Lecture 5
3D graphics part 3

Shading; applying lighting

Surface detail: Mappings

Texture mapping

Light mapping

Bump mapping

Surface detail

<u>Shading:</u>
takes away the surface detail of the polygons

<u>Texture mapping and other mappings:</u>
add the surface detail that we really want

Surface mapping techniques

Texture mapping
Billboards
Bump mapping
Light mapping
Environment mapping

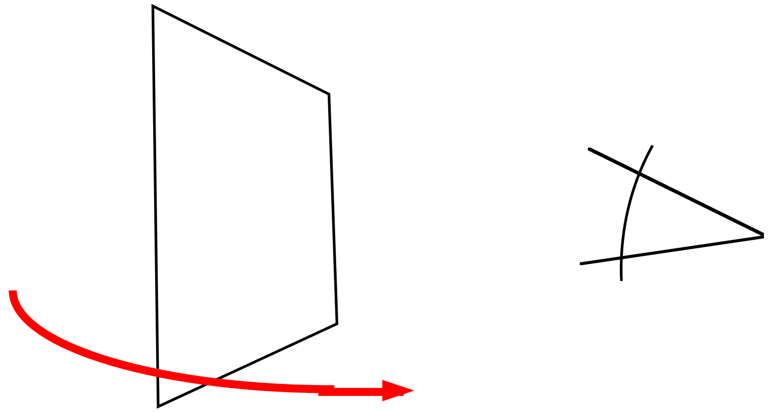
Texture mapping

In common use

Supported by the fixed pipeline and all GPU hardware - extremely fast and easy to use

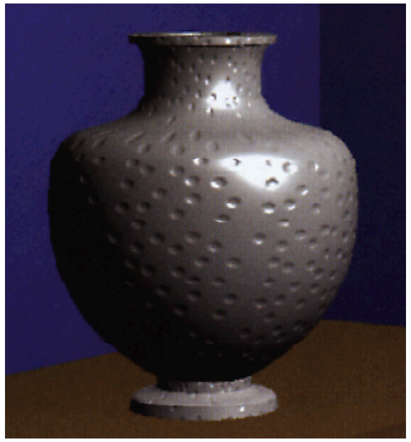
Billboards

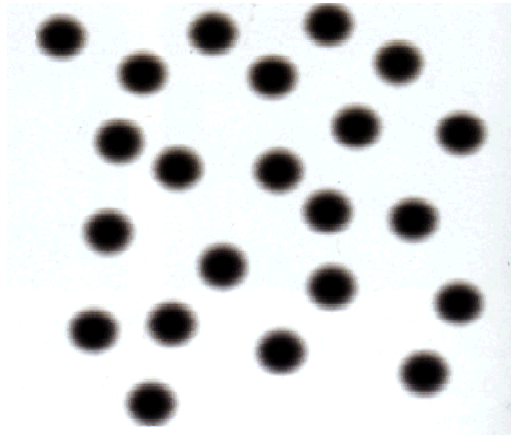
A texture mapped polygon, which always faces the viewer



