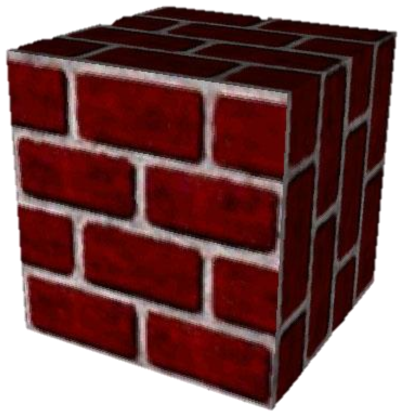
Texturing

Basic Idea

Paint pictures on all of your polygons

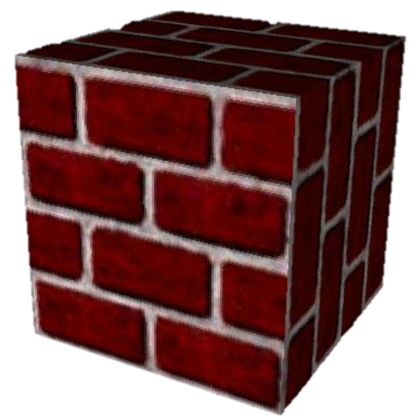
- adds color data
- adds (fake) geometric and texture detail



Basic Idea

Paint pictures on all of your polygons

- adds color data
- adds (fake) geometric and texture detail



one of **the** basic graphics techniques

- tons of hardware support

Sprites

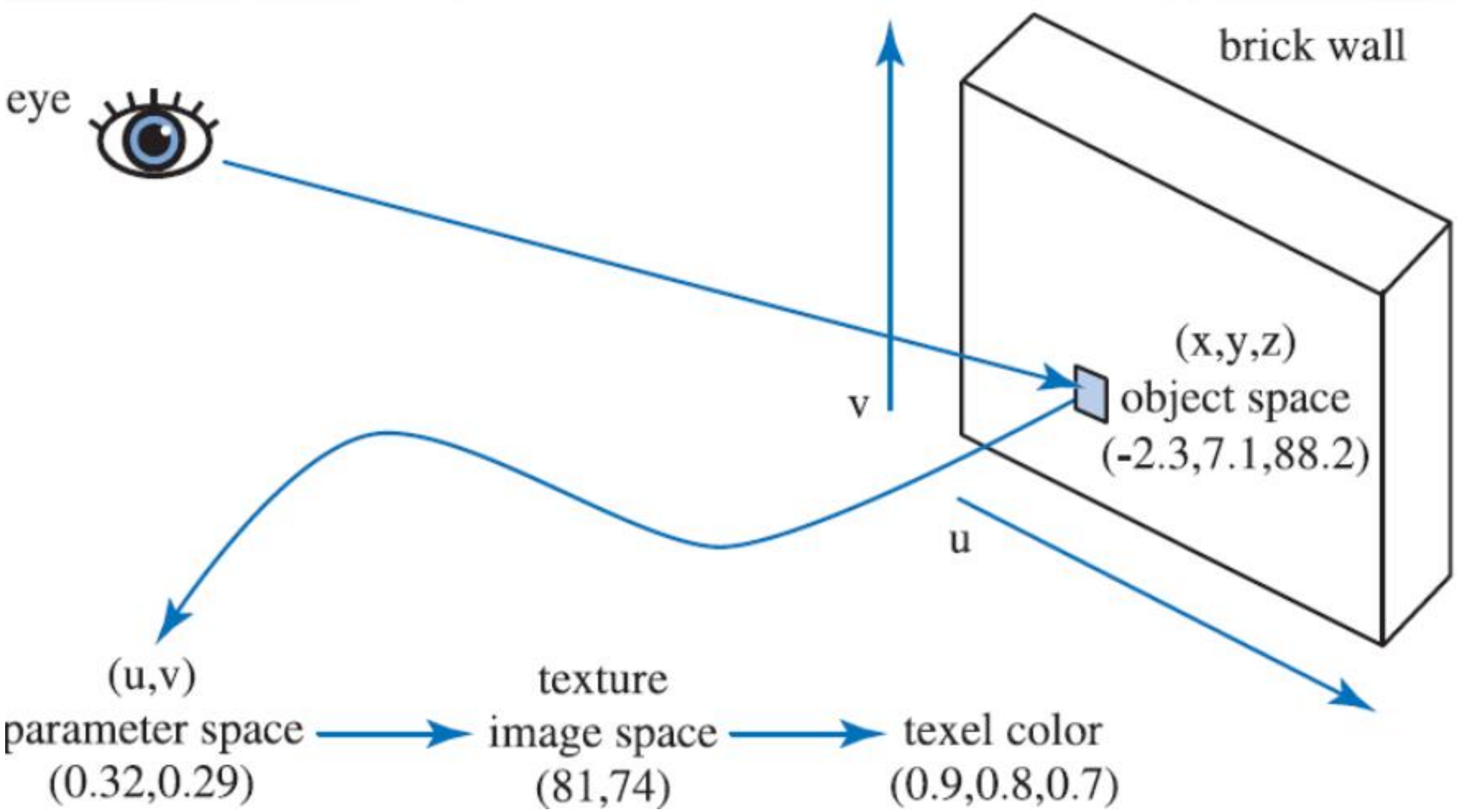
Draw object for a few states / from a few viewpoints

Sprites

Draw object for a few states / from a few viewpoints

Then copy closest matching image directly to screen

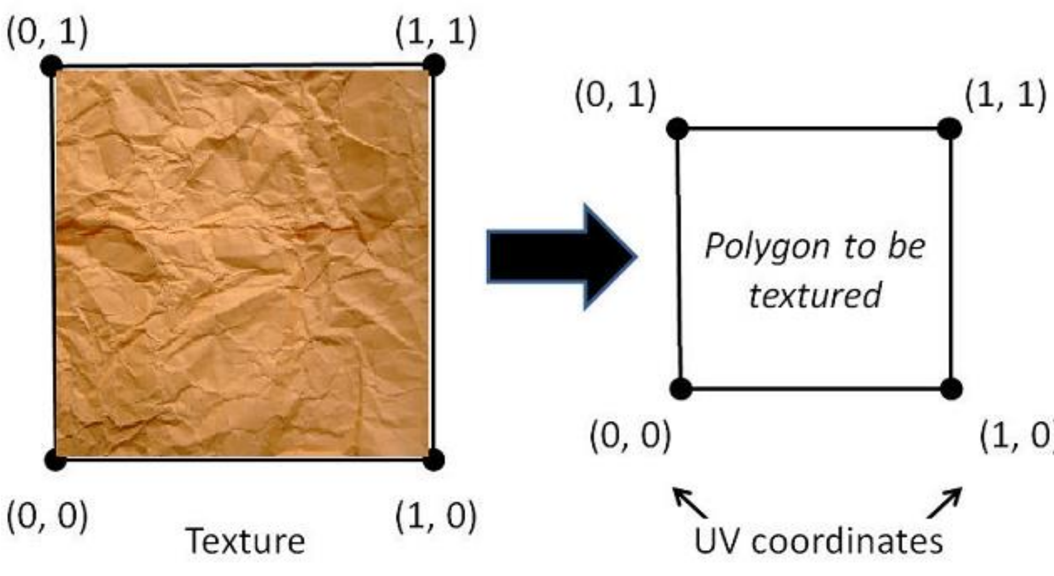
True Texture Mapping



True Texture Mapping

Texture map: map from **object** to **texture** coordinates (u,v) to **bitmap pixels**

- last lecture: how to generate UVs



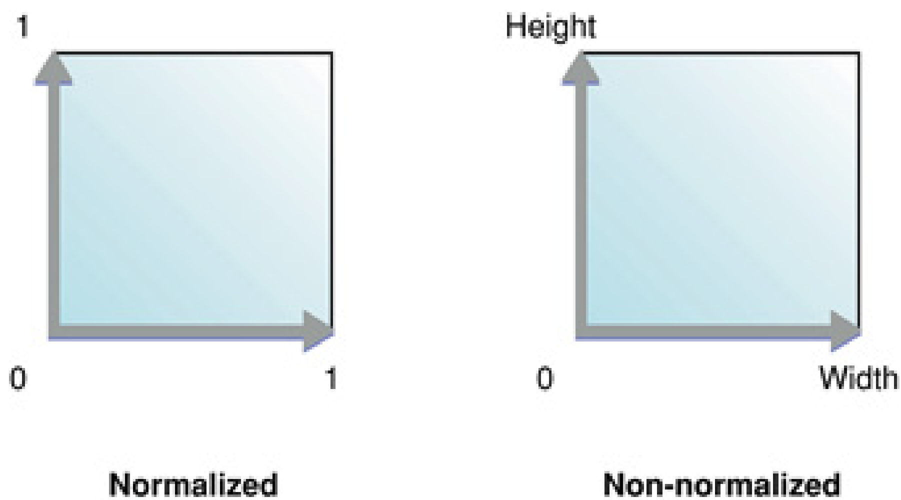
Texture Mapping Algorithm

For every pixel:

- compute pixel’s (u,v) using barycentric interpolation
- look up texture pixel (**texel**) at (u,v)
- copy color to pixel
- apply shading

Texture Mapping: Details

Two texture coordinate conventions:



Doesn’t matter which you use, but must be consistent

Texture Mapping: Details

What if (u,v) is out of range?

repeat

mirror

clamp

background

Texture Mapping: Details

What if (u,v) aren't integers?

Option 1: snap to nearest texel

Texture Mapping: Details

What if (u,v) aren't integers?

Option 2: linearly interpolate color

Texture Mapping: Details

What if (u,v) is out of range?

What if (u,v) aren't integers?

These are both parts of **sampling**

- extracting color from bitmap given (u,v)

Texture Mapping: Details

What if (u,v) is out of range?

What if (u,v) aren't integers?

