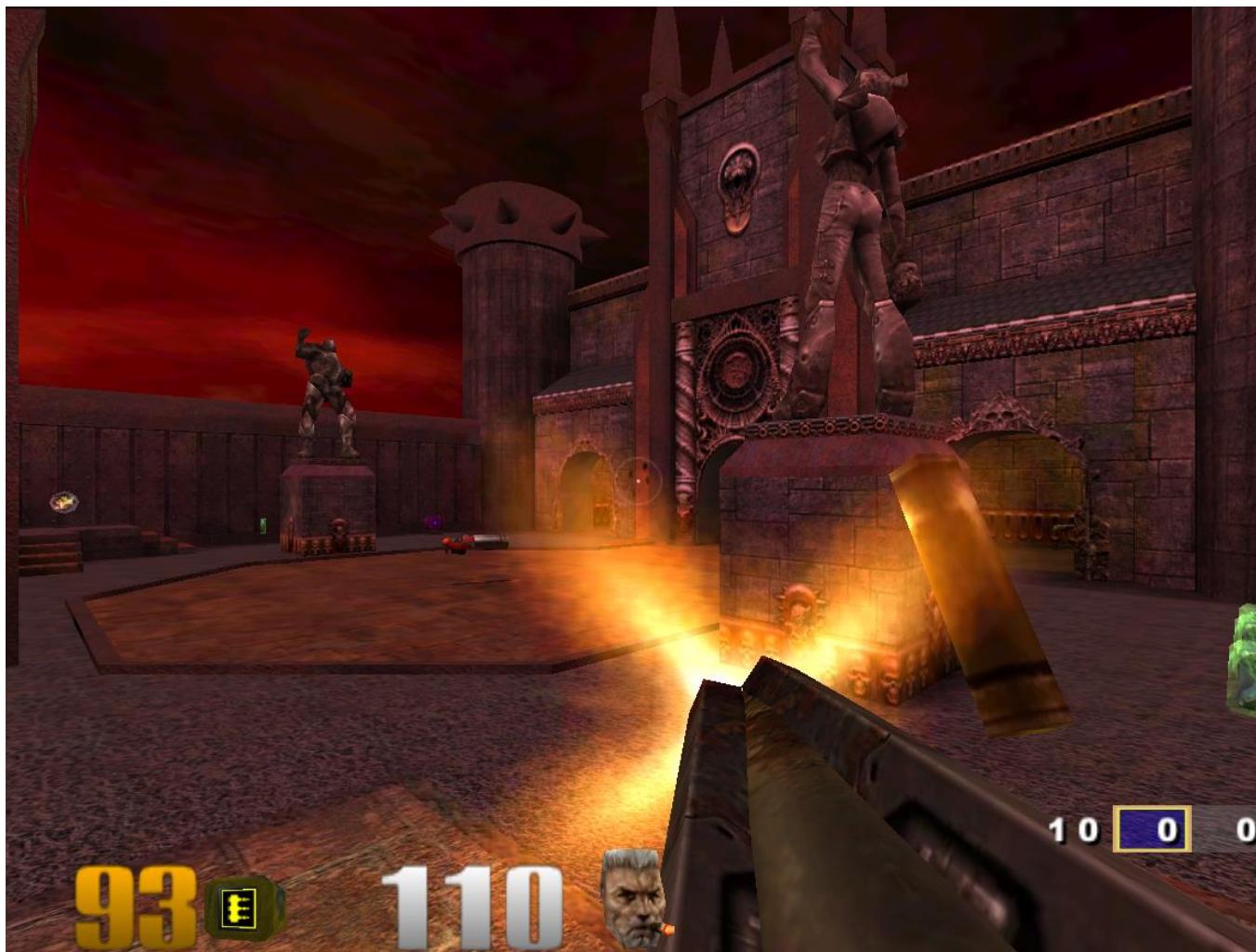
Co to jest tekstura?

- ▶ Rzut oka np. do wikipedii dostarcza nam różnych znaczeń, niekoniecznie informatycznych
- ▶ Nas interesuje *texture mapping*: nakładanie na powierzchnię obiektu:
 - ▶ Wzoru w postaci mapy bitowej lub obrazu rastrowego
 - ▶ Koloru
 - ▶ Wygenerowaną grafikę
 - ▶ Modele 3D

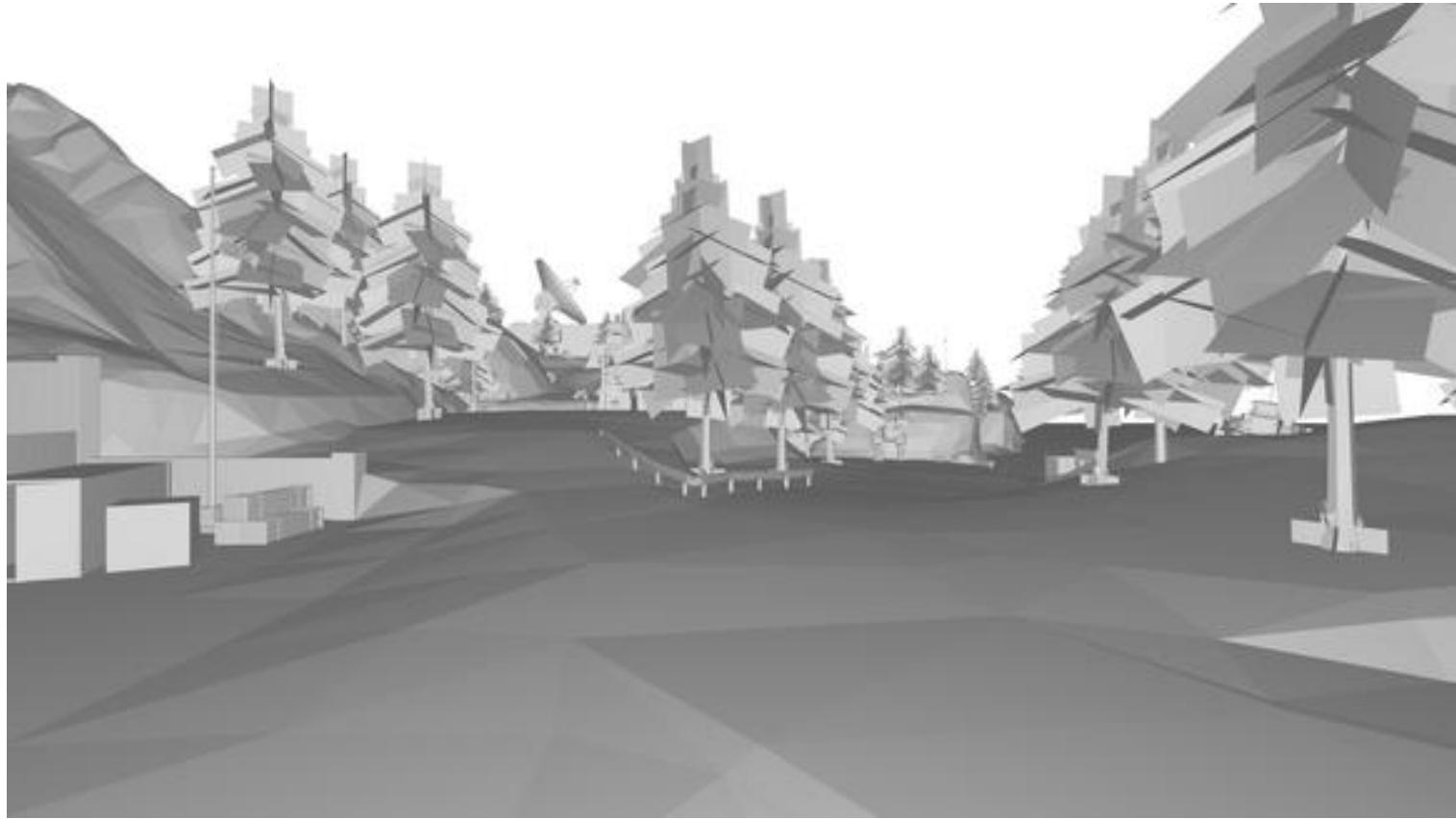
Scena z Quake 3 bez tekstur



Scena z Quake 3 z teksturami



Scena z Quake Wars bez tekstur



Scena z Quake Wars z teksturami i oświetleniem



mapowanie tekstur, teksturowanie

- ▶ Nakładanie na powierzchnię mapy parametrów charakteryzujących jej własności optyczne: kolor, sposób odbijania światła, stopień przezroczystości, współczynnik załamania światła i inne.
- ▶ Źródłem tekstuury może
 - ▶ mapa pikselowa reprezentująca najczęściej zdjęcie rzeczywistych przedmiotów (tekstura bitmapowa)
 - ▶ procedura generująca układ pikseli według wzoru
- ▶ Uwaga: ponieważ nazwa piksel jest elementem ekranu, odpowiadający mu element tekstuury nazywamy tekselem.



mapowanie tekstur, teksturowanie

- ▶ Prawie wszystkie obiekty są teksturowane
 - ▶ Słoje na drewnie, cegły, faktura tkaniny... etc.
 - ▶ Tekstury wzbogacają scenę o szczegóły



Model oparty na wielokątach

... i na teksturach

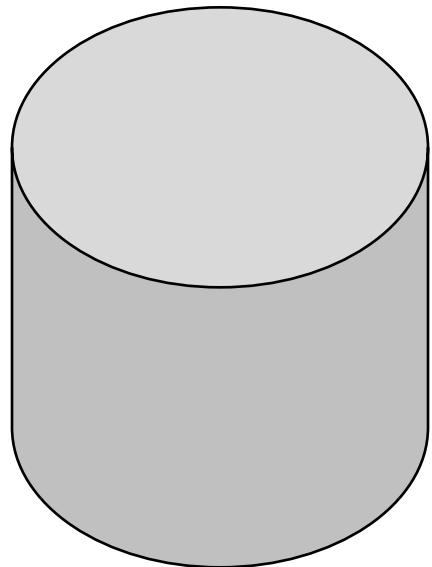


Mapowanie tekstur

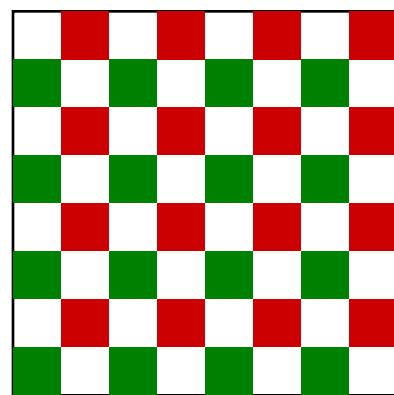
- ▶ **Texture mapping:**
Obrazy rasterowe nakładane na wielokąty.
- ▶ Mamy oddzielny bufor (*texture buffer*) na karcie graficznej, który przechowuje tekstury
 - ▶ Elementy tekstur nazywają się *teksele*
 - ▶ Tekstury mogą być 1-, 2-, 3- (lub 4-!) wymiarowe (mapowane na 1-, 2-, lub 3-wymiarowe powierzchnie)



Jak mapować tekstury?



+



=



geometria

obraz

mapowana tekstura

- *Pytanie: W jaki sposób decydujemy gdzie w geometrii powinien się znaleźć kolor z obrazu?*



Ogólne podejście do teksturowania

Przestrzeń tekstury **2D (u,v) lub 3D (u,v,w)**

2D (s,t) lub 3D (s,t,r)



Przestrzeń obiektu **3D (x,y,z)**

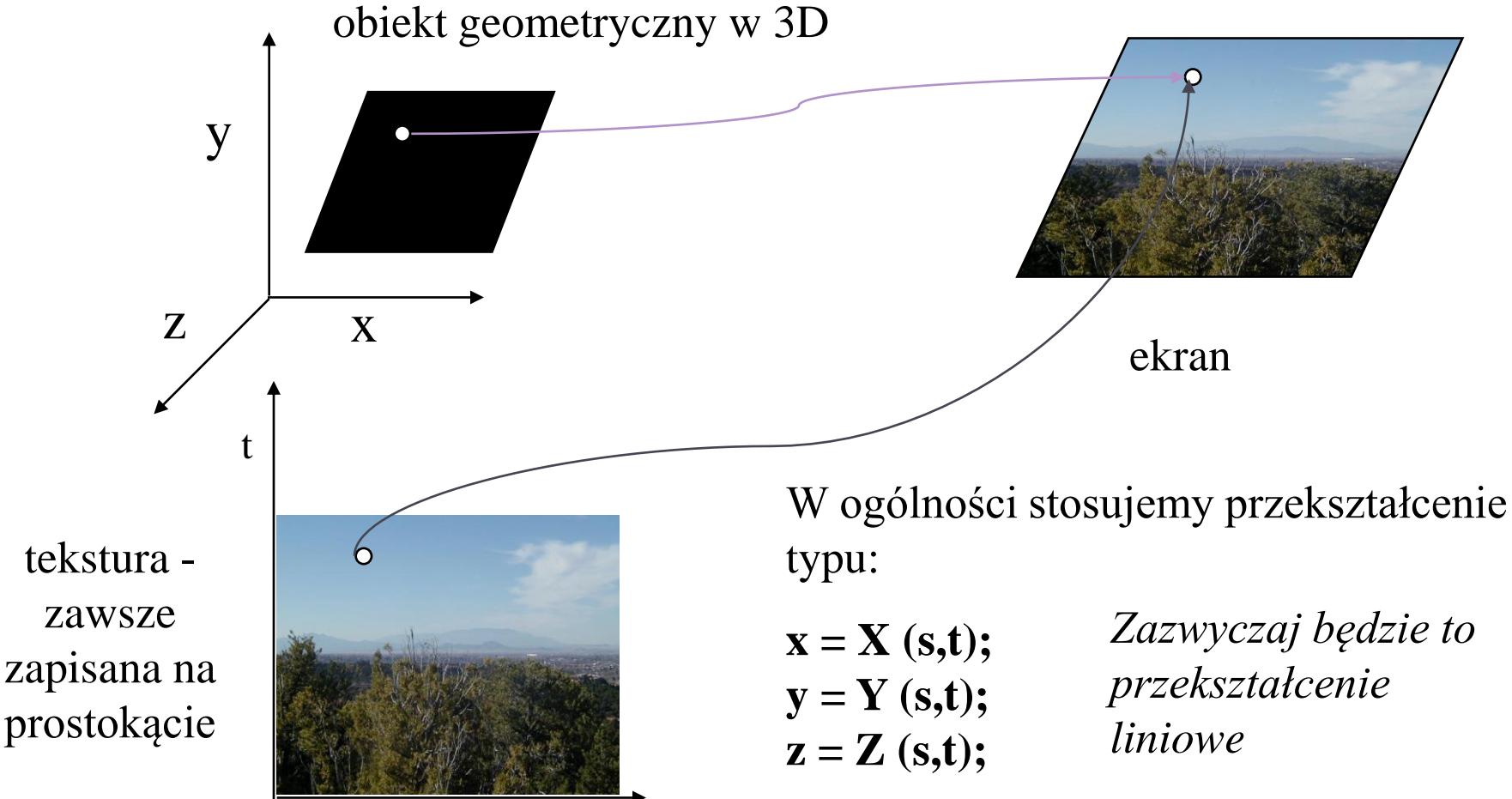


Przestrzeń ekranu **2D (x,y)**

Kluczowe problemy:

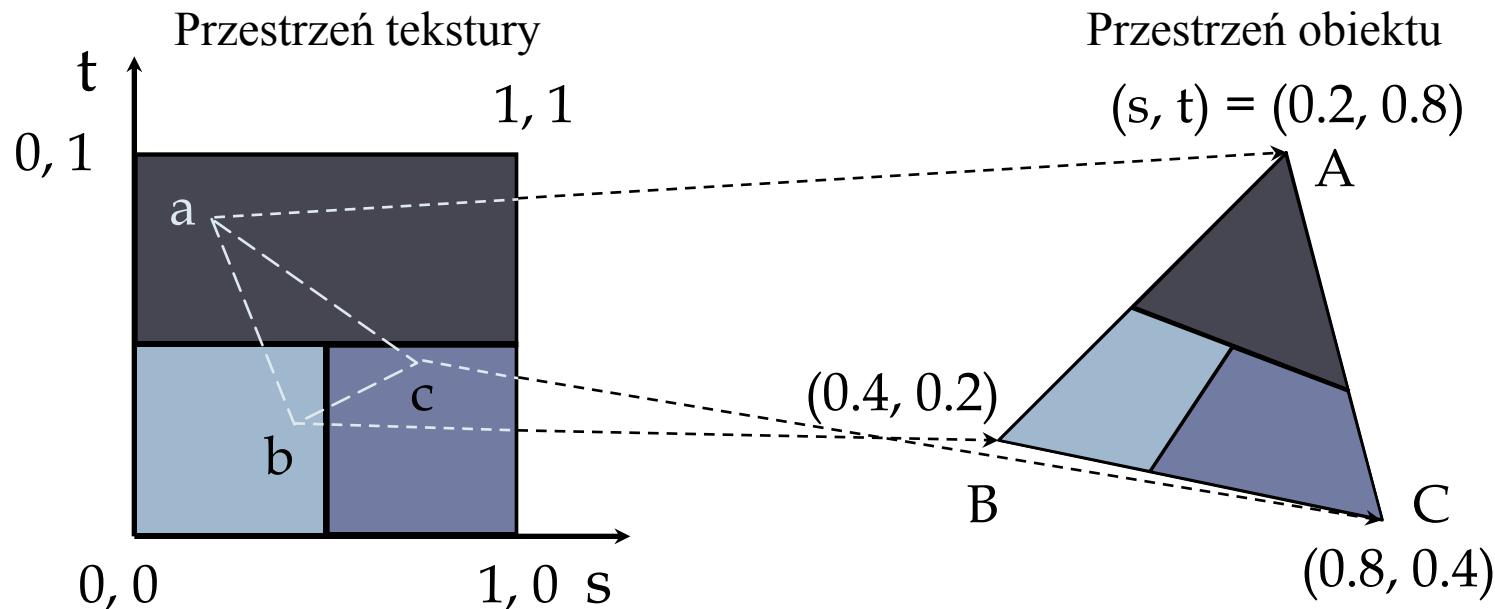
- jak mapować teksturę 2d na obiekt 3d
- jak przygotować obraz na teksturowe, żeby dobrze wyglądał po zmapowaniu.

