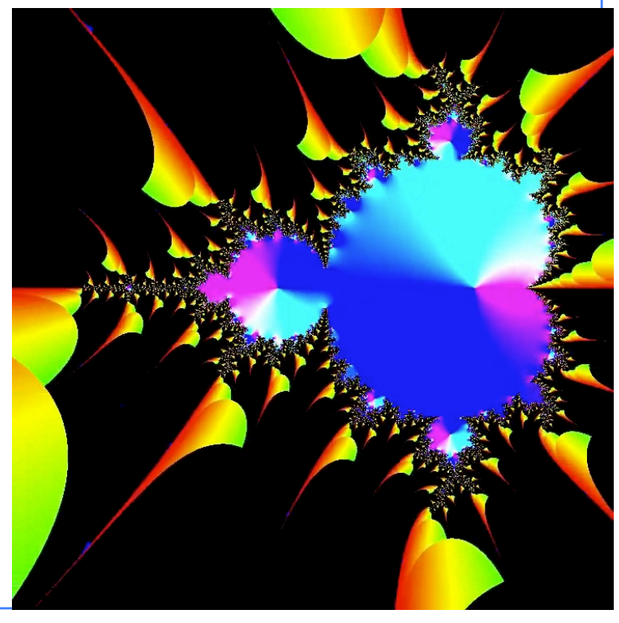
TSBK07/TSBK11

Computer Graphics

Ingemar Ragnemalm, ISY





Lecture 6

Texture mapping

Multitexturing

Skyboxes

Environment mapping

Labs and duggas

Dugga 2 went very well.

Many went from a weak #2 to a much stronger #2.

TSBK07: Many were at 7-10.
TSBK11: 6 points common.

Lab progress is just fine. Most are done with #1, many with #2.

Annoying bug in Mac version seems to be fixed!

Lecture questions

1. What happens if you access outside a texture?

2. How can you generate texture coordinates?

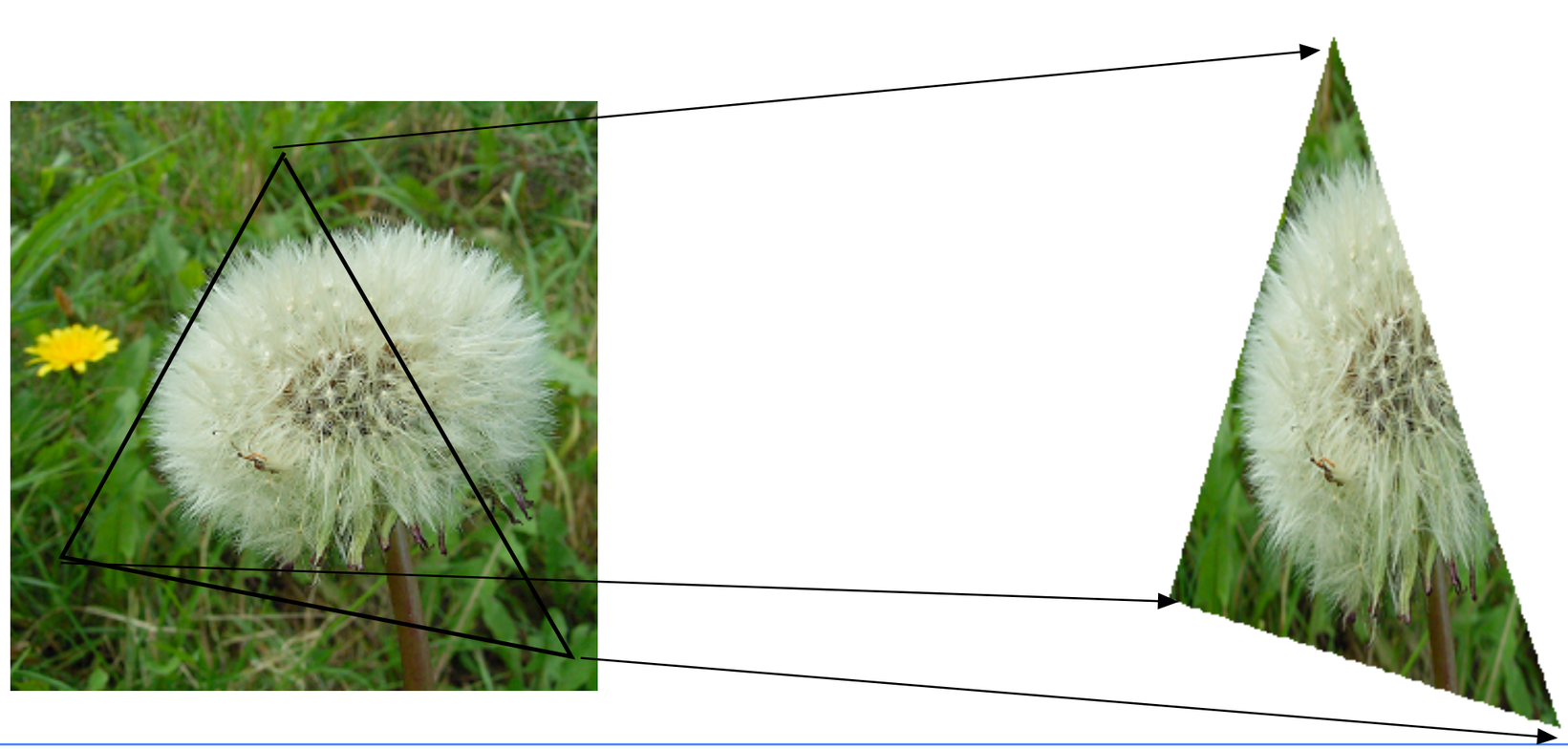
3. In multitexturing, how can you decide what texture to use?

4. How big is a skybox?

5. When do we get aliasing problems in graphics?

Texture mapping

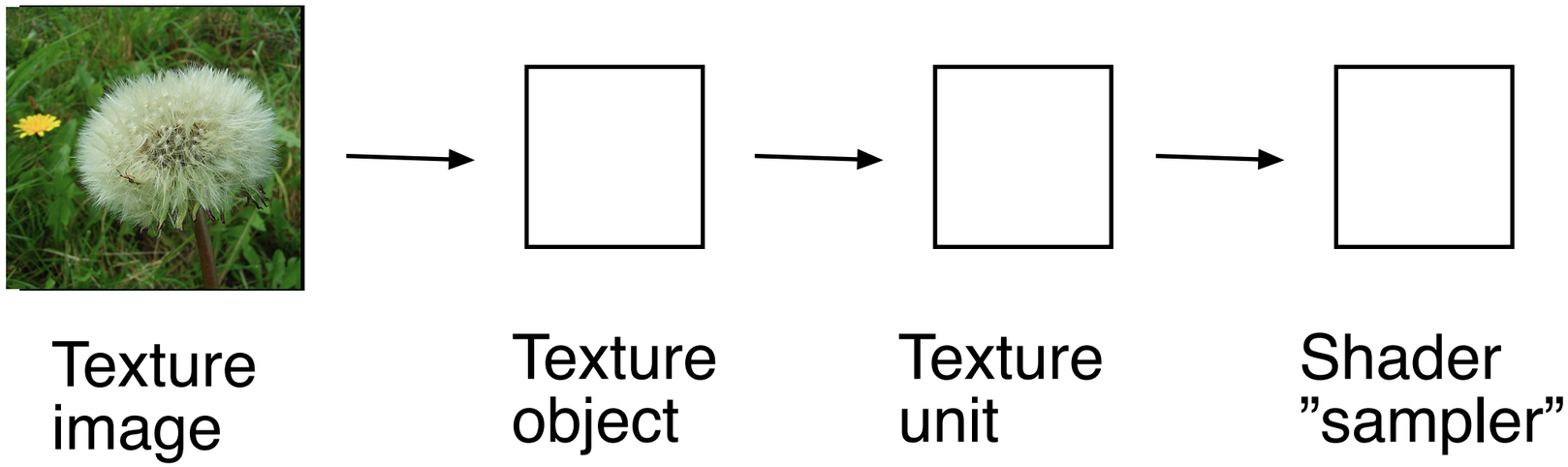
Texture coordinates from 0 to 1



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!



