Simple Texture Mapping Example (1)

```
void init( void ) {
  ...
  glPixelStorei(GL_UNPACK_ALIGNMENT, 1);

  glGenTextures(1, &texObj);

  glBindTexture(GL_TEXTURE_2D, texObj);

  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, imageWidth, imageHeight,
               0, GL_RGBA, GL_UNSIGNED_BYTE, texImage);
  ...
}
```

- set wrapping for S and T coords to repeat
- set linear interpolation for mipmapping

Simple Texture Mapping Example (2)

```
void display( void )  {
  ...
  glEnable(GL_TEXTURE_2D);

  glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);

  glBindTexture(GL_TEXTURE_2D, texObj);

  glBegin(GL_QUADS);
    glTexCoord2f(0.0, 0.0); glVertex3f(-2.0, -1.0, 0.0);
    glTexCoord2f(0.0, 1.0); glVertex3f(-2.0, 1.0, 0.0);
    glTexCoord2f(1.0, 1.0); glVertex3f(0.0, 1.0, 0.0);
    glTexCoord2f(1.0, 0.0); glVertex3f(0.0, -1.0, 0.0);
    glTexCoord2f(0.0, 0.0); glVertex3f(1.0, -1.0, 0.0);
    glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 1.0, 0.0);
    glTexCoord2f(1.0, 1.0); glVertex3f(2.4, 1.0, -1.4);
    glTexCoord2f(1.0, 0.0); glVertex3f(2.4, -1.0, -1.4);
  glEnd();
  ...
}
```


Reference

Example Explained

- **glBindTexture**(..., texObj) sets texObj as the current texture object for the current texture unit
- **glTexParameteri**(...) sets how texture in current texture object is treated
  - The states are attached to the bound texture object, and will be restored when the texture object is re-bound
- **glTexImage2D**(...) loads a texture image into the current texture object
- **glTexEnvf**(...) sets how a retrieved texture value is combined to the primary color of the fragment
  - The states are global (or attached to the current texture unit) and are not attached to any particular texture object

Reference

Specifying Texture Image

- void **glTexImage2D**(GLenum *target*, GLint *level*,
  GLint *internalFormat*,
  GLsizei *width*, GLsizei *height*,
  GLint *border*, GLenum *format*,
  GLenum *type*, const GLvoid *texImg);
  - *target* — GL_TEXTURE_2D, GL_TEXTURE_CUBE_MAP_POSITIVE_X, ...
  - *level* — mipmap level (0 for base level)
  - *internalFormat* — GL_RGB, GL_RGBA, GL_APLHA, GL_DEPTH_COMPONENT, and many more
  - *width, height* — width and height (in pixels) of the texture image
  - *border* — width of border around texture image: 0 or 1
  - *format* — components in the input image: GL_RGB, GL_RGBA, GL_RED, GL_APLHA, GL_DPETH_COMPONENT, etc.
  - *type* — data type of each component in input image: GL_UNSIGNED_BYTE, GL_BYTE, GL_FLOAT, etc.
  - *texImg* — pointer to the texture image data (and border if applicable)

Mipmap Generation

- int **gluBuild2DMipmaps**(GLenum *target*,
  GLint *internalFormat*,
  GLint *width*, GLint *height*,
  GLenum *format*, GLenum *type*, void *texImg);
- Example

```
void init( void ) {
  ...
  glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
  glGenTextures(1, &texObj);
  glBindTexture(GL_TEXTURE_2D, texObj);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                  GL_LINEAR_MIPMAP_LINEAR);
  ...
  gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGBA,
                    imageWidth, imageHeight,
                    GL_RGBA, GL_UNSIGNED_BYTE, texImage);
  ...
}
```


Using Framebuffer Data As Texture Image

- void **glCopyTexImage2D**(GLenum *target*, GLint *level*,
  GLint *internalFormat*, GLint *x*, GLint *y*,
  GLsizei *width*, GLsizei *height*, GLint *border*);
- Example

```
void display( void ) {
  ...
  drawScene();
  glReadBuffer( GL_BACK );
  glBindTexture(GL_TEXTURE_2D, texObj);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                  GL_LINEAR_MIPMAP_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_GENERATE_MIPMAP, GL_TRUE);
  ...
  glCopyTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 0, 0,
                   viewportWidth, viewportHeight, 0);
  ...
}
```

- Can also read **depth buffer** to texture object by using GL_DEPTH_COMPONENT as *internalFormat*