Prosty przypadek: płaskie mapowanie tekstur 2D na wielokąt

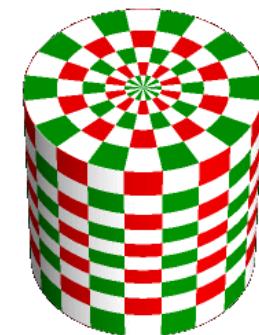
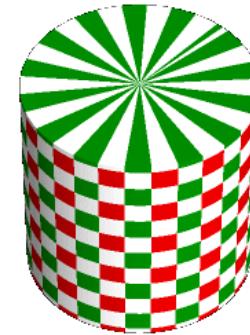
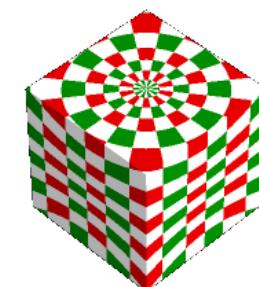
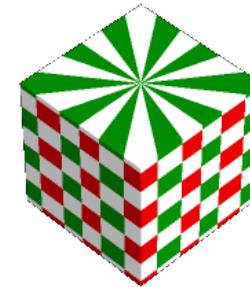
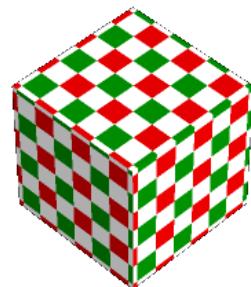
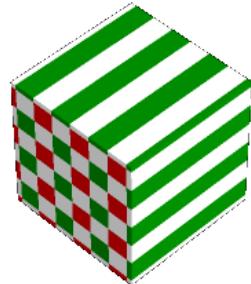
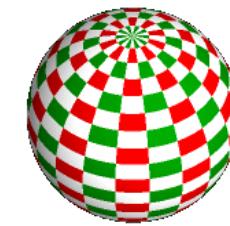
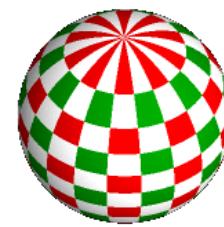
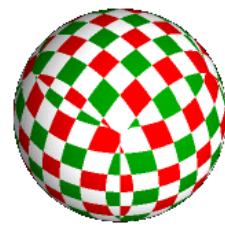


Jak działają współrzędne tekstury?

- ▶ Przestrzeń tekstury 2D jest określona parametrycznie, zmiennymi (s, t) z przedziału $[0, 1]$ (zazwyczaj).



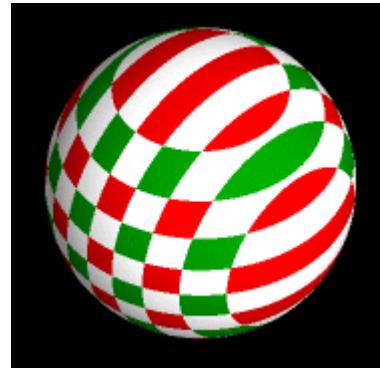
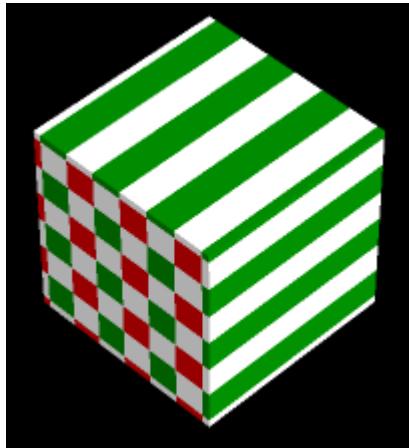
Mnogość rzutowań (pierwsza z możliwości)



[Paul Bourke]

Jak mapować płaskie tekstury na różne powierzchnie?

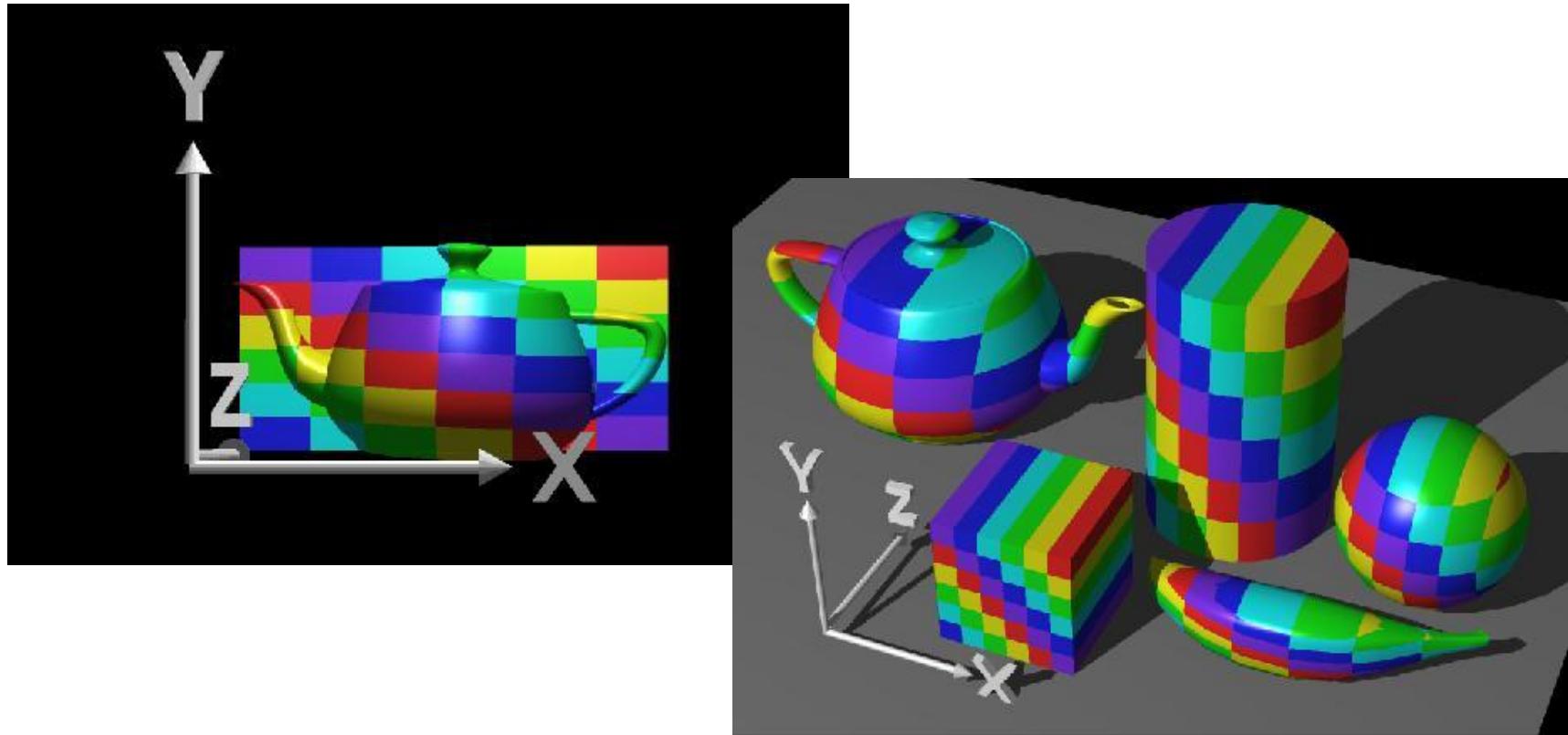
- ▶ Nie ma oczywiście jednej odpowiedzi:
 - ▶ Na początek: Planar (mapowanie płaskie)



- ▶ http://paulbourke.net/texture_colour/tiling/

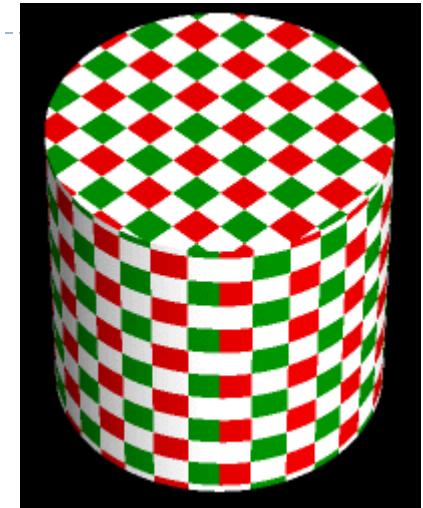
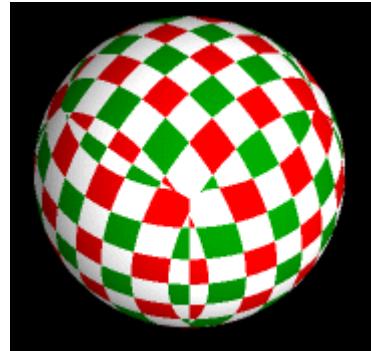
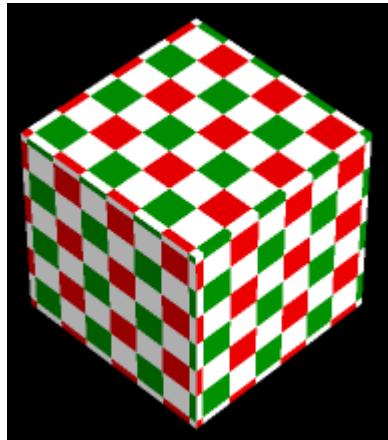
Planar mapping

- ▶ Jak w projekcji: zerujemy współrzędną $z(s,t) = (x,y)$
- ▶ Problemy: co się dzieje w pobliżu $z = 0$?

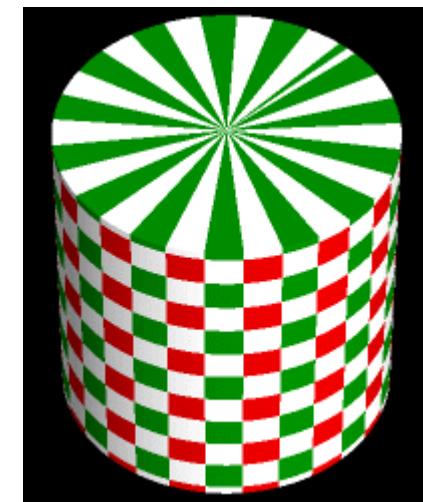
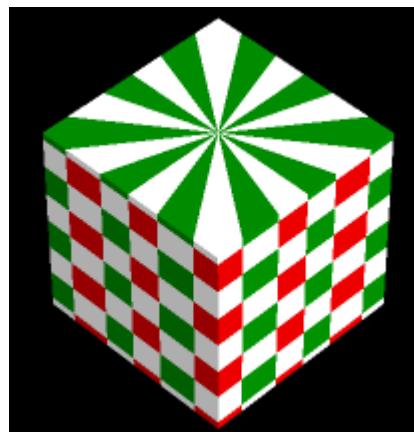


Mapowanie sześcienne i cylindryczne

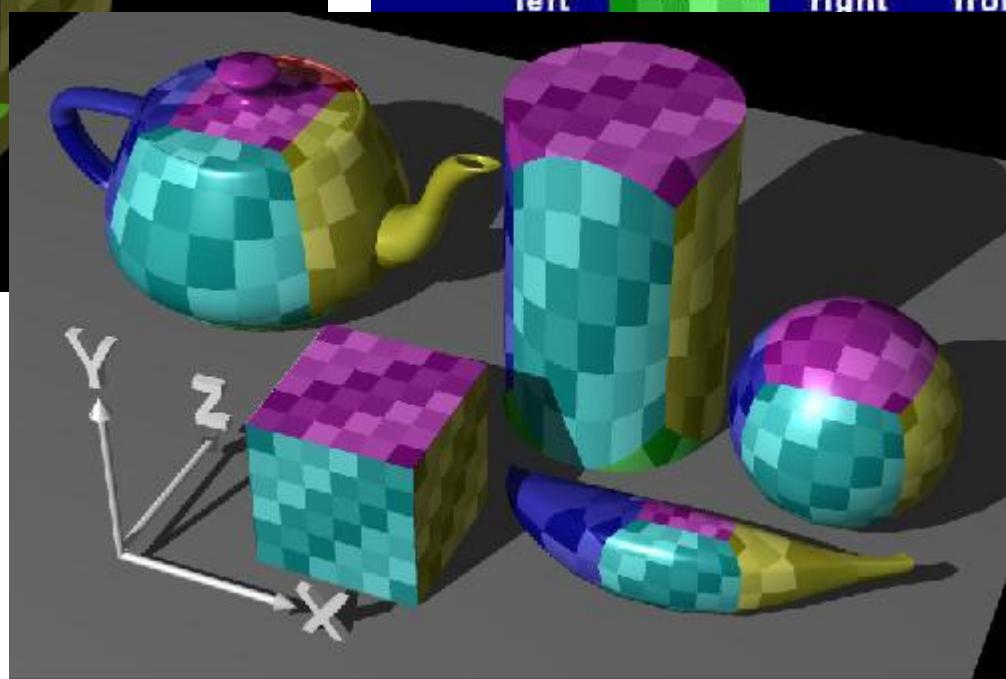
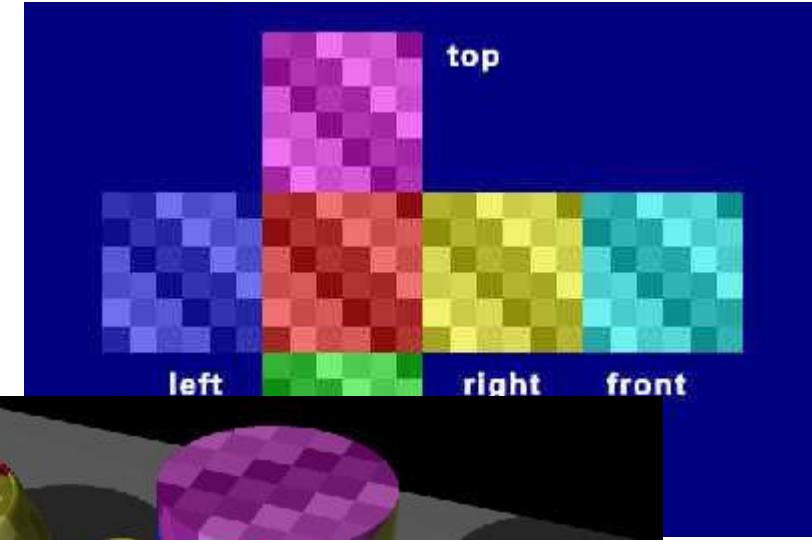
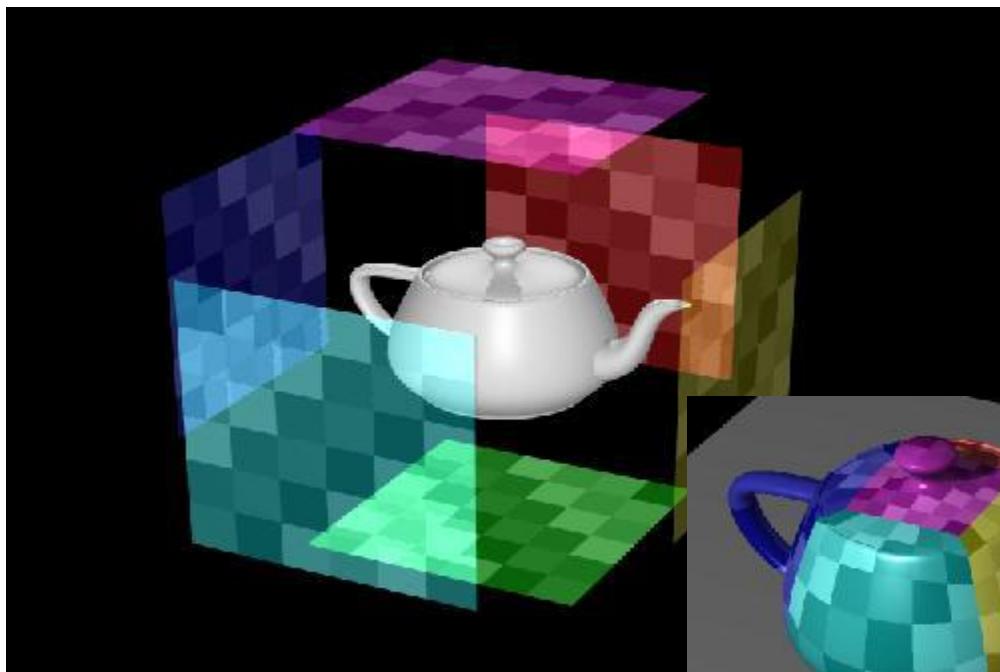
- ▶ Cubic