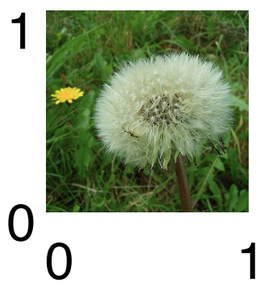
Texture parameters

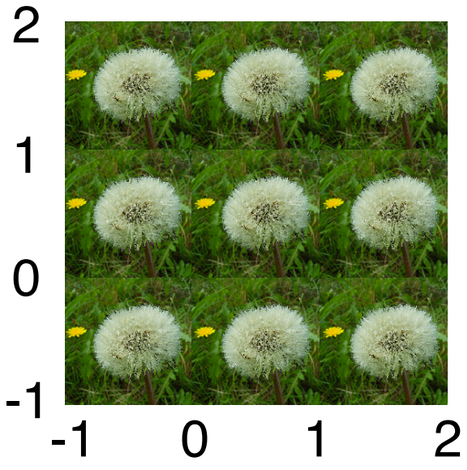
glTexParameter(...);

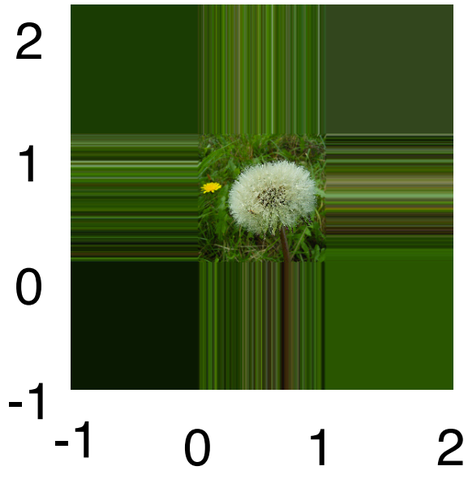
Most common ones:

GL_TEXTURE_WRAP_S
GL_TEXTURE_WRAP_T

GL_REPEAT
GL_MIRRORED_REPEAT
GL_CLAMP_TO_EDGE







Magnification and minification parameters:

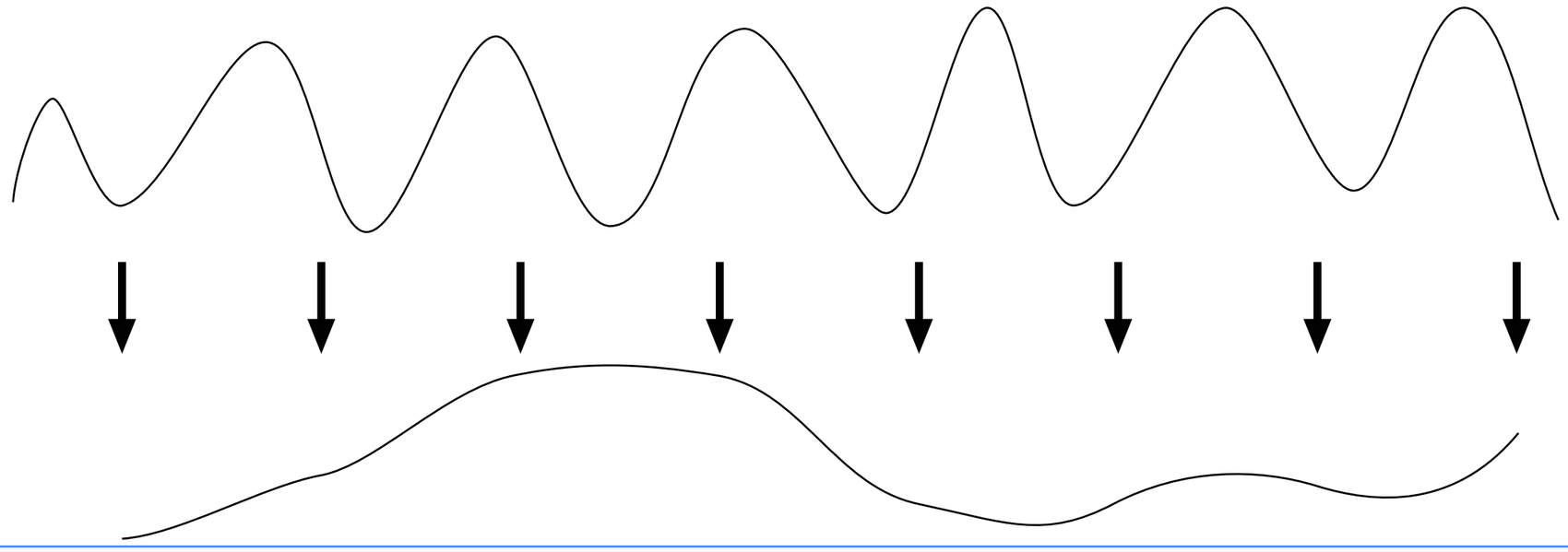
glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER, GL_NEAREST);

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 in texture mapping 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