These are both parts of **sampling**

- extracting color from bitmap given (u,v)
- sampling methods (**filters**) can improve quality/jagginess of textured objects

Texture Mapping: Details

A few more minor details:

- texture sizes traditionally powers of 2

Texture Mapping: Details

A few more minor details:

- texture sizes traditionally powers of 2
- textures usually compressed on GPU

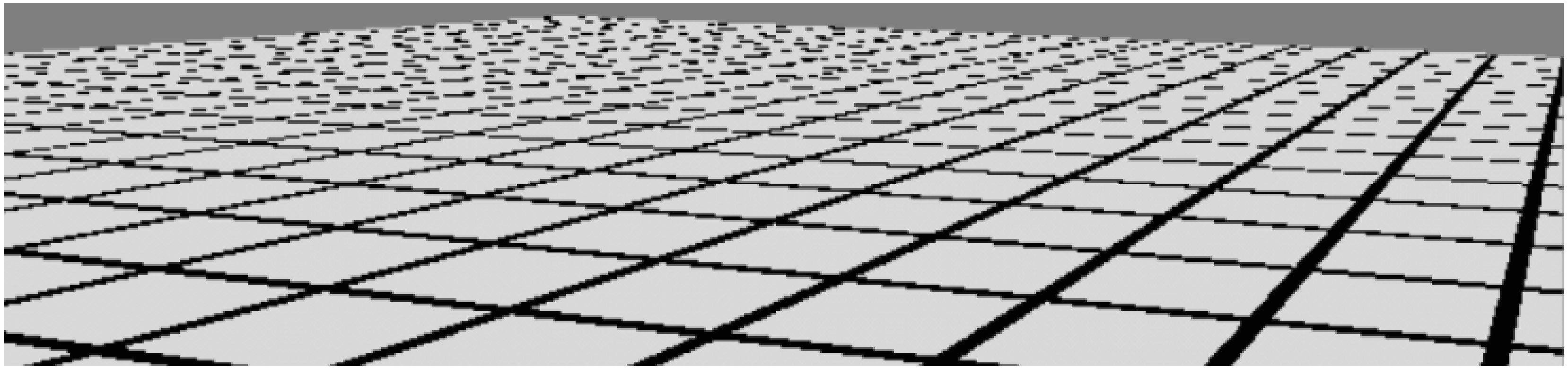
Texture Mapping: Details

A few more minor details:

- texture sizes traditionally powers of 2
- textures usually compressed on GPU
- textures can be 3D
  - huge memory hog!

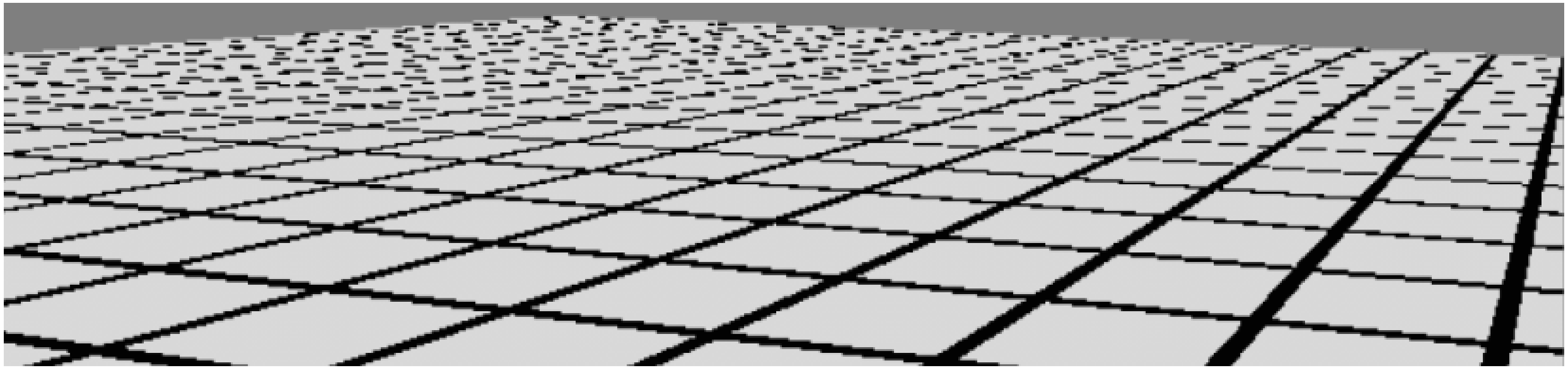
Advanced Sampling

Problem:



Advanced Sampling

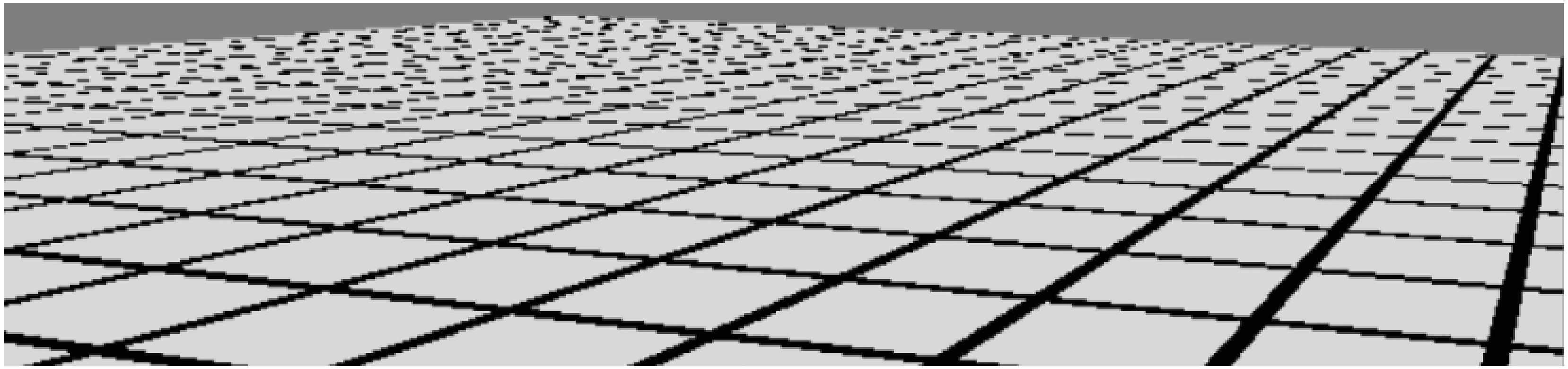
Problem:



No correlation between pixels and texels

Advanced Sampling

Problem:

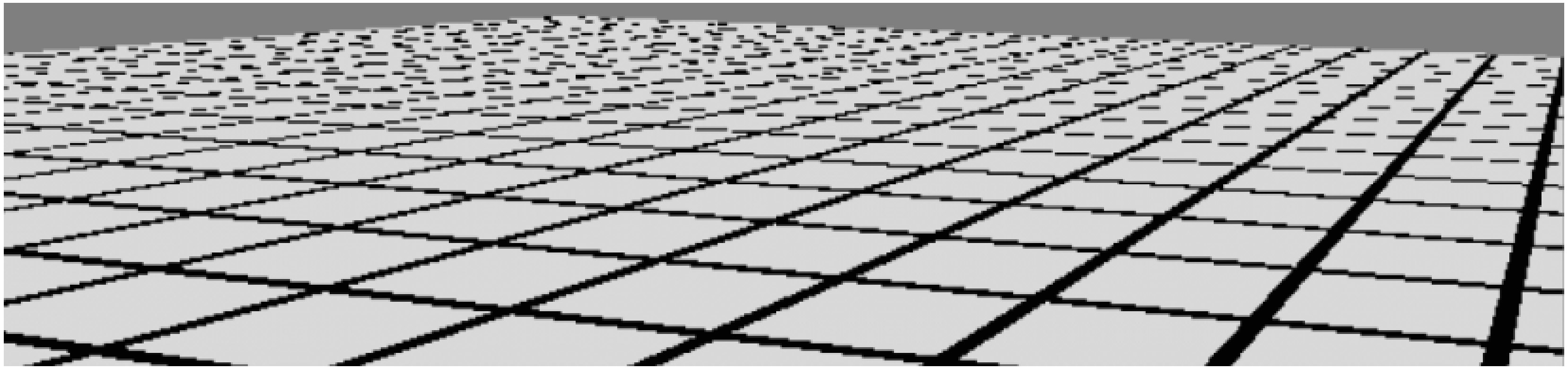


No correlation between pixels and texels

Too many texels per pixel: aliasing

Advanced Sampling

Problem:

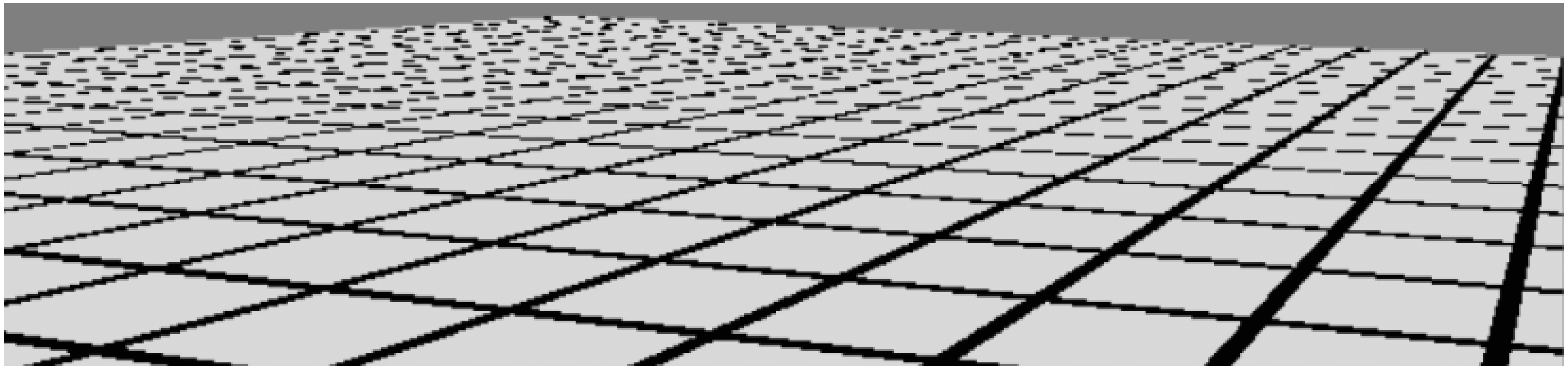


No correlation between pixels and texels

Sample many texels and average?

