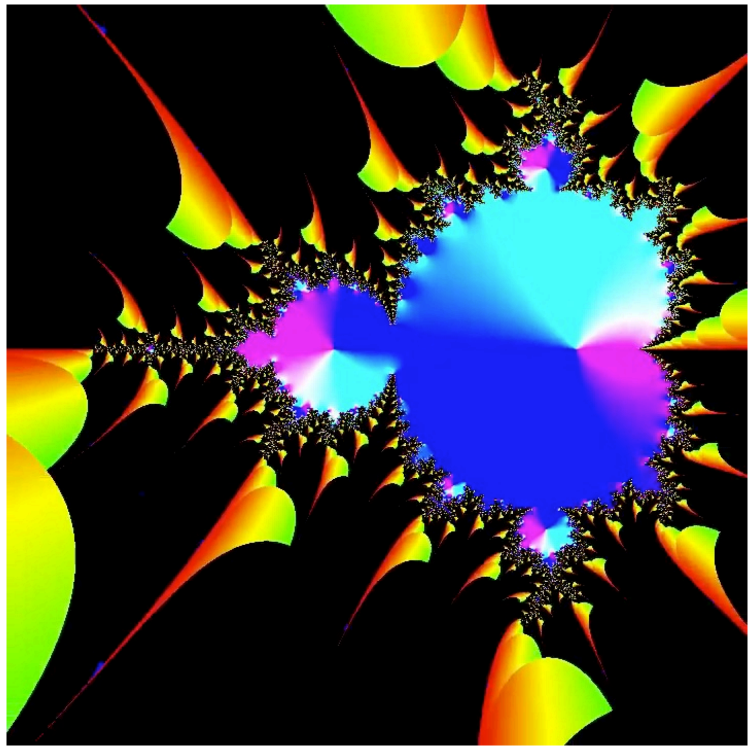
TSBK 07

Computer Graphics

Ingemar Ragnemalm, ISY



Lecture 6

Texture mapping

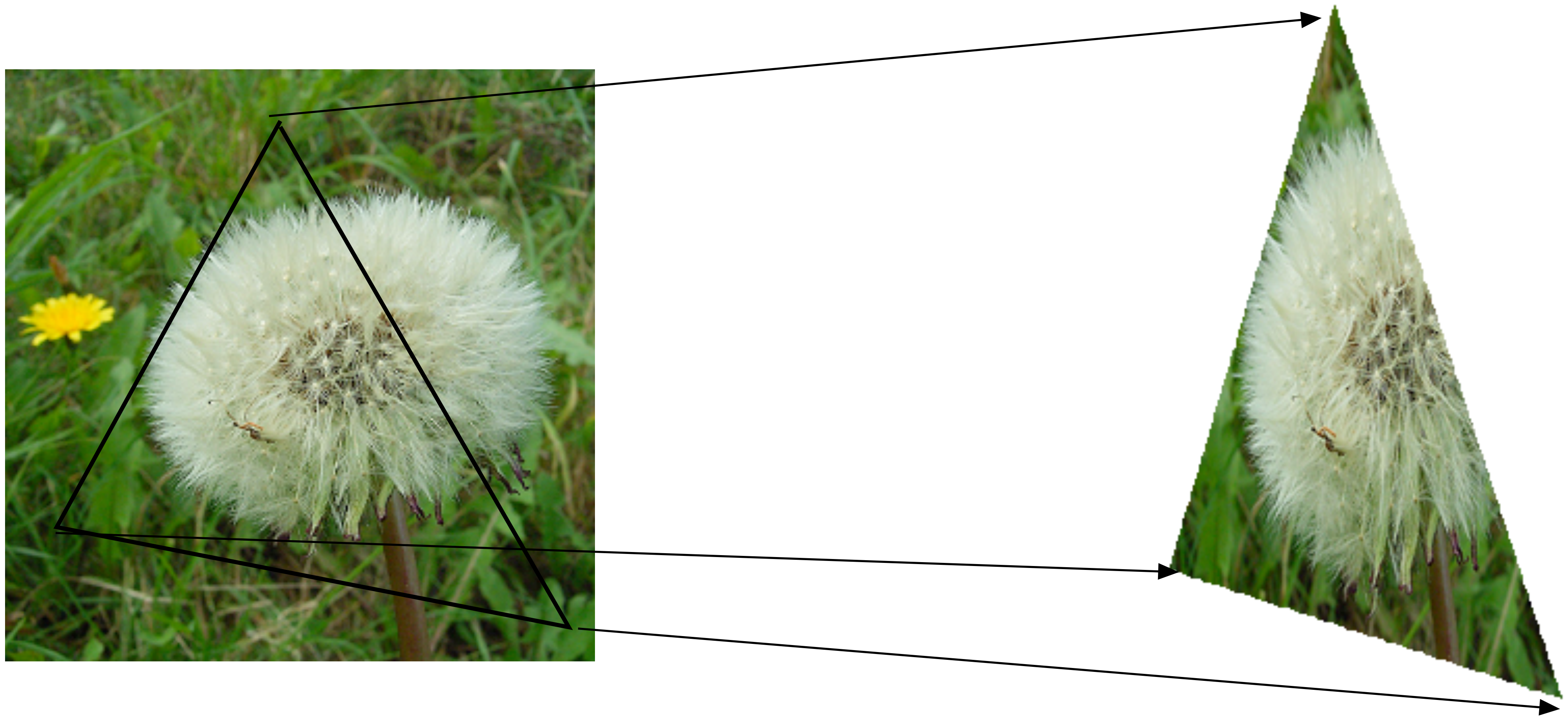
Skyboxes

Environment mapping

Bump mapping

Texture mapping

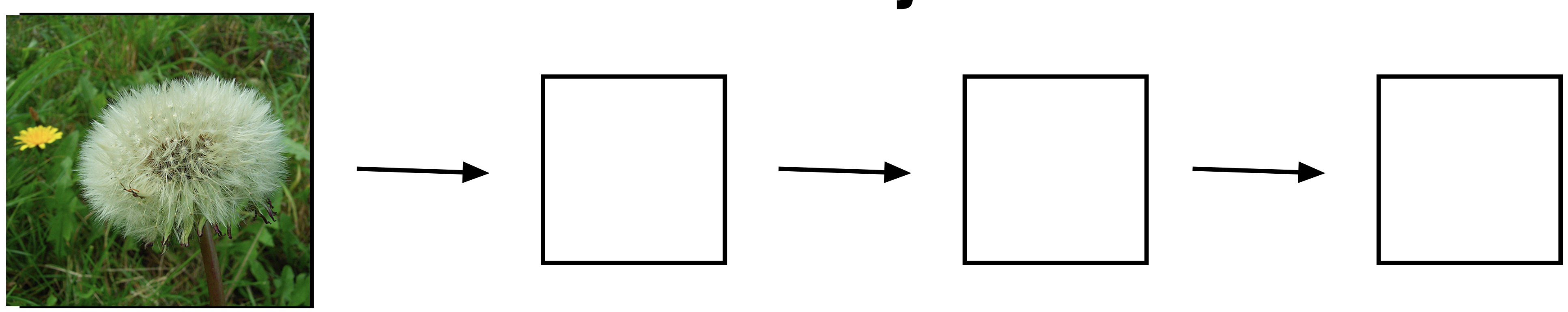
Texture coordinates from 0 to 1



Texture units

Textures are bound to ”texture units”, hardware resources for looking up textures

The shader uses the texture unit ID, not the texture object!