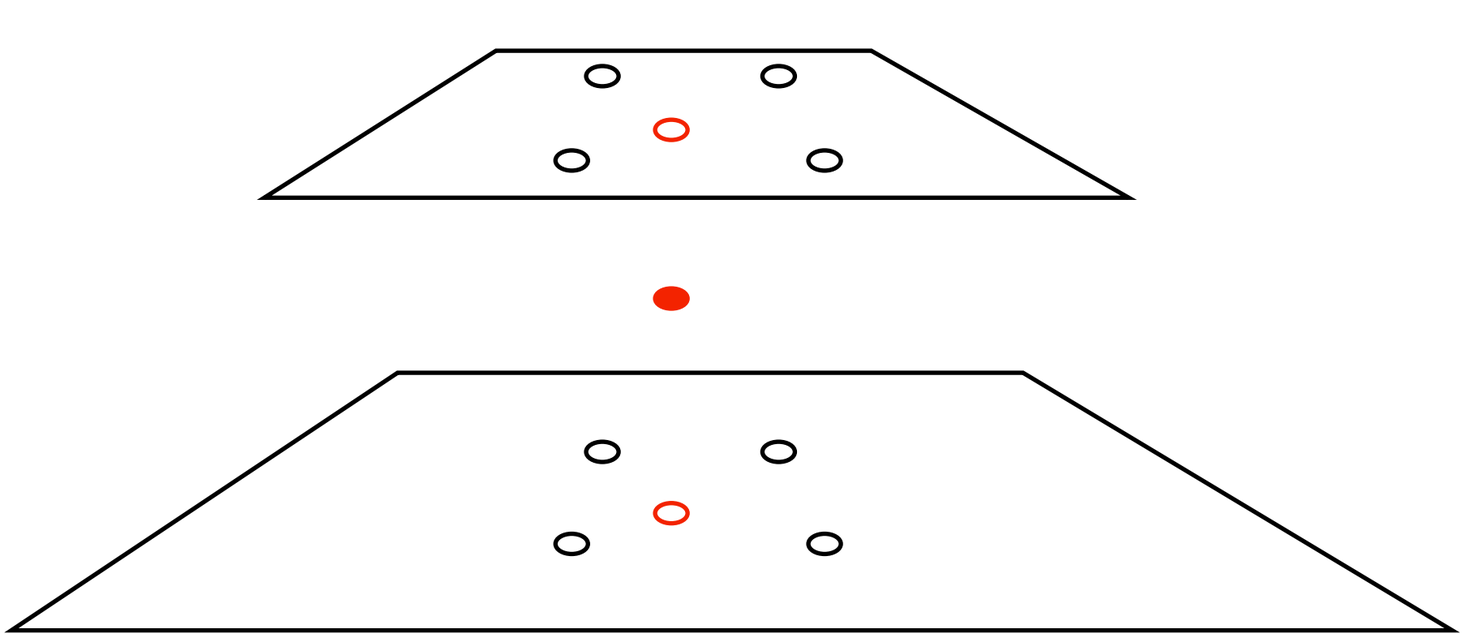
64x64

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

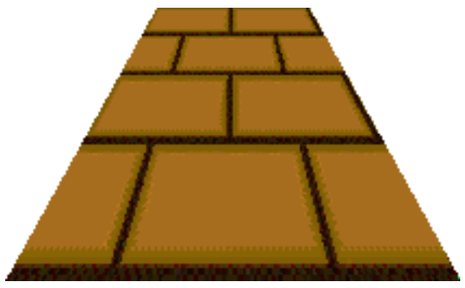
Gives anti-aliasing at a very low cost.

Good results in most situations.

Aliasing problems remain at steep angles.

Texture mapping is not just stretching:

Affine



Perspective correct

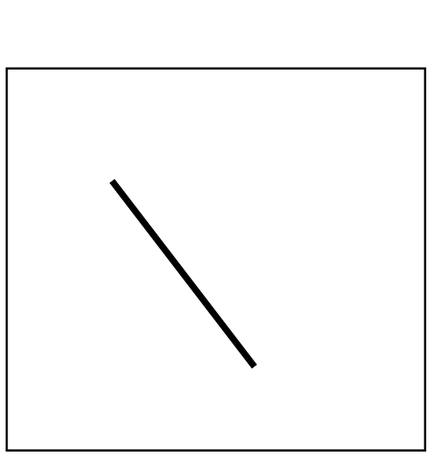
Generating texture coordinates

Now we know how to draw, but... where do the texture coordinates come from?

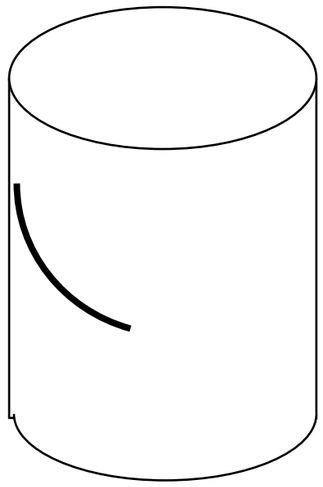
May be delivered with the model... but can we make them ourselves?

Yes!

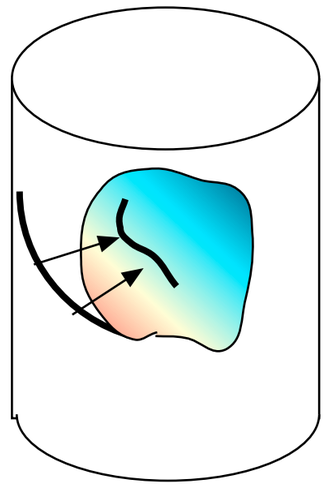
Pre-calculating (s,t) for every vertex in a model



Texture map (s,t)

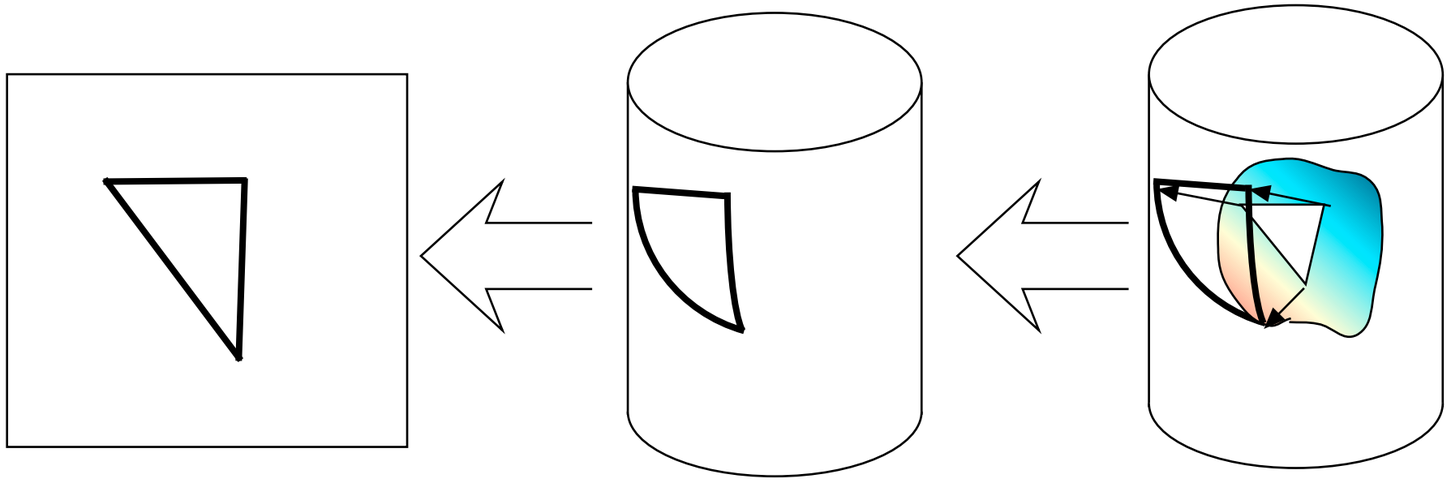


Map to intermediate surface (u,v)



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

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

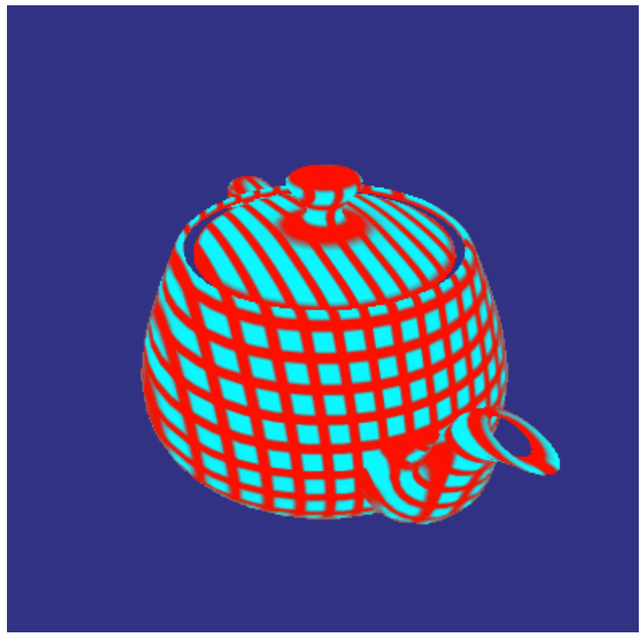


Use to index the texture map

Map to inter-mediate surface coordinates

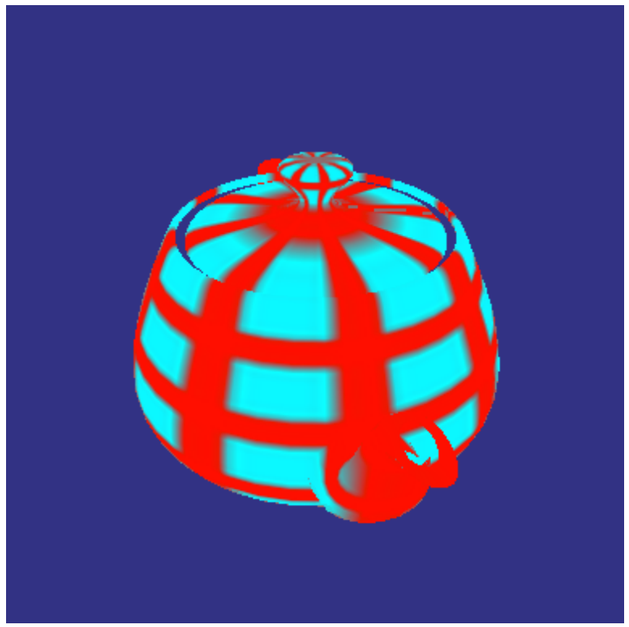
Polygon vertex coordinates

Examples



Planar
(Linear)