Bump mapping

Simulates surface structure by manipulating the normal vector



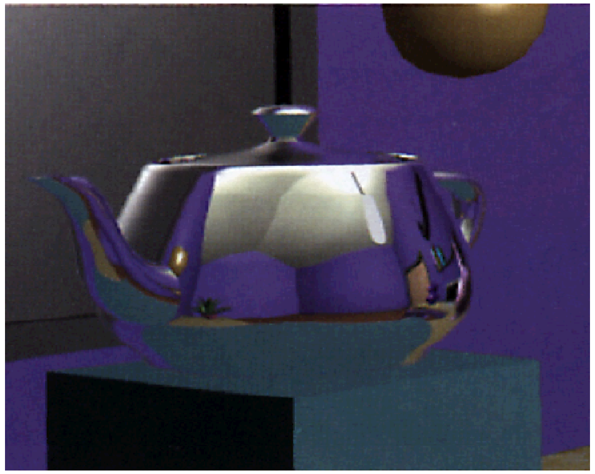


Light mapping

Applies pre-calculated light to surfaces

Environment mapping

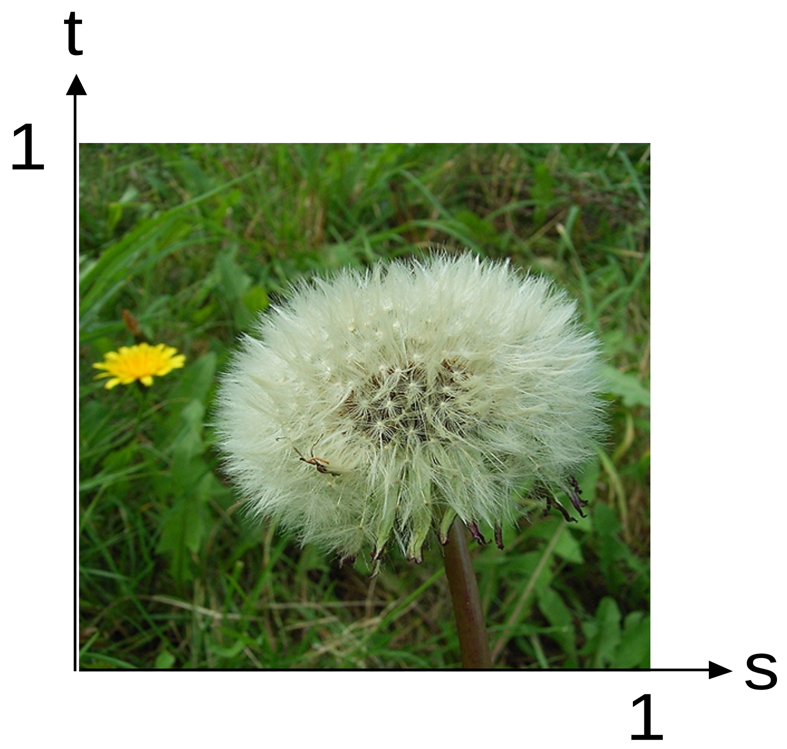
aps an pre-rendered image as a eflection in the object