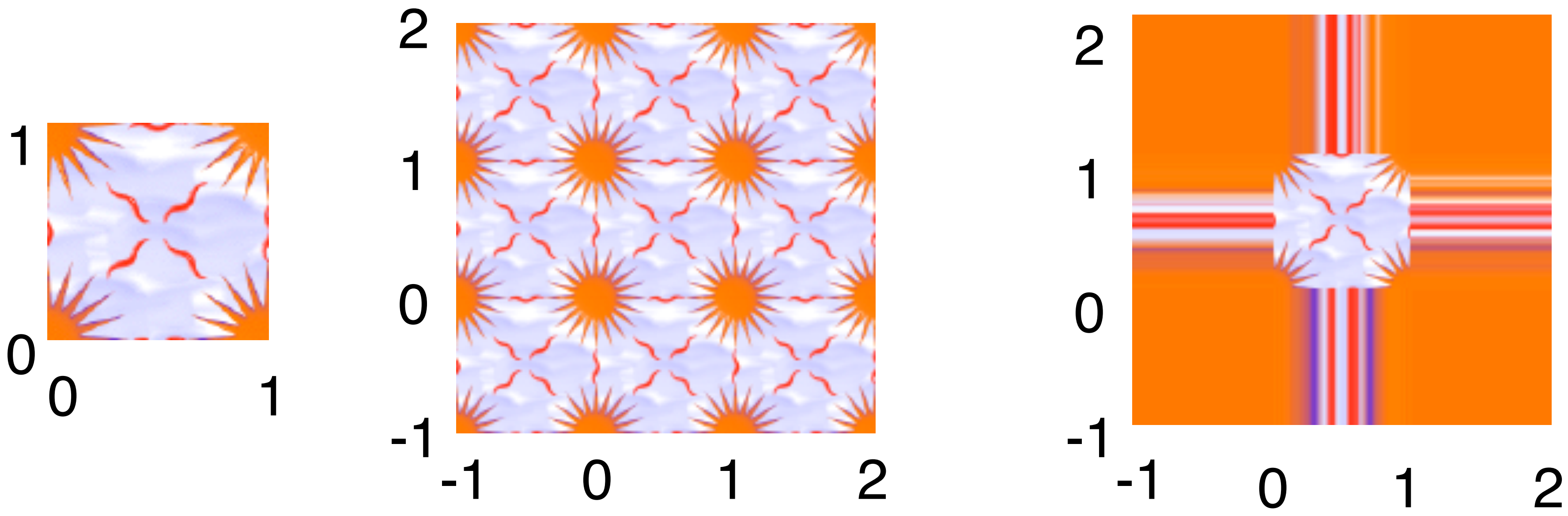


Texture parameters

glTexParameter(...);

GL_TEXTURE_WRAP_S
GL_TEXTURE_WRAP_T

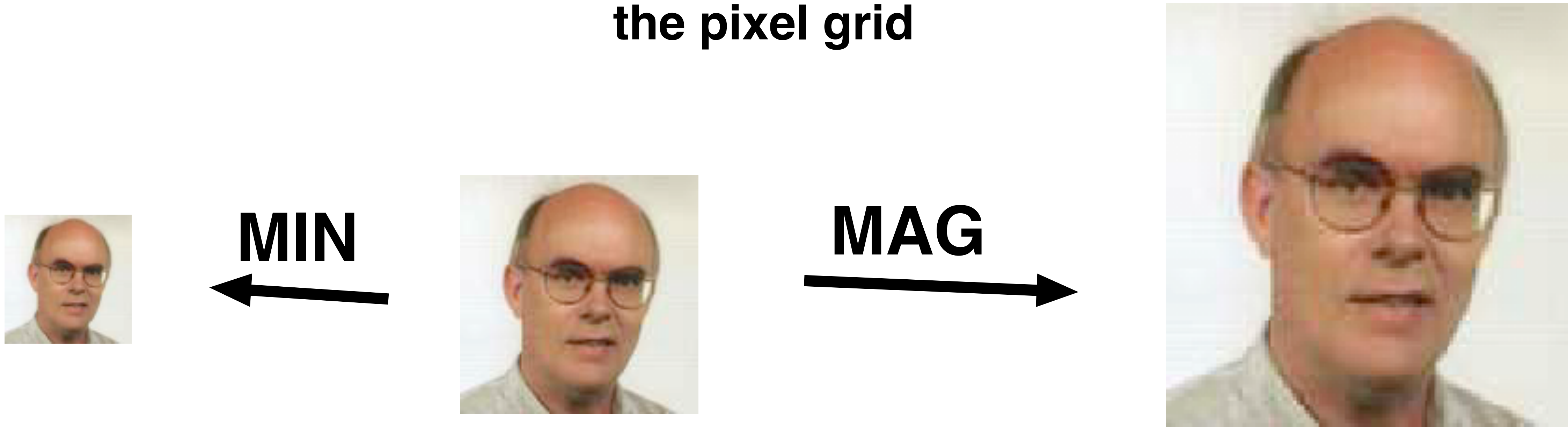
GL_REPEAT
GL_CLAMP_TO_EDGE



Magnification and minification parameters:

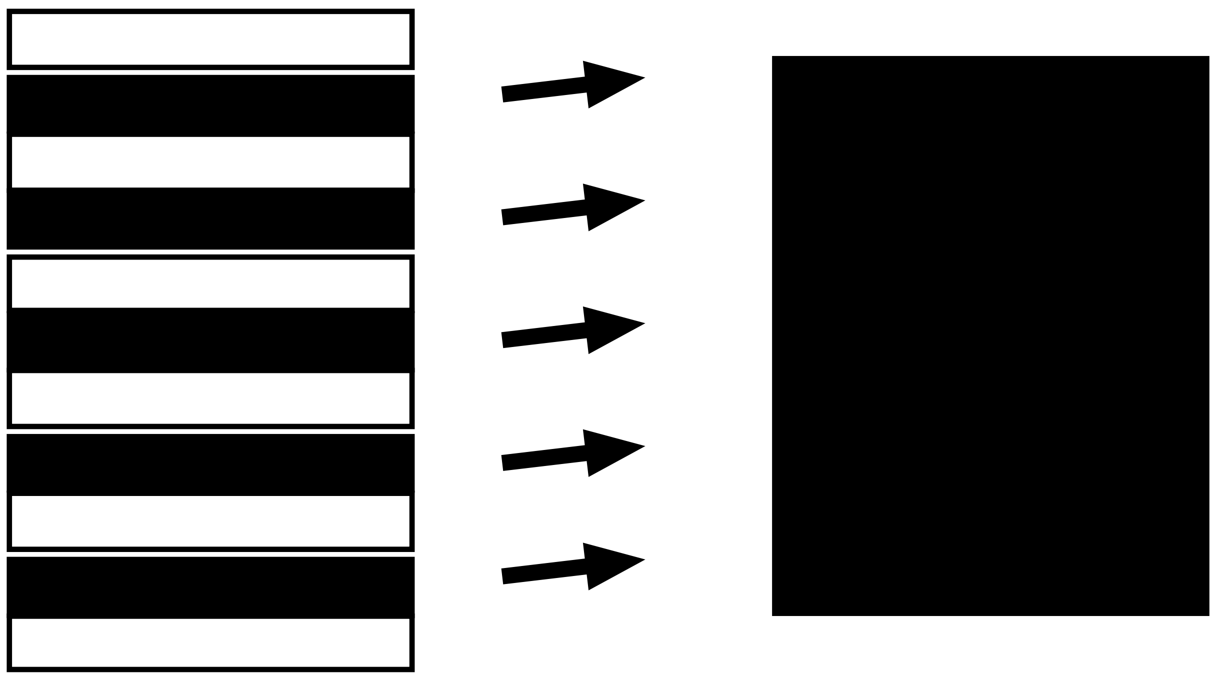
```
glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER, GL_NEAREST);

glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

Specifies what should happen when the texture doesn't match the pixel grid



Aliasing

A digital image is a sampled signal
If the signal is not band limited, aliasing will occur



Aliasing in texture mapping

At large distance, textures get smaller

=>

higher spatial frequencies on the screen

=>

increasing risk for aliasing!

Aliasing can be reduced by two methods:

Filtering

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
```