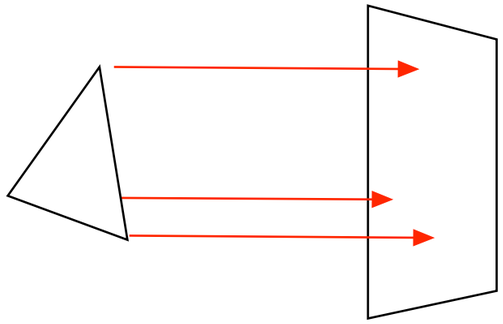
Cylinder



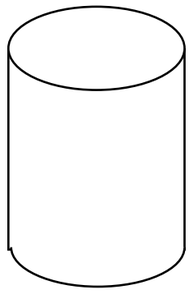
Sphere

Common mappings

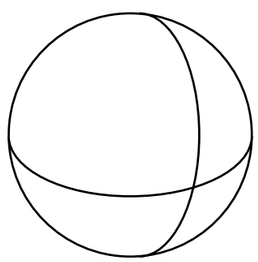
Planar



Cylinder

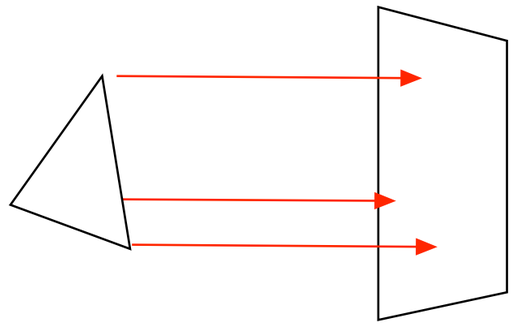


Sphere



Planar texture mapping

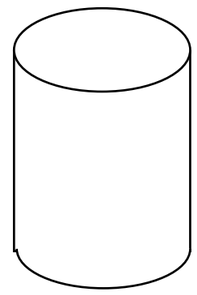
Along z:
u = x
v = y



Cylindrical texture mapping

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

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

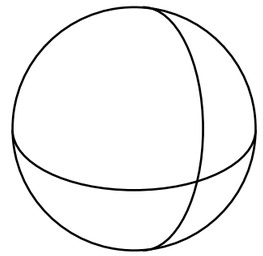


arctan(y,x) =
x > 0: tan-1(y/x)
x<0: π + tan-1(y/x)
x = 0, y > 0: π/2
x = 0, y < 0: -π/2

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

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

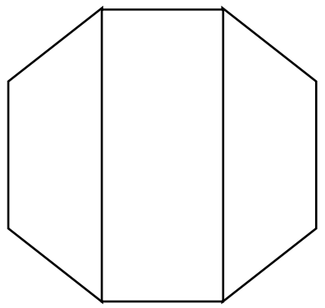
$u = \arctan(y,x)$
$v = z$

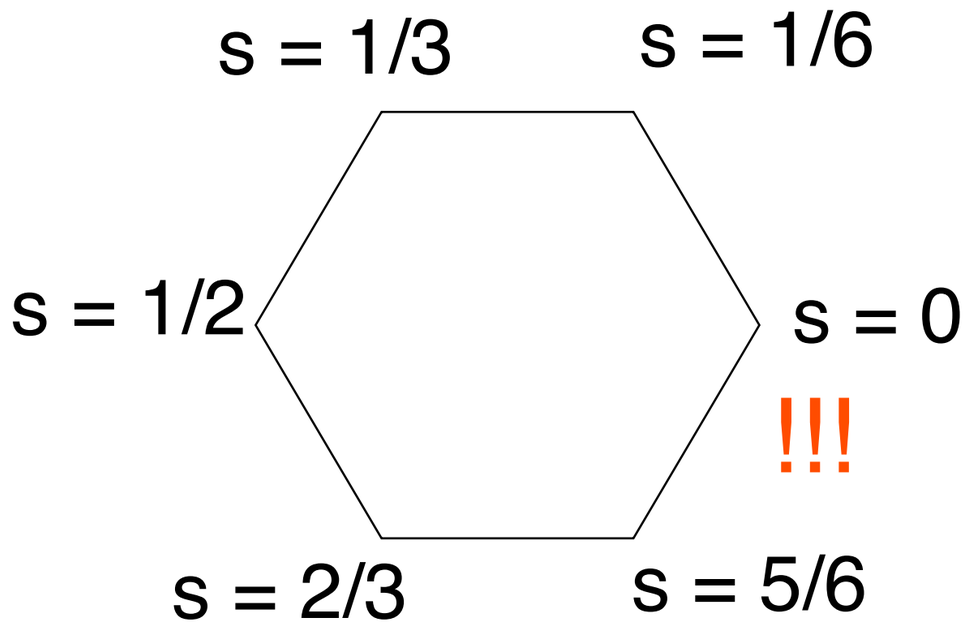
$s = (u + \pi/2) / 2\pi$

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

Watch the edges!

Example: six-sided barrel





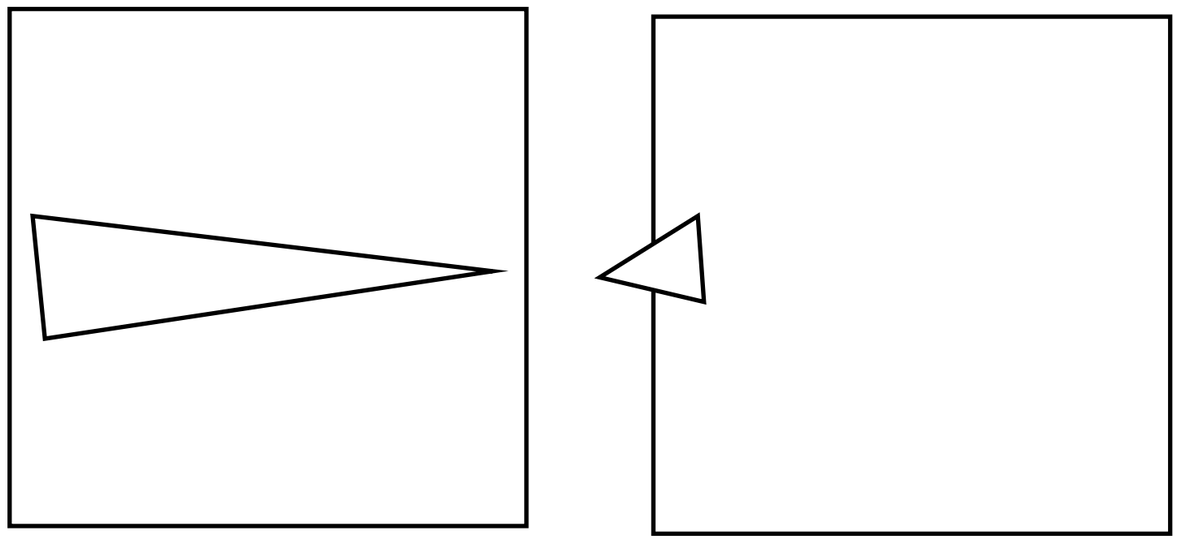
$x = R\cos(u)$
$y = R\sin(u)$

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

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

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

Problem is detected by checking proximity to the edge



Unlikely
(impossible)

Good

Duplicate vertices to solve

Texture mapping conclusions so far

Texture coordinates (s, t)

Texture objects and texture units

Filtering and mip-mapping

Texture coordinate generation

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.