Texture mapping

“Wrap” a specified part of “texture space” onto a polygon

Texture space

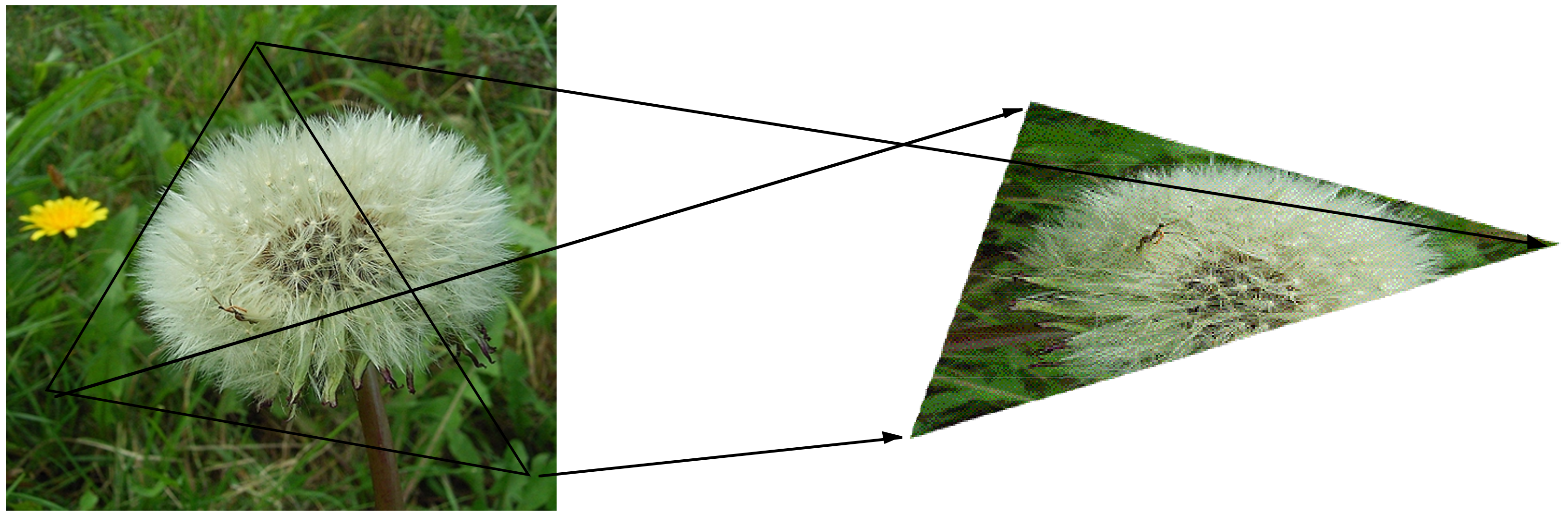


exture = image used
or texture mapping

exture space is
sually 2-
imensional, (s, t),
ith textures defined
n [0, 1]

Mapping from texture to surface

Each vertex has a texture coordinate, interpolate between.



Texture mapping in OpenGL (lab 1 style)

glEnable(GL_TEXTURE_2D);
turns on texture mapping

glTexImage2D(...);
loads a texture and makes it the current one

glTexCoord2f(s, t);
specifies texture coordinates for the following vertex

Arrays can also be used!

Texture objects

Referring to already loaded textures

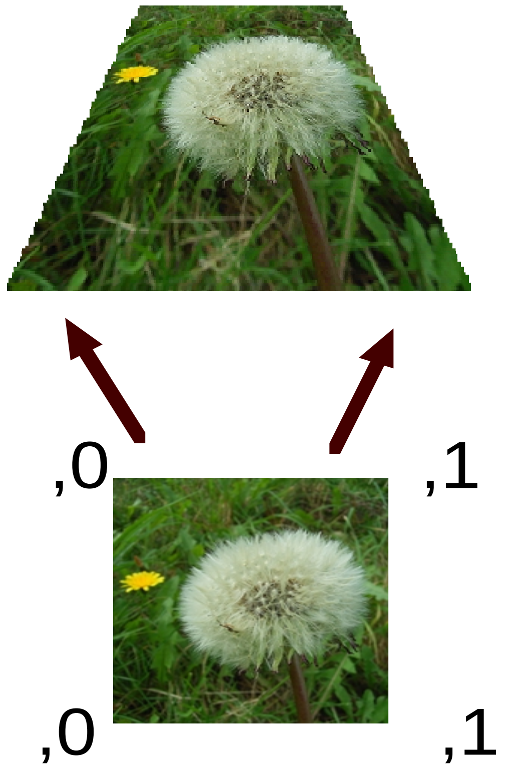
glGenTextures(...);
reserves texture numbers, making them available to use

glBindTexture(...);
makes a texture the current one

glTexImage2D(...);
loads a texture for the current texture number

A textured polygon

```
glTexImage2D(...);
...
glBindTexture(texNum);
glBegin(GL_POLYGON);
glTexCoord2f(0, 0);
glVertex3f(x1, y1, z1);
glTexCoord2f(1, 0);
glVertex3f(x2, y2, z2);
glTexCoord2f(1, 1);
glVertex3f(x3, y3, z3);
glTexCoord2f(0, 1);
glVertex3f(x4, y4, z4);
glEnd();
```

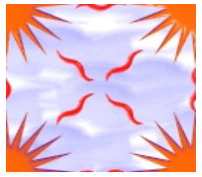


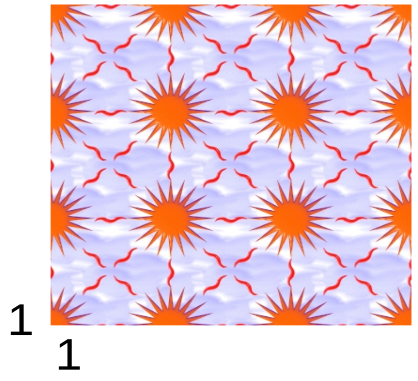
Texture parameters

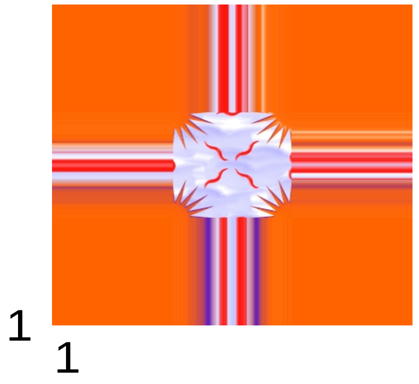
glTexParameter(...);