Mip-mapping

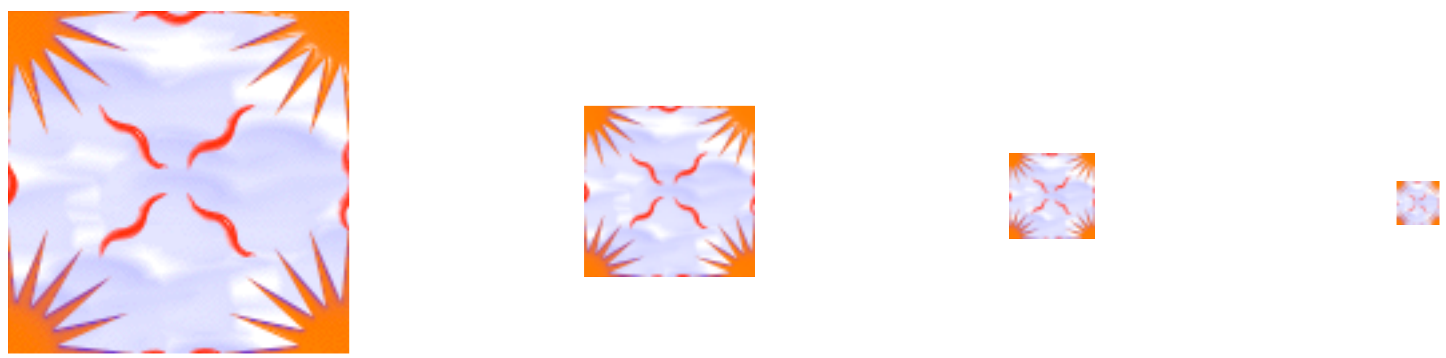
```
glGenerateMipmap();

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_NEAREST);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_LINEAR);
```

MIP mapping

Texture mapping with anti-aliasing.

A resolution pyramid is built from every texture.

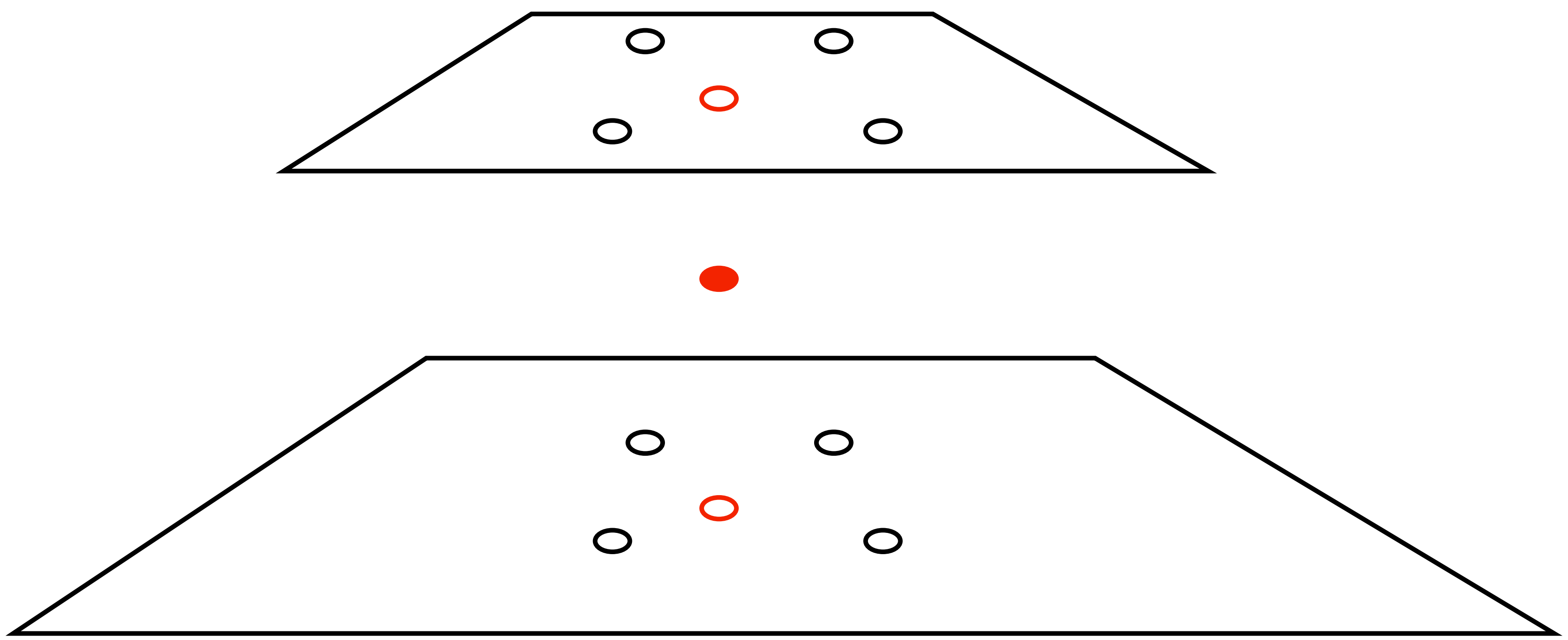
Memory cost: 33% more. Cheap!



128x128 64x64 32x32 16x16

MIP mapping filtering

Both within a level and between!



MIP mapping filtering

GL_NEAREST
GL_LINEAR
GL_NEAREST_MIPMAP_NEAREST
GL_LINEAR_MIPMAP_NEAREST
GL_NEAREST_MIPMAP_LINEAR
GL_LINEAR_MIPMAP_LINEAR

Preferred:
GL_LINEAR for magnification
GL_LINEAR_MIPMAP_LINEAR for minification

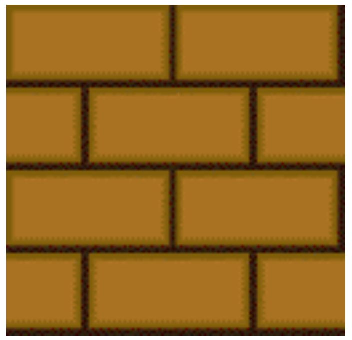
MIP mapping

Gives anti-aliasing at a very low cost.

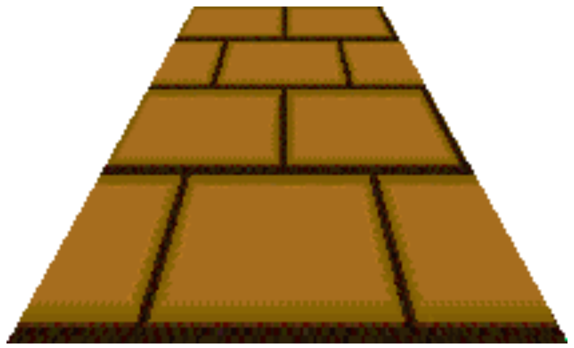
Good results in most situations.

Aliasing problems remain at steep angles.