- ▶ Cylindrical (mapowanie cylindryczne)

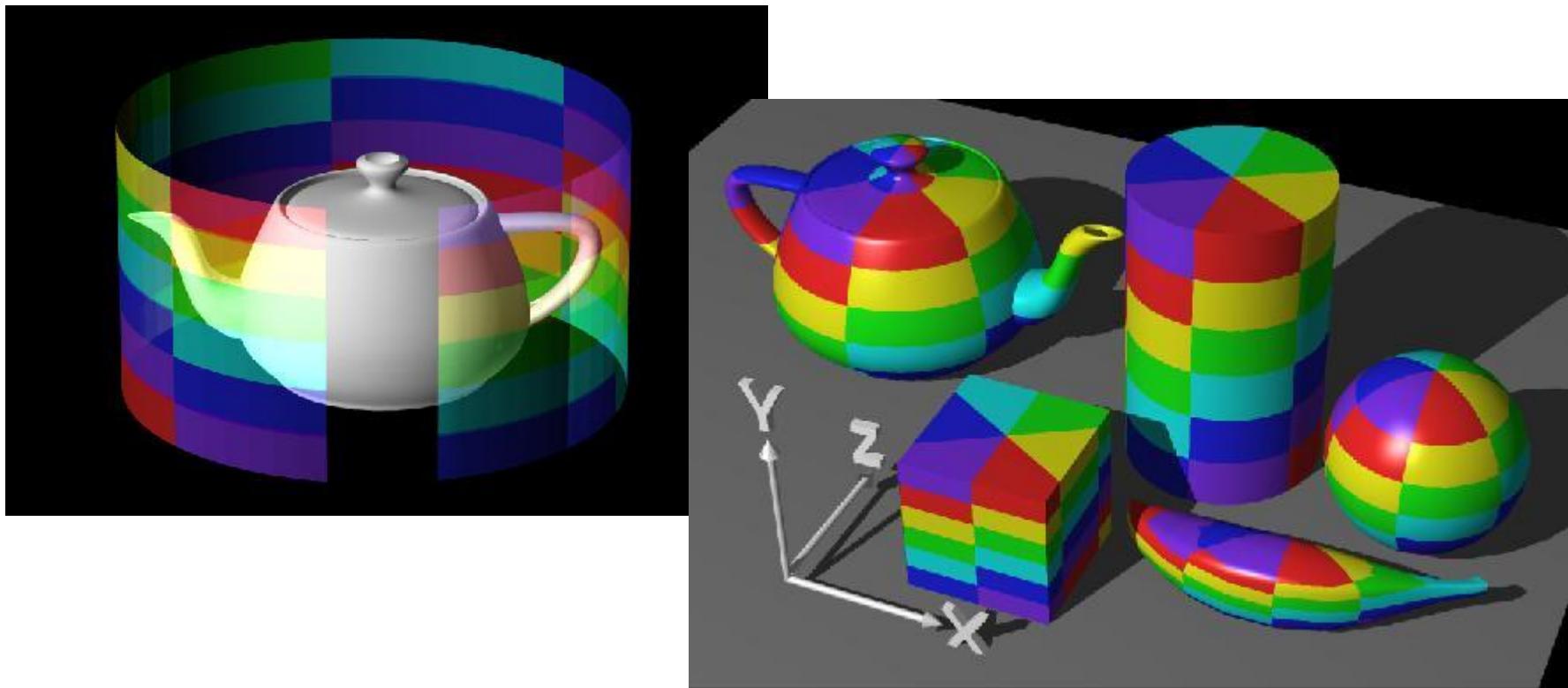


Cube Mapping



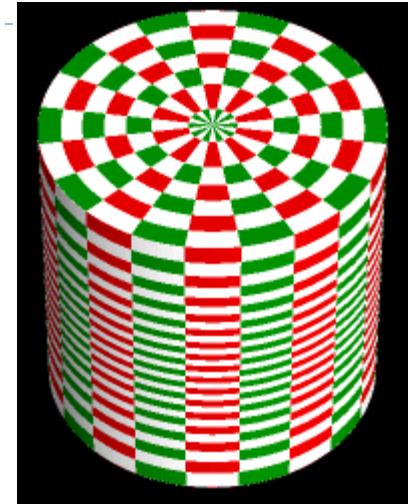
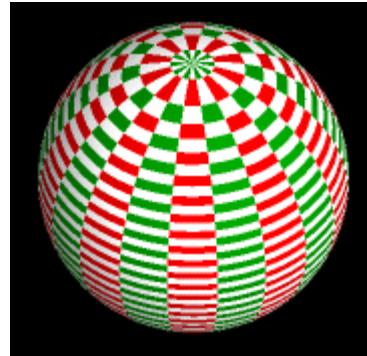
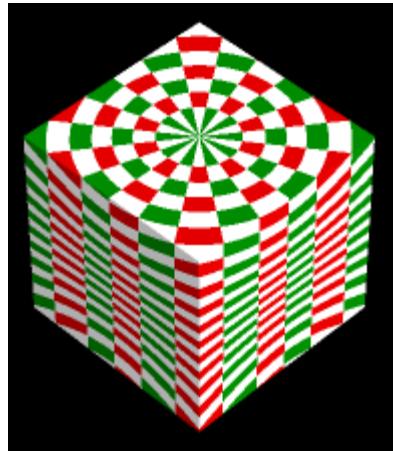
Cylindrical Mapping

- ▶ Cylinder: r, θ, z , $(s,t) = (\theta/(2\pi), z)$
- ▶ Problem gdy owijamy wokół ($\theta = 0$ or 2π)

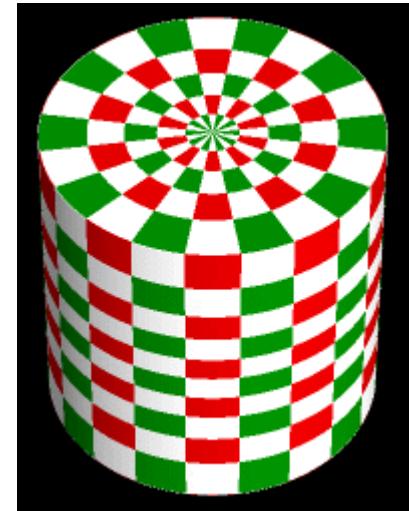
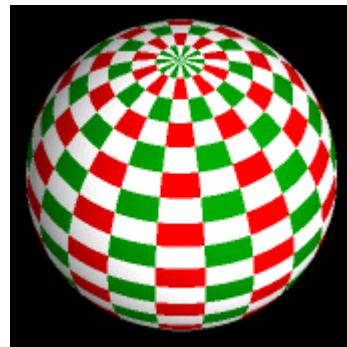
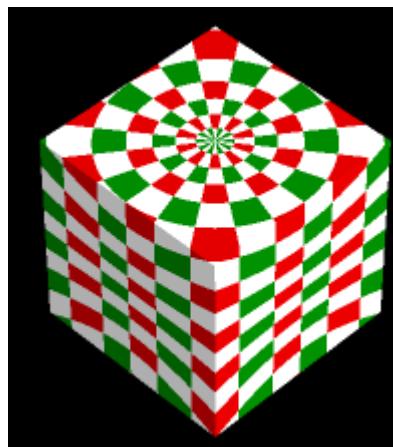


Mapowanie prostokątne cylindryczne i sferyczne

- ▶ Rectangular cylindrical

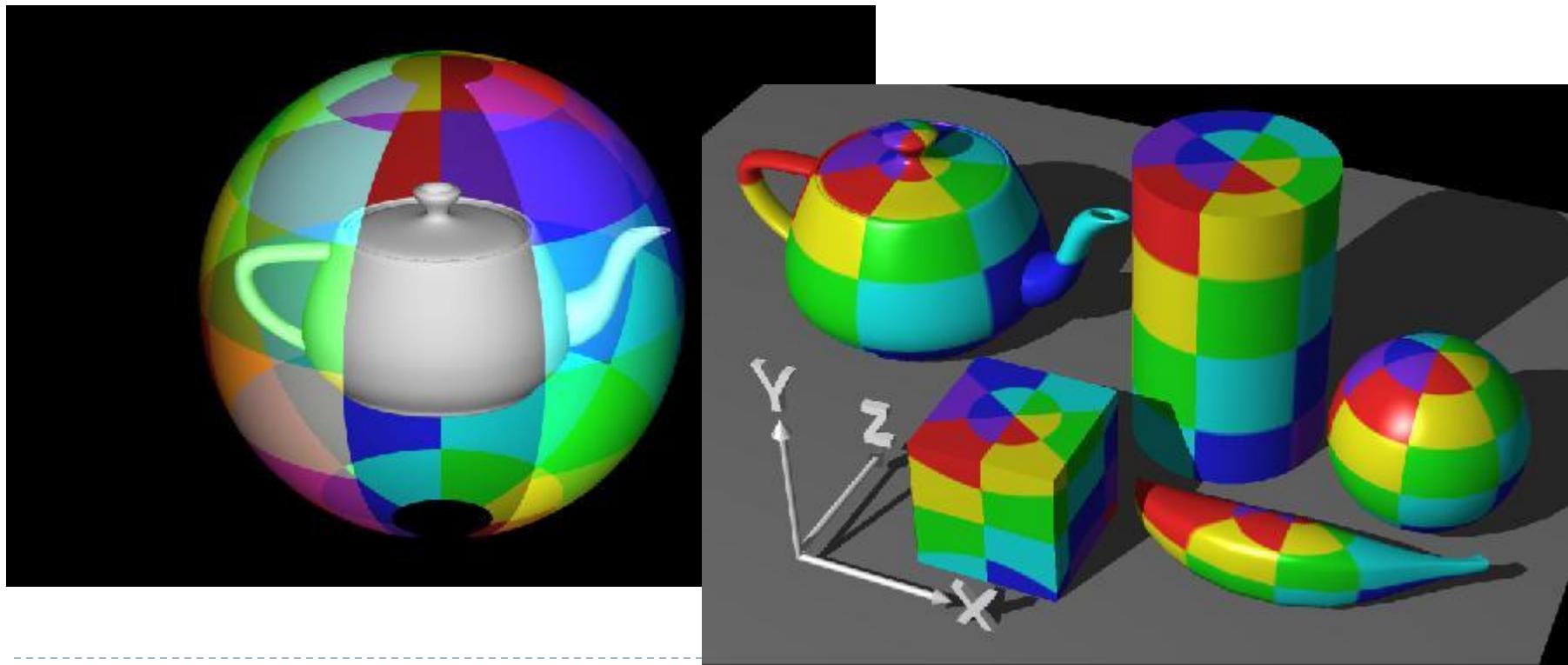


- ▶ Spherical



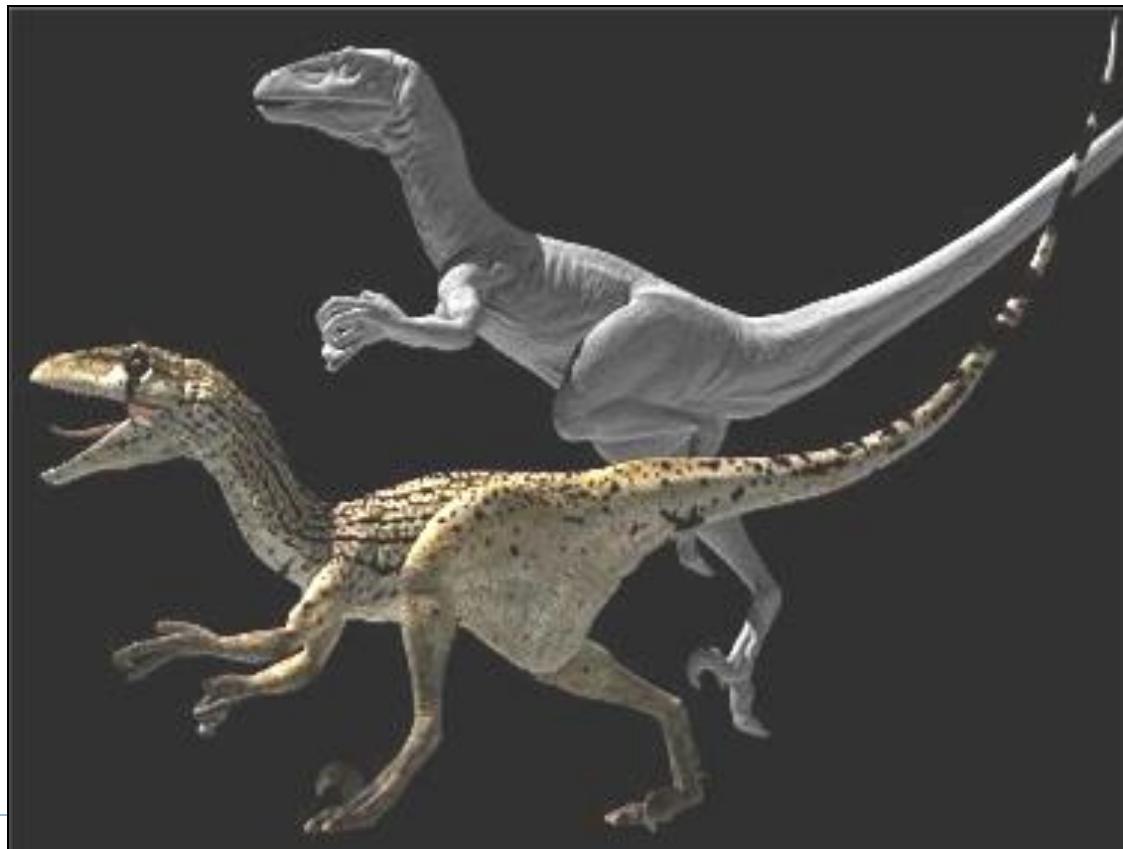
Spherical Mapping

- ▶ Używamy współrzędnych sferycznych (szerokość i wysokość geograficzna w stopniach)
 - ▶ Osobliwości wokół biegunów



Wzbogacanie o szczegóły

- ▶ Zasadniczy pomysł: do przedstawiania szczegółów należy użyć obrazów zamiast gęstej siatki wielokątów.

