

# *Cross Platform OpenGL Using SDL*

*Robert Gustafson*



# *What are those acronyms?*

- OpenGL : (Open Graphics Library) API for producing 2D and 3D computer graphics.
- SDL : Simple DirectMedia Layer, a cross-platform multimedia programming library.



# *Commercial uses*

- OpenGL with SDL
  - Quake 4, Doom 3, UT2k4
  - Second Life
  - Spore
- OpenGL with native windows
  - World Of Warcraft
  - Enemy Territory: Quake Wars

# *Cross Platform Problems*

- Compiler Setup
  - Window Creation
  - Input Handling
  - Endian Issues
  - Vsync
  - Timers
- 
-

# *Compiler Setup*

- Windows and Linux
  - Header files and static libs
- OSX
  - Header files and included Frameworks



# Tiny SDL Application

```
int main()
{
    SDL_Init(SDL_INIT_VIDEO);

    const SDL_VideoInfo* info = SDL_GetVideoInfo();
    int bpp = info->vfmt->BitsPerPixel;

    int vidFlags = SDL_OPENGL | SDL_GL_DOUBLEBUFFER
        | SDL_HWSURFACE;

    SDL_SetVideoMode(window_width, window_height,
        bpp, vidFlags);
    GL_Setup(window_width, window_height);

    while(!done)
    {
        events();
        renderSomething();
        SDL_GL_SwapBuffers();
    }
}

void events()
{
    SDL_Event event;
    if( SDL_PollEvent(&event) )
    {
        switch( event.type )
        {
            case SDL_KEYDOWN :/*do something */
            case SDL_KEYUP   :/* do something*/
            case SDL_QUIT    :/*shutdown app*/
            }
        }
    }

    // Initialize OpenGL perspective matrix
    void GL_Setup(int width, int height)
    {
        glViewport( 0, 0, width, height );
        glMatrixMode( GL_PROJECTION );
        glEnable( GL_DEPTH_TEST );
        gluPerspective( 45, (float)width/height, 0.1,
            100 );
        glMatrixMode( GL_MODELVIEW );
    }
}
```

# *renderSomething()*

```
void renderSomething()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0,0, -10);
    glRotatef(angle, 0, 0, 1);
    glBegin(GL_QUADS);
        glVertex2f(-1, 1);
        glVertex2f( 1, 1);
        glVertex2f( 1, -1);
        glVertex2f(-1, -1);
    glEnd();
}
```

# *Tech Demo*



# SDL & Timers

```
SDL_GetTicks();
```

A running count of ms passed since SDL initialization

Timers:

```
SDL_AddTimer(interval, callback, param)
```

```
SDL_RemoveTimer(t)
```

```
SDL_SetTimer(500, flash_callback);
```

```
unsigned int flash_callback(unsigned int interval){  
    lightOn = !lightOn;  
    return 500; // call again after interval passes  
}
```

# *SDL & User Input*

- Keyboard
- Mouse
- Joystick type devices



# SDL & Endian issues

- Determine the endianness of the current system
- Swap data on systems of differing endianness

```
void ReadScanline16(FILE *file, Uint16 *scanline, int length)
{
    fread(scanline, length, sizeof(Uint16), file);
    if ( SDL_BYTEORDER == SDL_BIG_ENDIAN ) {
        int i;
        for ( i=length-1; i >= 0; --i )
            scanline[i] = SDL_SwapLE16(scanline[i]);
    }
}
```

# Vsync example

```
#ifdef __APPLE__  
GLint sync = 1;  
CGLSetParameter(CGLGetCurrentContext(),  
    kCGLCPSwapInterval, &sync);  
#endif
```

# *Platform specific code*

```
void doSomethingPlatformSpecific(){  
  
#ifdef linux  
    SDL_WM_SetCaption( "sd1Demo1 - running on Linux", NULL );  
#endif  
  
#ifdef __APPLE__  
    SDL_WM_SetCaption( "sd1Demo1 - running on OSX", NULL );  
#endif  
  
#ifdef WIN32  
    SDL_WM_SetCaption( "sd1Demo1 - running on Windows", NULL );  
#endif  
}
```

---

---

# *SDL code for loading a bitmap*

```
GLuint texture;           // This is a handle to our texture object
SDL_Surface *surface;    // This surface holds details of the image
GLenum texture_format;

if ( (surface = SDL_LoadBMP("image.bmp")) ) {
    if (surface->format->Rmask == 0x000000ff)
        texture_format = GL_RGB;
    else
        texture_format = GL_BGR;

    glGenTextures( 1, &texture );

    glBindTexture( GL_TEXTURE_2D, texture );

    // Edit the texture object's image data using the SDL_surface
    glTexImage2D( GL_TEXTURE_2D, 0, 3, surface->w, surface->h, 0,
        texture_format, GL_UNSIGNED_BYTE, surface->pixels );
}
```

# *Discussion*

- ~~OpenGL vs DirectX~~
  - SDL vs GLUT vs Your Preference
  - Porting vs Platform Neutral from Day 1
- 
-