Texture mapping is not just stretching:

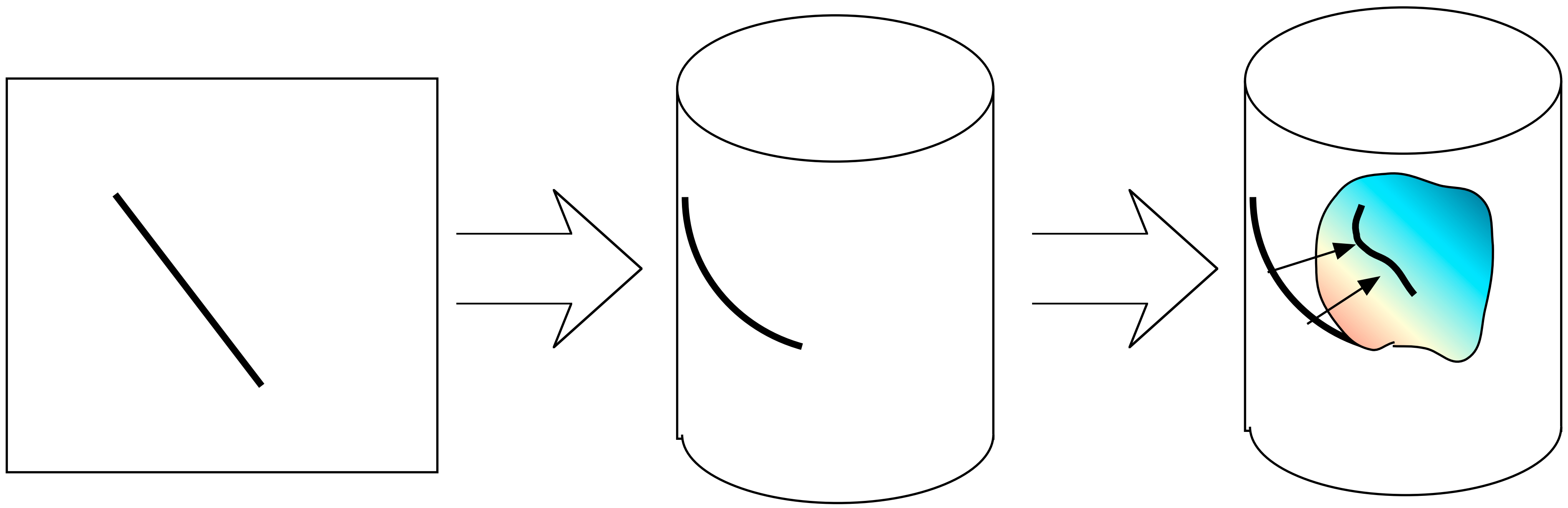


Affine



Perspective correct

Pre-calculating (s,t) for every vertex in a model

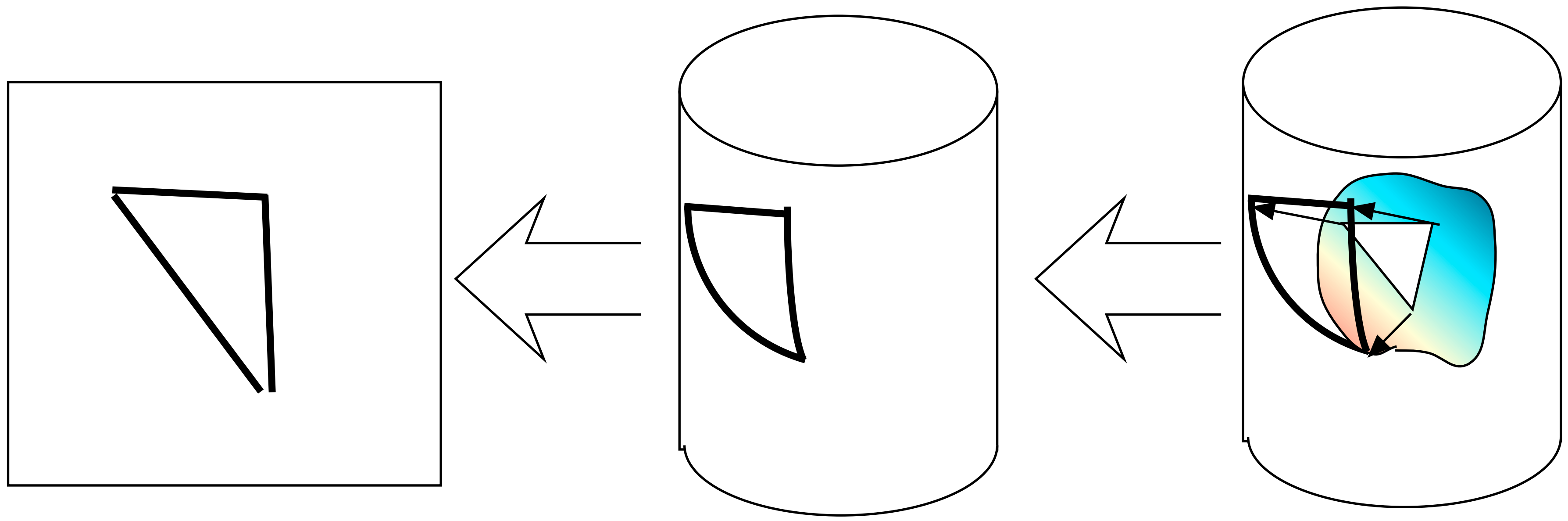


Texture map (s,t)

Map to intermediate surface (u,v)

Map to 3D object (x, y, z)

Pre-calculating (s,t) for every vertex in a model



Examples

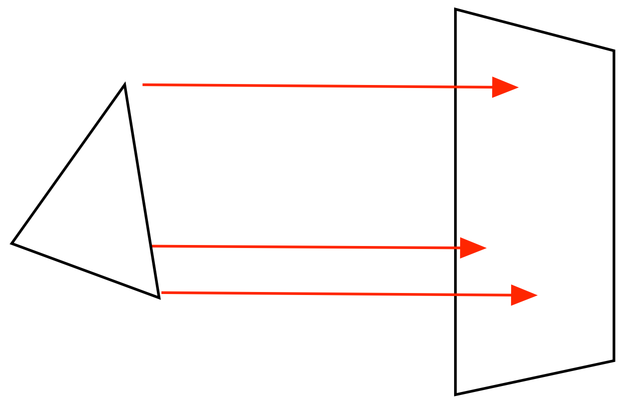
Planar

Cylinder

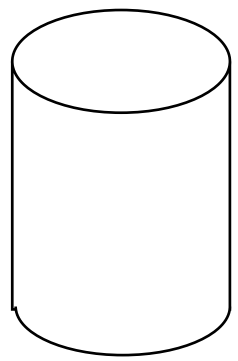
Sphere

Common mappings

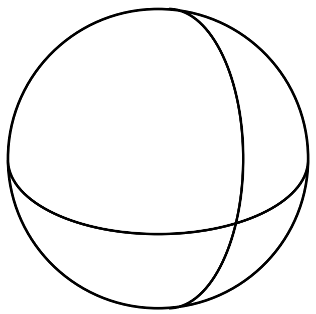
Planar



Cylinder

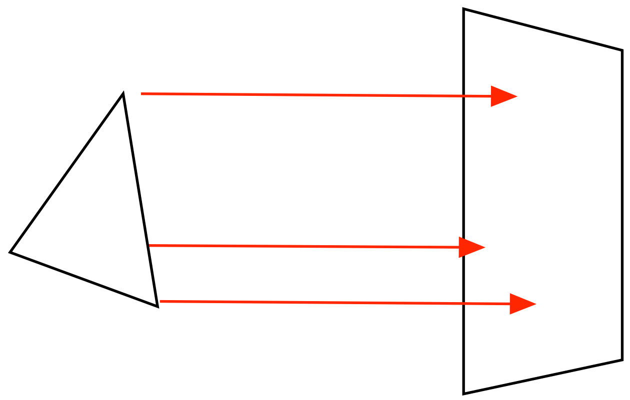


Sphere



Planar texture mapping

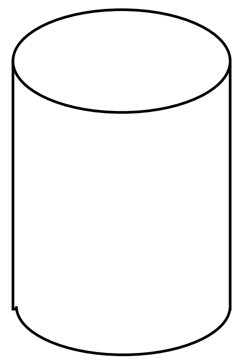
Along z:
u = x
v = y



Cylindrical texture mapping

$x = R\cos(\theta)$
$y = R\sin(\theta)$

$u = \theta = \arctan(y,x)$
$v = z$

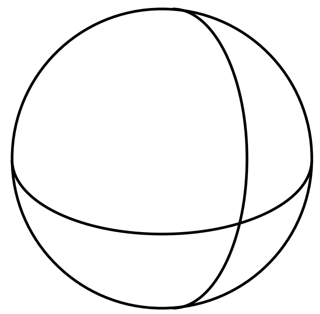


arctan(y,x) =
x > 0: tan-1(y/x)
x<0: π + tan-1(y/x)
x = 0, y > 0: π/2
x = 0, y < 0: -π/2

Spherical texture mapping

$x = R\cos(\phi)\cos(\theta)$
$y = R\cos(\phi)\sin(\theta)$
$z = R\sin(\phi)$

$u = \arctan(y,x)$
$v = \sin^{-1}(z/R)$



(Swap $\cos(\phi)$ and $\sin(\phi)$ if you define ϕ from the z axis rather than from the equator!)