Advanced Sampling

Problem:

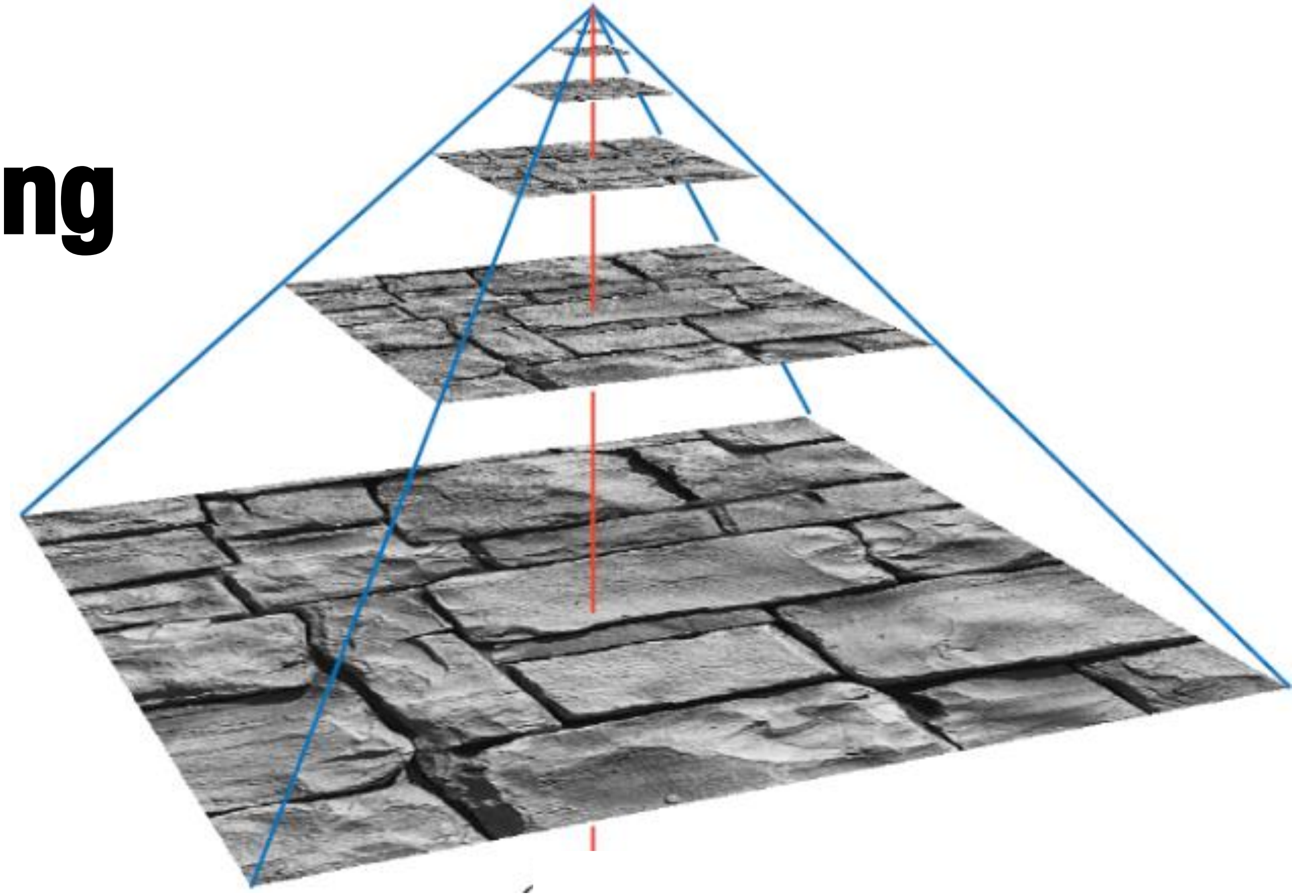


No correlation between pixels and texels

Sample many texels and average?

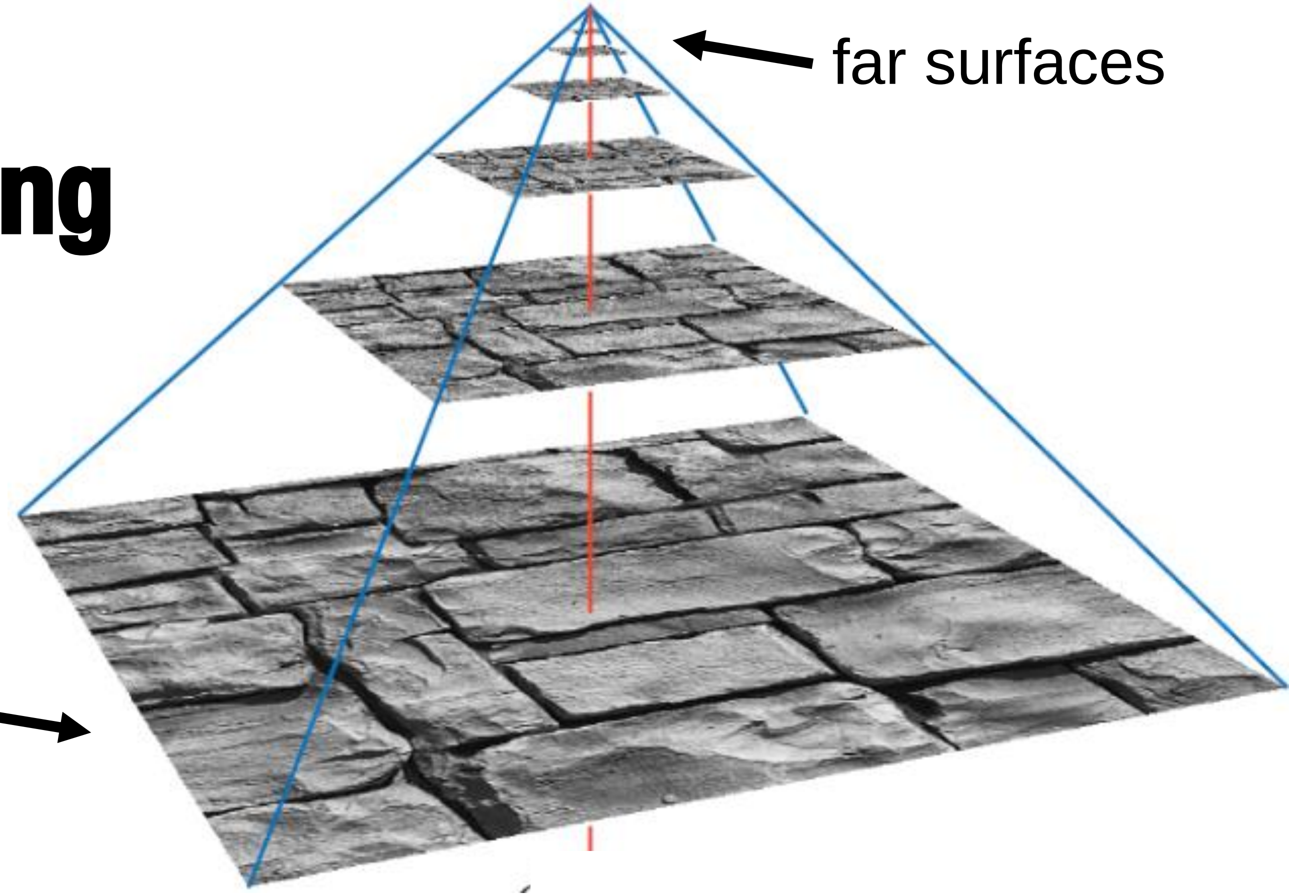
- works but very slow

Mipmapping



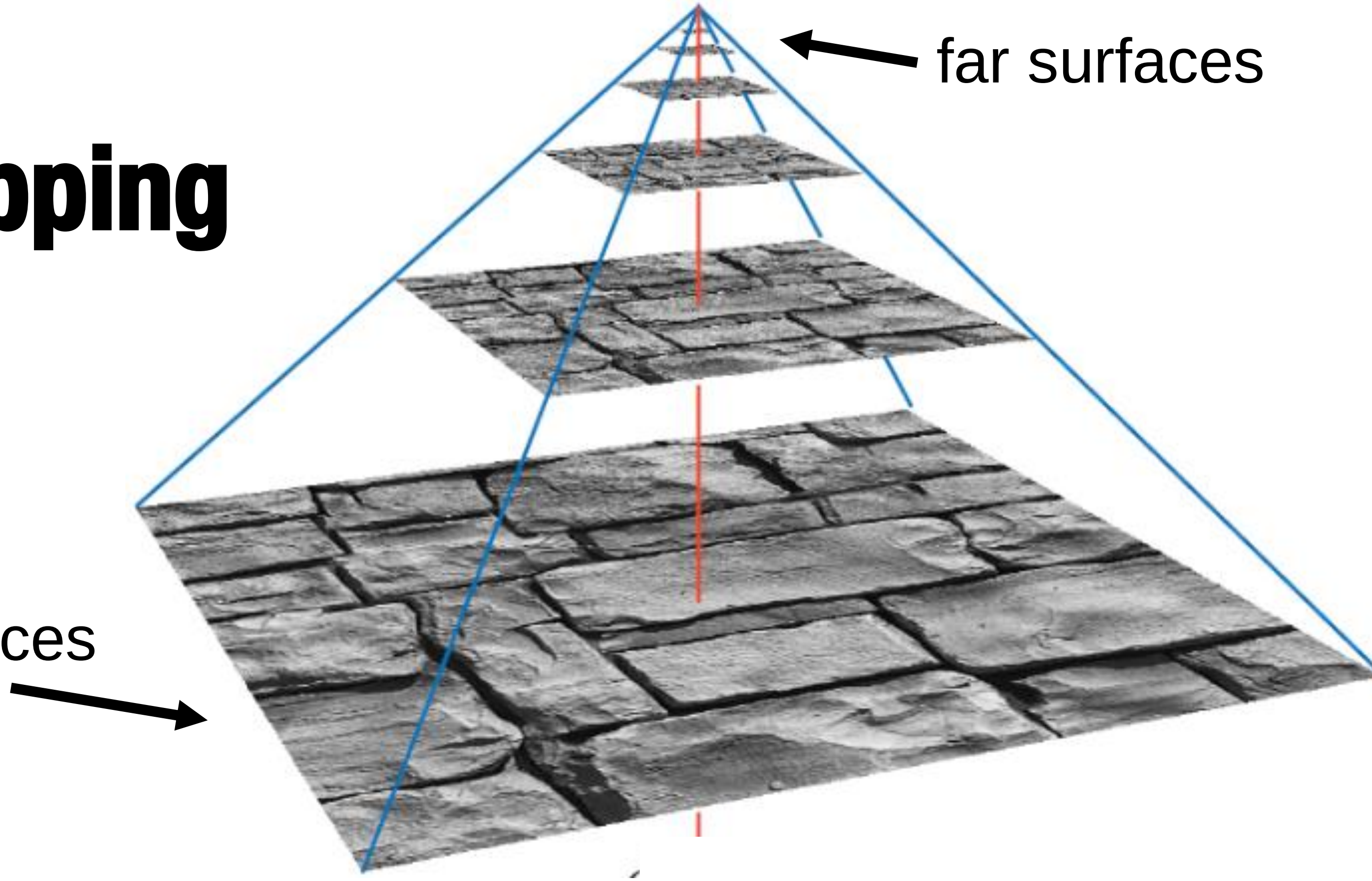
Main idea: store hierarchy of subsampled textures