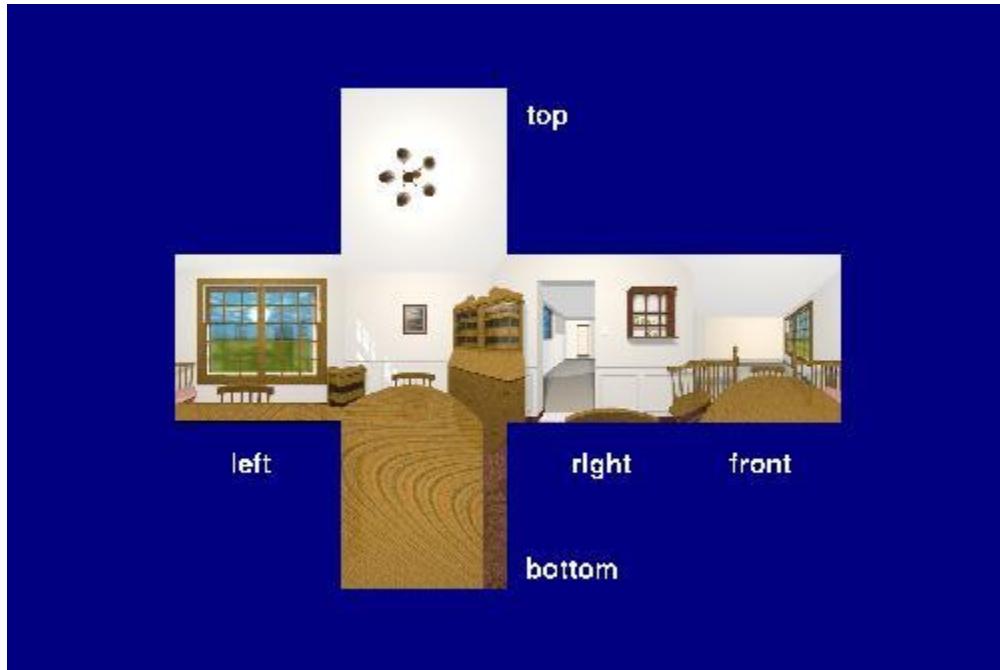


Współrzędne mapowania tekstuury

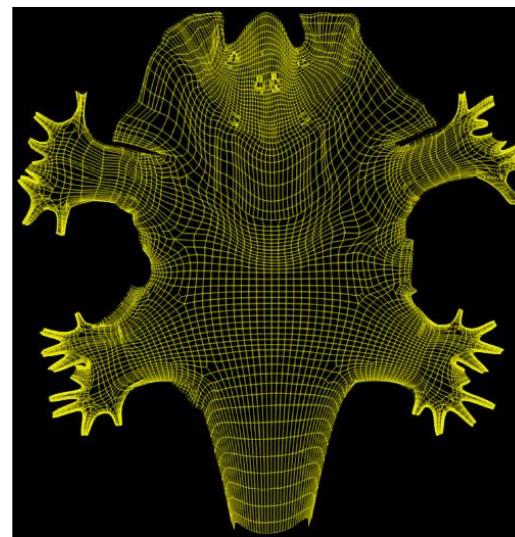
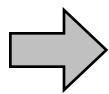
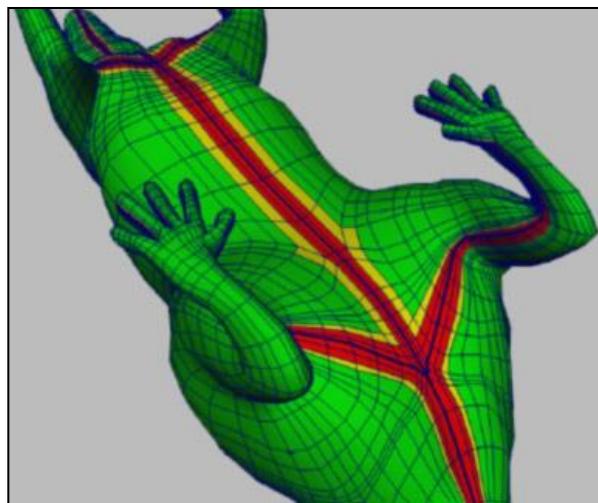
- ▶ Czasem mapowanie jest proste (prostokąt na prostokąt)
- ▶ Można użyć standardowych procedur (np. prostokąt na sferę)
- ▶ W innych przypadkach zwykle ręcznie
 - ▶ Automatycznie rozwijamy siatkę obiektu na płaszczyznę
 - ▶ Ręcznie malujemy mapę



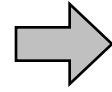
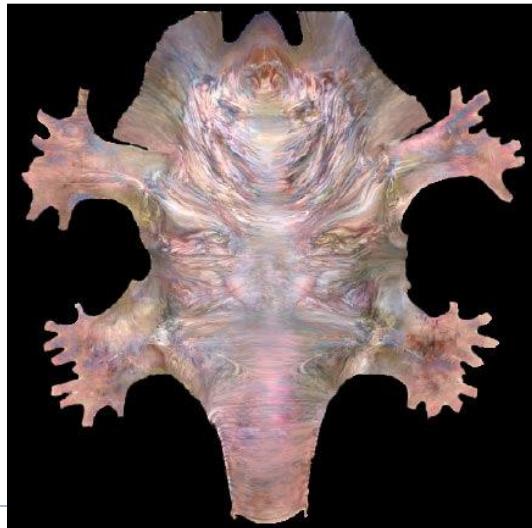
Cube Mapping



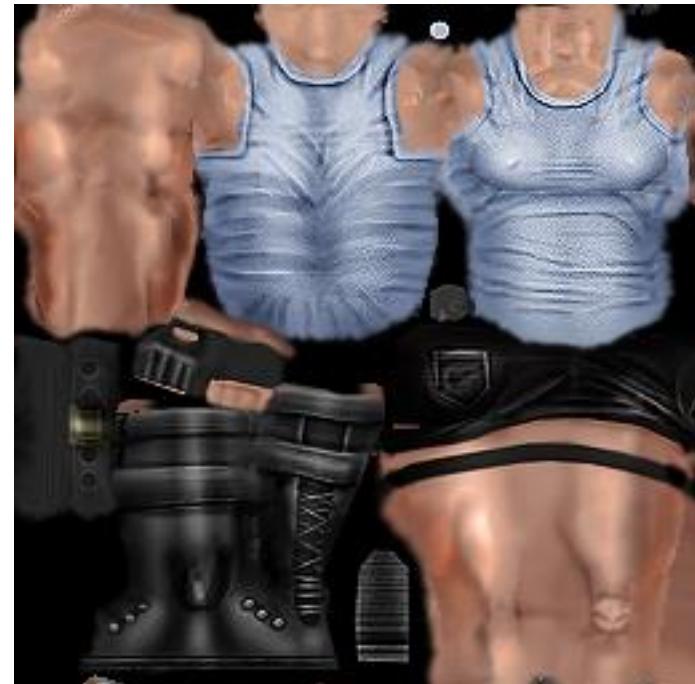
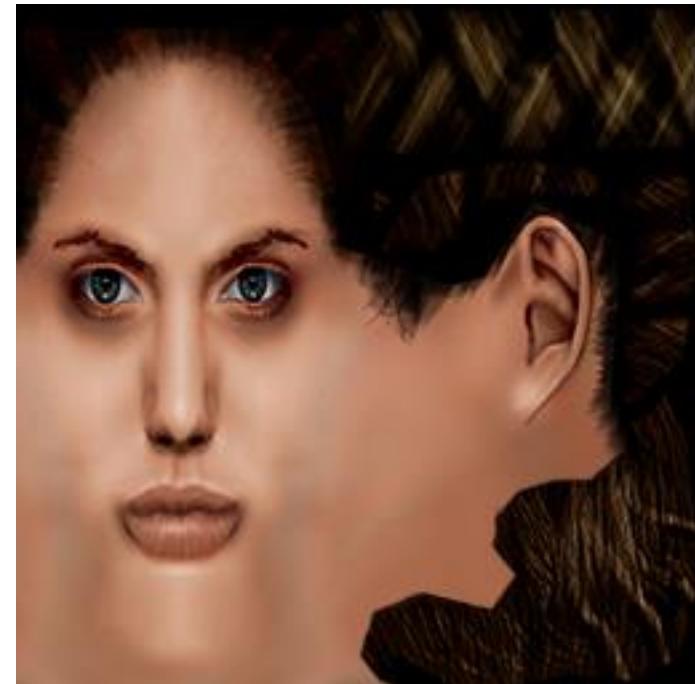
Rozwijanie powierzchni (druga możliwość)



[Piponi 2000]



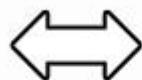
Kolejna możliwość – niech się artysta tym zajmie



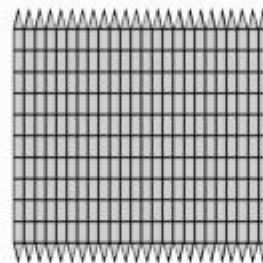
UV mapping – trzy etapy

- ▶ Mapowanie obrazu 2D na powierzchnię obiektu 3D

3-D Model



UV Map



$$p = (x, y, z)$$



1. Rozwinięcie siatki modelu

2. Nałożenie tekstuury na siatkę

Texture



3. Zwinięcie siatki z tekstyurą

<http://upload.wikimedia.org/wikipedia/commons/0/04/UVMapping.png>

Dygresja: Mapowanie na sfére (spokojnie można pominąć)

Załóżmy: α jest kątem od osi X ($0 \leq \alpha \leq 2\pi$)
 φ jest kątem od osi Z ($0 \leq \varphi \leq \pi$)

Równanie sfery wygląda:

$$X = R \sin(\varphi) * \cos(\alpha) = R \sin(\pi t) * \cos(2\pi s), \quad \varphi/\pi = t \quad (0.0 \leq t \leq 1.0)$$

$$Y = R \sin(\varphi) * \sin(\alpha) = R \sin(\pi t) * \sin(2\pi s), \quad \alpha/2\pi = s \quad (0.0 \leq s \leq 1.0))$$

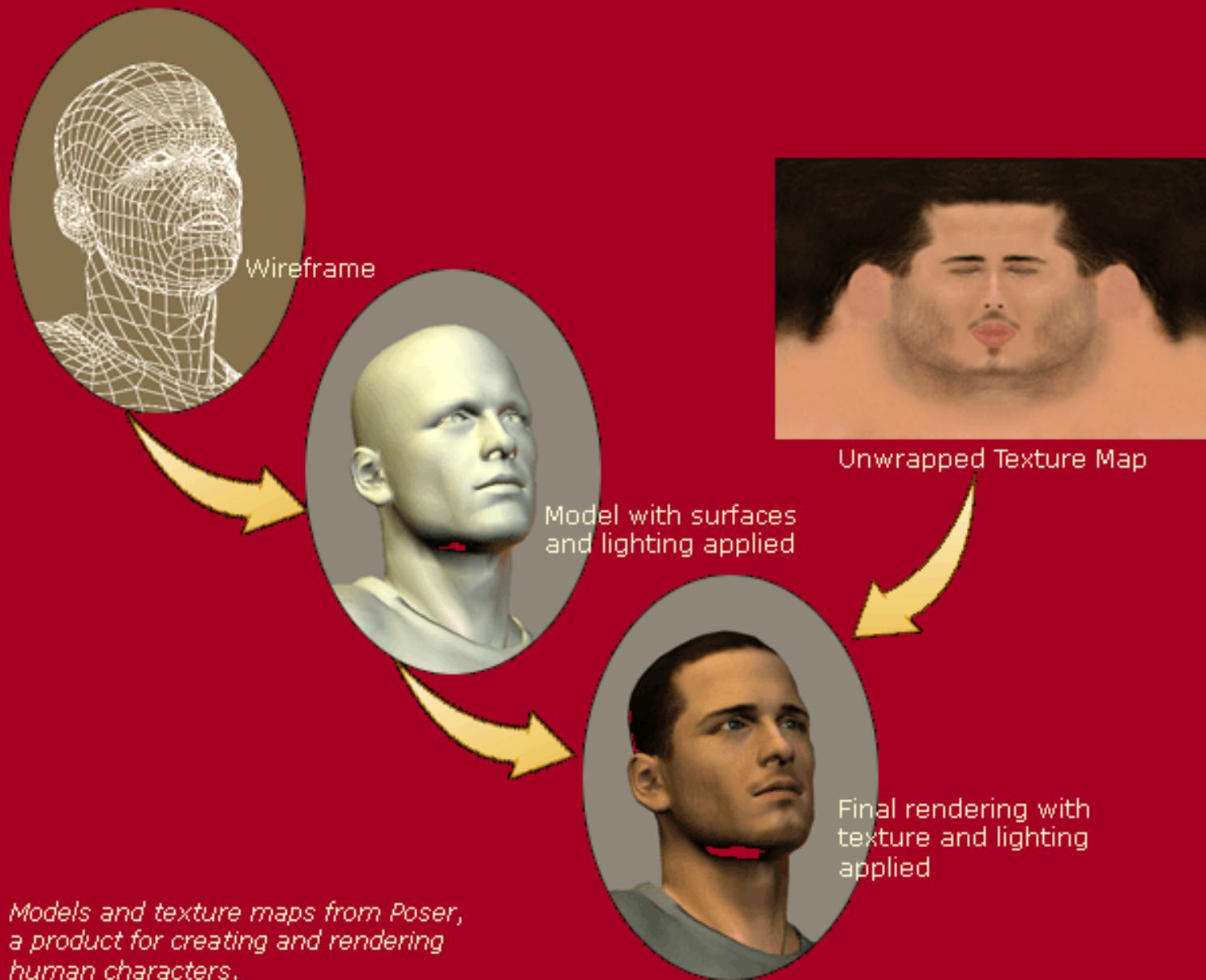
$$Z = R \cos(\varphi) = R \cos(\pi t)$$

$$\text{Z równania na } Z \text{ mamy: } t = \varphi / \pi = \arccos(Z/R) / \pi$$

$$\text{Z równania na } X \text{ mamy: } s = [\arccos(X/R \sin(\pi t))] / 2\pi$$

$$\text{gdzie: } \alpha = \arccos x \Rightarrow x = \cos \alpha$$

Zatem jeśli znamy współrzędne punktu na powierzchni sfery, X, Y, Z, możemy obliczyć odpowiadający mu punkt (s,t) w przestrzeni tekstury.



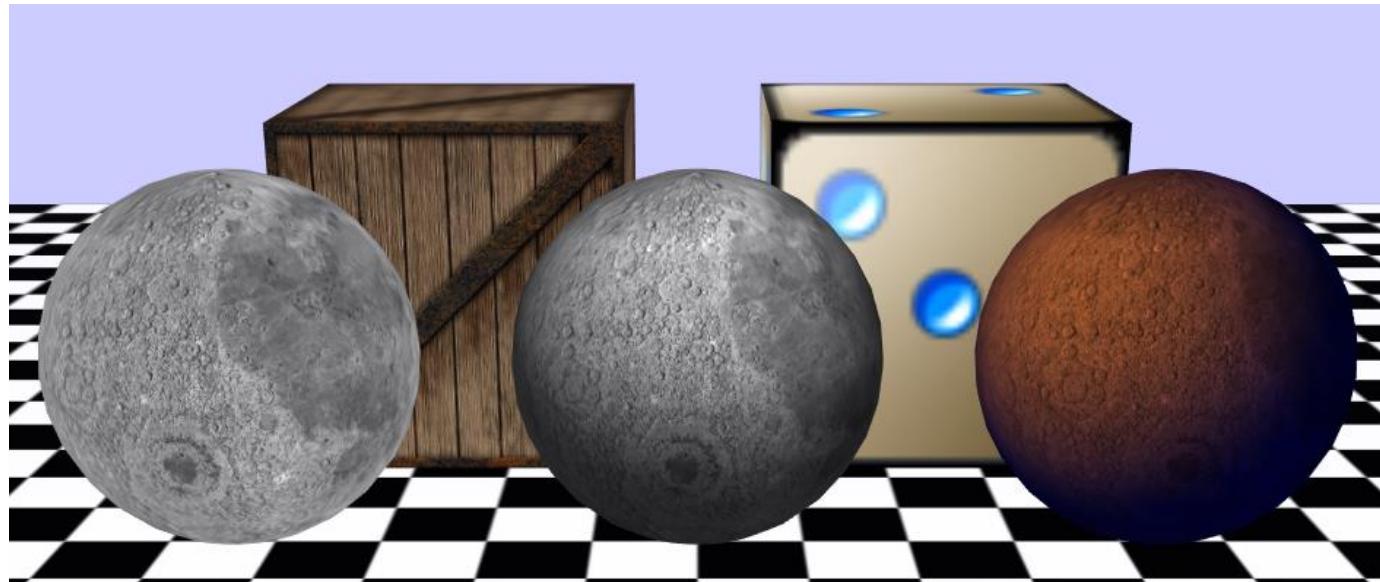
*Models and texture maps from Poser,
a product for creating and rendering
human characters.*

Example

Mapowanie tekstur w Three.js (przykład ze Stemkoskiego: texture.html)

- ▶ Specyfikujemy mapowanie tekstury jak parametry materiałowe

```
var moonTex = THREE.ImageUtils.loadTexture('images/moon.jpg');
var moonMat = new THREE.MeshLambertMaterial({ map: moonTex });
```



Mapowanie tekstur w Three.js cd.