$\arctan(y,x) =$
$x > 0$: $\tan^{-1}(y/x)$
$x<0$: $\pi + \tan^{-1}(y/x)$
$x = 0, y > 0$: $\pi/2$
$x = 0, y < 0$: $-\pi/2$

(u,v) => (s,t)

Normalize, typically 0 to 1
Example: Cylinder

$$x = R\cos(u)$$
$$y = R\sin(u)$$

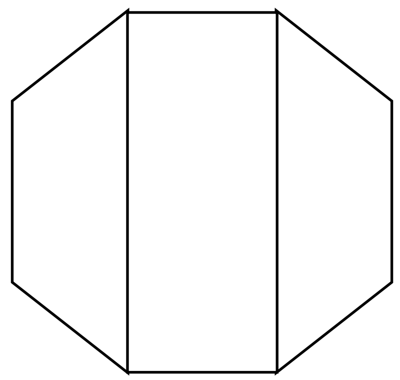
$$u = \arctan(y,x)$$
$$v = z$$

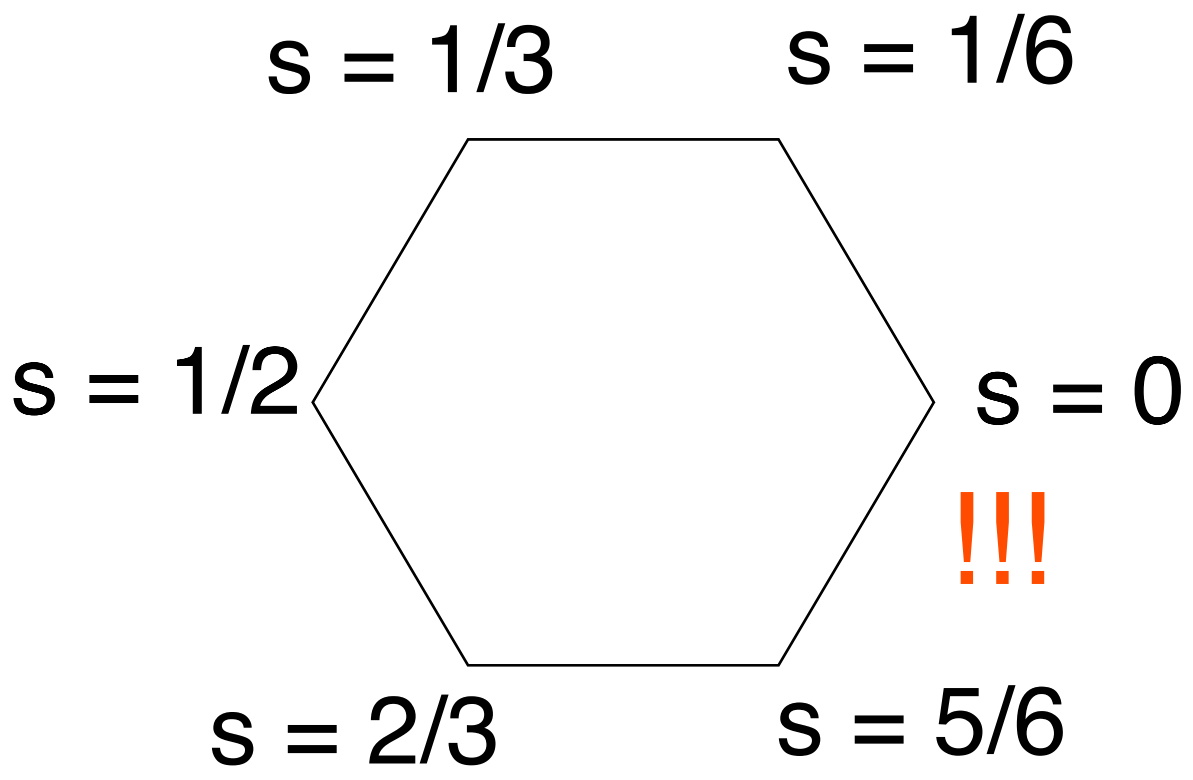
$$s = (u + \pi/2) / 2\pi$$

$$t = (v - z_{min}) / (z_{max} - z_{min})$$

Watch the edges!

Example: six-sided barrel





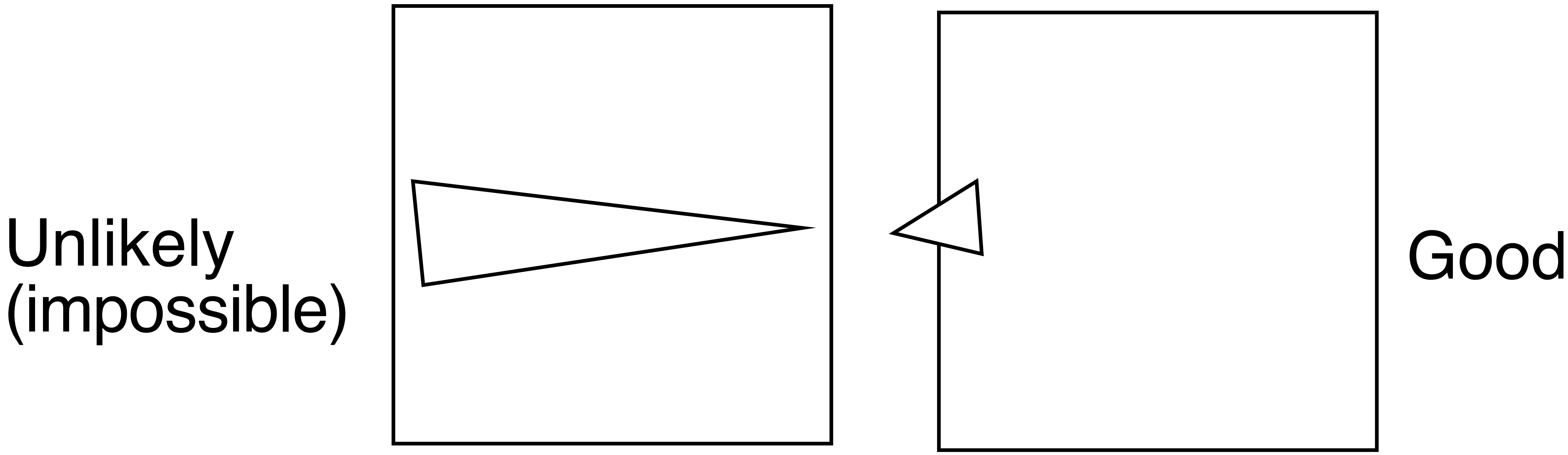
$x = R\cos(u)$
$y = R\sin(u)$

$u = \arctan(y,x)$
$v = z$

$s = (u + \pi/2) / 2\pi$

$t = (v - z_{min}) / (z_{max} - z_{min})$

Problem is detected by checking proximity to the edge



Duplicate vertices to solve

Multitexturing

The GPU has several texture units. Each texture unit can have one texture attached to it, available to the shader.

glActiveTexture() selects a texture unit to work on. Then glBindTexture can bind a texture to that unit.

When needed, pass several texture coordinates arrays

Multitexturing

Applications:

Bump mapping
Gloss mapping
Light mapping
Blending between textures (e.g. mountains)
Putting non-texture data in textures

Any case where you need more data than a single texture!