Mipmapping



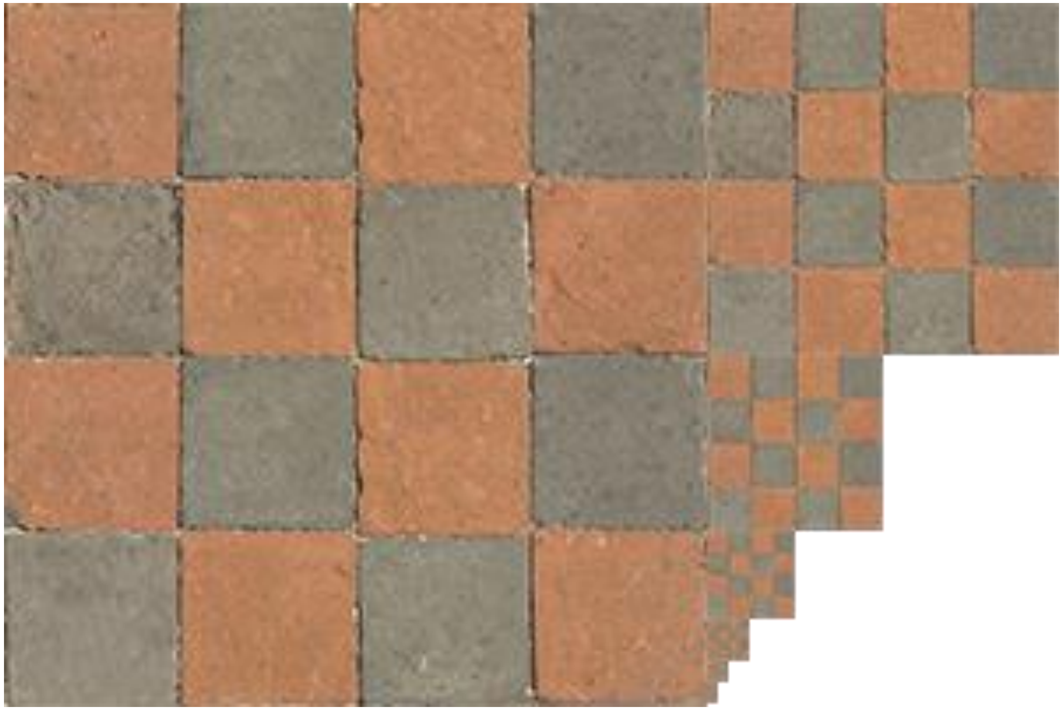
Main idea: store hierarchy of subsampled textures

Mipmapping



How expensive is this?

Mipmapping



~50% more memory consumed

Bilinear Filtering

Average four nearest texels

Eliminates “blockiness”/pixellation

Trilinear Filtering

Classic problem in games: popping

Trilinear Filtering

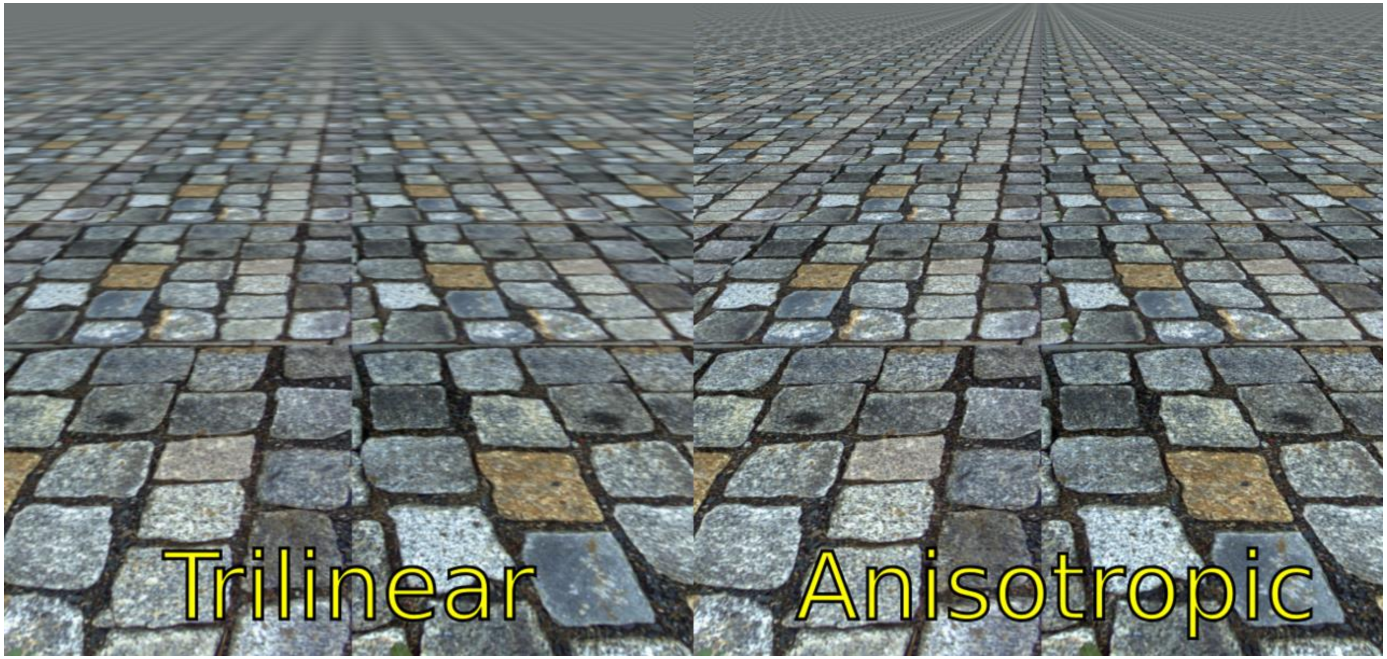
Classic problem in games: popping

different mipmap levels

Can fix by averaging neighboring levels

Anisotropic Filtering

Use non-square pyramid levels



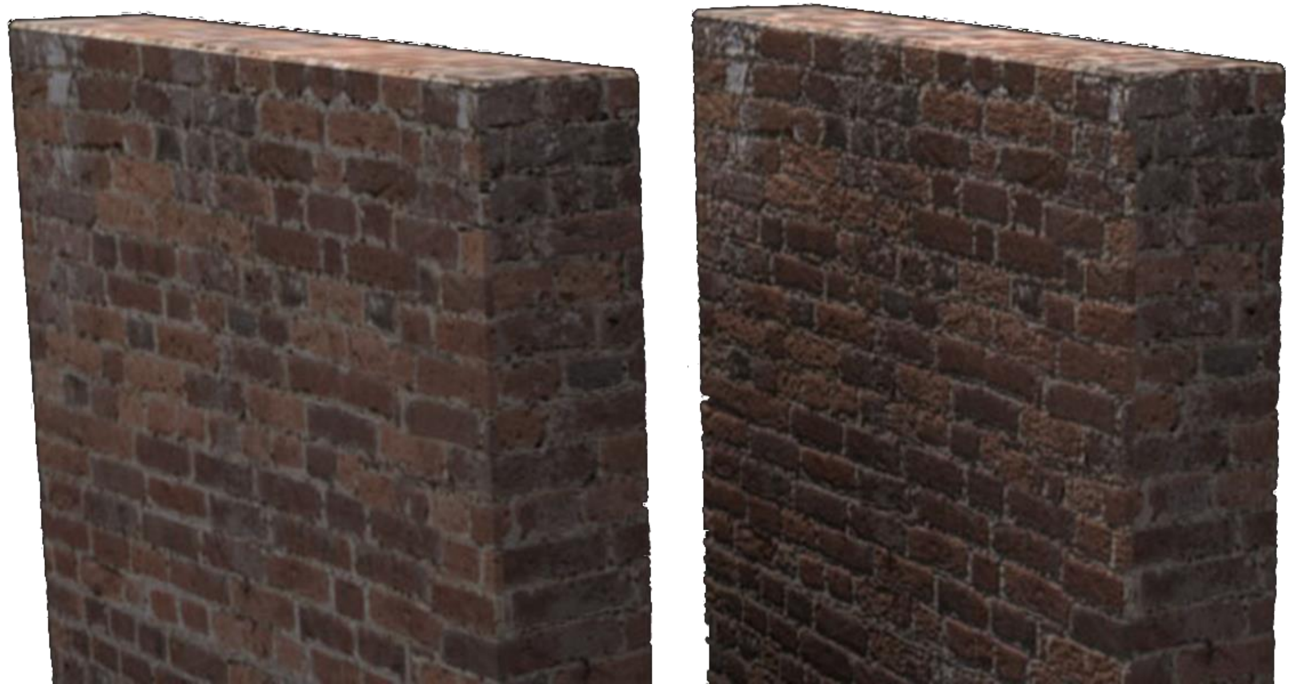
Compute them on the fly

Texture Mapping Flaws

Texture map adds fake geometric detail

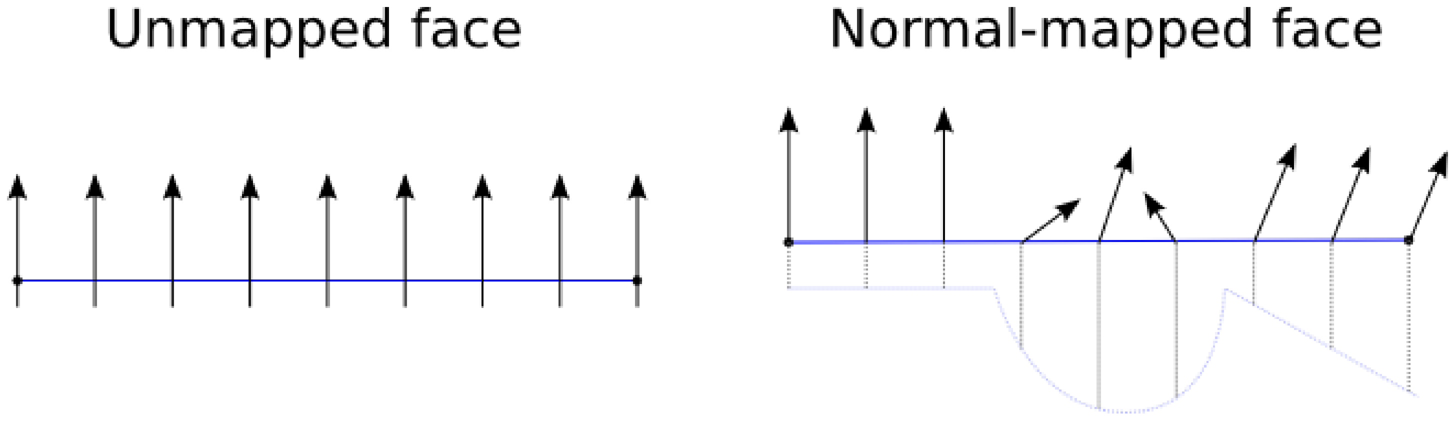
- but still looks **flat**

Why?