GL_TEXTURE_WRAP_S
GL_TEXTURE_WRAP_T

GL_REPEAT
GL_CLAMP







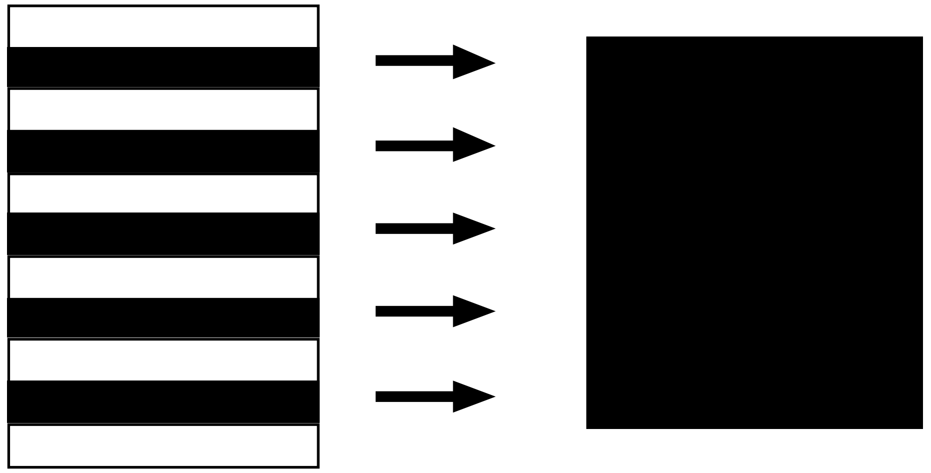
Magnification and minification parameters:

```
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

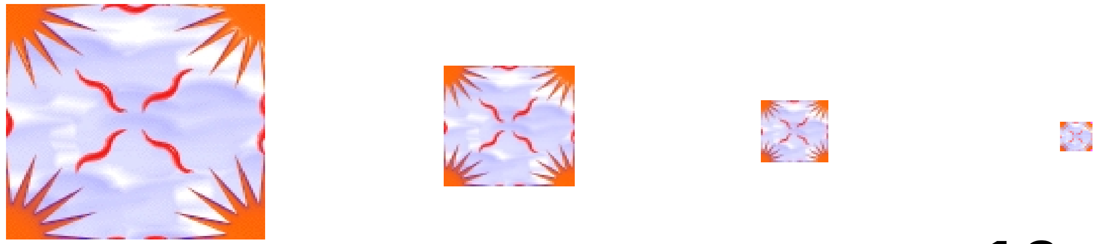
```
gluBuildMipmaps();

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!



MIP mapping

Gives anti-aliasing at a very low cost.

Good results in most situations.

Aliasing problems remain at steep angles.

Why texture size had to be power of 2

128x128, 64x256, 32x8, 1024x1…

Makes texture wrapping faster to calculate

Without: address = (s * w) mod w
With: address = (s << 7) & 127

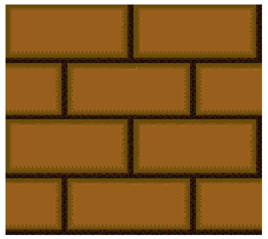
Binary AND and shifts instead of multiplication and division

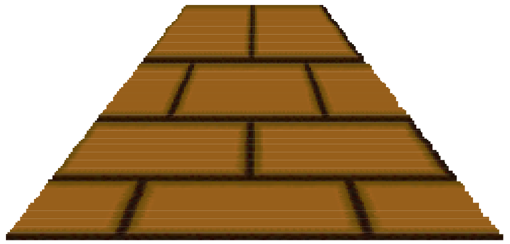
Recent GPUs do not have this limitation

Texture mapping is not just stretching:

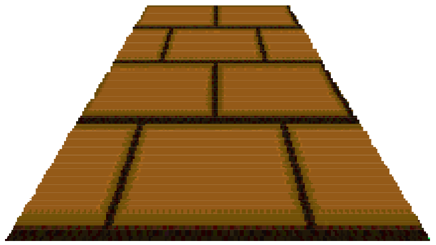
exture not
orrectly placed!

his is called
affine texture
apping"



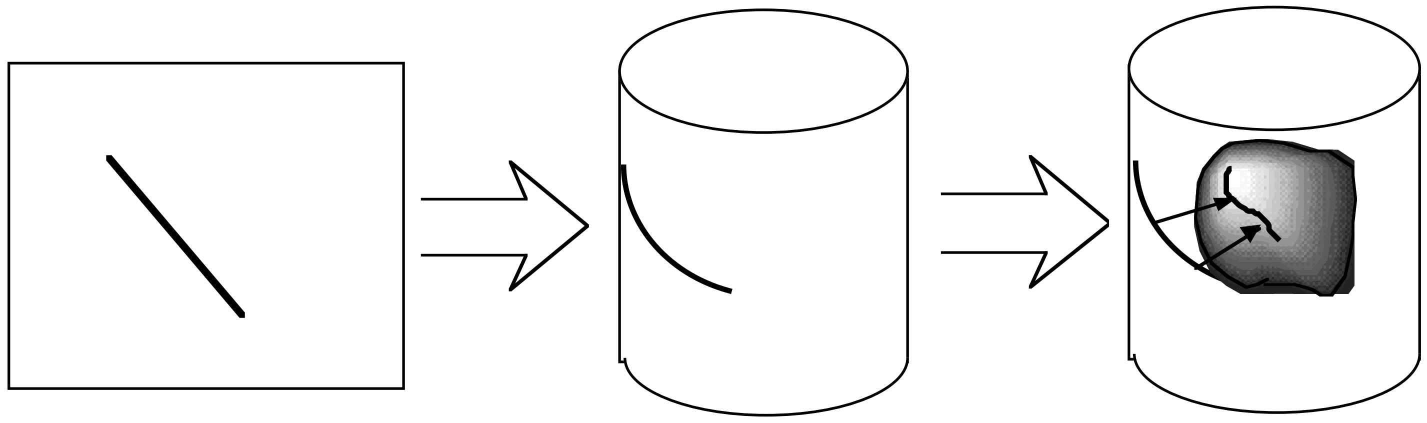


Affine

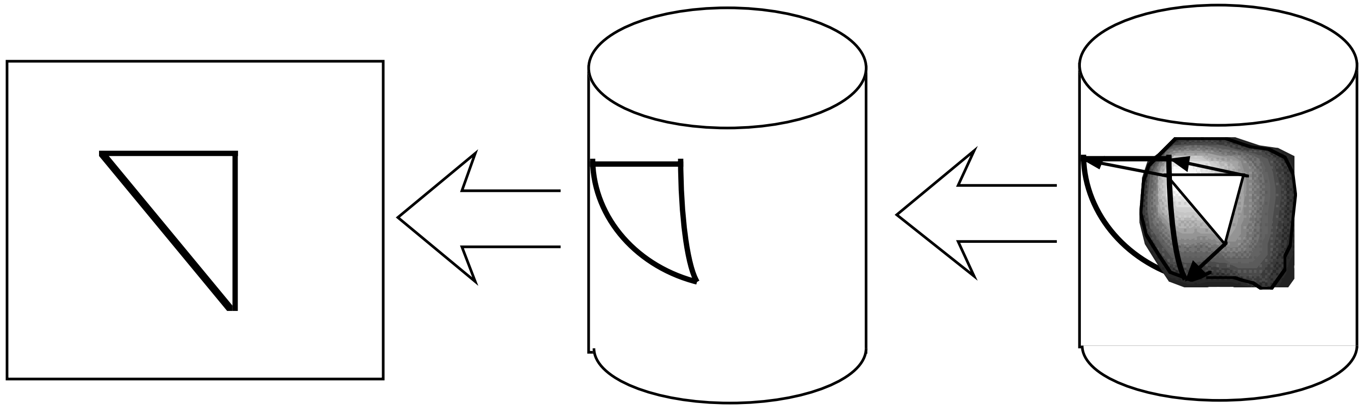


Perspective correct

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



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

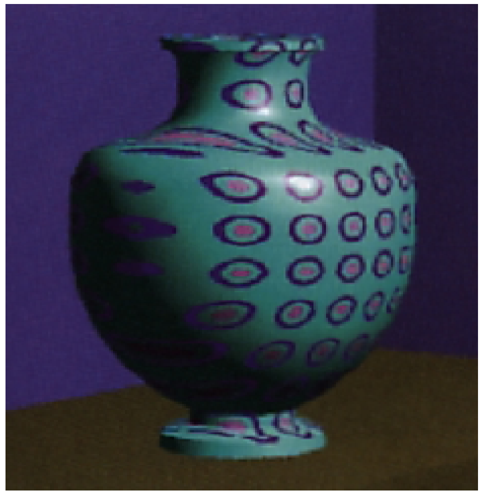


se to index the exture map

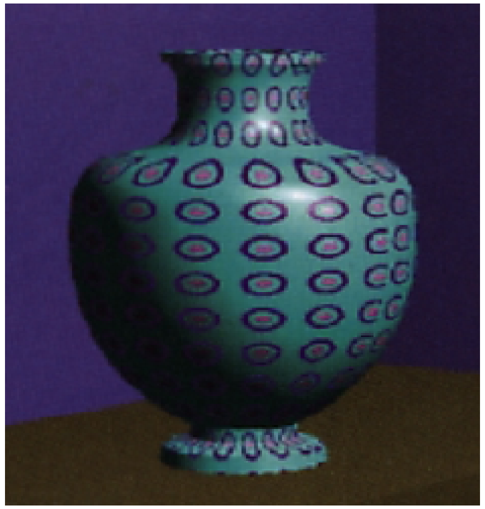
ap to inter- ediate surface oordinates

olygon ertex oordinates

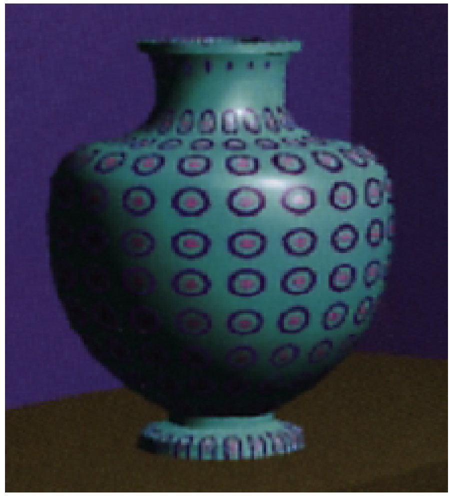
Examples



Planar



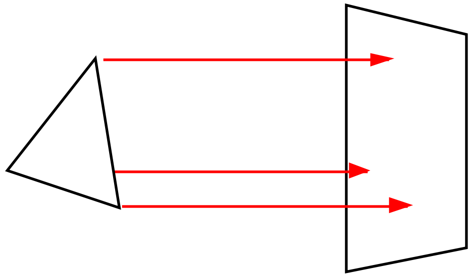
Cylinder



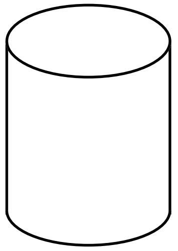
Sphere

Common mappings

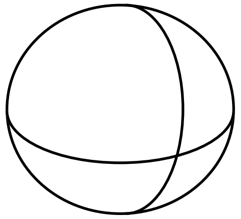
lanar



ylinder

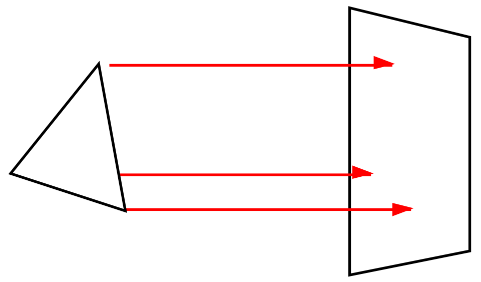