Multitexturing in GLSL

- Bind one texture per texturing unit
- Pass GLSL unit number and name
  - Declare as samplers in GLSL

Multitexturing in GLSL

Bind textures

Pass unit numbers

Declare as samplers

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(flowerTex);
glActiveTexture(GL_TEXTURE1);
glBindTexture(bumpTex);

glUseProgramObject(shader);
GLint loc = glGetUniformLocation(shader, "flowerTex");
glUniform1i(loc, 0); // texture unit 0
loc = glGetUniformLocation(shader, "bumpTex");
glUniform1i(loc, 1); // texture unit 1
loc = glGetUniformLocation(shader, "noiseTex");

uniform sampler2D flowerTex;
uniform sampler2D bumpTex;
```

Example: Multitexturing

Simple example: blend depending on model coordinate

```
uniform sampler2D worldTex;
uniform sampler2D flowerTex;
in vec4 pixelPos;
in vec2 texCoord;
out vec4 outColor;

void main()
{
outColor = sin(pixelPos.z*10.0) * texture(flowerTex,
    texCoord.st) + (1.0 - sin(pixelPos.z*10.0)) *
    texture(worldTex, texCoord.st);
}
```

Example: Multitexturing

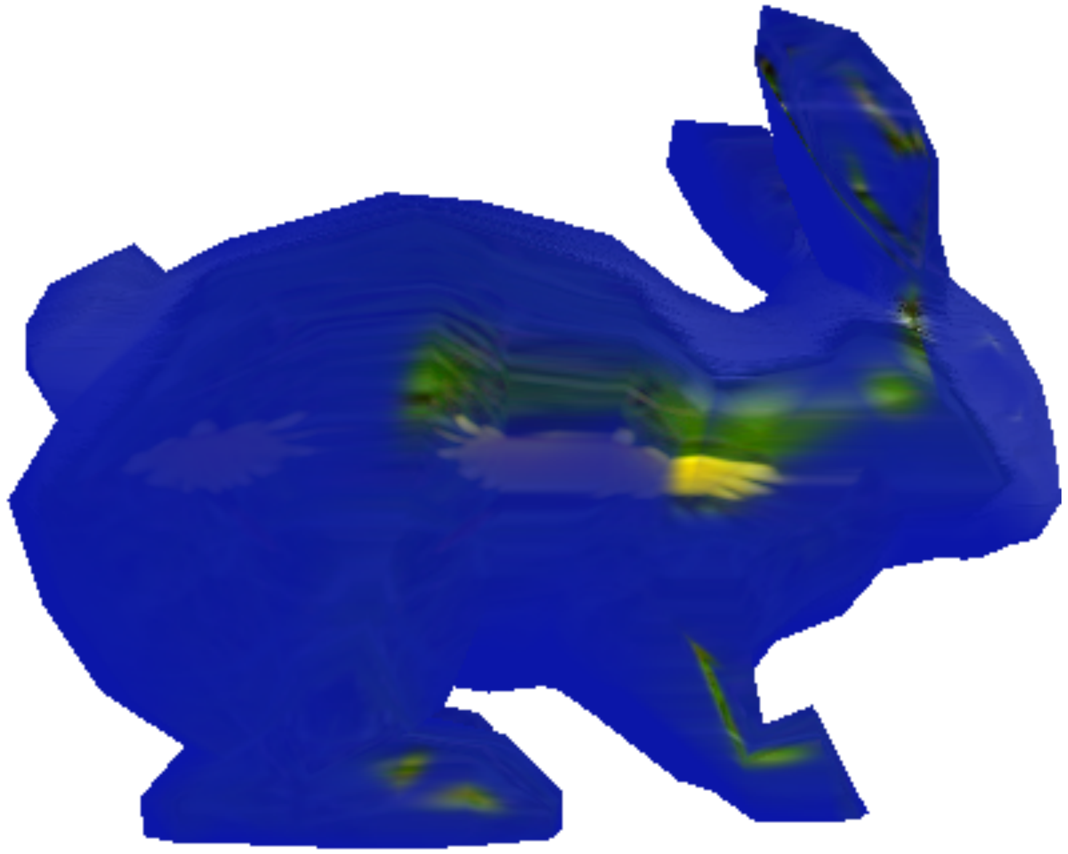
Example: Multitexturing

(Lighting omitted, calculates light from two light sources, spec/spec2)

```
uniform sampler2D world;
uniform sampler2D flower;
out vec4 outColor;
...
float tot = diffuseStrength*0.2 + specularStrength*0.9;
tot = max(0.0, tot);
outColor = vec4(1.0 - tot) * world*0.7 + vec4(tot)* flower;
```

Example: Multitexturing

Switches texture depending on light



Texture mapping conclusions

Texture coordinates (s, t)

Texture objects and texture units

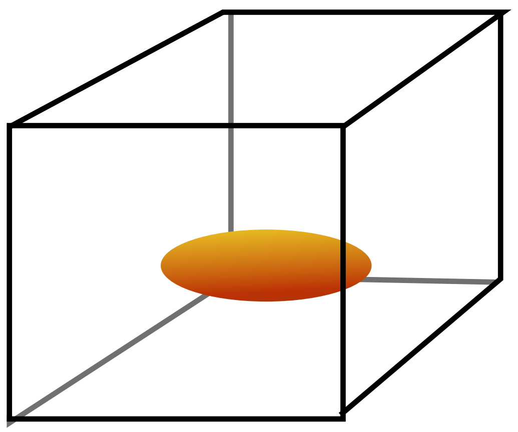
Filtering and mip-mapping

Texture coordinate generation

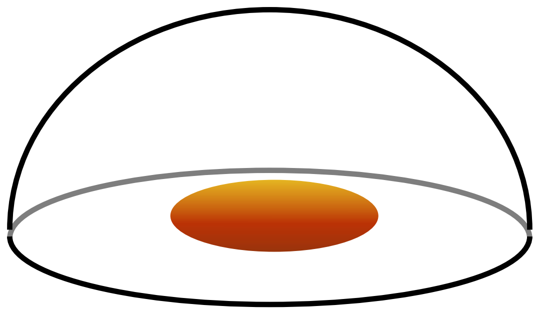
Multi-texturing

Skyboxes

A backdrop that makes your virtual world look bigger than it is.

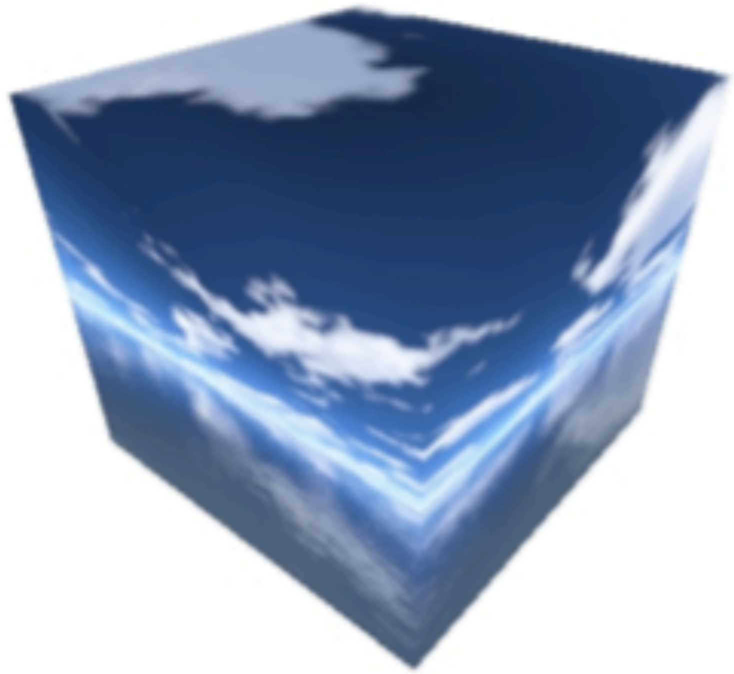


Skybox



Skydome

Skybox, example



Skyboxes

Infinite distance

=

the camera should always be in the center

=>

Always draw the skybox centered on the camera!

Skybox implementation

- Must be at infinite distance
- Must follow camera
- Must rotate with world

How big should it be?

The skybox does not have to be big!

"Maximum size" will waste frustum space!

- **Draw any size between "near" and "far" planes!**
- **Draw with Z buffer turned off!**

