What is a ‘texture unit’?

PrestoChung **Jan 2011**

I guess what I really want to know is there 1 texture per texture unit? Or can I generate multiple textures in the same texture unit? Thanks.

Alfonse_Reinheart Senior Member **Jan 2011**

> I guess what I really want to know is there 1 texture per texture unit? Or can I generate multiple textures in the same texture unit?

These two questions have nothing to do with each other, and the fact that you asked them both at the same time betrays a misunderstanding of how texturing works.

When you execute the following code:

```
GLuint tex;
glGenTextures(1, &tex);
```

You have (for all intents and purposes) created a texture. You have created a space that will hold all of the data associated with a texture. And the variable ‘tex’ now stores the reference to this texture.

When you call:

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, tex);
```

You are doing a number of things. You are attaching the texture object ‘tex’ to the texture unit number 0. You are telling OpenGL that the texture object ‘tex’ is a 2D texture, and you will refer to it (while it is bound) with GL_TEXTURE_2D.

When you call:

Skip to main content

```
glTexImage(GL_TEXTURE_2D, ...);
```

You are telling OpenGL to create image data, associate that image data with the texture that is currently bound to the active texture unit (GL_TEXTURE0) and is bound to the GL_TEXTURE_2D target. When called immediately after our previous functions, that will refer to the texture object 'tex'.

So this function will cause image data to be attached to the texture object 'tex'. But only because it is the texture currently bound to the current active texture unit and the GL_TEXTURE_2D target.

Now, when you call:

```
glBindTexture(GL_TEXTURE_2D, 0);
```

You are telling OpenGL that the currently bound texture object (aka: 'tex') is no longer bound to the current texture unit and GL_TEXTURE_2D target. This means that if you call glTexImage again, it will **not affect** the texture object 'tex'. Not unless you bind 'tex' again.

In short, 'tex' is a free-standing memory object. It stores all of the data associated with the texture. Unbinding it from a texture unit will have no affect on the **contents** of the object.

You can repeat this process for as long as you like. Well, until OpenGL runs out of memory. You can create textures with glGenTextures, bind them, create image data for them, and unbind them.

Now, when it comes time to **use** textures as source for image data, you must bind them again. But only the ones you need for that **particular** drawing operation.

Let's say you have object A and object B. Object A needs texture 'texA' bound to texture unit 0, and object B needs textures 'texB' and 'texC', bound to texture units 0 and 1 respectively.

To render object A, you do this:

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texA);
glEnable(GL_TEXTURE_2D); //Only use this if not using shaders.

glDrawElements(...);

//Cleanup texture binds.
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, 0);
glEnable(GL_TEXTURE_2D); //Only use this if not using shaders.
```

Skip to main content

To render object B, you do this:

```
glActiveTexture(GL_TEXTURE0 + 0);
glBindTexture(GL_TEXTURE_2D, texB);
glEnable(GL_TEXTURE_2D); //Only use this if not using shaders.
glActiveTexture(GL_TEXTURE0 + 1);
glBindTexture(GL_TEXTURE_2D, texC);
glEnable(GL_TEXTURE_2D); //Only use this if not using shaders.

glDrawElements(...);

//Cleanup texture binds.
glActiveTexture(GL_TEXTURE0 + 0);
glBindTexture(GL_TEXTURE_2D, 0);
glEnable(GL_TEXTURE_2D); //Only use this if not using shaders.
glActiveTexture(GL_TEXTURE0 + 1);
glBindTexture(GL_TEXTURE_2D, 0);
glEnable(GL_TEXTURE_2D); //Only use this if not using shaders.
```

Does that answer your question? Even though binding a texture to a texture unit is necessary for creating it and for using it, that does not mean that texture units and texture objects are permanently associated with each other.

If we wanted, when rendering objectB, ‘texA’ could have been bound to texture unit 1 instead of ‘texC’.

---

imported_Kelvin **Jan 2011**

You can think of the “texture unit” as “a piece of hardware that takes a sample of a texture” (using on texture coordinates). When drawing a particular object, each texture unit can sample one texture, but you can change the texture each sampler uses in between drawing objects.

The details of how texture sampling is controlled depends on whether you are using fixed-function OpenGL or GLSL for your fragment processing.

PrestoChung **Jan 2011**

I think I almost have the idea. I always assumed that the texture coordinates would refer to the active texture unit. So how do I specify which texture unit the texture coords are for? Is that what specifying a uniform sampler is doing?:

Skip to main content

cation(mShaderProgramHandle, “texsampler”);

glUniform1i(loc, 0);

IE, if I changed it to **glUniform1i(loc, 1);**, the texture coords would refer to unit 1 instead?

Alfonse_Reinheart Senior Member **Jan 2011**

> Is that what specifying a uniform sampler is doing?:

No. You should have said you were using shaders to begin with. That changes things.

The texture coordinates that are used to sample a texture are defined by the **shader**. The shader doesn't have to use texture coordinates passed as vertex attributes; it can make them up or generate them from arbitrary code.

When using shaders, the only thing the texture unit does is define which texture is used. By setting a sampler uniform to point to a particular texture unit, you are saying that any use of that sampler in the shader refers to the texture bound to that texture unit.

---

imported_Kelvin **Jan 2011**

My experience with the programmable pipeline is 99% theoretical at this point, since I've worked mostly with fixed function.

But here's how I read it:

> PrestoChung:
>
> So how do I specify which texture unit the texture coords are for? Is that what specifying a uniform sampler is doing?

Not quite. The association of texture coords with a texture unit is done in the shader. The uniform calls just set values that can be used in the shader:

e.g., in

```
vec4 texture2D (sampler2D sampler, vec2 coord)
```

the "sampler" variable specifies the texture unit you use, and "coord", the texture coordinate, comes from some appropriate attribute you fed into your shader. This call in the shader (or one like it) is how the GL knows which samplers get used with what texture coordinates.

> PrestoChung:
>
> **loc = glGetUniformLocation(mShaderProgramHandle, "texsampler");**

Skip to main content

> if I changed it to glUniform1i(loc, 1);, the texture coords would refer to unit 1 instead?

Yes, if you use “texsampler” as the texture unit selector for a texture lookup function in your shader code.

The actual texture the unit will sample is controlled by glBindTexture (using glActiveTexture to pick the unit you will bind to).

EDIT: Prettify quotes and code
EDIT: Add context to 2nd quote

PrestoChung **Jan 2011**

I see, thanks