phere



Planar texture mapping

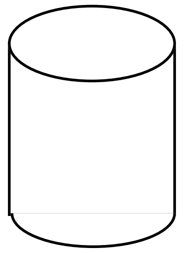
long z:
= x
= y



Cylindrical texture mapping

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

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

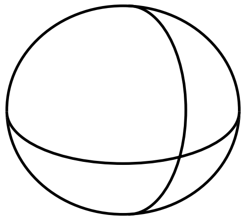


rctan(y,x) =
> 0: $\tan^{-1}(y/x)$
<0: $\pi + \tan^{-1}(y/x)$
$= 0$, $y > 0$: $\pi/2$
$= 0$, $y < 0$: $-\pi/2$

Spherical texture mapping

= Rcos(ϕ)cos(θ)
= Rcos(ϕ)sin(θ)
= Rsin(ϕ)

= arctan(y,x)
= sin-1(z/R)



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

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

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

ormalize, typically 0 to 1
xample: Cylinder

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

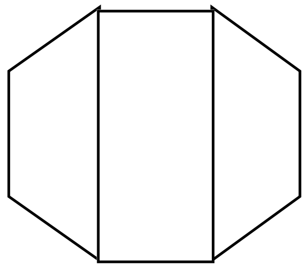
$= \arctan(y,x)$
$= z$

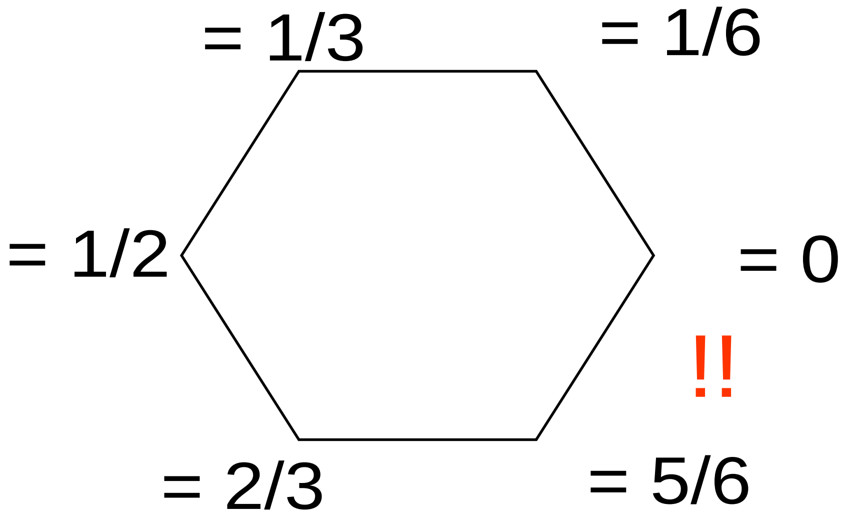
$= (u + \pi/2) / 2\pi$

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

Watch the edges!

xample: six-sided barrel





= Rcos(u)
= Rsin(u)

= arctan(y,x)
= z

= (u + π/2) / 2π

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

Automatic texture generation in OpenGL

```
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE,
GL_OBJECT_LINEAR);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE,
GL_OBJECT_LINEAR);
```

Can also calculate environment mapping!

Adjusting automatic texture generation in OpenGL

```
GLfloat planeS = {1, 0, 0, 0}; // 1
GLfloat planeT[] = {0, 0, 1, -1.5};

glTexGenfv(GL_S, GL_OBJECT_PLANE,
           &planeS[0]);
glTexGenfv(GL_T, GL_OBJECT_PLANE,
           &planeT[0]);
```