Texturing in GLSL

Mike Bailey
mjb@cs.oregonstate.edu

2D Texturing

Vertex shader:

```glsl
#version 330 compatibility
out vec2 vST;

void main()
{
    vST = gl_MultiTexCoord0.st;
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

Fragment shader:

```glsl
#version 330 compatibility
in vec2 vST;
uniform sampler2D uTexUnit;

void main()
{
    vec3 newcolor = texture(uTexUnit, vST).rgb;
    gl_FragColor = vec4(newcolor, 1.);
}
```

2D Texturing in the OpenGL API

```glsl
GLuint TexName; // a global
GLuint uTexUnit;

// in InitGraphics():
...) glGenTextures(1, &TexName);
...) glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
...) glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
...) glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
...) glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
...) glTexImage2D(GL_TEXTURE_2D, 0, 3, ...);
...) Pattern.Use();

// in Display():
...) glActiveTexture(GL_TEXTURE0 + uTexUnit);
...) Pattern.SetUniformVariable("uTexUnit", uTexUnit);
...
```

2D Texturing in glman

```glsl
Texture2D 6 worldtex.bmp

Vertex pattern.vert
Fragment pattern.frag
Program World uTexUnit 6
```

The OsuSphere Has Sensible s,t Coordinates Assigned

The GLUT Teapot Doesn't
What if You Want to Use Two Textures in One Shader?

C++ Program:

```cpp
// In Display():
Pattern.Use();
glActiveTexture(GL_TEXTURE5);
 glBindTexture(GL_TEXTURE_2D, TexName0);
 glActiveTexture(GL_TEXTURE6);
 glBindTexture(GL_TEXTURE_2D, TexName1);
 Pattern.SetUniformVariable("uTexUnit0", 5);
 Pattern.SetUniformVariable("uTexUnit1", 6);
<< draw something >>
 Pattern.UnUse();
```

```cpp
// Fragment shader:

#version 330 compatibility

in vec2 vST;
uniform sampler2D uTexUnit0;
uniform sampler2D uTexUnit1;

void main()
{
  vec3 newColor0 = texture(uTexUnit0, vST);
  vec3 newColor1 = texture(uTexUnit1, vST);
  gl_FragColor = ...
}
```

Why Might You Want to Use Two Textures in One Shader?

Once the RGBs have been read from a texture, they are just numbers. You can do any arithmetic you want with the texture RGBs, other colors, lighting, etc. Here is an example of blending two textures at once:

Fragment shader:

```cpp
// Fragment shader:

#version 330 compatibility

uniform float uBlend;
uniform float uSunLng;
uniform sampler2D uTexUnitDay, uTexUnitNight;
in float vLightIntensity;
in vec2 vST;
const float OFFSETS = 0.0; // how much the images are off horizontally

void main()
{
  float sunlng = uSunLng; // -180. to +180.
  float sunlngp = sunlng - 180.; // -180. to +180.
  vec2 nightST = vST;
  vec2 dayST = vST;
  dayST.s += OFFSETS;
  if (dayST.s < 0.0) dayST.s += 1.0;
  vec3 earthlng = 360.0 * (dayST.s - 0.5); // -180. to +180.
  float delta0 = earthlng - sunlng;
  if (delta0 < -180.0) delta0 += 360.0;
  if (delta0 > 180.0) delta0 -= 360.0;
  float delta1 = earthlng - sunlngp;
  if (delta1 < -180.0) delta1 += 360.0;
  if (delta1 > 180.0) delta1 -= 360.0;
  float delta = delta0;
  if (abs(delta1) < abs(delta0))
    delta = -delta1;
  vec3 rgbday = texture(uTexUnitDay, dayST).rgb;
  vec3 rgbnight = texture(uTexUnitNight, nightST).rgb;
  float t = smoothstep(-uBlend, uBlend, delta);
  vec3 newcolor = mix(rgbday, rgbnight, t);
  gl_FragColor = vLightIntensity * newcolor;
}
```

Using Two Textures in One Shader, I

```cpp
// Fragment shader:

#version 330 compatibility

uniform float uBlend;
uniform float uSunLng;
uniform sampler2D uTexUnitDay, uTexUnitNight;
in float vLightIntensity;
in vec2 vST;
const float OFFSETS = 0.0; // how much the images are off horizontally

void main()
{
  float sunlng = uSunLng; // -180. to +180.
  float sunlngp = sunlng - 180.; // -180. to +180.
  vec2 nightST = vST;
  vec2 dayST = vST;
  dayST.s += OFFSETS;
  if (dayST.s < 0.0) dayST.s += 1.0;
  vec3 earthlng = 360.0 * (dayST.s - 0.5); // -180. to +180.
  float delta0 = earthlng - sunlng;
  if (delta0 < -180.0) delta0 += 360.0;
  if (delta0 > 180.0) delta0 -= 360.0;
  float delta1 = earthlng - sunlngp;
  if (delta1 < -180.0) delta1 += 360.0;
  if (delta1 > 180.0) delta1 -= 360.0;
  float delta = delta0;
  if (abs(delta1) < abs(delta0))
    delta = -delta1;
  vec3 rgbday = texture(uTexUnitDay, dayST).rgb;
  vec3 rgbnight = texture(uTexUnitNight, nightST).rgb;
  float t = smoothstep(-uBlend, uBlend, delta);
  vec3 newcolor = mix(rgbday, rgbnight, t);
  gl_FragColor = vLightIntensity * newcolor;
}
```

Why Might You Want to Use Two Textures in One Shader, II

```cpp
// Fragment shader:

#version 330 compatibility

uniform float uBlend;
uniform float uSunLng;
uniform sampler2D uTexUnitDay, uTexUnitNight;
in float vLightIntensity;
in vec2 vST;
const float OFFSETS = 0.0; // how much the images are off horizontally

void main()
{
  float sunlng = uSunLng; // -180. to +180.
  float sunlngp = sunlng - 180.; // -180. to +180.
  vec2 nightST = vST;
  vec2 dayST = vST;
  dayST.s += OFFSETS;
  if (dayST.s < 0.0) dayST.s += 1.0;
  vec3 earthlng = 360.0 * (dayST.s - 0.5); // -180. to +180.
  float delta0 = earthlng - sunlng;
  if (delta0 < -180.0) delta0 += 360.0;
  if (delta0 > 180.0) delta0 -= 360.0;
  float delta1 = earthlng - sunlngp;
  if (delta1 < -180.0) delta1 += 360.0;
  if (delta1 > 180.0) delta1 -= 360.0;
  float delta = delta0;
  if (abs(delta1) < abs(delta0))
    delta = -delta1;
  vec3 rgbday = texture(uTexUnitDay, dayST).rgb;
  vec3 rgbnight = texture(uTexUnitNight, nightST).rgb;
  float t = smoothstep(-uBlend, uBlend, delta);
  vec3 newcolor = mix(rgbday, rgbnight, t);
  gl_FragColor = vLightIntensity * newcolor;
}
```

Shaders Can Combine More than Two Textures

Just assign each texture a different texture unit