You can use several texture units in the same fragment shader! Just use multiple "sampler" variables.

Multitexturing applications

Bump mapping
Gloss mapping
Light mapping
Blending between textures (e.g. mountains)
Splatting (blend map)
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 (e.g. by name)
- Declare as samplers in GLSL

Multitexturing in GLSL

Bind textures →

Pass unit numbers →

Declare as samplers →

```
bindTextureToTextureUnit(flowerTex, GL_TEXTURE0);
bindTextureToTextureUnit(bumpTex, GL_TEXTURE1);
bindTextureToTextureUnit(noiseTex, GL_TEXTURE2);

uploadUniformIntToShader(shader, "flowerTex", 0);
uploadUniformIntToShader(shader, "bumpTex", 1);
uploadUniformIntToShader(shader, "noiseTex", 2);
```

```
uniform sampler2D flowerTex;
uniform sampler2D bumpTex;
uniform sampler2D noiseTex;
```

My code calling OpenGL calls:

```
void bindTextureToTextureUnit(GLuint tex, int unit)
{
    glActiveTexture(GL_TEXTURE0 + unit);
    glBindTexture(GL_TEXTURE_2D, tex);
}
```

```
void uploadUniformIntToShader(GLuint shader,
        const char *nameInShader, GLint i)
{
    if (nameInShader == NULL) return;
    glUseProgram(shader);
    GLint loc = glGetUniformLocation(shader, nameInShader);
    if (loc >= 0)
        glUniform1i(loc, i);
    else
        ReportError("uploadUniformIntToShader", nameInShader);
}
```