Texture Parameters

- Some of the texture parameters that can be set using **`glTexParameter*()`**

Parameter	Values

`GL_TEXTURE_WRAP_S`
`GL_TEXTURE_WRAP_T`
`GL_TEXTURE_WRAP_R`
	`GL_CLAMP`, `GL_CLAMP_TO_EDGE`, `GL_CLAMP_TO_BORDER`, `GL_REPEAT`, `GL_MIRRORED_REPEAT`

`GL_TEXTURE_MAG_FILTER`	`GL_NEAREST`, `GL_LINEAR`
`GL_TEXTURE_MIN_FILTER`	`GL_NEAREST`, `GL_LINEAR`, `GL_NEAREST_MIPMAP_NEAREST`, `GL_NEAREST_MIPMAP_LINEAR`, `GL_LINEAR_MIPMAP_NEAREST`, `GL_LINEAR_MIPMAP_LINEAR`
`GL_TEXTURE_BORDER_COLOR`	any 4 values in [0.0, 1.0]
`GL_GENERATE_MIPMAP`	`GL_TRUE` or `GL_FALSE`

Texture Functions / Environments

- Texture function can be set using

```
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, f );
```

where `f` is `GL_DECAL`, `GL_REPLACE`, `GL_MODULATE`, `GL_BLEND`, `GL_ADD`, or `GL_COMBINE`

- Defines how the texture color is combined with the underlying primary color of the fragment

Texture Base Internal Format	Texture source color C_s	A_s
`ALPHA`	$(0, 0, 0)$	A_t
`LUMINANCE`	(L_t, L_t, L_t)	1
`LUMINANCE_ALPHA`	(L_t, L_t, L_t)	A_t
`INTENSITY`	(I_t, I_t, I_t)	I_t
`RGB`	(R_t, G_t, B_t)	1
`RGBA`	(R_t, G_t, B_t)	A_t

Reference

Texture Functions (continued)

Texture Base Internal Format	`REPLACE` Function	`MODULATE` Function	`DECAL` Function
`ALPHA`	$C_v = C_p$ $A_v = A_s$	$C_v = C_p$ $A_v = A_p A_s$	*undefined*
`LUMINANCE` (or 1)	$C_v = C_s$ $A_v = A_p$	$C_v = C_p C_s$ $A_v = A_p$	*undefined*
`LUMINANCE_ALPHA` (or 2)	$C_v = C_s$ $A_v = A_s$	$C_v = C_p C_s$ $A_v = A_p A_s$	*undefined*
`INTENSITY`	$C_v = C_s$ $A_v = A_s$	$C_v = C_p C_s$ $A_v = A_p A_s$	*undefined*
`RGB` (or 3)	$C_v = C_s$ $A_v = A_p$	$C_v = C_p C_s$ $A_v = A_p$	$C_v = C_s$ $A_v = A_p$
`RGBA` (or 4)	$C_v = C_s$ $A_v = A_s$	$C_v = C_p C_s$ $A_v = A_p A_s$	$C_v = C_p(1 - A_s) + C_s A_s$ $A_v = A_p$

Useful for combining lighting result with texture map

Reference

Texture Functions (continued)

Texture Base Internal Format	`BLEND` Function	`ADD` Function
`ALPHA`	$C_v = C_p$ $A_v = A_p A_s$	$C_v = C_p$ $A_v = A_p A_s$
`LUMINANCE` (or 1)	$C_v = C_p(1 - C_s) + C_c C_s$ $A_v = A_p$	$C_v = C_p + C_s$ $A_v = A_p$
`LUMINANCE_ALPHA` (or 2)	$C_v = C_p(1 - C_s) + C_c C_s$ $A_v = A_p A_s$	$C_v = C_p + C_s$ $A_v = A_p A_s$
`INTENSITY`	$C_v = C_p(1 - C_s) + C_c C_s$ $A_v = A_p(1 - A_s) + A_c A_s$	$C_v = C_p + C_s$ $A_v = A_p + A_s$
`RGB` (or 3)	$C_v = C_p(1 - C_s) + C_c C_s$ $A_v = A_p$	$C_v = C_p + C_s$ $A_v = A_p$
`RGBA` (or 4)	$C_v = C_p(1 - C_s) + C_c C_s$ $A_v = A_p A_s$	$C_v = C_p + C_s$ $A_v = A_p A_s$