- ▶ W wersji 87 biblioteki (w tej chwili bieżącej) mamy też inną wersję ładowarki tekstur, o nieco większych możliwościach:

```
var moonTex = THREE.TextureLoader().load('images/moon.jpg');  
var moonMat = new THREE.MeshLambertMaterial({ map: moonTex });
```

- ▶ Szczegóły możliwości... do podpatrzenia w dokumentacji na threejs.org



Objekt klasy Texture

- ▶ Obiekt załadowany przez `TextureLoader().load` może być następnie modyfikowany.
- ▶ Konstruktor:

```
Texture( image, mapping, wrapS,  
wrapT, magFilter, minFilter,  
format, type, anisotropy )
```

Basic Texture Mapping (Stemkoski cd)

```
//Cubes
// Note: when using a single image, it will appear on each of the faces.
// Six different images (one per face) may be used if desired.

var cubeGeometry = new THREE.CubeGeometry( 85, 85, 85 );
var crateTexture = new THREE.ImageUtils.loadTexture( 'images/crate.gif' );
var crateMaterial = new THREE.MeshBasicMaterial( { map: crateTexture } );
var crate = new THREE.Mesh( cubeGeometry.clone(), crateMaterial );
crate.position.set(-60, 50, -100);
scene.add( crate );
```

Użycie **cubeGeometry.clone()**
pozwala na stworzenie większej
liczby obiektów o tej samej
nazwie



```
// Cubes
// Note: when using a single image, it will appear on
// each of the faces.
// Six different images (one per face) may be used if
// desired.
```

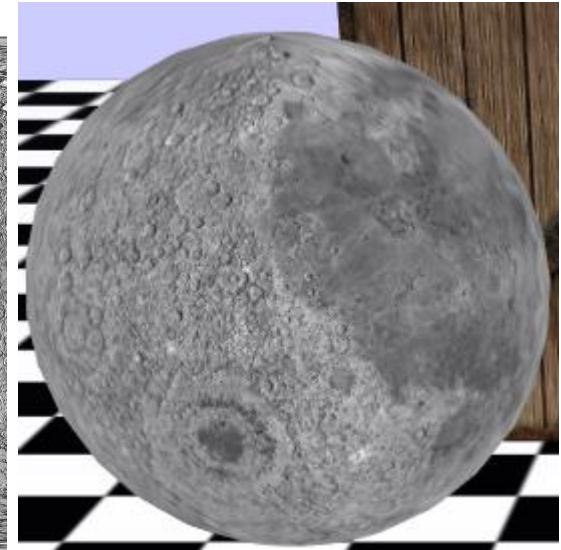
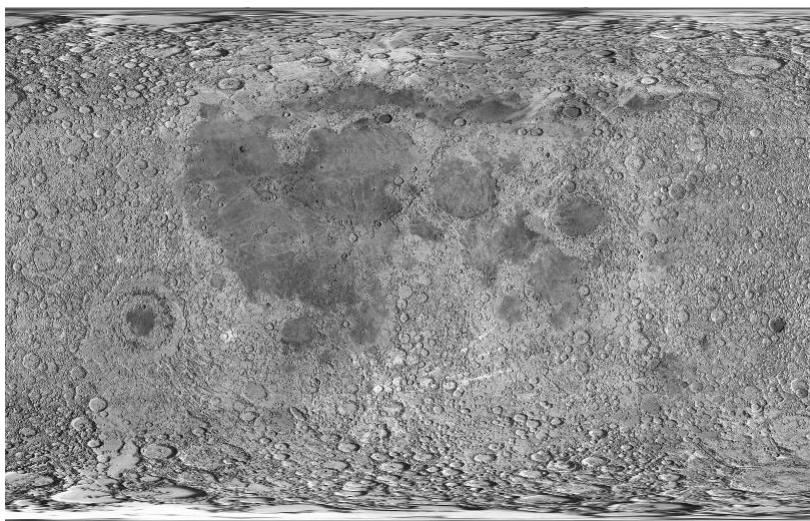
```
var cubeGeometry = new THREE.CubeGeometry( 85, 85, 85 );

var crateTexture = new THREE.ImageUtils.loadTexture(
'images/crate.gif' );
var crateMaterial = new THREE.MeshBasicMaterial( { map:
crateTexture } );
var crate = new THREE.Mesh( cubeGeometry.clone(),
crateMaterial );
crate.position.set(-60, 50, -100);
scene.add( crate );
```



Basic Spherical Texture Mapping

```
// basic moon
var moonTexture = THREE.ImageUtils.loadTexture( 'images/moon.jpg' );
var moonMaterial = new THREE.MeshBasicMaterial( { map: moonTexture } );
var moon = new THREE.Mesh( sphereGeom.clone(), moonMaterial );
moon.position.set(-100, 50, 0);
scene.add( moon );
```



MeshBasicMaterial – sama tekstura bez oświetlenia



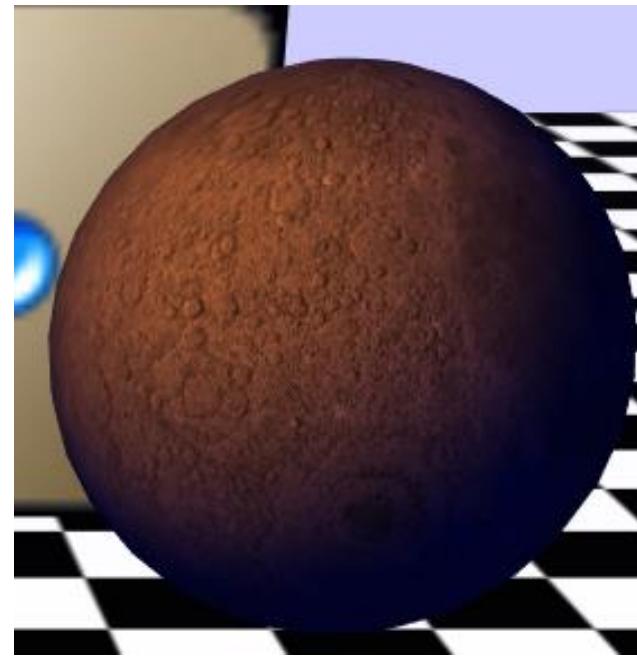
Lighting and Shading

```
// shaded moon -- side away from light picks up AmbientLight's color.  
var moonTexture = THREE.ImageUtils.loadTexture( 'images/moon.jpg' );  
var moonMaterial = new THREE.MeshLambertMaterial( {map: moonTexture} );  
var moon = new THREE.Mesh( sphereGeom.clone(), moonMaterial );  
moon.position.set(0, 50, 0);  
scene.add( moon );
```



Texture and Color

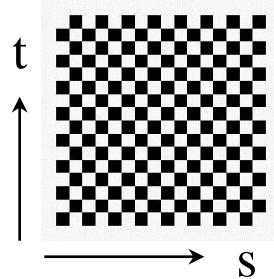
```
// colored moon
var moonTexture = THREE.ImageUtils.loadTexture( 'images/moon.jpg' );
var moonMaterial = new THREE.MeshLambertMaterial( {map: moonTexture,
color: 0xff8800, ambient: 0x0000ff} );
var moon = new THREE.Mesh( sphereGeom.clone(), moonMaterial );
moon.position.set(100, 50, 0);
scene.add( moon );
```



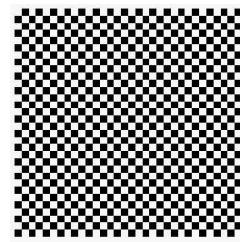
Filtrowanie tekstur

- ▶ Filtrowanie tekstur jest dosyć rozciągliwym pojęciem i obejmuje różne przekształcenia związane z teksturami. Omówimy najważniejsze.

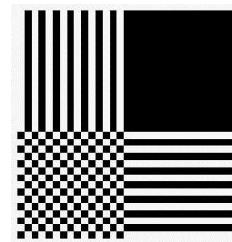
Filtrowanie tekstur: powtarzanie i rozciąganie (**REPEAT** i **CLAMP**)



tekstura



REPEAT

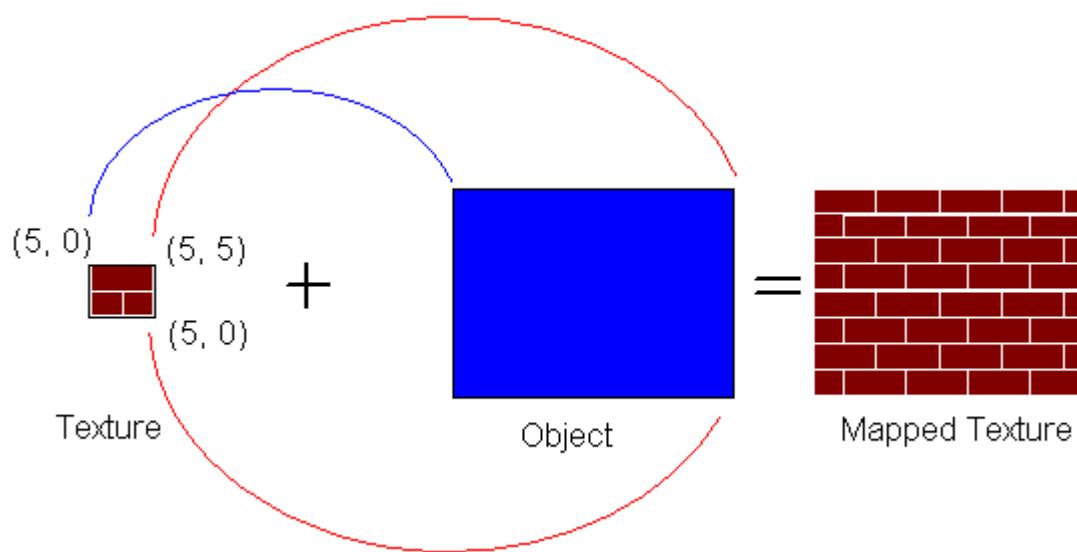


CLAMP



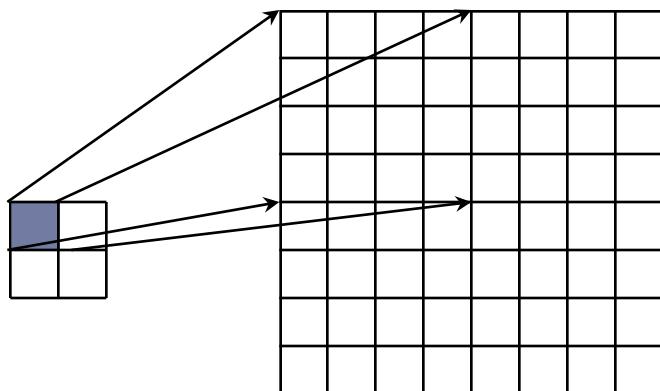
Filtrowanie tekstur: powtarzanie i rozciąganie

Tryb **REPEAT** jest znacznie bardziej popularny



Filtrowanie tekstur

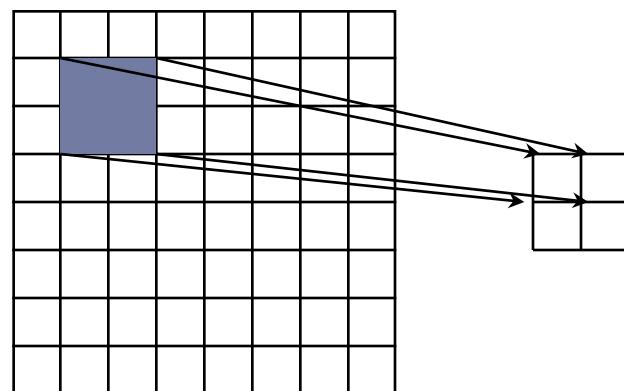
Drugie zastosowanie: Ustalenie algorytmu interpolacji przy powiększaniu/pomniejszaniu obiektów



Tekstura

Powiększenie – jeden
teksel jest mapowany
na wiele pikseli

Wielokąt



Tekstura

Pomniejszenie – wiele
tekseli jest mapowanych
na jeden piksel

Wielokąt



Filtrowanie tekstur

Filtrowanie określa w jaki sposób tekstura będzie nakładana na obiekt oraz jak będzie się zmieniać przy zmianie położenia obiektu względem obserwatora.

Filtrowanie tekstur: powiększanie i pomniejszanie

Pojęcia z jakimi się spotykamy:

NEAREST, które może być zastąpione przez **LINEAR**

NEAREST (Point Sampling)

Wybierz teksel, którego środek jest najbliższym danemu pikselowi

LINEAR (Bilinear Sampling)

Użyta jest ważona średnia tablicy 2x2 najbliższych tekseli

W przypadku *mipmappingu* mamy do dyspozycji więcej trybów interpolacji.



Filtrowanie tekstur: wartości parametrów w three.js

Wartość

THREE.NearestFilter

THREE.LinearFilter

THREE.NearestMipMapNearestFilter

THREE.NearestMipMapLinearFilter

THREE.LinearMipMapNearestFilter

THREE.LinearMipMapLinearFilter



Filtrowanie tekstur

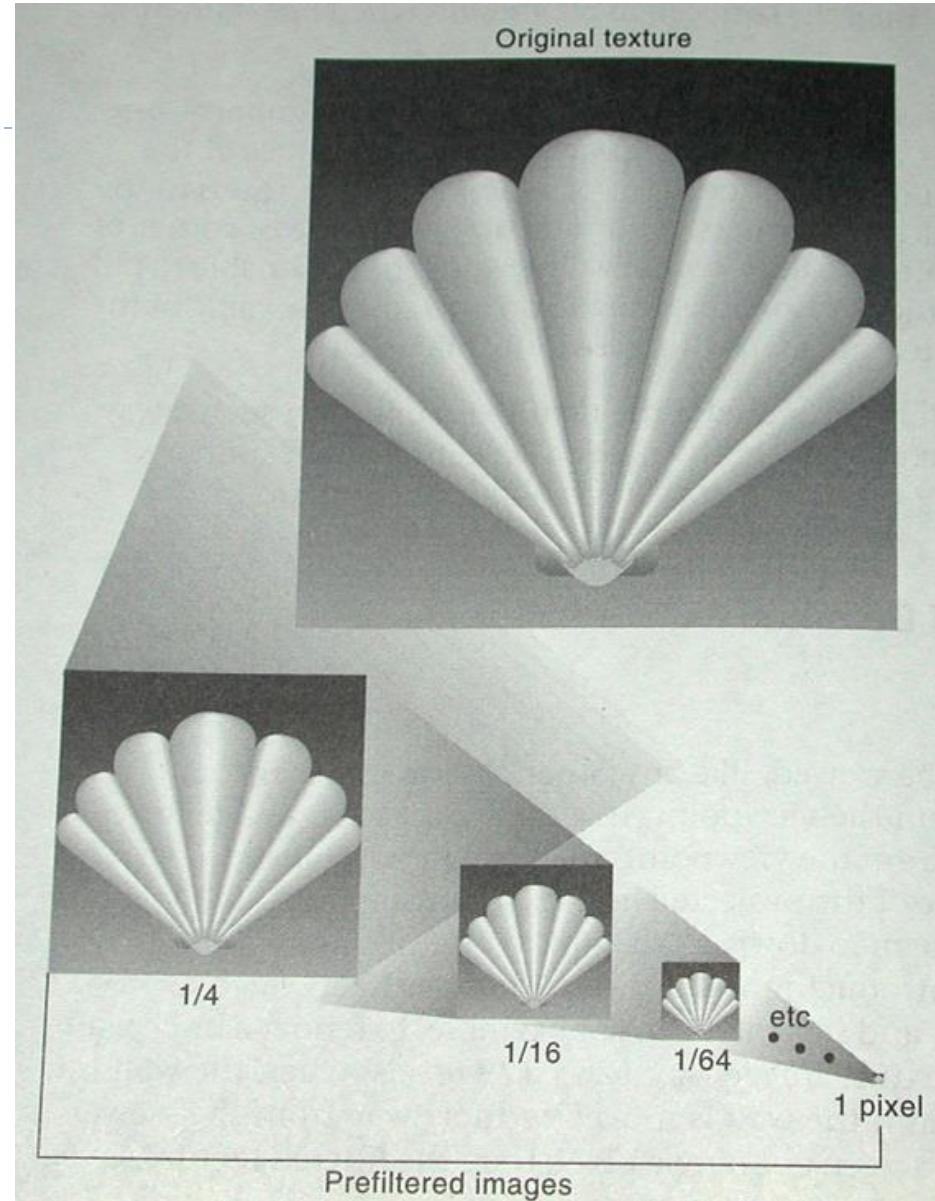
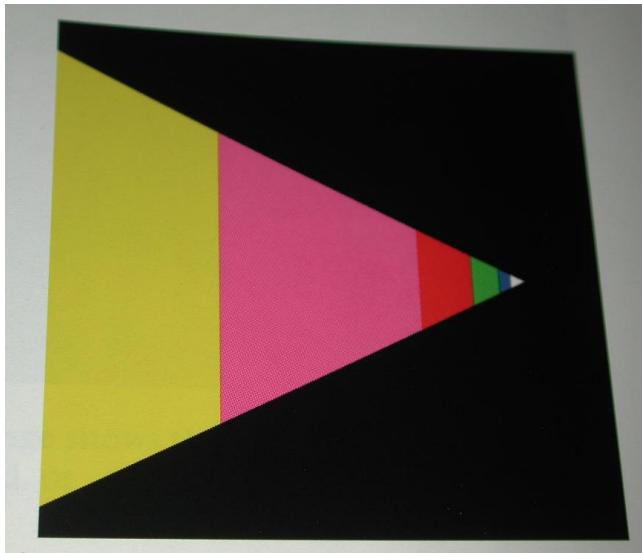
- ▶ Co się dzieje gdy tekstura jest zbyt mała lub zbyt duża?
- ▶ Co się dzieje gdy zbliżamy się lub oddalamy do/od teksturowanego obiektu?