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

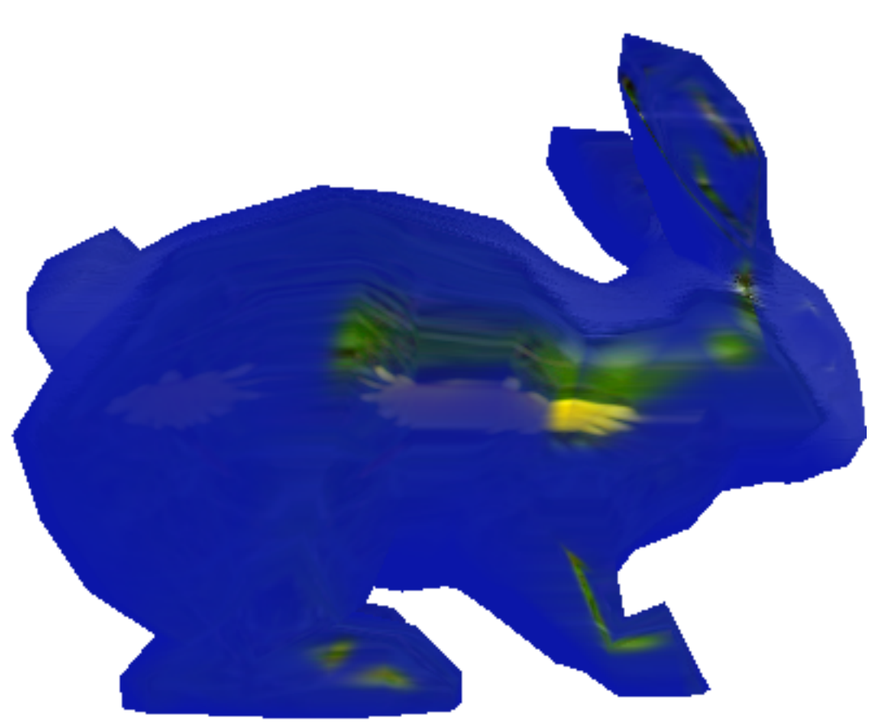
Example: Multitexturing 2

(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;
```

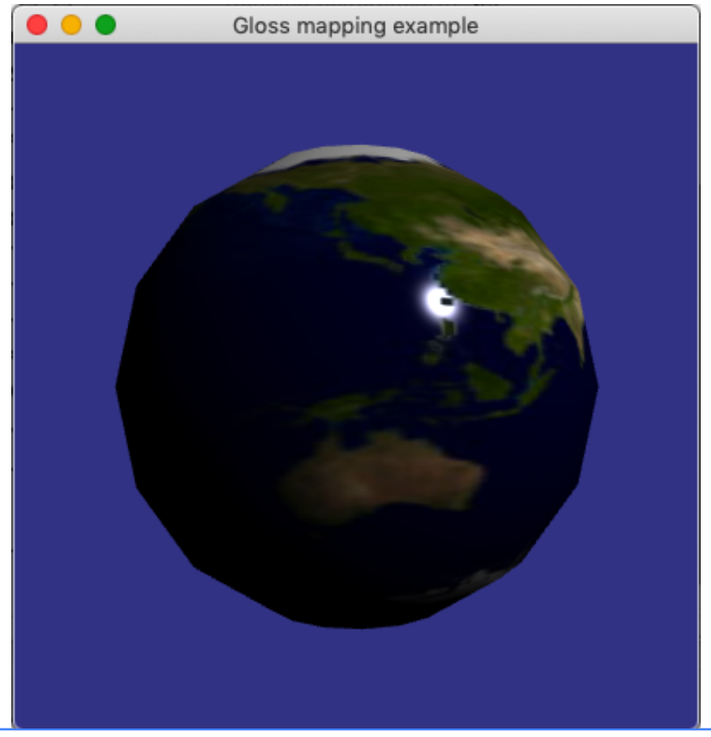
Example: Multitexturing

Switches texture depending on light



Gloss mapping

Map material parameters over a surface for varied reflectivity



Splatting

Multitexture with a blend map.

Using a texture as "map" for multitexturing.

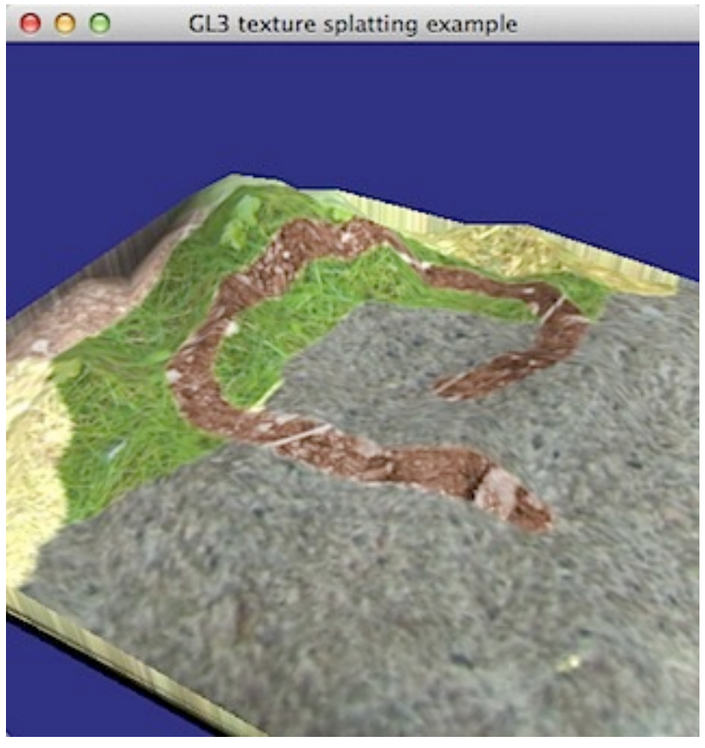
Each channel R, G, B (and possibly A) used for one texture.

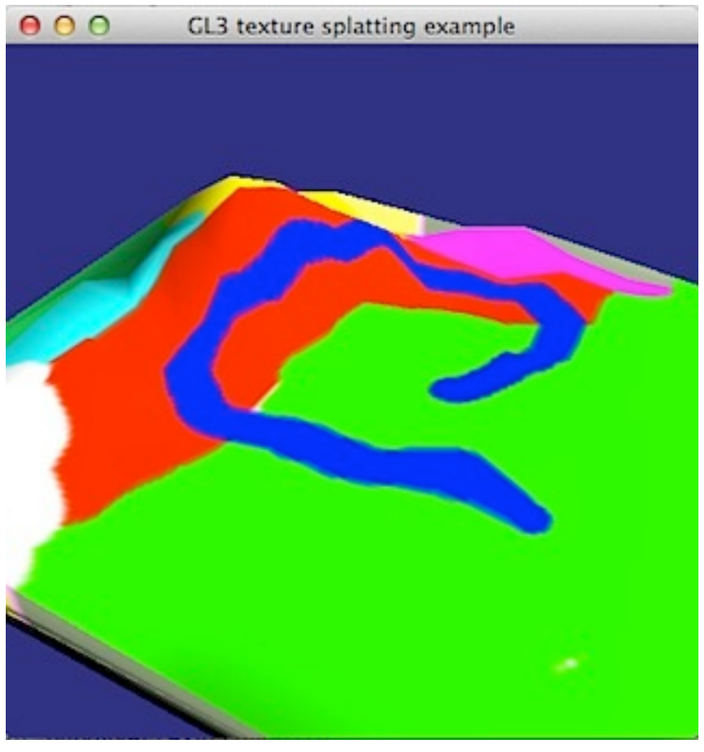
Smooth transitions between textures.

Combinations of textures possible.

Input

Result


GL3 texture splatting example


GL3 texture splatting example

Number of textures

RGB: Three textures. Easiest!

RGB plus 1-magnitude: Four textures

RGBA: Four textures

RGBA plus 1-magnitude: Five textures

Need more? Use multiple splatmaps!

Beware texture scaling

Splatmaps are often large scale, rarely repeated

The textures it controls is often repeated

Triplanar texure mapping

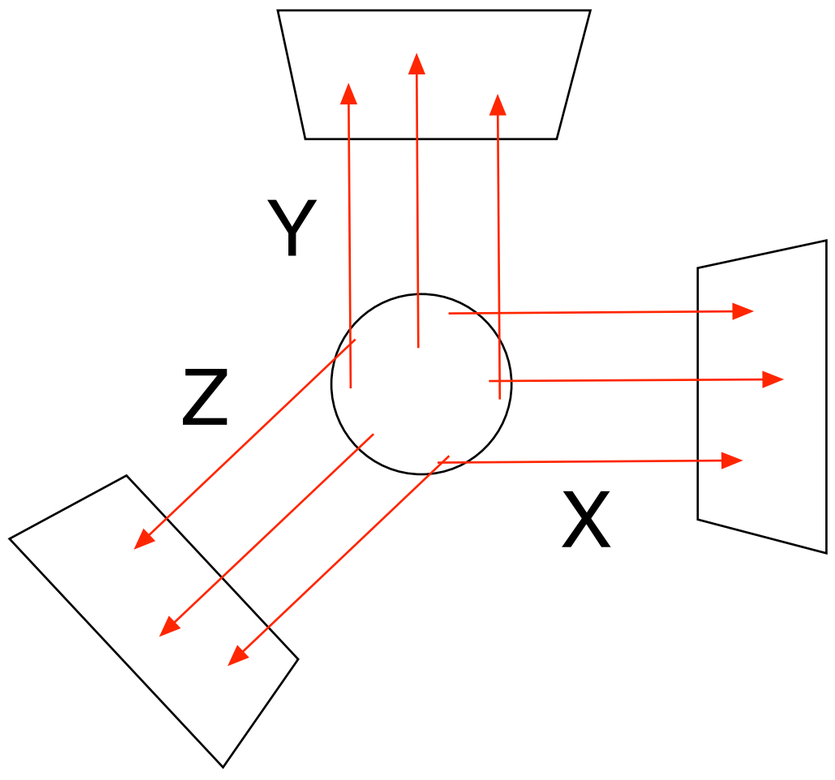
"Multitexture" with a single texture.

Combine planar mapping with multitexturing

Apply textures - possibly the same - along all three axes!

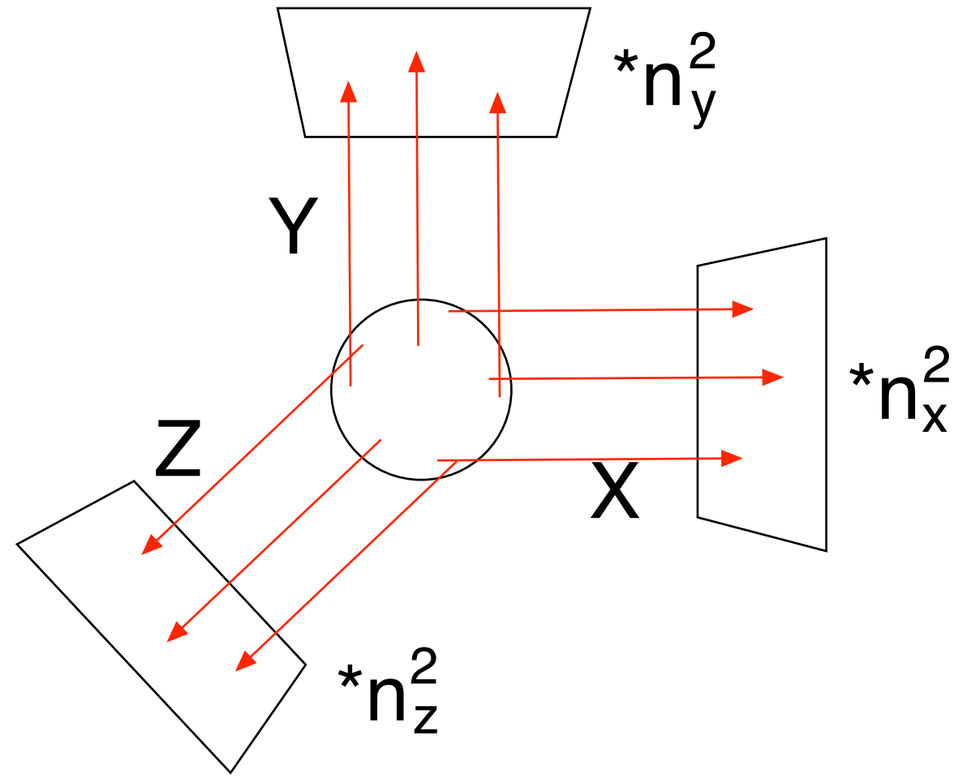
Sum together with suitable function

Map texture on object along all three axes!

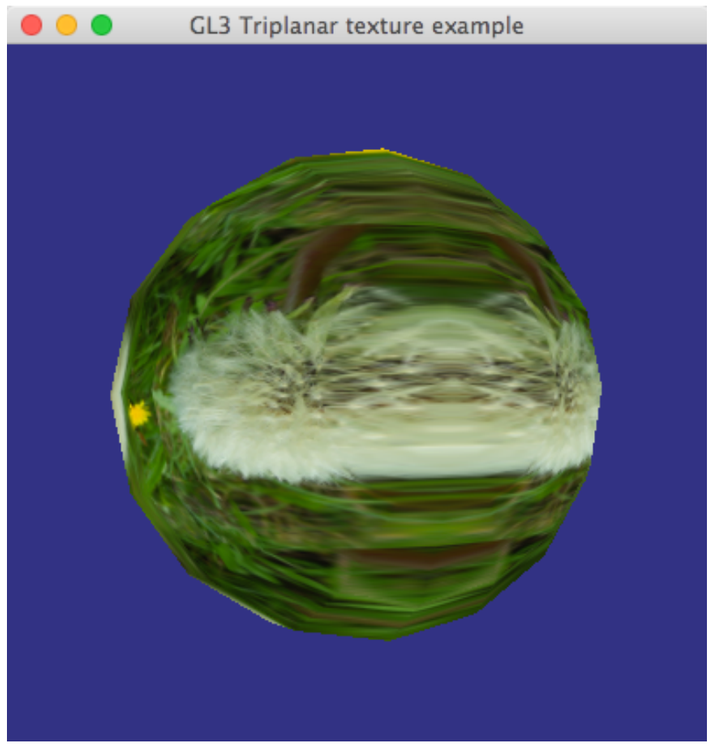


Blend together based on normal vector!

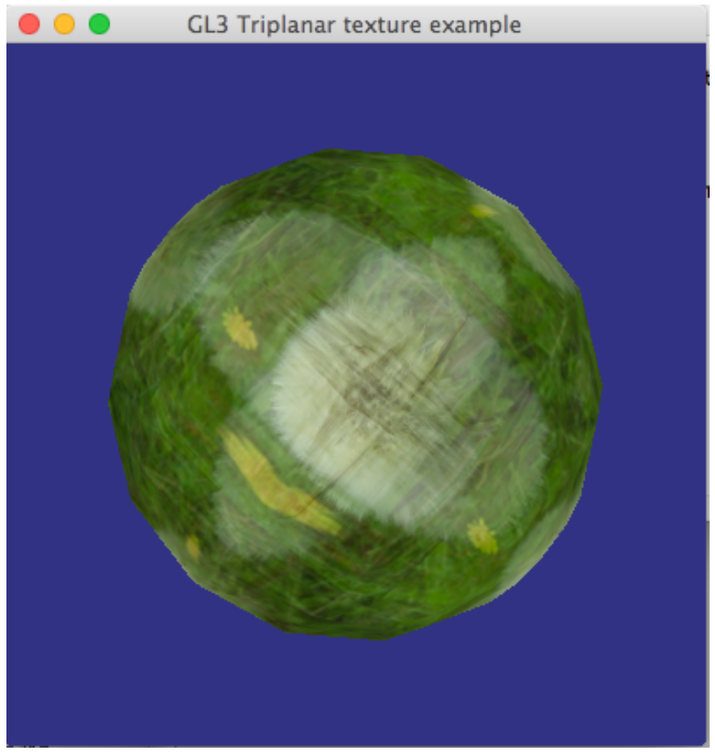
Example: Square of each component



Blend together properly



Simple planar



