Lazy Foo' Productions

Getting an Image on the Screen

Last Updated 10/06/13

Now that you've already got a window open, let's put an image on it.

Note: From now on the tutorials will only cover key parts of source code. For the full program, you will have to download the full source code.

```
//Starts up SDL and creates window
bool init();

//Loads media
bool loadMedia();

//Frees media and shuts down SDL
void close();
```

In the first tutorial, we put everything in the main function. Since it was a small program we can get away with that, but in real programs (like video games) you want to have your code as modular as possible. This means you want your code to be in neat chunks that are each easy to debug and reuse.

Here it means we have functions to handle initialization, loading media, and closing down the SDL application. We declare these near the top of our source file.

```
//The window we'll be rendering to
SDL_Window* gWindow = NULL;

//The surface contained by the window
SDL_Surface* gScreenSurface = NULL;

//The image we will load and show on the screen
SDL_Surface* gHelloWorld = NULL;
```

Here we declare some global variables. Typically you want to avoid using global variables in large programs. The reason we're doing it here is because we want the source code to be as simple as possible, but in large projects global variables make things more complicated. Since this is a single source file program we don't have to worry about it too much.

Here's a new data type called an SDL Surface. An SDL surface is just an image data type that contains the pixels of an image along with all data needed to render it. SDL surfaces use software rendering which means it uses the CPU to render. It is possible to render hardware images but it's a bit more difficult so we're going to learn it the easy way first. In future tutorials we'll cover how to render GPU accelerated images.

The images we're going to be dealing with here are the screen image (what you see inside of the window) and the image we'll be loading from a file.

Notice that these are pointers to SDL surfaces. The reason is that 1) we'll be dynamically allocating memory to load images and 2) it's better to reference an image by memory location. Imagine you had a game with a brick wall that consisted of the same brick image being rendered multiple times (like Super Mario Bros). It's wasteful to have dozens of copies of the image in memory when you can have one copy of the image and render it over and over again.

Also, always remember to initialize your pointers. We set them to NULL immediately when declaring them.

```
bool init()
{
    //Initialization flag
    bool success = true;
```

```
    //Initialize SDL
    if( SDL_Init( SDL_INIT_VIDEO ) < 0 )
    {
        printf( "SDL could not initialize! SDL_Error: %s\n", SDL_GetError() );
        success = false;
    }
    else
    {
        //Create window
        gWindow = SDL_CreateWindow( "SDL Tutorial", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, SCREEN_WIDTH, SCREEN_HEIGHT, SDL_WINDOW
        if( gWindow == NULL )
        {
            printf( "Window could not be created! SDL_Error: %s\n", SDL_GetError() );
            success = false;
        }
        else
        {
            //Get window surface
            gScreenSurface = SDL_GetWindowSurface( gWindow );
        }
    }

    return success;
}
```

As you can see here, we've taken the SDL initialization and the window creation code and put it in its own function. What's new is that there's a call to SDL_GetWindowSurface.

We want to show images inside of the window and in order to do that we need to get the image inside of the window. So we call SDL_GetWindowSurface to grab the surface contained by the window.

```
bool loadMedia()
{
    //Loading success flag
    bool success = true;

    //Load splash image
    gHelloWorld = SDL_LoadBMP( "02_getting_an_image_on_the_screen/hello_world.bmp" );
    if( gHelloWorld == NULL )
    {
        printf( "Unable to load image %s! SDL Error: %s\n", "02_getting_an_image_on_the_screen/hello_world.bmp", SDL_GetError() );
        success = false;
    }

    return success;
}
```

In the load media function we load our image using SDL_LoadBMP. SDL_LoadBMP takes in the path of a bmp file and returns the loaded surface. If the function returns NULL, that means it failed so we print to the console an error using SDL_GetError.

An important thing to note is that this piece of code assumes you have a directory called "02_getting_an_image_on_the_screen" that contains an image named "hello_world.bmp" in your working directory. The working directory is where your application thinks it is operating. Typically, your working directory is the directory where your executable is at but some programs like Visual Studio change the working directory to where the vcxproj file is located. So if your application can't find the image, make sure it is in the right place.

```
void close()
{
    //Deallocate surface
    SDL_FreeSurface( gHelloWorld );
    gHelloWorld = NULL;

    //Destroy window
    SDL_DestroyWindow( gWindow );
    gWindow = NULL;

    //Quit SDL subsystems
    SDL_Quit();
}
```

In our clean up code, we destroy the window and quit SDL like before but we also have to take care of the surface we loaded. We do this by freeing it with SDL_FreeSurface. Don't worry about the screen surface, SDL_DestroyWindow will take care of it.

Make sure to get into the habit of having your pointers point to NULL when they're not pointing to anything.

```
int main( int argc, char* args[] )
{
    //Start up SDL and create window
    if( !init() )
    {
        printf( "Failed to initialize!\n" );
    }
    else
    {
        //Load media
        if( !loadMedia() )
        {
            printf( "Failed to load media!\n" );
```

```
        }
    else
    {
        //Apply the image
        SDL_BlitSurface( gHelloWorld, NULL, gScreenSurface, NULL );
```

In our main function we initialize SDL and load the image. If that succeeded we blit the loaded surface onto the screen surface using SDL_BlitSurface.

What blitting does is take a source surface and stamps a copy of it onto the destination surface. The first argument of SDL_BlitSurface is the source image. The third argument is the destination. We'll worry about the 2nd and 4th arguments in future tutorials.

Now if this was the only code for drawing we had, we still wouldn't see the image we loaded on the screen. There's one more step.

```
        //Update the surface
        SDL_UpdateWindowSurface( gWindow );
```

After drawing everything on the screen that we want to show for this frame we have to update the screen using SDL_UpdateWindowSurface. See when you draw to the screen, you are not typically drawing to the image on the screen you see. By default, most rendering systems out there are double buffered. These two buffers are the front and back buffer.

When you make draw calls like SDL_BlitSurface, you render to the back buffer. What you see on the screen is the front buffer. The reason we do this is because most frames require drawing multiple objects to the screen. If we only had a front buffer, we would be able to see the frame as things are being drawn to it which means we would see unfinished frames. So what we do is draw everything to the back buffer first and once we're done we swap the back and front buffer so now the user can see the finished frame.

This also means that you don't call SDL_UpdateWindowSurface after every blit, only after all the blits for the current frame are done.

```
        //Wait two seconds
        SDL_Delay( 2000 );
    }
    }

    //Free resources and close SDL
    close();

    return 0;
}
```

Now that we've rendered everything to the window, we delay for two seconds so the window doesn't just disappear. After the wait is done, we close out our program.

Download the media and source code for this tutorial here.

Back to SDL Tutorials