- ▶ (Moire patterns)

Mipmapy

- ▶ Przygotowujemy zestaw tekstur o rozmiarach 1×1 , 2×2 , 4×4 , ..., $n \times n$
- ▶ Wybieramy najlepszą
- ▶ Przykłady: [1][2]



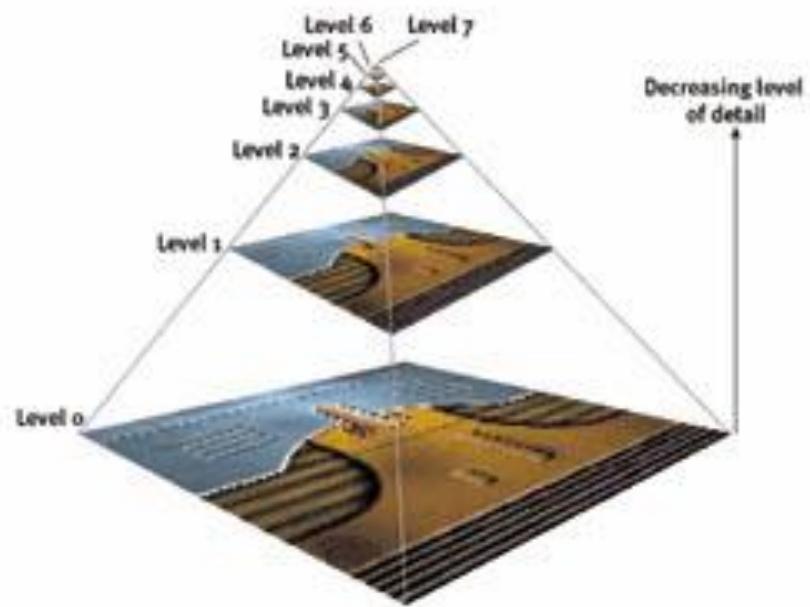
Mipmaps, mipmapping

Zastosowanie techniki LOD (Level of Detail) do tekstur.

Polega na przygotowaniu szeregu tekstur o różnej rozdzielczości i używaniu ich w zależności od oddalenia teksturowanego obiektu.

Wymaga więcej pamięci na tekstury

Eliminuje drgania i migotanie tekstury w poruszających się obiektach.



Mipmapy c.d.

W jaki sposób przełączane są tekstury w mipmappingu?
Ta uwaga jest raczej teoretyczna

$$\rho = \text{max texture size} / \text{max polygon size}$$

$$\lambda = \log_2 \rho$$

λ określa poziom mipmappingu

Jeżeli $\lambda < 0.0 \rightarrow$ tekstura jest mniejsza niż wielokąt i użyty jest filtr powiększający – nie ma mipmappingu

Jeżeli $\lambda > 0.0 \rightarrow$ tekstura jest większa od wielokąta i użyty jest filtr pomniejszający – stosujemy mipmapping



Filtrowanie z mipmappingiem

OpenGL próbuje dobrać najlepszy poziom mipmapy

Pytanie: Który teksel odpowiada określonemu pikselowi?

NEAREST_MIPMAP_NEAREST

Wybierana jest tekstura o najbliższej rozdzielczości i stosowany filtr **GL_NEAREST** w obrębie tekstury

NEAREST_MIPMAP_LINEAR

Wybierana jest tekstura o najbliższej rozdzielczości i stosowany filtr **GL_LINEAR** w obrębie tekstury

LINEAR_MIPMAP_NEAREST

Średnia teksceli z dwóch sąsiednich poziomów mipmapy

LINEAR_MIPMAP_LINEAR (Trilinear)

Średnia z dwóch uśrednionych teksceli z dwóch sąsiednich poziomów mipmapy.



Mipmapy w Three.js

```
E.Texture = function ( image, mapping, wrapS, wrapT, magFilter, minFilter, format, type, anisotropy ) {  
    THREE.TextureLibrary.push( this );  
  
    this.id = THREE.TextureIdCount++;  
  
    this.image = image;  
  
    this.mapping = mapping !== undefined ? mapping : new THREE.UVMapping();  
  
    this.wrapS = wrapS !== undefined ? wrapS : THREE.ClampToEdgeWrapping;  
    this.wrapT = wrapT !== undefined ? wrapT : THREE.ClampToEdgeWrapping;  
  
    this.magFilter = magFilter !== undefined ? magFilter : THREE.LinearFilter;  
    this.minFilter = minFilter !== undefined ? minFilter : THREE.LinearMipMapLinearFilter;  
  
    this.anisotropy = anisotropy !== undefined ? anisotropy : 1;  
  
    this.format = format !== undefined ? format : THREE.RGBAFormat;  
    this.type = type !== undefined ? type : THREE.UnsignedByteType;  
  
    this.offset = new THREE.Vector2( 0, 0 );  
    this.repeat = new THREE.Vector2( 1, 1 );  
}
```



Multitekstury w Three.js

```
var materials = [];
for (var i=0; i<6; i++) {
    var img = new Image();
    img.src = i + '.png';
    var tex = new THREE.Texture(img);
    img.tex = tex;
    img.onload = function() {
        this.tex.needsUpdate = true;
    };
    var mat = new THREE.MeshBasicMaterial({color: 0xffffffff, map: tex});
    materials.push(mat);
}
var cubeGeo = new THREE.CubeGeometry(400,400,400,1,1,1, materials);
var cube = new THREE.Mesh(cubeGeo, new THREE.MeshFaceMaterial());
```

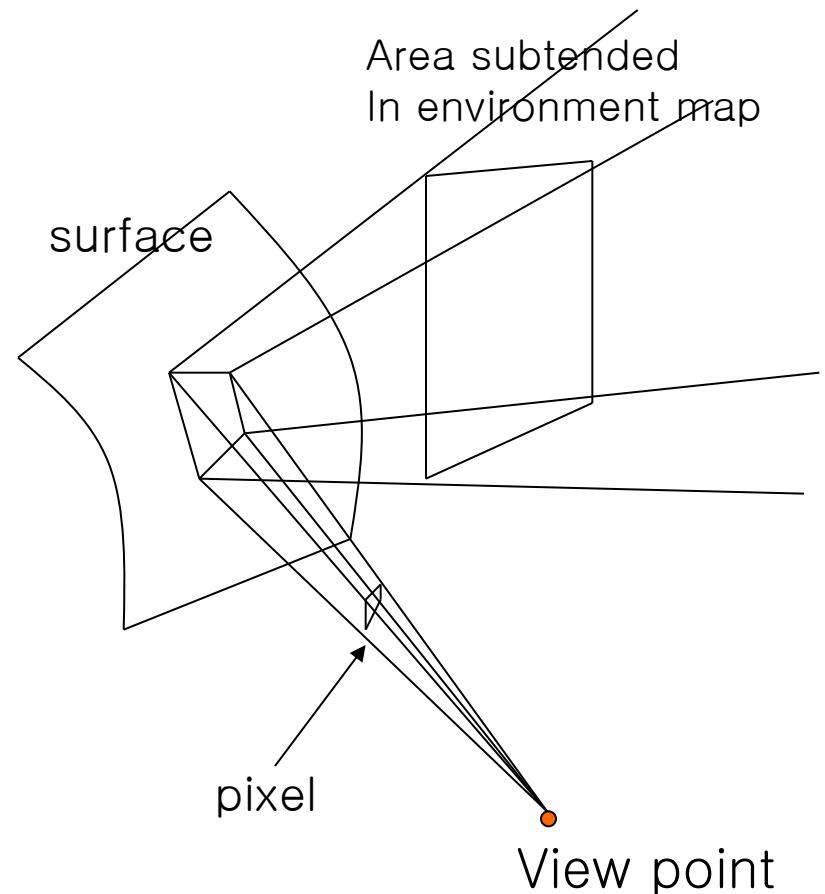


Spherical Environment Mapping

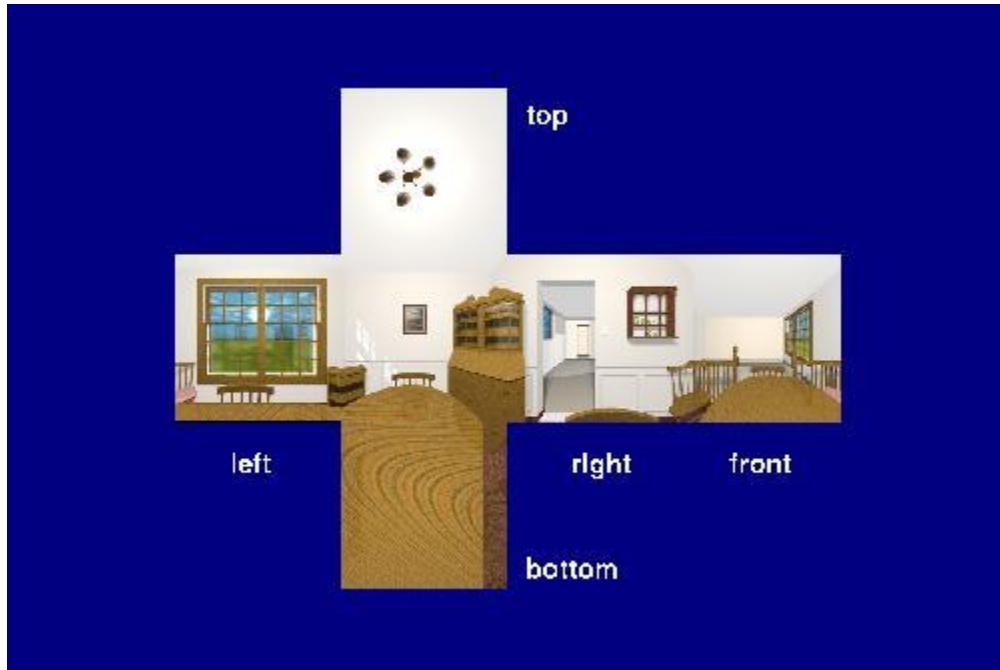
*Litografia
M.C. Eschera z 1935 r.
jest przykładem sferycznego
odbicia*

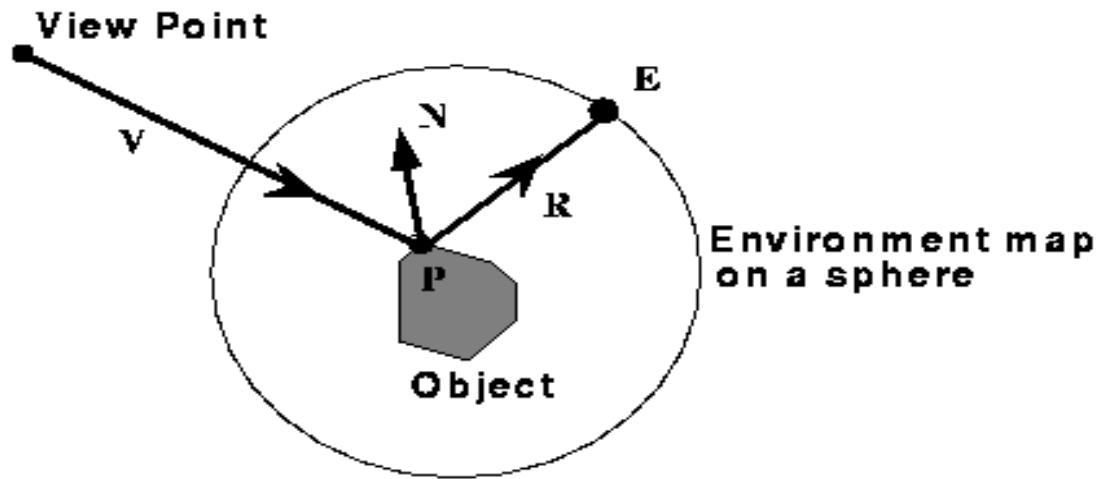


Environment mapping



Cube Environment Mapping





$$d = V \cdot N$$

$$R = V - 2dN$$

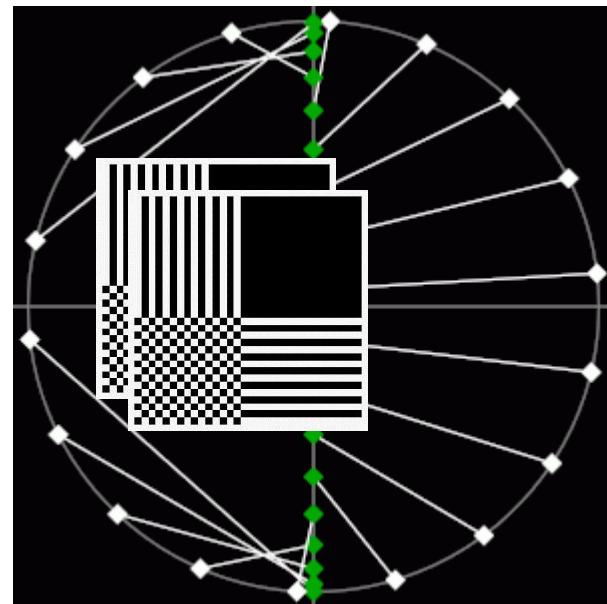
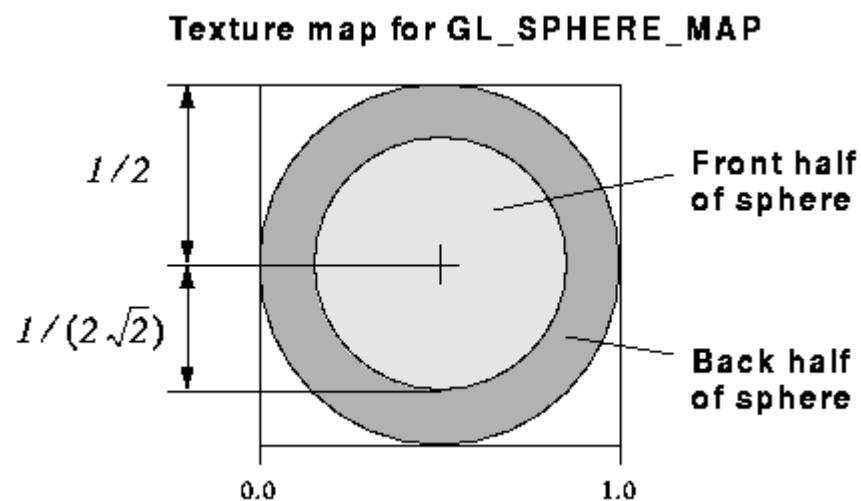
Sphere mapping - założenia

- ▶ Znajdujemy się wewnątrz sfery, na którą nałożona jest tekstura.
- ▶ Sfera jest nieskończonym wielkiem – zakładamy tak, żeby obserwator zawsze znajdował się w jej wnętrzu. Możemy też tak założyć, gdyż...
- ▶ przyjmujemy rzutowanie **ortogonalne**.