Linear by component - not same sum everywhere!

Square of component

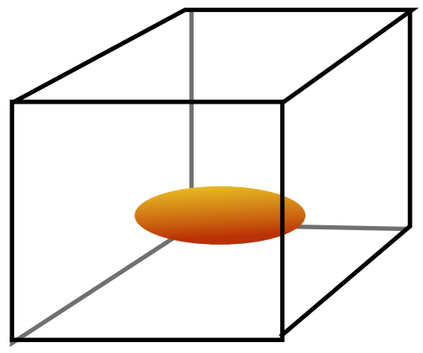
Triplanar texture mapping

Great alternative for procedurally generated geometry

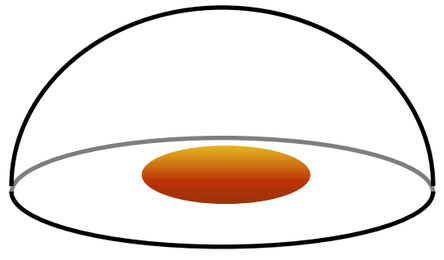
also a general purpose mapping for general materials

Skyboxes

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

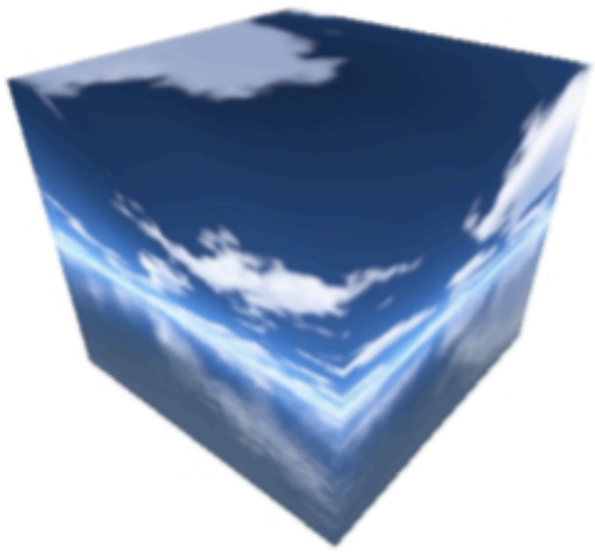


Skybox



Skydome

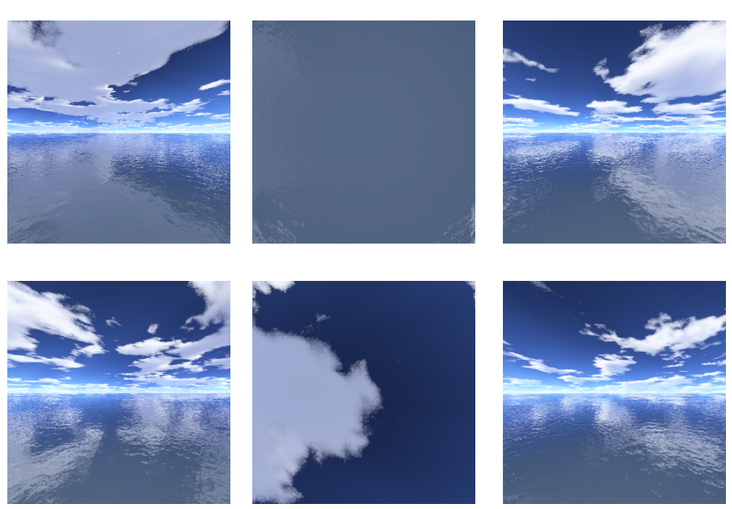
Skybox, example



Different kinds

Single texture, cross

Lab skybox, limited cross



Six separate sides

Skyboxes are at 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 matrix for the rest of the scene, but keep only rotations!

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

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?)

Skybox in lab 3

Will be in both TSBK07 and TSBK11

Full skyboxes also provided. Exactly how might change before monday.

Related to skyboxes:

Environment mapping

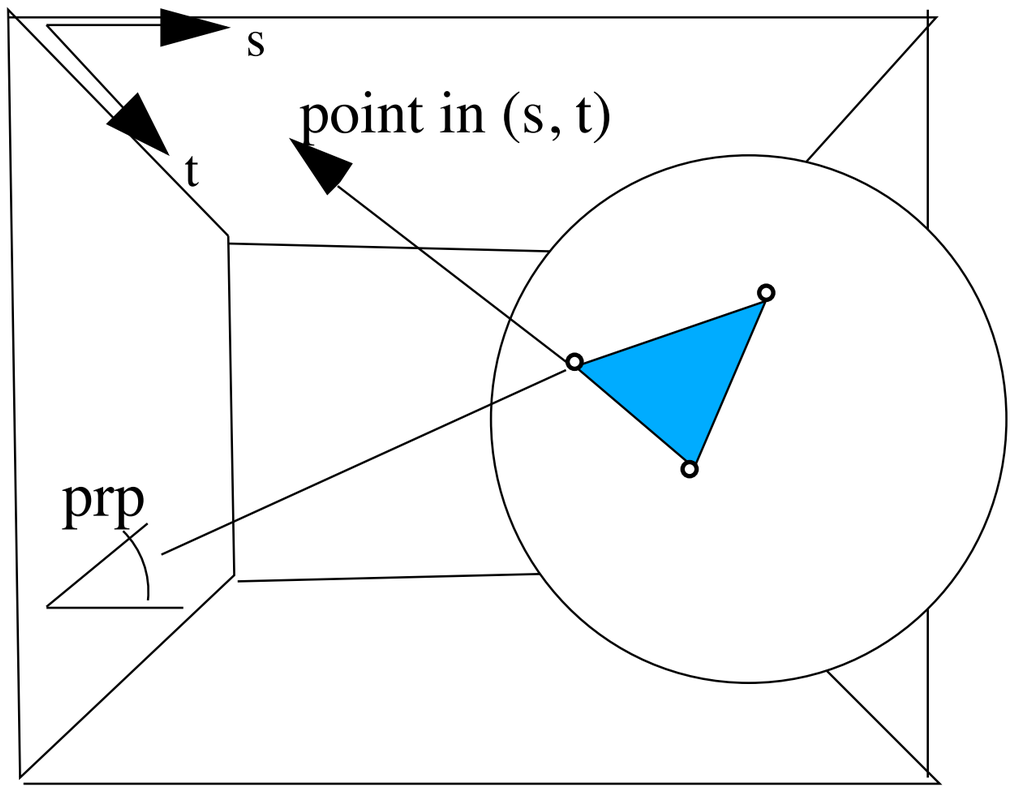
Reflects an environment texture in the surface

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!

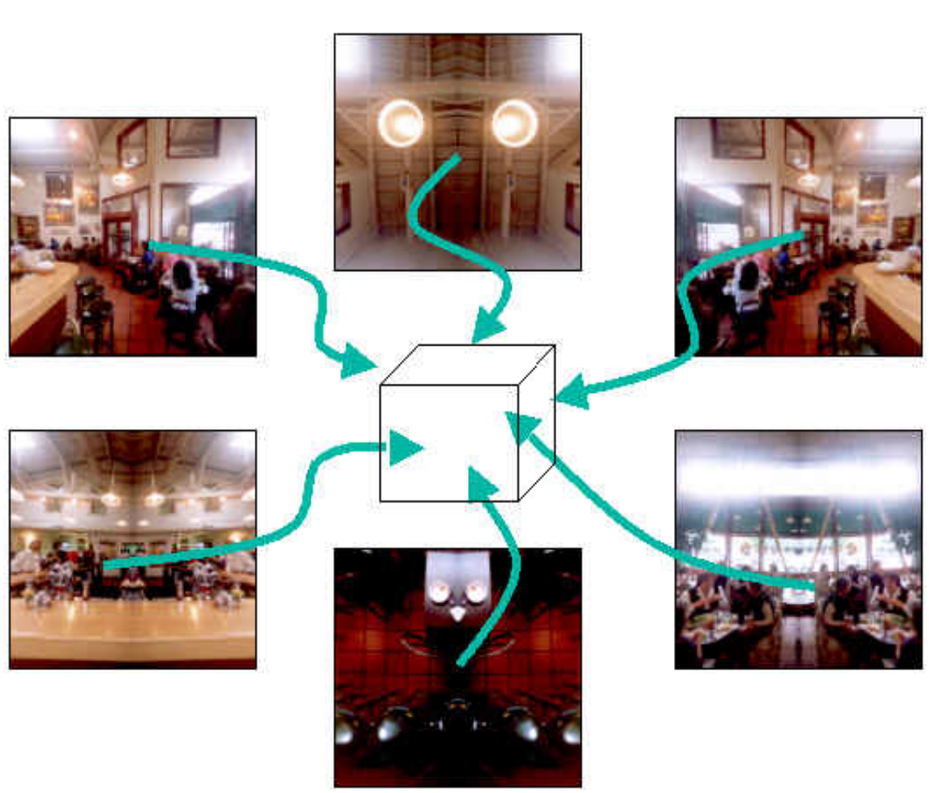
Simplification: Only direction, not reflection point, is used!

Sphere environment mapping

First hardware supported method. Simple and straight-forward.

Requires view dependent distorsion of the texture.

Better method:
Cube environment mapping



Cube mapping

Hardware support on modern hardware

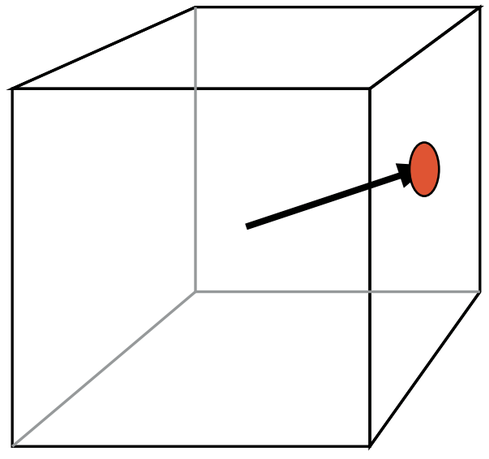
Six separate textures is one cube map

Can be used both for skyboxes and environment mapping

