Texture Mapping

Texture Mapping

- Map a 2D image with a 3D surface.
- Can significantly increase the visual realism.
- Very important and popular in graphics.

Textures

A texture is a 2D array of pixels:

- Can be obtained by:
  - Photography images or paintings.
  - Procedural methods.
  - Texture synthesis.

Texture Mapping

Map a 2D image with a 3D surface.

- Can be done:
  - Manually.
  - Surface parameterization.
  - Texture synthesis.
- Ideally: no distortion.
- Practice: find a way to minimize the distortion.
  - Still a very active, open research problem!

Texture Mapping in OpenGL

- Textures can be 1D, 2D, 3D, …
- Mainly use 2D texture:

```
glEnable(GL_TEXTURE_2D);
```

- Several main steps:
  - Get an image, create a texture object
  - Compute texture coords for each vertex
  - Describe how to apply textures
  - Draw the surface with texture coordinates

Texture Mapping in OpenGL

- Step 1:
  - Get an image, create a texture object

```
glTexImage2D(GL_TEXTURE_2D, level, components,
  width, height, border, format, type, tarray);
```

  - An example:

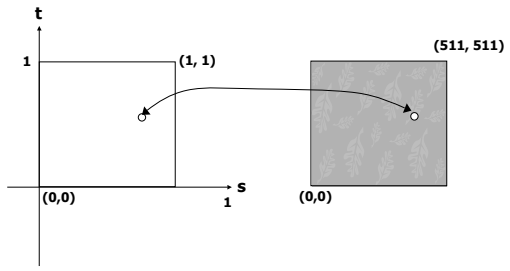
```
Glubyte my_textels[512][512];

glTexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, 0,
  GL_RGB, GL_UNSIGNED_BYTE, my_textels);
```

Texture Mapping in OpenGL

```
glBegin{GL_QUAD}
  glTexCoord2f(0.0, 0.0);
  glVertex3f(x1, y1, z1);
  glTexCoord2f(1.0, 0.0);
  glVertex3f(x2, y2, z2);
  glTexCoord2f(1.0, 1.0);
  glVertex3f(x3, y3, z3);
  glTexCoord2f(0.0, 1.0);
  glVertex3f(x4, y4, z4);
glBegin{GL_QUAD}
```


Texture Coordinates



Texture Coordinates

For texture coordinates value over the range of (0,1), we can either wrap it:

```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);
```

Or clamp it:

```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_t,GL_CLAMP);
```


Mipmapping

- For objects that project to an area of screen space that is small compared with the size of the texel array, we can create a series of texture arrays at reduced sizes through GLU function:

```
gluBuild2DMipmaps(GL_TEXTURE_2D, 3, 64, 64, GL_RGB, GL_UNSIGNED_BYTE, my_texels);
```

- We can also set up the maps directly using the `level` in `glTexImage2D()`.

Interaction between texture and shading

- The texture can modulate the shade by multiplying the color components of the texture by the color components of the shader.

```
glTexEnv(GL_TEX_ENV, GL_TEX_ENV_MODE, GL_MODULATE);
```

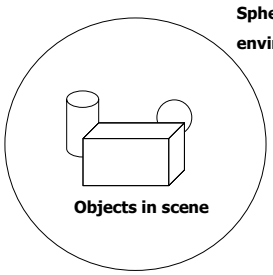
- Or, the color of the texture determines the color of the object completely-decaling.

```
glTexEnv(GL_TEX_ENV, GL_TEX_ENV_MODE, GL_DECAL);
```


Environment Mapping

- Also called reflection mapping.
- Define an array of intensity values that describes the environment around a single object or a set of objects.
- A fast approximation of the more accurate/expensive global illumination method such as ray tracing.

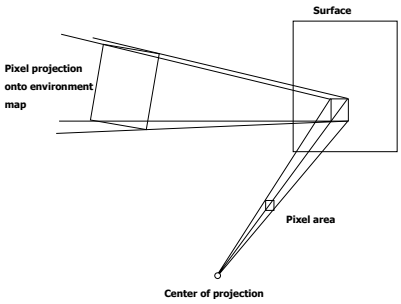
Environment Mapping



Environment Mapping

- To render the surface of an object, we project pixel areas onto the surface and then reflect the projected pixel area onto the environment map to pick up the surface-shading attributes for each pixel.
- Pixel intensity is then determined by averaging the intensity values within the intersected region of the environment map.

Environment mapping



Bump Mapping

- Texture mapping can be used to add fine surface detail, however, it is not good enough to model surface roughness such as orange.
- Bump mapping can create surface bumpiness by applying a perturbation function to the surface normal and use the perturbed normal in the illumination model calculations.

Bump Mapping

For a point P(u,v), the normal vector is $n = P_u \times P_v$

The perturbed position is P' (u,v), with bump function b(u,v)

$P'(u,v) = P(u,v) + b(u,v)\, n$

The perturbed normal vector is:

$n' = P_u' \times P_v'$

$P_u' = P_u + b_u n + b n_u \approx P_u + b_u n$, and $P_v' \approx P_v + b_v n$,

So, $n' = P_u \times P_v + b_u(n \times P_v) + b_v(P_u \times n) + b_u b_v (n \times n)$

And $n \times n = 0$, hence,

$n' = n + b_u(n \times P_v) + b_v(P_u \times n)$, and $n' = n' / \| n' \|$