```
glDisable(GL_DEPTH_TEST);
Draw
glEnable(GL_DEPTH_TEST);
```

Then anything else will be drawn on top!

Follow and rotate

Use the world-to-view atrix for the rest of the scene, but keep only rotations!

$$M = \begin{bmatrix} u_x & u_y & u_z & \cancel{t_x} \\ v_x & v_y & v_z & \cancel{t_y} \\ n_x & n_y & n_z & \cancel{t_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Zero these!

More on skyboxes

- No light!

If you put light on a skybox, it will look like a box!

- Clamp to edge

Repeating textures will cause visible errors at edges!

(Why?)

Environment mapping

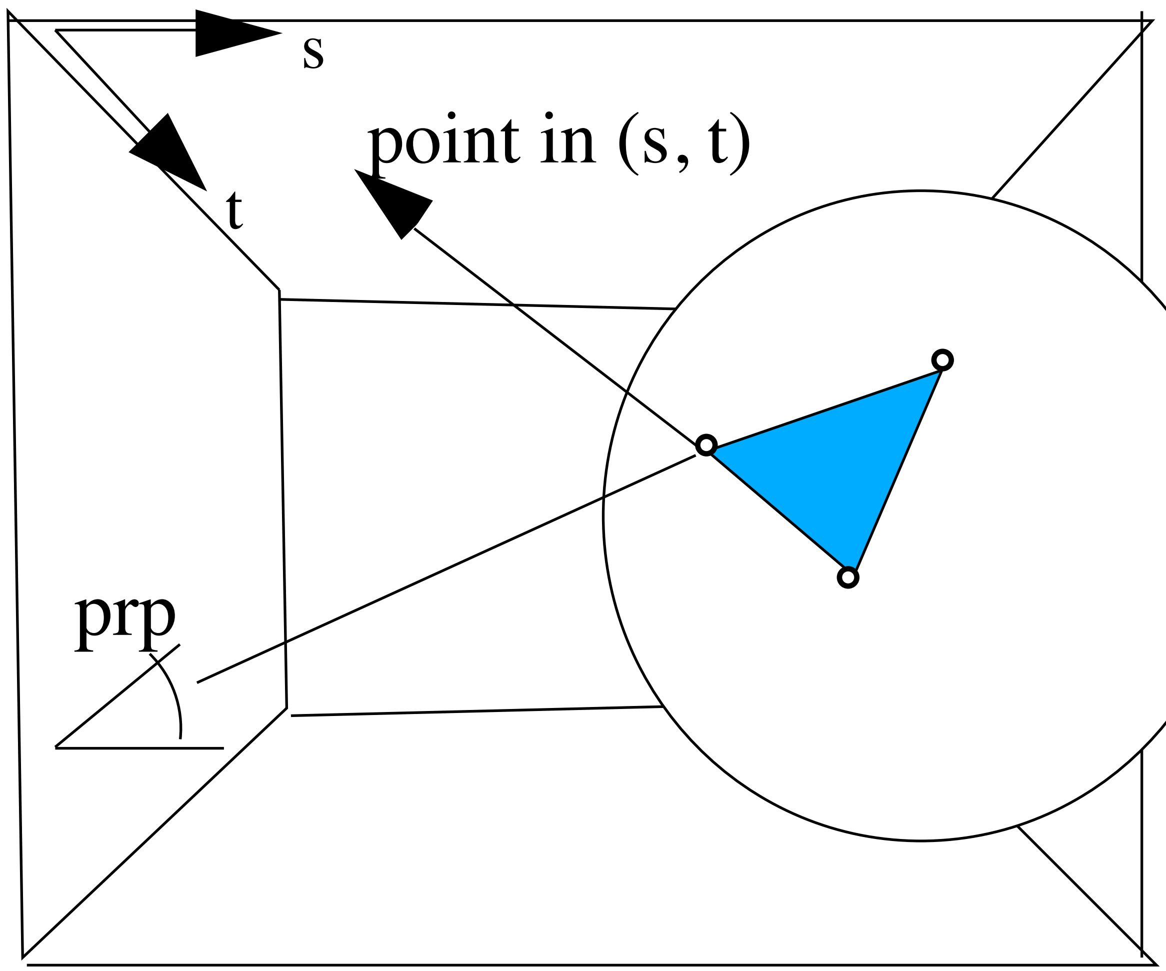
Reflects an environment texture in the surface

Mimicks mirroring reflections

In OpenGL:

Spherical environment mapping (old)

Cube map environment mapping



Bounding shape, environment (cube, sphere etc)

Shape with environment mapping

Reflected ray direction is converted to texture coordinates!

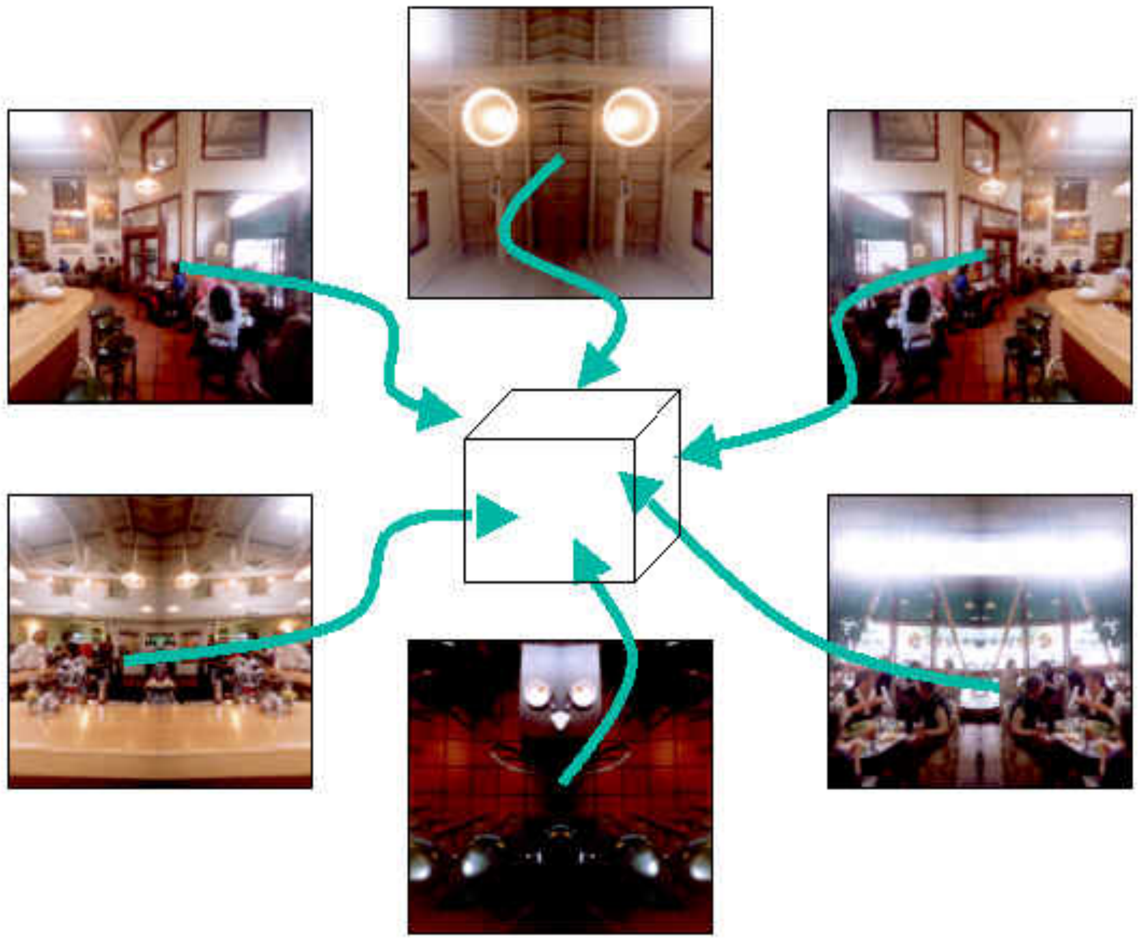
Simplification: Only direction, not reflection point, is used!

Sphere environment mapping

First hardware supported method. Simple and straight-forward.

Requires view dependent distorsion of the texture.

Better method: Cube environment mapping



Cube environment mapping

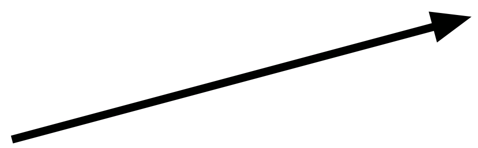
Hardware support on modern hardware

Six separate textures is one cube map

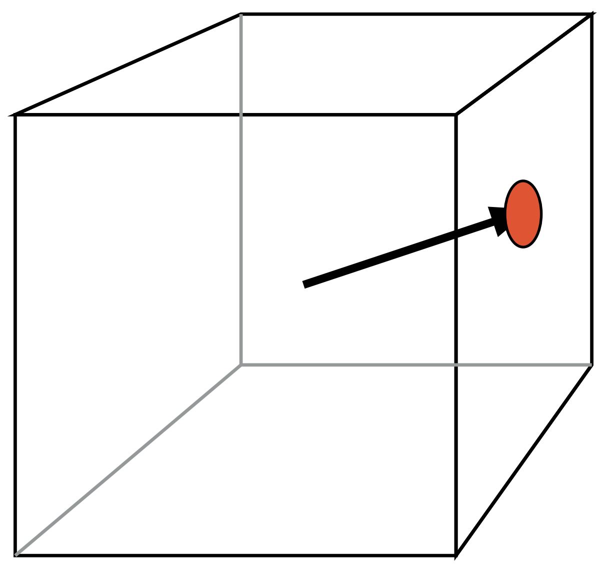
```
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X, 0, 3, 4, 4, 0, GL_RGB, GL_UNSIGNED_BYTE, minitex1);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, 0, 3, 4, 4, 0, GL_RGB, GL_UNSIGNED_BYTE, minitex2);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, 0, 3, 4, 4, 0, GL_RGB, GL_UNSIGNED_BYTE, minitex3);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, 3, 4, 4, 0, GL_RGB, GL_UNSIGNED_BYTE, minitex4);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z, 0, 3, 4, 4, 0, GL_RGB, GL_UNSIGNED_BYTE, minitex5);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, 3, 4, 4, 0, GL_RGB, GL_UNSIGNED_BYTE, minitex6);
```

Cube map lookup

Done with direction vector!



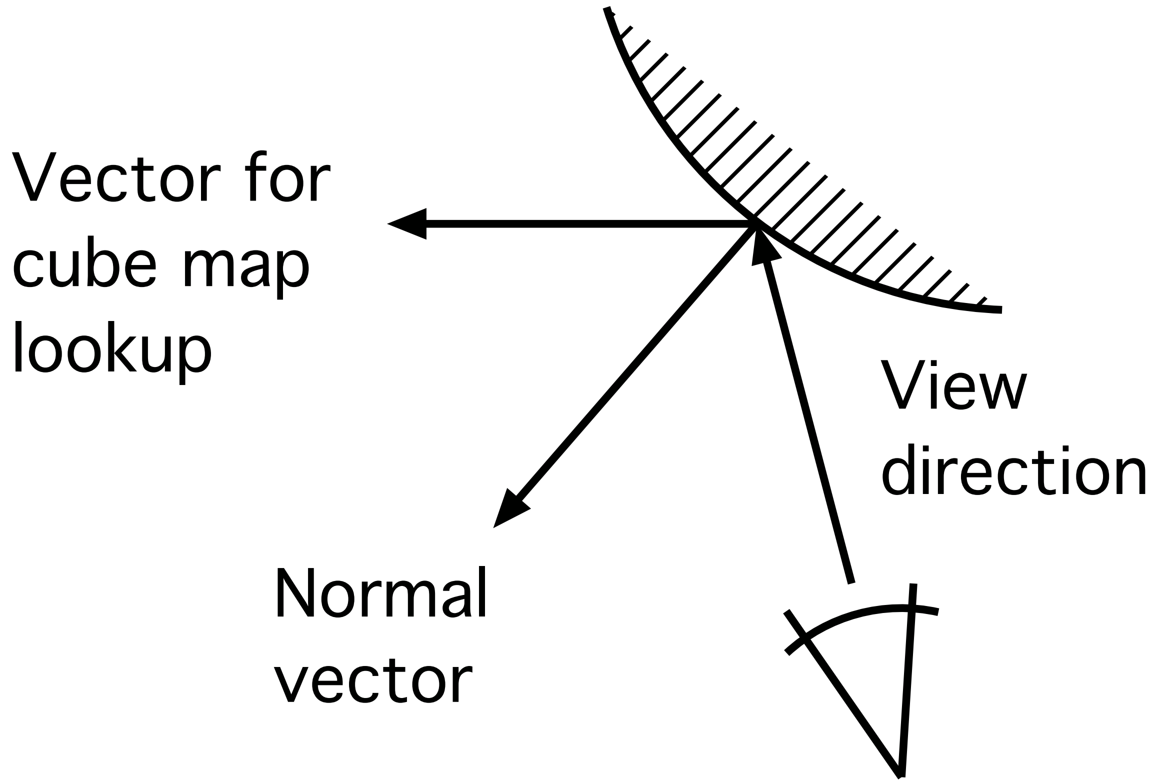
A direction...



...is converted to a position in the cube

Environment mapping with cube maps

Mirror view direction in surface to get direction vector



Can be done in vertex shader -> fast!

Environment mapping shaders

Vertex shader: Transform tricks!