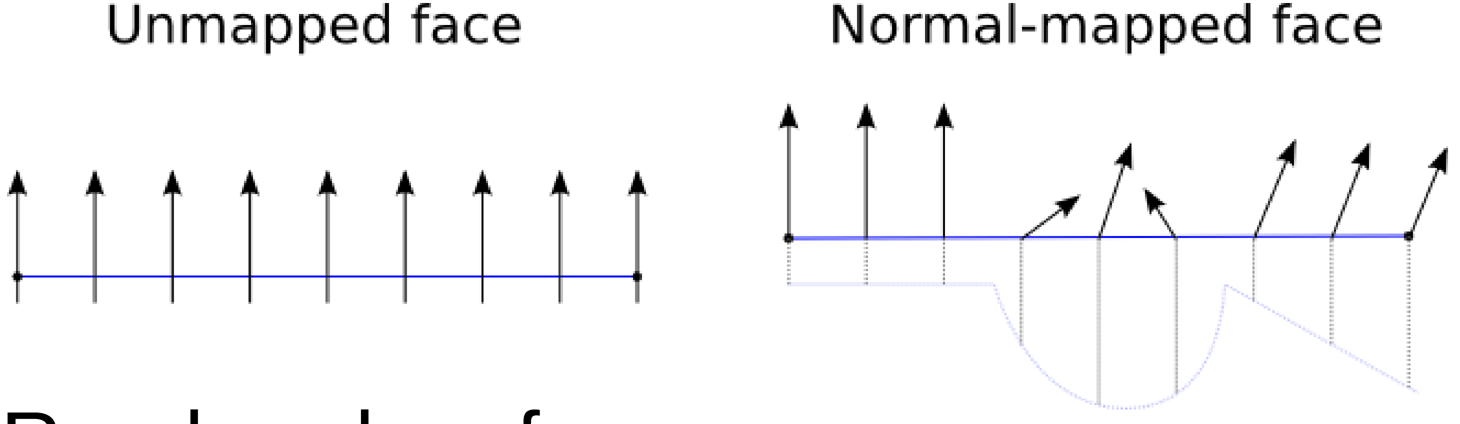
Normal Map

Key idea: modify normals of flat face



Normal Map

Key idea: modify normals of flat face

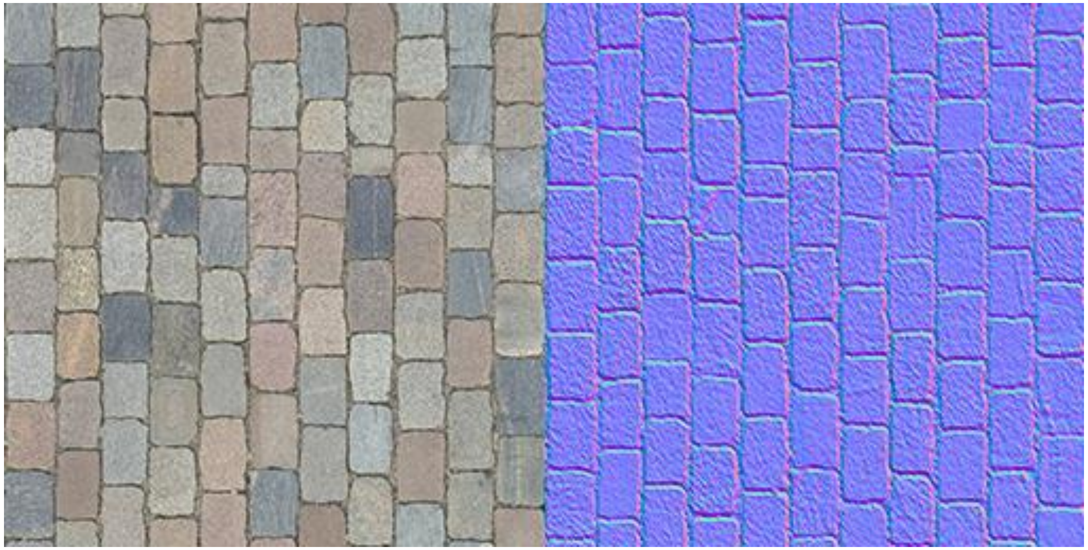


Rendered surface

- **is** flat
- shaded as if it were bumpy

Normal Map

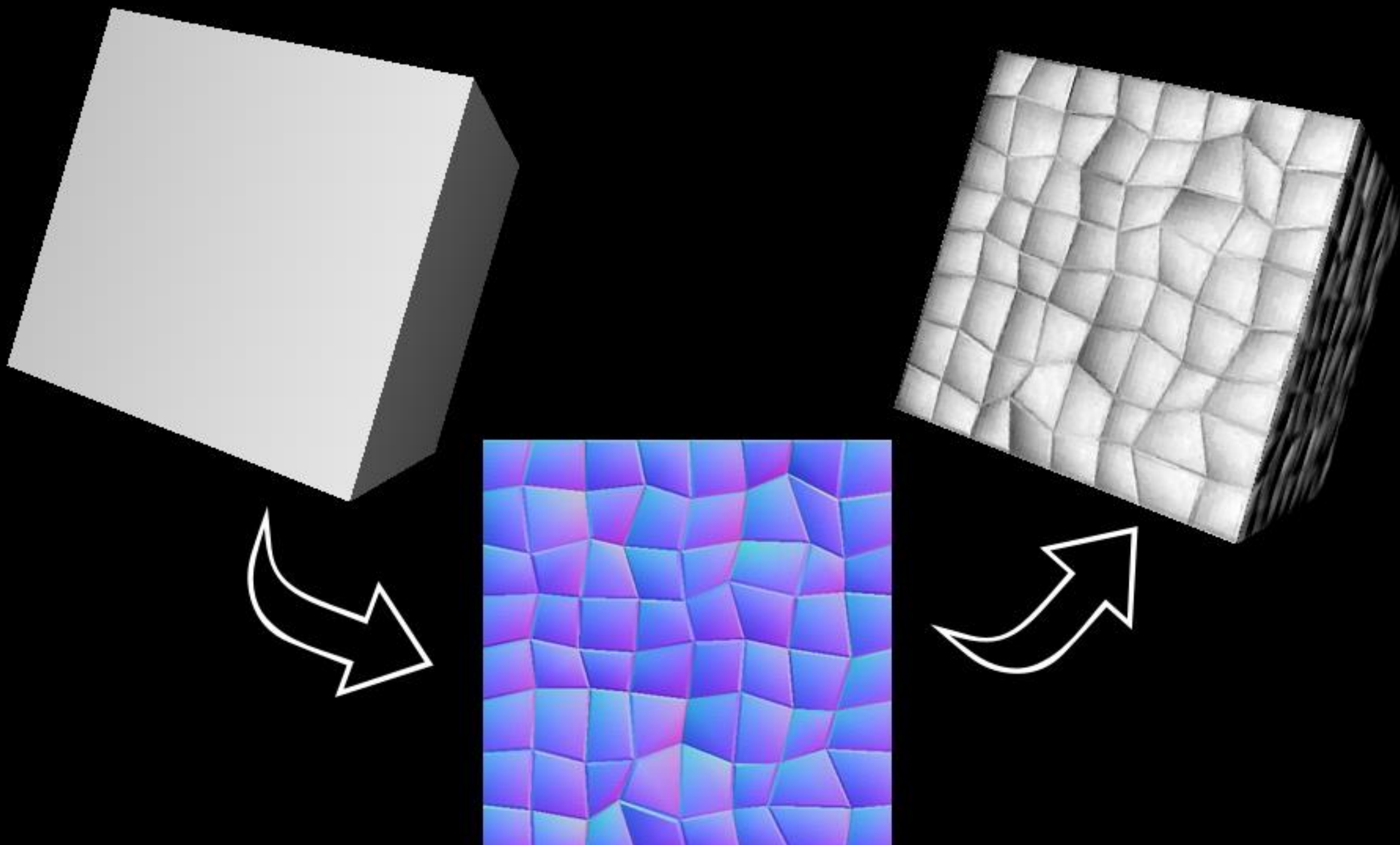
How to represent normals?



Encode as second texture (same size)

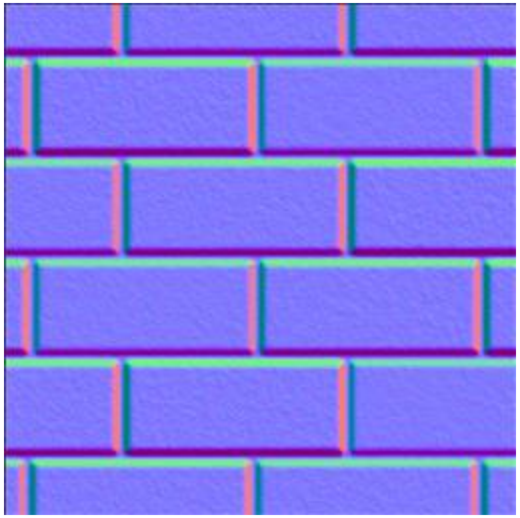
- (r,g,b) encodes coordinates of normal

Applying Normal Map



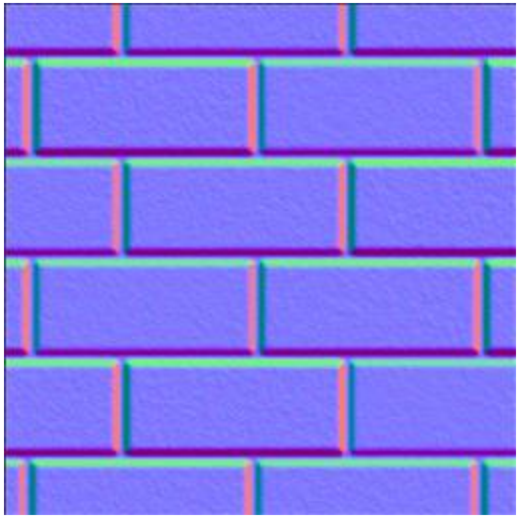
Hold On

What coordinate system does normal map use?



