SDL:Tutorials:Using SDL with OpenGL

From GPWiki

Using SDL with OpenGL

Prerequistes:

1. Knowledge of SDL
2. Knowledge of C
3. Knowledge of OpenGL
4. Read and understood Displaying a Bitmap With SDL

How To Include

Before using SDL's and OpenGL's functions, we first need to include the appropriate headers. SDL provides a header that makes this very easy:

```
#include "SDL.h"
#include "SDL_opengl.h"
```

You should recognize what SDL.h is for already. SDL_opengl.h, however, will include all of the

headers required by a specific platform needed to use OpenGL. When compiling on Windows, for instance, SDL_opengl.h will include <windows.h> before GL/gl.h and GL/glu.h to avoid errors.

How To Initialize SDL with OpenGL

SDL with OpenGL is slightly different from regular SDL initialization:

```
if ( SDL_Init(SDL_INIT_VIDEO) != 0 ) {
        printf("Unable to initialize SDL: %s\n", SDL_GetError());
        return 1;
}

SDL_GL_SetAttribute( SDL_GL_DOUBLEBUFFER, 1 ); // *new*

screen = SDL_SetVideoMode( 640, 480, 16, SDL_OPENGL | SDL_FULLSCREEN ); // *changed*
```

Notice the new line, *SDL_GL_SetAttribute(SDL_GL_DOUBLEBUFFER, 1);* (enables double buffering with OpenGL), and the new initialization flag, *SDL_OPENGL* (this is what tells SDL to use OpenGL).

Note: **DO NOT** use the initialization flag SDL_OPENGLBLIT! It is only there for backwards compatibility.

How To Set the OpenGL State

This tutorial will only cover how to set OpenGL for drawing in two dimensions. For three dimensions, see other tutorials. Keep in mind that every call to *SDL_SetVideoMode* destroys the OpenGL state, and therefore the OpenGL state needs to be set again.

Note: Even textures need to be loaded again after the OpenGL state is destroyed.

```
glEnable( GL_TEXTURE_2D );

glClearColor( 0.0f, 0.0f, 0.0f, 0.0f );
```

```
glViewport( 0, 0, 640, 480 );

glClear( GL_COLOR_BUFFER_BIT );

glMatrixMode( GL_PROJECTION );
glLoadIdentity();

glOrtho(0.0f, 640, 480, 0.0f, -1.0f, 1.0f);

glMatrixMode( GL_MODELVIEW );
glLoadIdentity();
```

This sets the clear color to black, sets the viewport, creates an orthogonal projection matrix, and sets the matrix back to modelview. These OpenGL states have nothing to do with SDL, or using SDL with OpenGL. They are purely OpenGL.

How To Load an OpenGL Texture from an SDL_Surface

We'll create an OpenGL texture using information SDL can provide us about a image using the SDL_Surface structure:

```
GLuint texture;                 // This is a handle to our texture object
SDL_Surface *surface;   // This surface will tell us the details of the image
GLenum texture_format;
GLint  nOfColors;

if ( (surface = SDL_LoadBMP("image.bmp")) ) {

        // Check that the image's width is a power of 2
        if ( (surface->w & (surface->w - 1)) != 0 ) {
                printf("warning: image.bmp's width is not a power of 2\n");
        }

        // Also check if the height is a power of 2
        if ( (surface->h & (surface->h - 1)) != 0 ) {
                printf("warning: image.bmp's height is not a power of 2\n");
        }

        // get the number of channels in the SDL surface
        nOfColors = surface->format->BytesPerPixel;
        if (nOfColors == 4)     // contains an alpha channel
        {
                if (surface->format->Rmask == 0x000000ff)
                        texture_format = GL_RGBA;
                else
                        texture_format = GL_BGRA;
        } else if (nOfColors == 3)     // no alpha channel
        {
                if (surface->format->Rmask == 0x000000ff)
```

```
                        texture_format = GL_RGB;
                else
                        texture_format = GL_BGR;
        } else {
                printf("warning: the image is not truecolor..  this will probably break
\n");
                // this error should not go unhandled
        }

        // Have OpenGL generate a texture object handle for us
        glGenTextures( 1, &texture );

        // Bind the texture object
        glBindTexture( GL_TEXTURE_2D, texture );

        // Set the texture's stretching properties
        glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR );
        glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR );

        // Edit the texture object's image data using the information SDL_Surface gives us
        glTexImage2D( GL_TEXTURE_2D, 0, nOfColors, surface->w, surface->h, 0,
                      texture_format, GL_UNSIGNED_BYTE, surface->pixels );
}
else {
        printf("SDL could not load image.bmp: %s\n", SDL_GetError());
        SDL_Quit();
        return 1;
}

// Free the SDL_Surface only if it was successfully created
if ( surface ) {
        SDL_FreeSurface( surface );
}
```

This is standard texture creation with OpenGL. If you do not understand how this works, you should read about OpenGL Textures. SDL_LoadBMP returns a SDL_Surface that stores the pixels as RGB. This is why GL_RGB is used when creating the texture.

How To Draw with OpenGL

Again, using OpenGL to draw a texture on the screen has nothing to do with SDL, but I'll include it just for the sake of making this tutorial more complete. We'll draw the image by texturing a polygon:

```
// Bind the texture to which subsequent calls refer to
glBindTexture( GL_TEXTURE_2D, texture );

glBegin( GL_QUADS );
```

```
        //Top-left vertex (corner)
        glTexCoord2i( 0, 0 );
        glVertex3f( 100, 100, 0.0f );

        //Bottom-left vertex (corner)
        glTexCoord2i( 1, 0 );
        glVertex3f( 228, 100, 0 );

        //Bottom-right vertex (corner)
        glTexCoord2i( 1, 1 );
        glVertex3f( 228, 228, 0 );

        //Top-right vertex (corner)
        glTexCoord2i( 0, 1 );
        glVertex3f( 100, 228, 0 );
glEnd();
```

Make sure that two-dimensional texturing is enabled using the glEnable(GL_TEXTURE_2D); function. Since we enabled this while creating the OpenGL state, there's no need to call it again.

How To Flip Buffers

When using OpenGL with SDL the screen buffer may be flipped with the following function:

```
SDL_GL_SwapBuffers();
```

How To Delete a Texture

Delete the OpenGL texture once we're finished with it:

```
glDeleteTextures( 1, &texture );
```

Source Code

Download the source code (sdl_ogl.zip contents: image.bmp, LICENSE.TXT, sdl_ogl.c)

Make sure your compiler is linking with OpenGL and SDL.

See also

- OpenGL Extensions with SDL

Retrieved from "http://gpwiki.org/index.php/SDL:Tutorials:Using_SDL_with_OpenGL"

Categories: C and OpenGL and SDL | All C articles | All OpenGL articles | All SDL articles

---