```
in  vec3 inPosition;
in  vec3 inNormal;
out vec3 reflectedView;

uniform mat4 projMatrix;
uniform mat4 worldToView;
uniform mat4 modelToWorld;

void main(void)
{
 mat3 normalMatrix1 = mat3(worldToView * modelToWorld);
 gl_Position = projMatrix * worldToView * modelToWorld * vec4(inPosition, 1.0);

 vec3 posInViewCoord = vec3(worldToView * modelToWorld * vec4(inPosition, 1.0));
 vec3 viewDirectionInViewCoord = normalize(posInViewCoord);
 vec3 viewDirectionInWorldCoord = inverse(mat3(worldToView)) * viewDirectionInViewCoord;

 vec3 wcNormal = mat3(modelToWorld) * inNormal;
 reflectedView = reflect(viewDirectionInWorldCoord, normalize(wcNormal));
}
```

Trivial fragment shader

```
#version 150

out vec4 outColor;
uniform samplerCube cubemap;
in vec3 reflectedView;

void main(void)
{
// Cubemap texture only:
 outColor = texture(cubemap, normalize(reflectedView));
}
```

Lookup in skybox coordinates = world coordinates!

Combine skybox with environment map

Immersive and cheap!

Advanced environment mapping

Recursive environment mapping
”Fake ray-tracing”
Advanced multi-pass method
Render the scene from the object
Use the result as texture

Result: Possible to see reflections of moving objects.

Much higher performance cost

Gloss mapping

Map material parameters over a surface for varied reflectivity