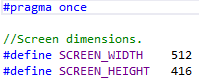
SDL Tutorial 2

Opening an SDL Window

Before starting this tutorial make sure you have completed Tutorial 1 – Setting up SDL.

In this tutorial we are going to go through the basics of opening a simple window in SDL. The code will be supplied at each step along with explanations.

1. Open your SDL project in Visual Studio.
2. As games will tend to accumulate a lot of constants we will need to set up a file to hold these.
   1. Create ‘Constants.h’ and input the following code:



1. Back in the main source file include the constants file file.
2. We are going to be using the functionality provided by iostream.h, so include this in to your source file. Remember to setup the std namespace with:



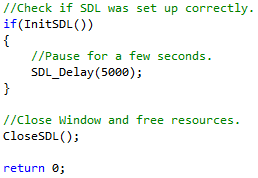
1. Add the following just below the include declarations:



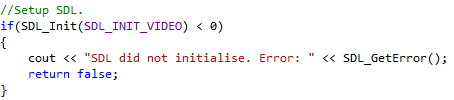
1. We will be splitting the code up in to separate functions. This will give us bite size chunks to work through. Add the following function prototypes to your file:



1. In the main function add the following:



1. Now we will look at the initialise function for SDL: InitSDL().
   1. To be able to call any SDL functions we must first initialise SDL. If for some reason SDL does not set up correctly then we will inform the user and return false.



The SDL\_INIT\_VIDEO flag passed in to the SDL\_Init function is to specify that we are using the video subsystems.

The other flags available are:

SDL\_INIT\_TIMER The Timer subsystem.

SDL\_INIT\_AUDIO The Audio subsystem.

SDL\_INIT\_CDROM The CDRom subsystem.

SDL\_INIT\_JOYSTICK The Joystick subsystem

SDL\_INIT\_EVERYTHING All of the above

SDL\_INIT\_NOPARACHUTE Prevents SDL from catching fatal signals

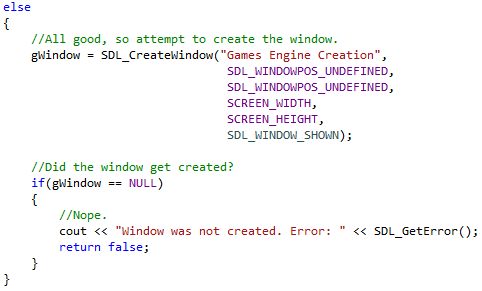
SDL\_INIT\_EVENTTHREAD Runs the event manager in a separate thread.

Multiple flags can be set by using the | operator.

For example: SDL\_Init(SDL\_INIT\_VIDEO | SDL\_INIT\_AUDIO)

SDL\_Init will return -1 to signify the initialisation failed and 0 on success. On failure we need to inform the user what went wrong. Using the SDL\_GetError() function allows us to detail what the error was.

* 1. If setup goes well we need to create a window. This will use the dimensions we set in the Constants header file.



The SDL\_CreateWindow() function will create us a window and return a pointer to a SDL\_Window. The parameters are as follows:

* The first is a string that will be displayed in the bar at the top of the window.
* The next two are screen positions, X and Y. SDL\_WINDOW\_UNDEFINED allows us to not set a particular position.
* The fourth is an integer to define the width of the new window.
* The fifth is also an integer, but this one specifies the height of the new window.
* The final is a flag to state whether the window should be instantly shown.

The other available window flags are:

SDL\_WINDOW\_FULLSCREEN Fullscreen window

SDL\_WINDOW\_FULLSCREEN\_DESKTOP Fullscreen window at current desktop resolution

SDL\_WINDOW\_OPENGL Window usable with OpenGL context

SDL\_WINDOW\_HIDDEN Window is not visible

SDL\_WINDOW\_BORDERLESS No window decoration

SDL\_WINDOW\_RESIZABLE Window can be resized

SDL\_WINDOW\_MINIMIZED Window is minimized

SDL\_WINDOW\_MAXIMISED Window is maximised

SDL\_WINDOW\_INPUT\_GRABBED Window has grab input focus

SDL\_WINDOW\_INPUT\_FOCUS Window has input focus

SDL\_WINDOW\_MOUSE\_FOCUS Window has mouse focus

SDL\_WINDOW\_FORIEGN Window not created by SDL

SDL\_WINDOW\_ALLOW\_HIGHDPI Window should be created with High-DPI format

(if supported)

Back to the code, we make a check to see if the window was created. It the pointer returned was NULL there was an error. We need to catch this error and output the error to the user.

* 1. If the code progresses without returning out of the InitSDL() function then all must have been set up correctly. So return true.



1. The remaining function to be discussed is the CloseSDL() function. This is where we need to free up the memory used and close the SDL subsytems that have been set up.
   1. First free the memory from the window.



* 1. Next call the SDL functions to quit the subsytems.



1. Now build and run the program.

Upon completion of this tutorial a plain window should open and remain on the screen for 5 seconds before automatically closing. In the next tutorial we will be looking at using SDL Events to allow us to close the window using the X in the window.