Hold On

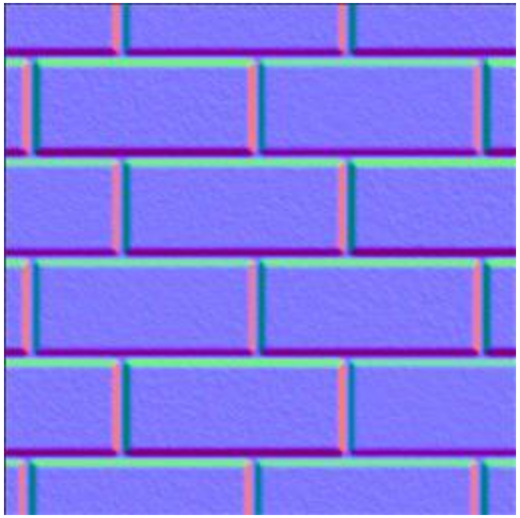
What coordinate system does normal map use?



$$(r, g, b) \mapsto r\hat{u} + g\hat{v} + b\hat{n}$$

Hold On

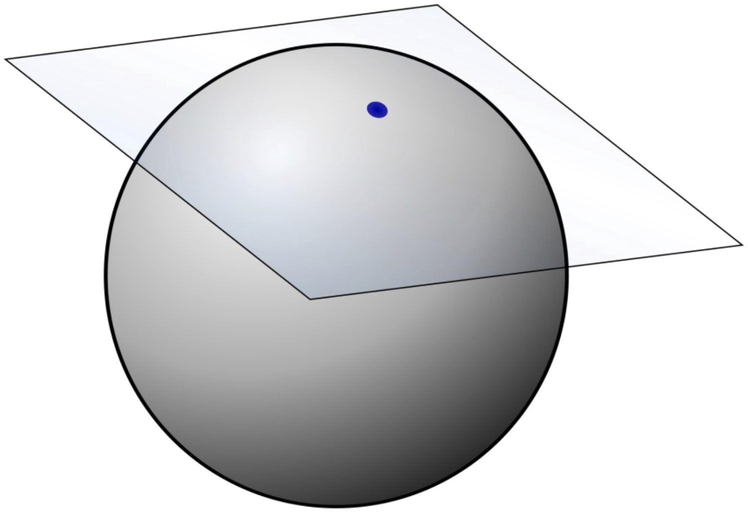
What coordinate system does normal map use?



$$(r, g, b) \mapsto r\hat{u} + g\hat{v} + b\hat{n}$$

texture or **tangent space** coordinates

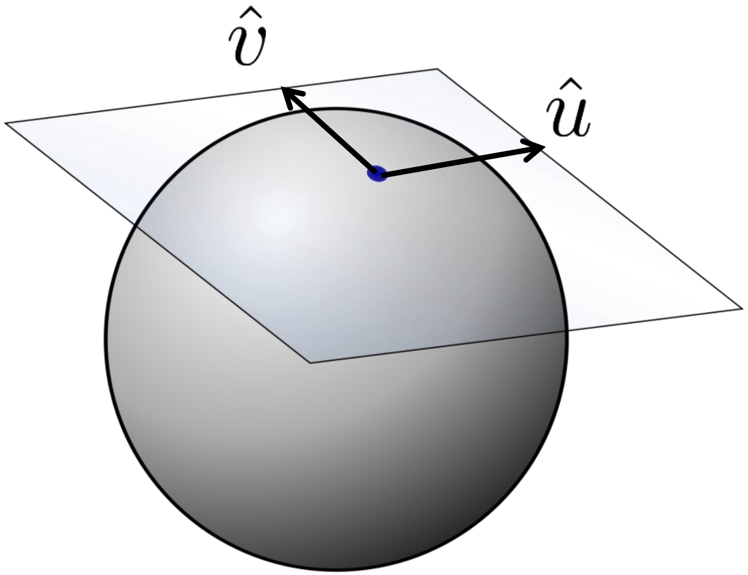
Tangent Space



At every point of surface:

- one **normal** direction
  - (we know how to compute this)

Tangent Space

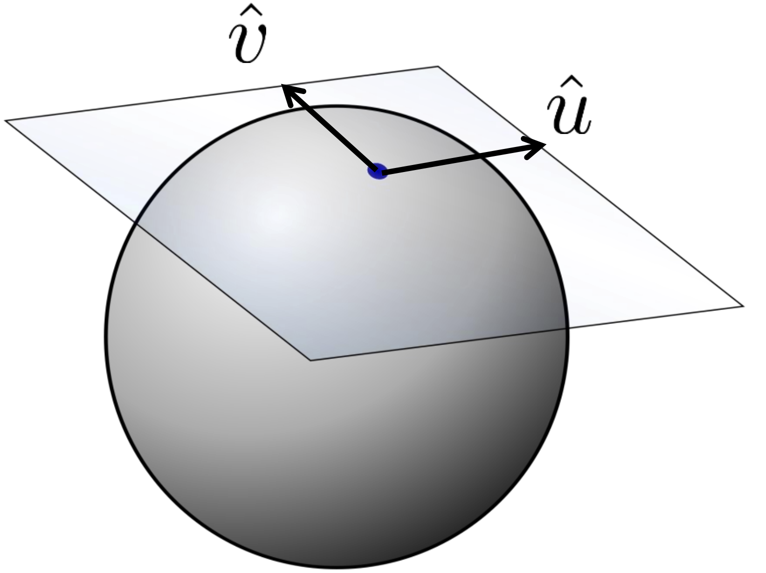


At every point of surface:

- one **normal** direction
  - (we know how to compute this)

- two-dimensional **tangent space**
  - how to pick basis vectors?

Tangent Space



At every point of surface:

- one **normal** direction
  - (we know how to compute this)

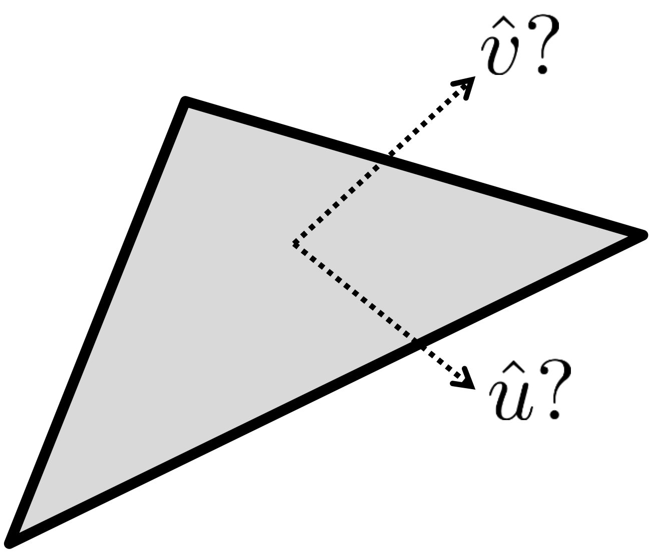
- two-dimensional **tangent space**
  - how to pick basis vectors?
  - many possible ways (parameterization lecture)

Texture Coordinates

What are basis vectors
at each point?