```
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

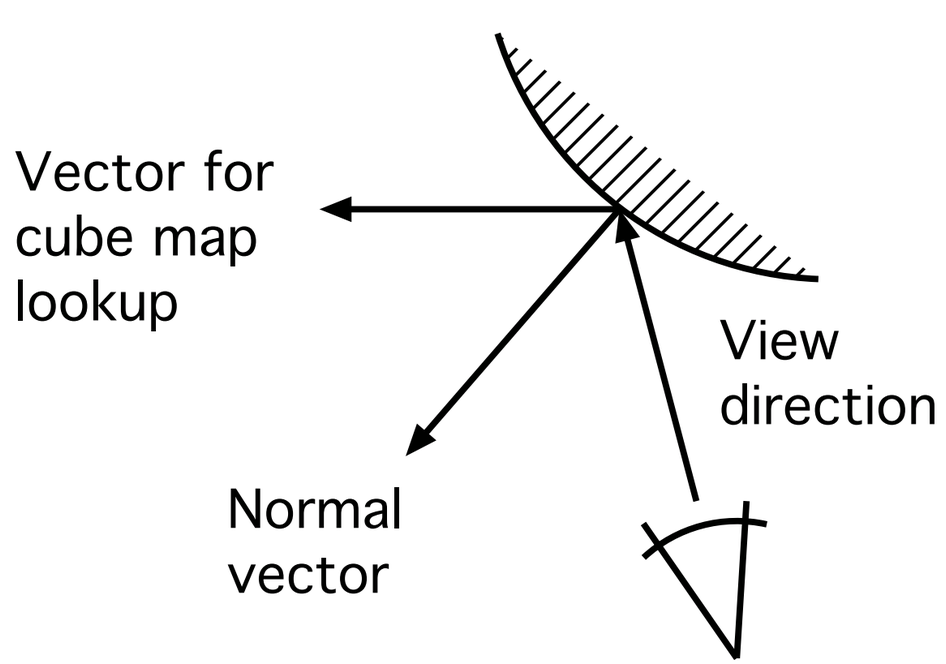
Done with direction vector!



...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

Next time

Bump mapping

Normal matrix

More visible surface detection