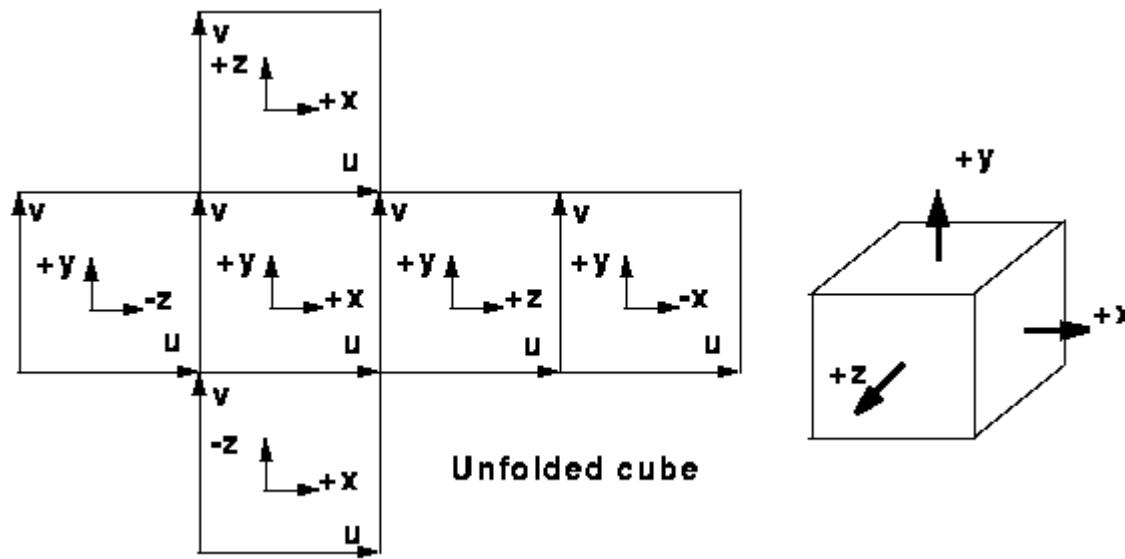
- ▶ Wyobraźmy sobie, że naszym obiektem lustrzanym jest również sfera, o promieniu $\frac{1}{2}$ (objaśnienie na tablicy)

Sphere mapping, cd.

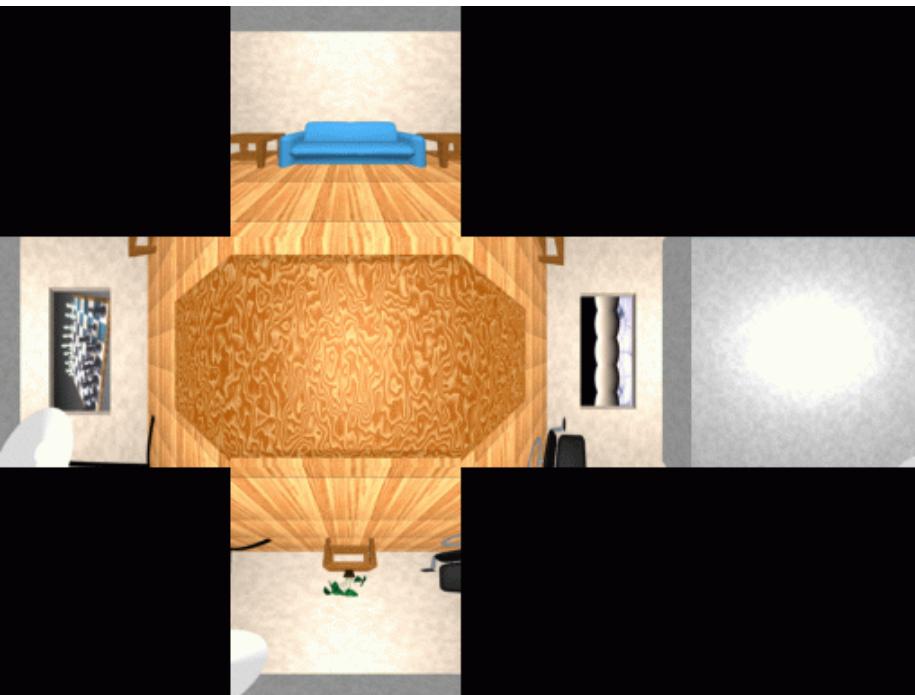
*W rzeczywistości nie widzimy kuli
tylko jej rzut:*



Cube mapping



Cube mapping





Cube Mapping w Three.js

```
mirrorCubeCamera = new THREE.CubeCamera(0.1,5000,512);
```

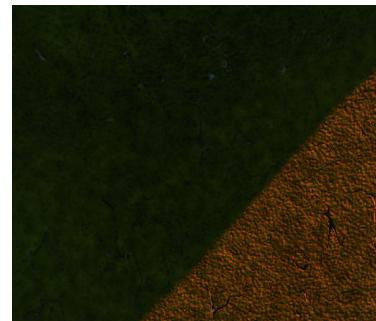
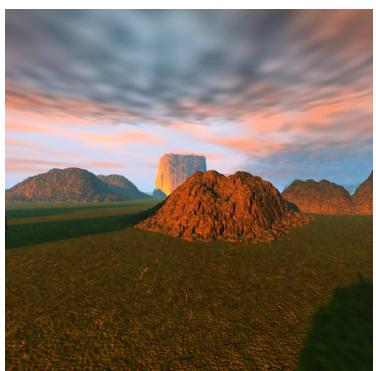
```
THREE.MeshBasicMaterial( { envMap:  
    mirrorCubeCamera.renderTarget } );
```

```
var cubeGeom = new THREE.CubeGeometry(100, 100, 100, 1, 1, 1);  
mirrorCubeCamera = new THREE.CubeCamera( 0.1, 5000, 512 );  
// mirrorCubeCamera.renderTarget.minFilter = THREE.LinearMipMapLinearFilter;  
scene.add( mirrorCubeCamera );  
var mirrorCubeMaterial = new THREE.MeshBasicMaterial( { envMap: mirrorCubeCamera.renderTarget } );  
mirrorCube = new THREE.Mesh( cubeGeom, mirrorCubeMaterial );  
mirrorCube.position.set(-75,50,0);  
mirrorCubeCamera.position = mirrorCube.position;  
scene.add(mirrorCube);  
  
var sphereGeom = new THREE.SphereGeometry( 50, 32, 16 ); // radius, segmentsWidth, segmentsHeight  
mirrorSphereCamera = new THREE.CubeCamera( 0.1, 5000, 512 );  
// mirrorSphereCamera.renderTarget.minFilter = THREE.LinearMipMapLinearFilter;  
scene.add( mirrorSphereCamera );  
var mirrorSphereMaterial = new THREE.MeshBasicMaterial( { envMap: mirrorSphereCamera.renderTarget } );  
mirrorSphere = new THREE.Mesh( sphereGeom, mirrorSphereMaterial );  
mirrorSphere.position.set(75,50,0);  
mirrorSphereCamera.position = mirrorSphere.position;  
scene.add(mirrorSphere);
```

<https://stemkoski.github.io/Three.js/Reflection.html>



Skybox



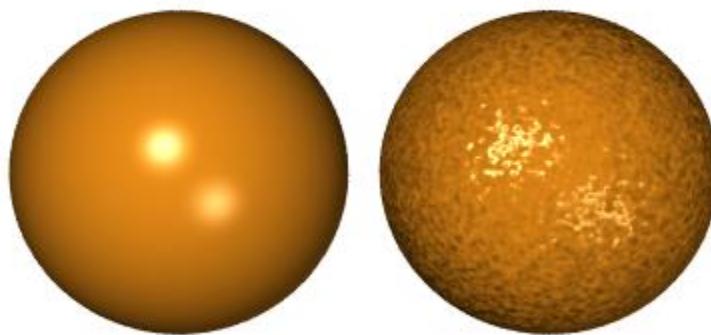
```
var materialArray = [];
materialArray.push(new THREE.MeshBasicMaterial( { map: THREE.ImageUtils.loadTexture( 'images/dawnmountain-xpos.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map: THREE.ImageUtils.loadTexture( 'images/dawnmountain-xneg.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map: THREE.ImageUtils.loadTexture( 'images/dawnmountain-ypos.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map: THREE.ImageUtils.loadTexture( 'images/dawnmountain-yneg.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map: THREE.ImageUtils.loadTexture( 'images/dawnmountain-zpos.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map: THREE.ImageUtils.loadTexture( 'images/dawnmountain-zneg.png' ) }));
var skyboxGeom = new THREE.CubeGeometry( 5000, 5000, 5000, 1, 1, 1, materialArray );
var skybox = new THREE.Mesh( skyboxGeom, new THREE.MeshFaceMaterial() );
skybox.flipSided = true;
scene.add( skybox );
```



Techniki tworzenia wrażenia nierównomiernych powierzchni

Bump mapping - mapowanie wypukłości

- ▶ Technika polega na zaburzeniu kierunku wektora normalnego i w konsekwencji zaburzeniu równomiernego oświetlenia i sprawieniu wrażenia wypukłości na gładkiej powierzchni.



Bump mapping – dwie techniki

- ▶ Techniki wiążą się z ustaleniem źródła zaburzeń normalnych
- ▶ Pierwsza technika odwołuje się do pracy Jamesa F. Blinna z 1978 r.: Simulation of Wrinkled Surfaces
<https://www.microsoft.com/en-us/research/wp-content/uploads/1978/01/p286-blinn.pdf> . Oparta jest na teksturach.
- ▶ Metoda Blinna została ponownie przeanalizowana i rozszerzona w pracy Mikkelsena: Simulation of Wrinkled Surfaces Revisited (2008).
- ▶ Druga technika bezpośrednio zaburza normalne metodą proceduralną.

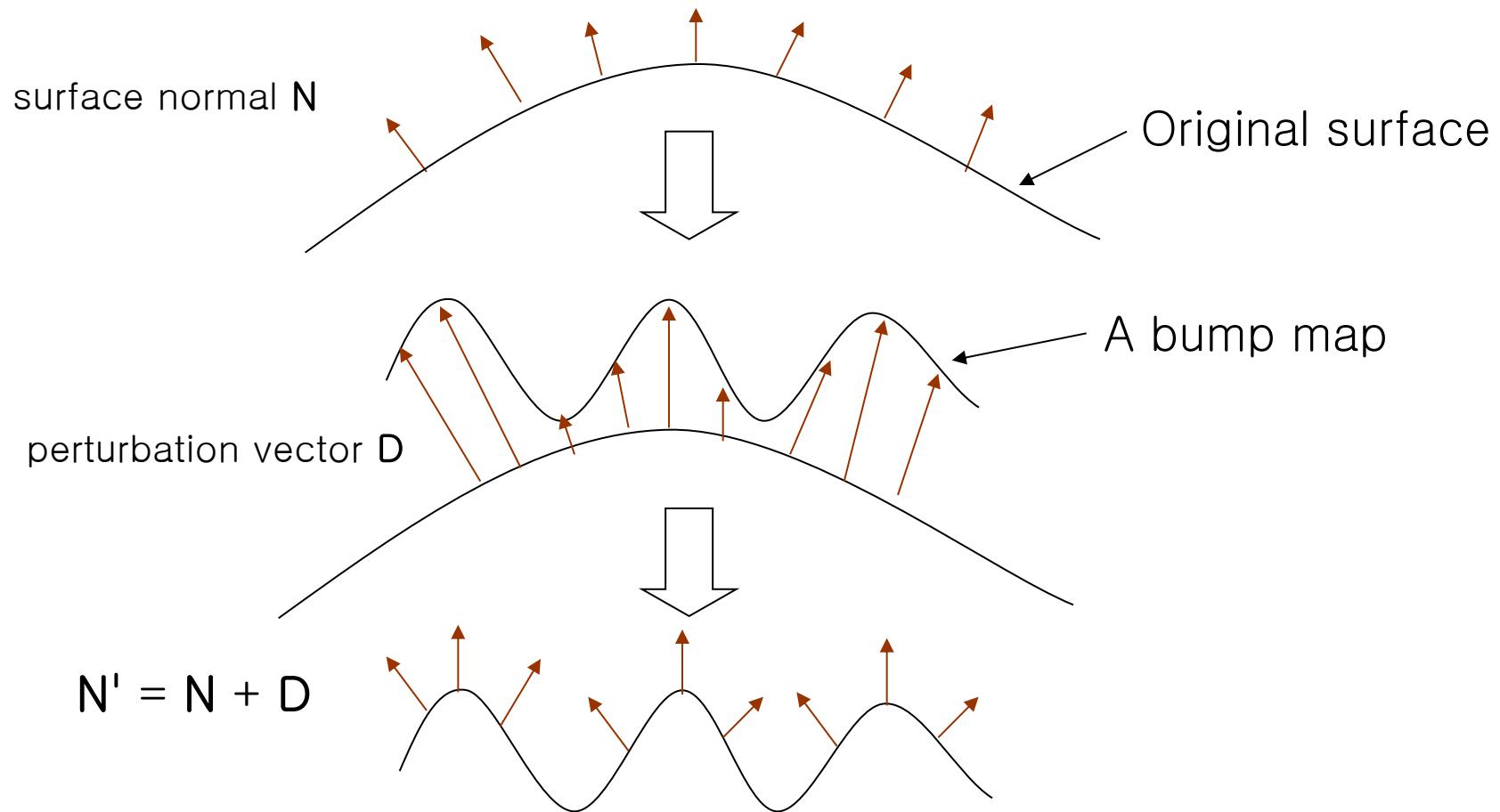
Bump mapping - technika Blinna

- ▶ Technika Blinna wykorzystuje monochromatyczną teksturę, potraktowaną jako mapę wysokości.

Bump Mapping – kroki obliczeń

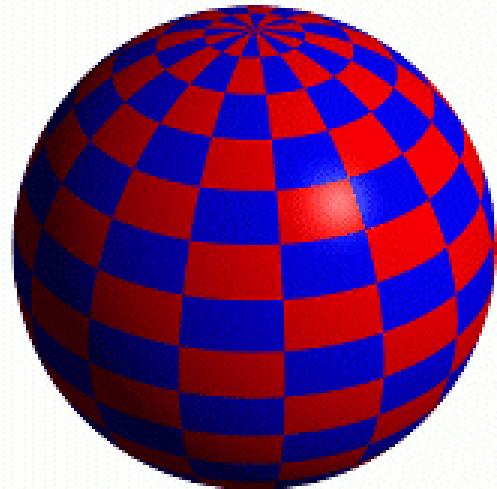
- ▶ Odczytaj wysokość z mapy wysokości w punkcie odpowiadającym położeniu na powierzchni.
 - ▶ Oblicz wektor normalny do mapy wysokości (zwyczajnie używa się metody różnic skończonych).
 - ▶ Połącz obie normalne – tę z mapy wysokości i z rzeczywistej powierzchni.
 - ▶ Oblicz oświetlenie według nowych normalnych.
-
- ▶ Poglądowy schemat na następnym slajdzie.

Bump mapping - technika Blinna

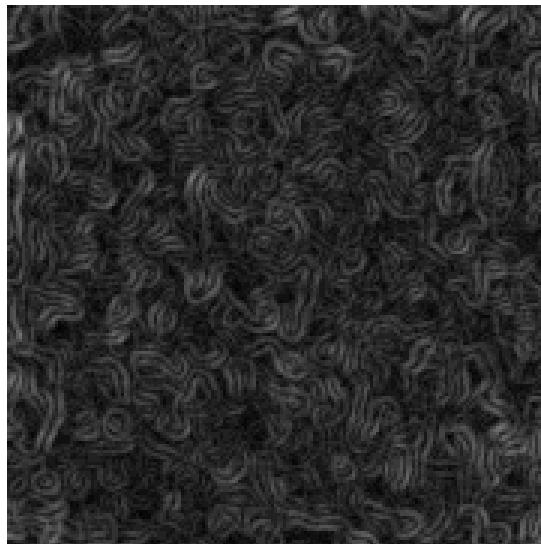


Przykład bump mappingu

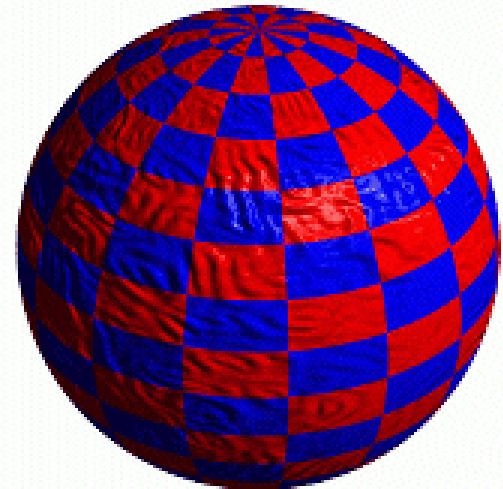
- ▶ W tym przykładzie mapa wysokości łączona jest z nakładaną teksturą, co jest typowym zabiegiem.