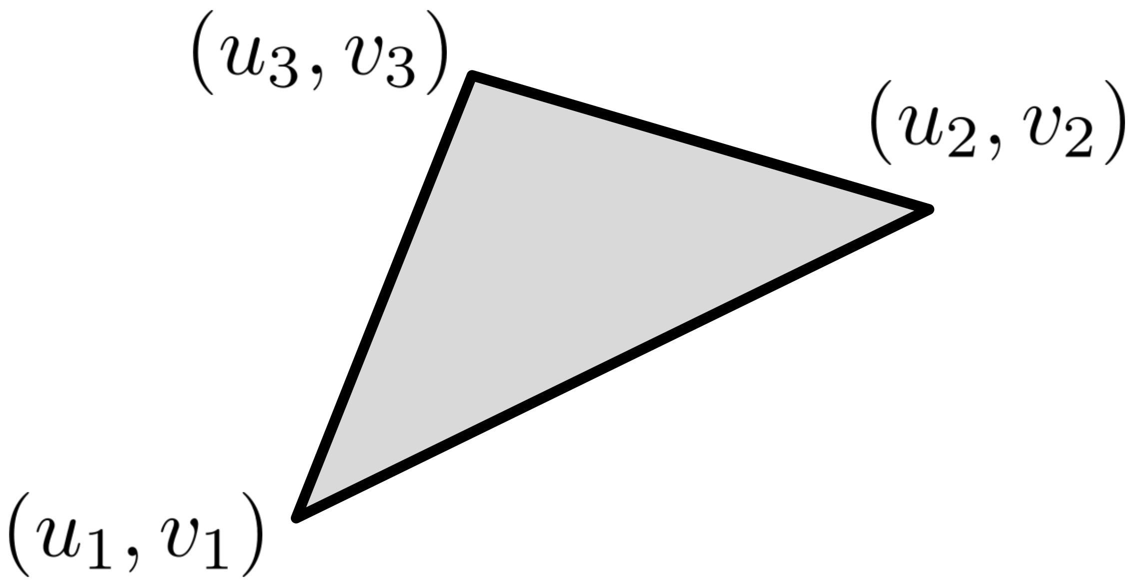
Computing Tangent Basis

On each triangle, **precompute** u and v directions



Computing Tangent Basis

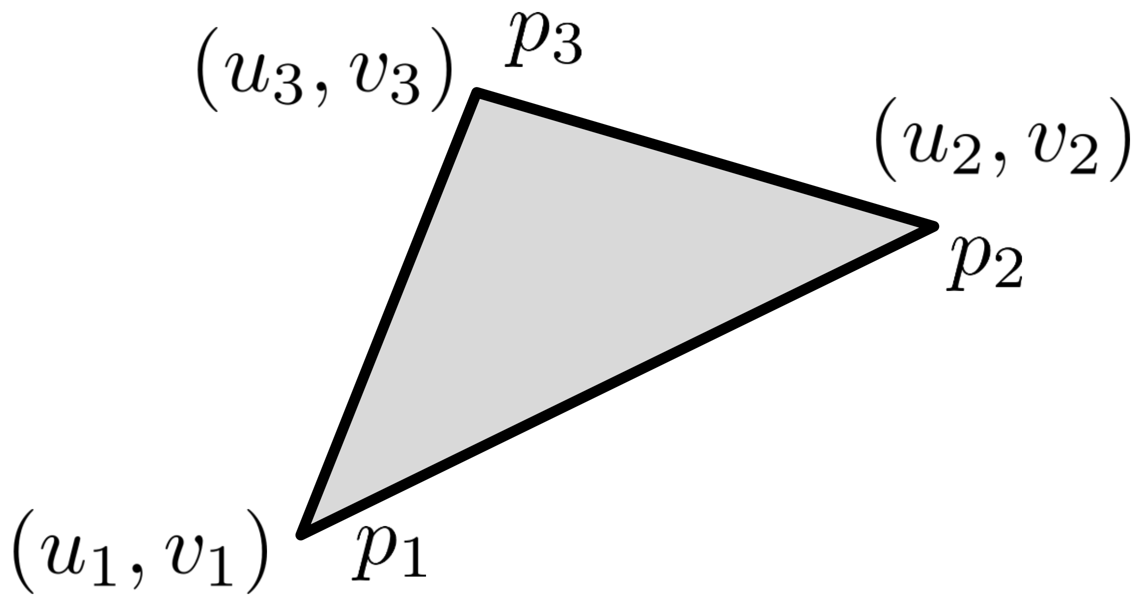
On each triangle, **precompute** u and v directions



Already know: three (u,v) pairs from parameterization

Computing Tangent Basis

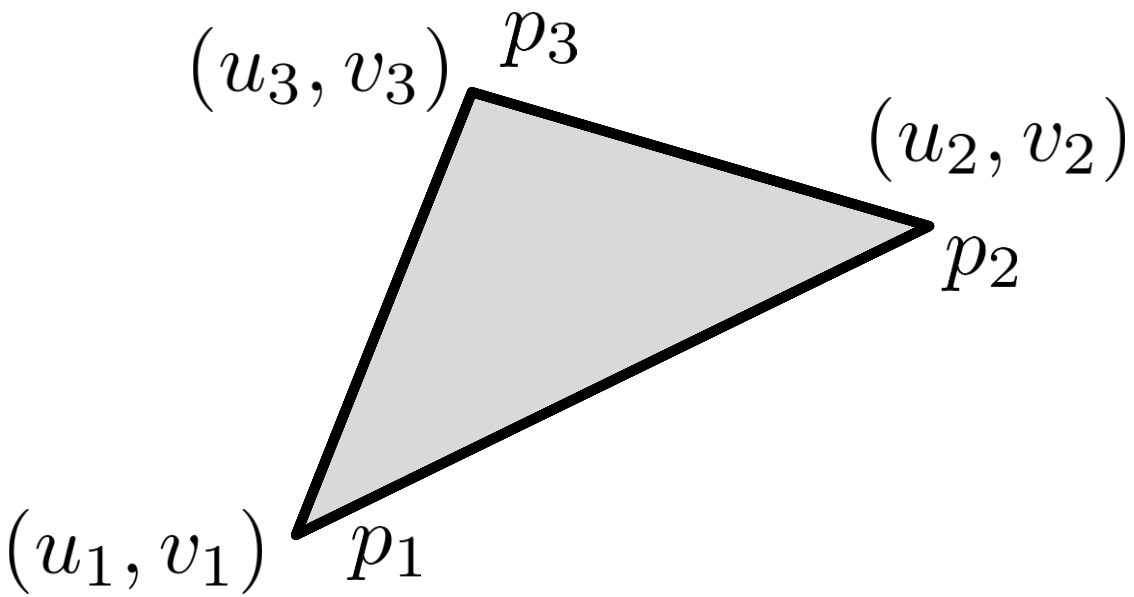
On each triangle, **precompute** u and v directions



Already know: vertex positions

Computing Tangent Basis

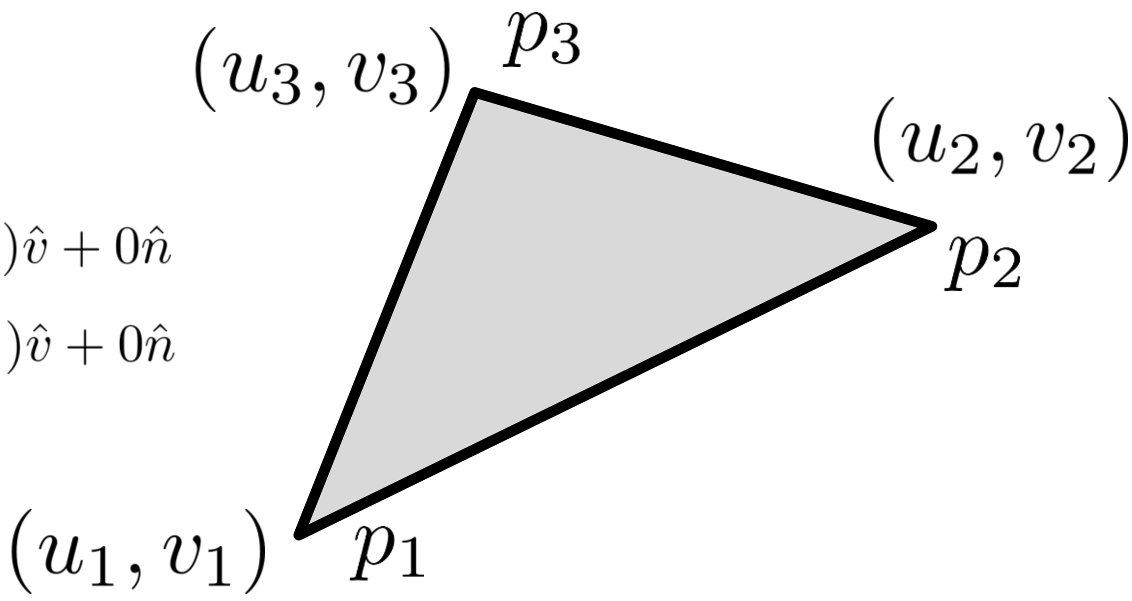
On each triangle, **precompute** u and v directions



$$p_2 = p_1 + (u_2 - u_1)\hat{u} + (v_2 - v_1)\hat{v}$$
$$p_3 = p_1 + (u_3 - u_1)\hat{u} + (v3 - v_1)\hat{v}$$

Computing Tangent Basis

On each triangle, **precompute** u and v directions



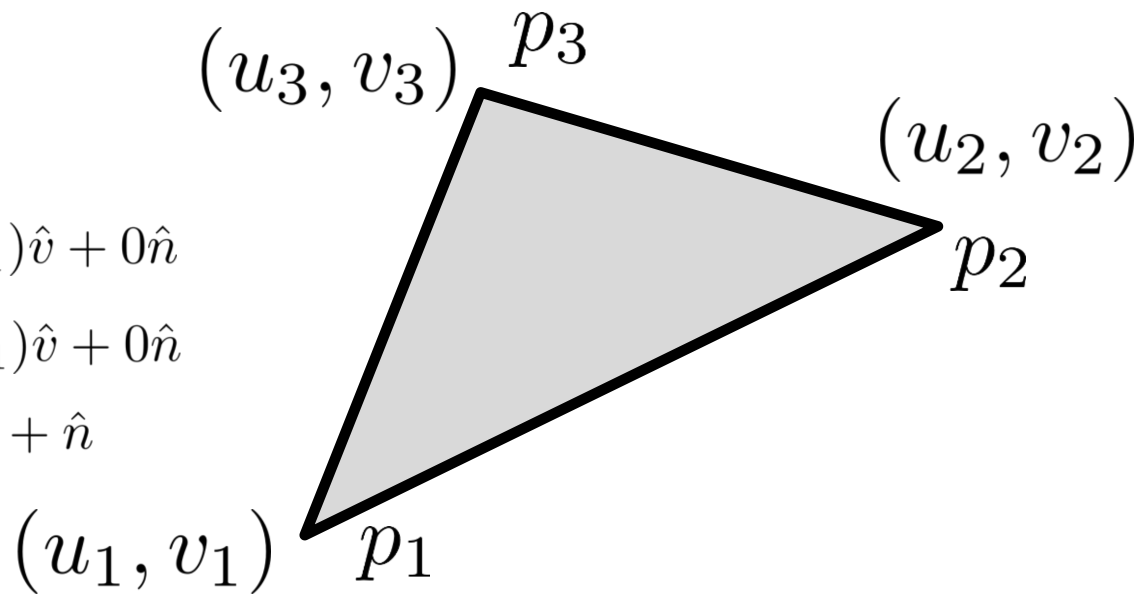
$$p_2 = p_1 + (u_2 - u_1)\hat{u} + (v_2 - v_1)\hat{v} + 0\hat{n}$$

$$p_3 = p_1 + (u_3 - u_1)\hat{u} + (v_3 - v_1)\hat{v} + 0\hat{n}$$

Need third equation…

Computing Tangent Basis

On each triangle, **precompute** u and v directions



$$p_2 = p_1 + (u_2 - u_1)\hat{u} + (v_2 - v_1)\hat{v} + 0\hat{n}$$

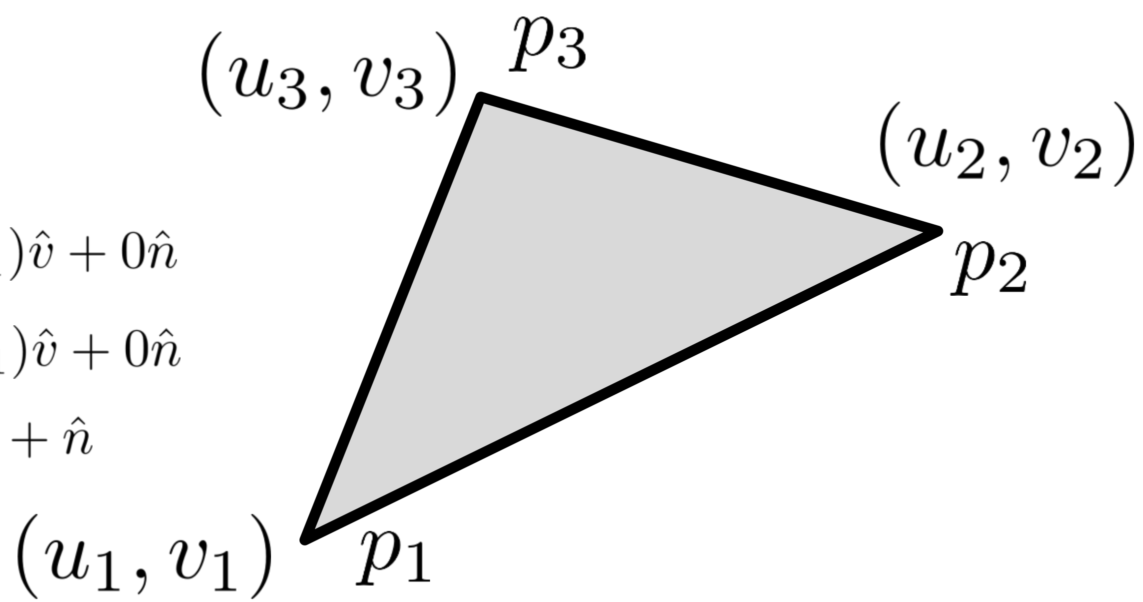
$$p_3 = p_1 + (u_3 - u_1)\hat{u} + (v_3 - v_1)\hat{v} + 0\hat{n}$$

$$\frac{(p_2 - p_1) \times (p_3 \times p_1)}{\|(p_2 - p_1) \times (p_3 \times p_1)\|} = 0\hat{u} + 0\hat{v} + \hat{n}$$