Sphere w/ diffuse texture



Swirly bump map



*Sphere w/ diffuse texture
and swirly bump map*

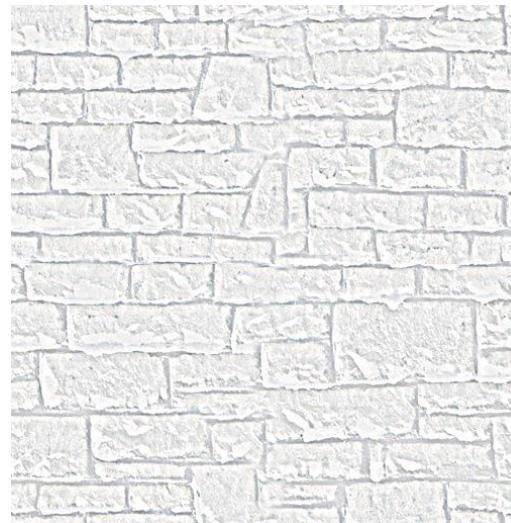
Bump mapping w three.js

```
var texture =  
THREE.ImageUtils.loadTexture("stone.jpg");  
  
var mat = new THREE.MeshPhongMaterial();  
mat.map = texture;  
var bump =  
THREE.ImageUtils.loadTexture("s_bump.jpg");  
mat.bumpMap = bump;  
mat.bumpScale = 0.2;  
var mesh = new THREE.Mesh(geom, mat);
```

Bump mapping – oparty na mapie wysokości

- ▶ Często mapa wysokości jest tworzona jest na podstawie mapowanej tekstury.
- ▶ Przykład:

<http://www.smartjava.org/lts/chapter-10/02-bump-map.html>



To jest mapa wysokości

Jak zrobić mapę wysokości dla danej tekstury.

- ▶ Jest oczywiście wiele sposobów. Nie wszystkie są skuteczne dla każdej tekstyury
- ▶ Najprostszy schemat w Gimpie (wersja ~2.8)
 1. Otwieramy nowy pusty obrazek, który na końcu stanie się mapą wysokości
 2. Otwieramy obrazek z teksturą , do które chcemy dorobić mapę wysokości, jako warstwę: *Plik → Otwórz jako warstwy*
 3. W okienku Warstwy wskazujemy podstawowy obrazek i filtry : *Filtr → Odwzorowania → Mapa wypukłości* . W oknie dialogowym ustalamy parametry mapy wysokości.
 4. Ukrywamy warstwę z tekstonią – w ten sposób na obrazku widoczna jest wyłącznie mapa wysokości.
 5. Eksportujemy obrazek.

Normal mapping (Dot3 bump mapping)

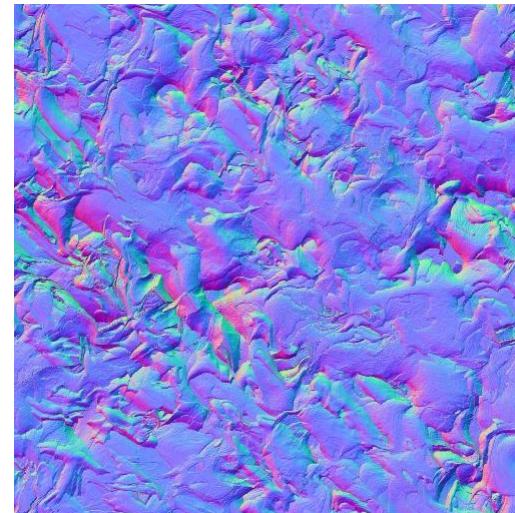
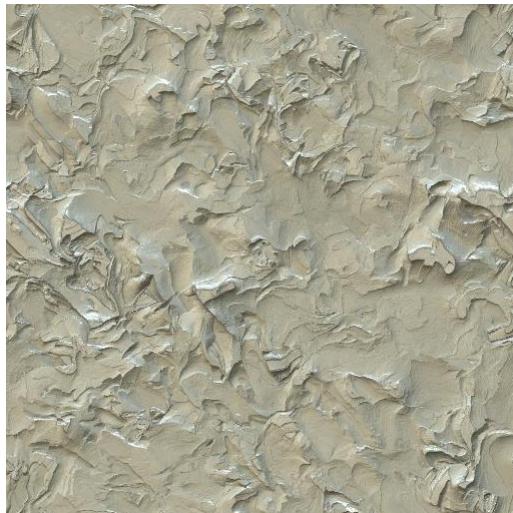
– udoskonalone mapowanie wypukłości

- ▶ Zmodyfikowane kierunki wektorów normalnych zapisane są za pomocą tekstury. Składowe elementu tekstury R,G,B przekładają się na współrzędne X,Y, Z .

- ▶ Podstawowe idee zawarte w:
 - ▶ Krishnamurthy i Levoy: *Fitting Smooth Surfaces to Dense Polygon Meshes*, SIGGRAPH 1996
 - ▶ Cohen, Olano i Manocha. :*Appearance Preserving Simplification*, SIGGRAPH 1998
 - ▶ Cignoni et al. :*A general method for preserving attribute values on simplified meshes*, SIGGRAPH 1998

Normal mapping

- ▶ <http://www.smartjava.org/ltjs/chapter-10/03-normal-map.html>



Normal mapping w three.js

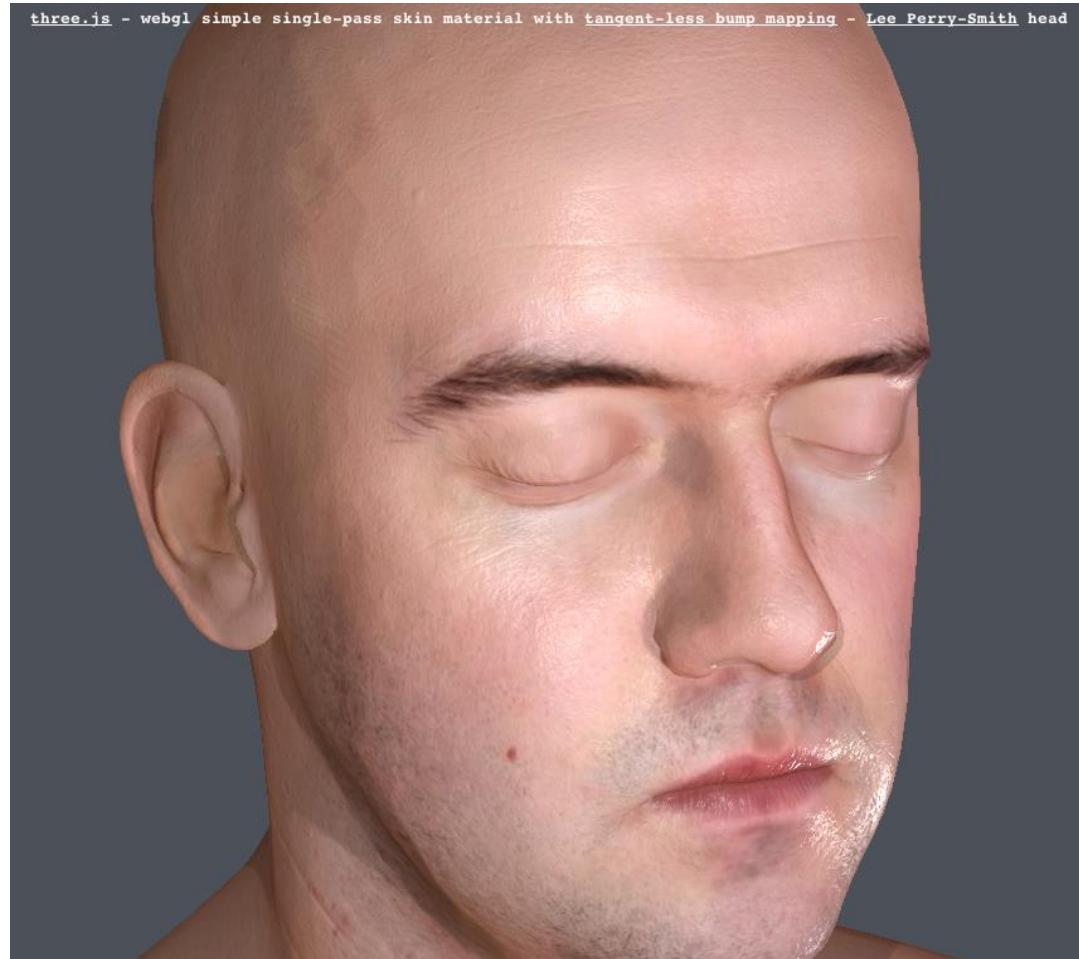
```
▶ var tex =  
THREE.ImageUtils.loadTexture(" ");  
  
var mat =  
THREE.ImageUtils.loadTexture("normal.jpg");  
  
var mat2 = new THREE.MeshPhongMaterial();  
mat2.map = tex;  
mat2.normalMap = mat;  
  
var mesh = new THREE.Mesh(geom, mat2);
```

Jak zrobić mapę normalnych dla danej tekstury

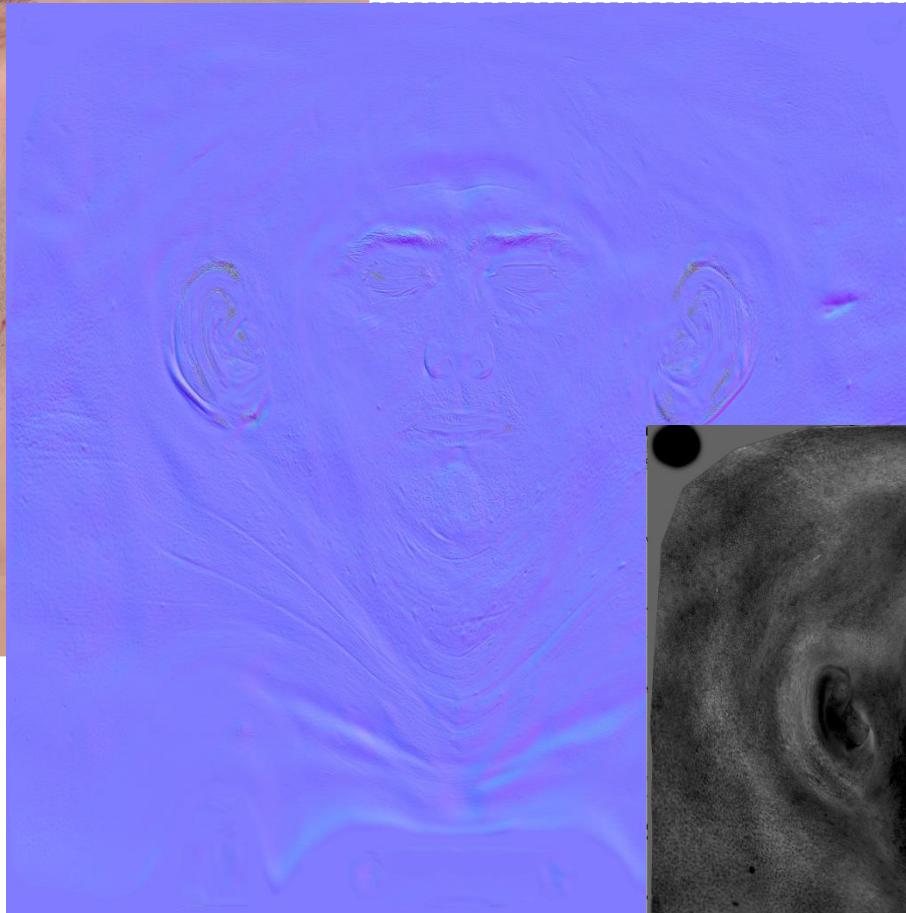
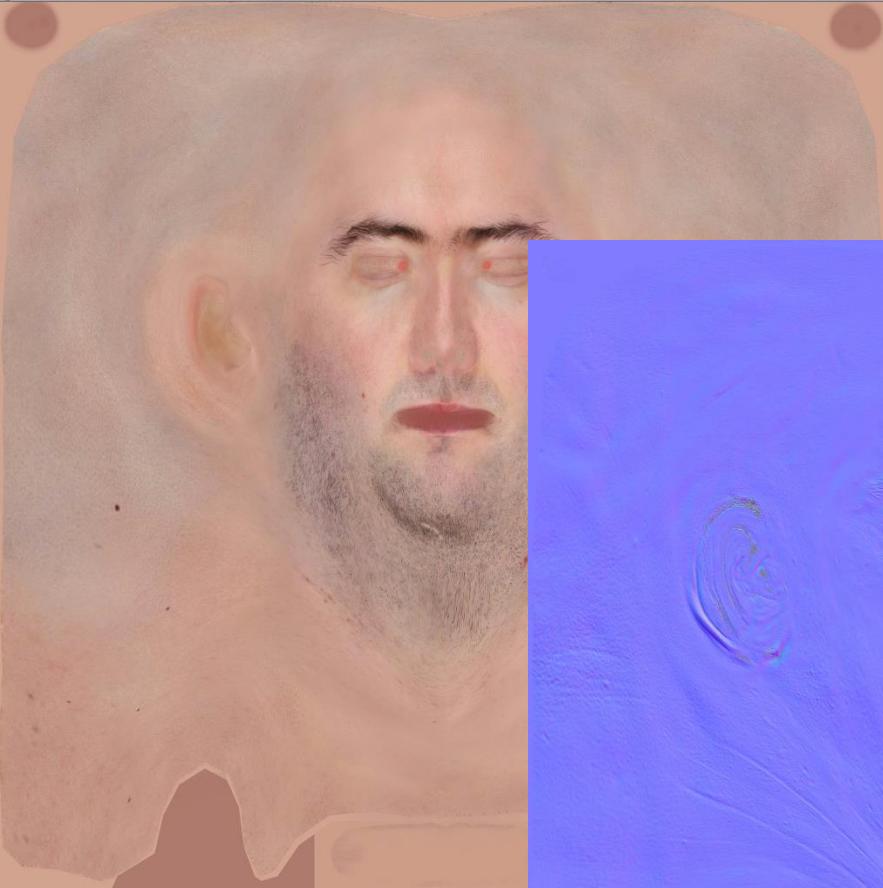
- ▶ Wiele różnych podejść. Nie dla każdej tekstury udaje się znaleźć poprawną mapę normalnych.
- ▶ W programie Gimp trzeba doładować wtyczkę normalmap: <http://registry.gimp.org/node/69>

Przykład Bump Mappingu w Three.js

- ▶ 3D Head Scan by Lee Perry-Smith



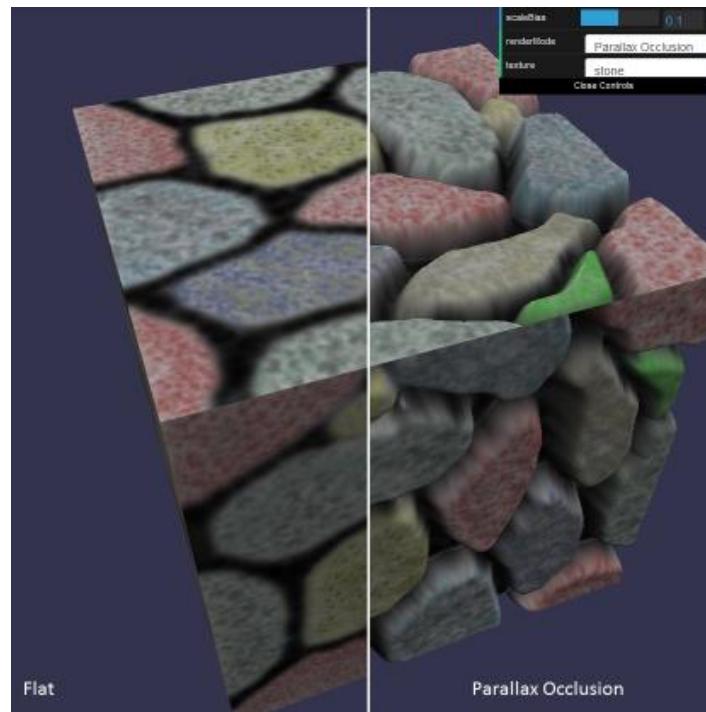
Tekstura, mapa normalnych i mapa wypukłości



Ch

Parallax mapping

- ▶ Rozszerzenie Bump Mappingu / Normal Mappingu.
- ▶ Polega na dynamicznym modyfikowaniu tekstury w zależności od kąta patrzenia, np. na translacji jasnych teksceli
- ▶ Porównanie:



Porównanie mapowań



Texture Mapped



Normal Mapped



Parallax Mapped



Steep Parallax Mapped

Displacement Mapping

- ▶ Polega na rzeczywistej modyfikacji geometrii na podstawie tekstury