Need third equation…

Computing Tangent Basis

On each triangle, **precompute** u and v directions



$$p_2 = p_1 + (u_2 - u_1)\hat{u} + (v_2 - v_1)\hat{v} + 0\hat{n}$$

$$p_3 = p_1 + (u_3 - u_1)\hat{u} + (v_3 - v_1)\hat{v} + 0\hat{n}$$

$$\frac{(p_2 - p_1) \times (p_3 \times p_1)}{\|(p_2 - p_1) \times (p_3 \times p_1)\|} = 0\hat{u} + 0\hat{v} + \hat{n}$$

$$\left[\begin{array}{c|c|c} p_2 - p_1 & p_3 - p_1 & \frac{(p_2 - p_1) \times (p_3 \times p_1)}{\|(p_2 - p_1) \times (p_3 \times p_1)\|} \end{array}\right] = \left[\begin{array}{c|c|c} \hat{u} & \hat{v} & \hat{n} \end{array}\right] \begin{bmatrix} u_2 - u_1 & u_3 - u_1 & 0 \\ v_2 - v_1 & v_3 - v_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Bump Mapping

Older technique: give **offset height** only

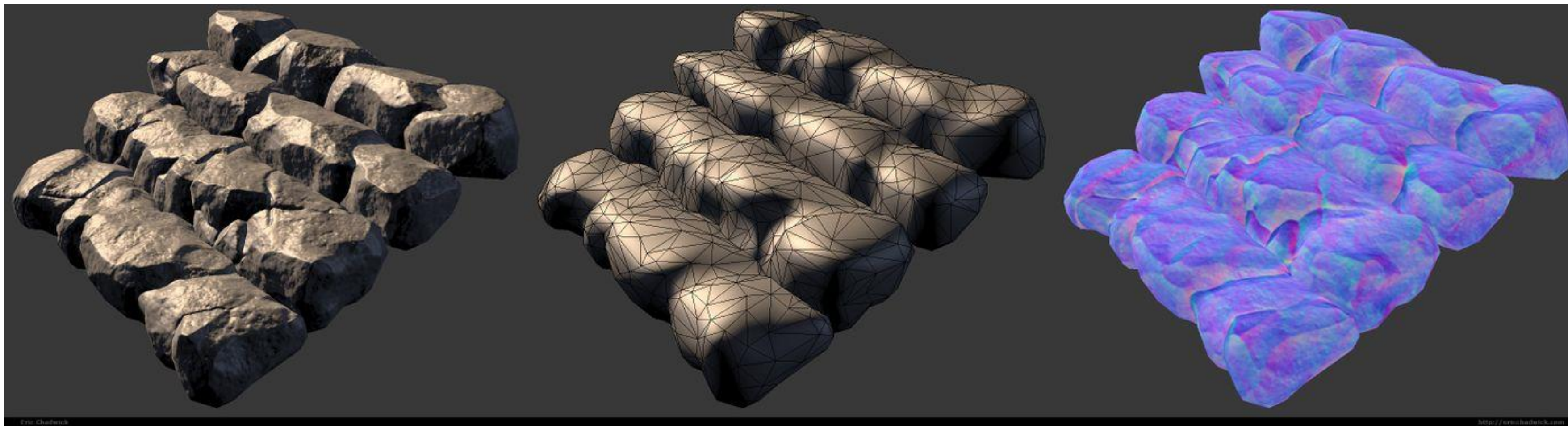
Bump Mapping

Older technique: give **offset height** only

Less flexible than normal map

To use, convert to normal map

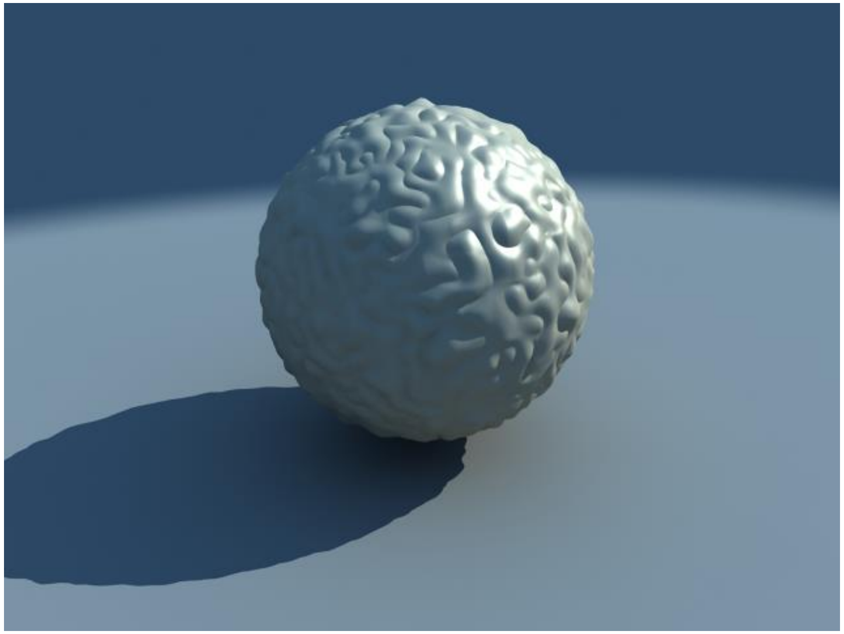
More Examples



Displacement Map

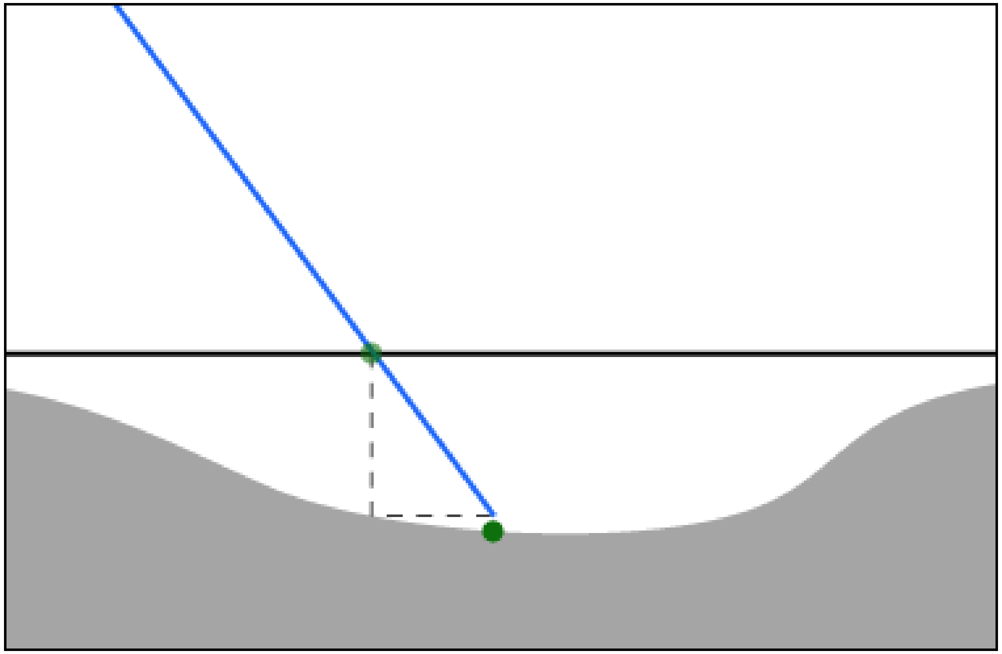
Like normal map, but change normals **and** geometry

- Fully correct
- **Slow**



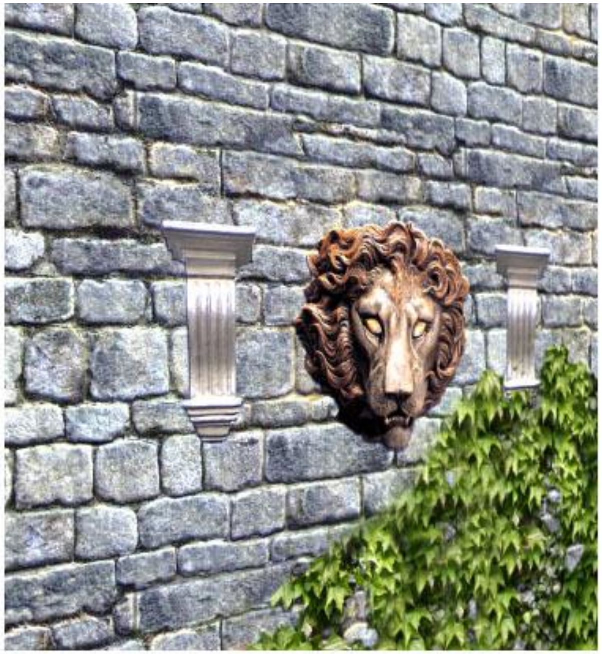
Parallax Map

Take into account **shift in texture coordinates**



Parallax Map Example

Texture Mapped



Normal Mapped

Parallax Mapped