# Lazy Foo' Productions

## Resizable Windows and Window Events

Last Updated 11/15/07

Window events are things like resizing, maximizing, or minimizing the window. Up until now we couldn't do much with our window, mainly because it's more work to deal with it being resizable. This tutorial shows you how to make an adjustable window and handle the events from the window changes.

```
//Our window
class Window
{
    private:
    //Whether the window is windowed or not
    bool windowed;

    //Whether the window is fine
    bool windowOK;

    public:
    //Constructor
    Window();

    //Handle window events
    void handle_events();

    //Turn fullscreen on/off
    void toggle_fullscreen();

    //Check if anything's wrong with the window
    bool error();
};
```

Now we have a window class to manage all the things a resizable window can do.

For variables we have two flags. The "windowed" flag keeps track of whether the screen is windowed or fullscreen. "windowOK" is whether the window is operational.

Then we have a constructor, a function to handle events, a function to toggle windowed/fullscreen modes and an error checking function.

```
bool init()
{
    //Initialize all SDL subsystems
    if( SDL_Init( SDL_INIT_EVERYTHING ) == -1 )
    {
        return false;
    }

    //If eveything loads fine
    return true;
}
```

Here's our init() function.

Because everything associated with the window is in a seperate class, all we do is initialize SDL.

```
Window::Window()
{
    //Set up the screen
    screen = SDL_SetVideoMode( SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_BPP, SDL_SWSURFACE | SDL_RESIZABLE );
    //If there's an error
    if( screen == NULL )
    {
        windowOK = false;
        return;
    }
    else
    {
        windowOK = true;
    }

    //Set the window caption
    SDL_WM_SetCaption( "Window Event Test", NULL );

    //Set window flag
    windowed = true;
}
```

In the window class' constructor we create a window. Since we want it to be resizeable, we pass the SDL_RESIZABLE flag.

Then we check if the screen is NULL. If it is we set "windowOK" to false so we can tell that there was a problem.

If everything set up fine, we set "windowOK" to true, set the caption and since we're starting out windowed we set the "windowed" flag to true.

```
void Window::toggle_fullscreen()
{
    //If the screen is windowed
    if( windowed == true )
    {
        //Set the screen to fullscreen
        screen = SDL_SetVideoMode( SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_BPP, SDL_SWSURFACE | SDL_RESIZABLE | SD

        //If there's an error
        if( screen == NULL )
        {
            windowOK = false;
            return;
        }

        //Set the window state flag
        windowed = false;
    }
    //If the screen is fullscreen
    else if( windowed == false )
    {
        //Window the screen
        screen = SDL_SetVideoMode( SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_BPP, SDL_SWSURFACE | SDL_RESIZABLE );

        //If there's an error
        if( screen == NULL )
        {
            windowOK = false;
            return;
        }

        //Set the window state flag
        windowed = true;
    }
}
```

In this program I added the ability to switch from windowed to fullscreen and back when you press enter, which you can only do with resizable window. Here's the function that toggles windowed/fullscreen modes.

When enter is pressed we check the if the screen is windowed or not. If it's windowed, we call SDL_SetVideoMode() with the SDL_FULLSCREEN flag to switch to fullscreen and then change "windowed" to false. If we're already fullscreen we call SDL_SetVideoMode() normally to switch to windowed mode and then change the "windowed" flag to true.

```
void Window::handle_events()
{
    //If there's something wrong with the window
    if( windowOK == false )
    {
        return;
    }

    //If the window resized
    if( event.type == SDL_VIDEORESIZE )
    {
        //Resize the screen
        screen = SDL_SetVideoMode( event.resize.w, event.resize.h, SCREEN_BPP, SDL_SWSURFACE | SDL_RESIZABLE );

        //If there's an error
        if( screen == NULL )
        {
            windowOK = false;
            return;
        }
    }
```

Here we start our event handling and the first thing we do is check if the window is operational. Then we handle is a video resize event.

Whenever the window is resized you must set the video over again using SDL_SetVideoMode(). The dimensions of the new window is stored in our event structure in event.resize.w and event.resize.h so you can easily set the screen to the new size.

```
    //If enter was pressed
    else if( ( event.type == SDL_KEYDOWN ) && ( event.key.keysym.sym == SDLK_RETURN ) )
    {
        //Turn fullscreen on/off
        toggle_fullscreen();
    }
```

When enter is pressed we toggle window/fullscreen modes.

```
    //If the window focus changed
    else if( event.type == SDL_ACTIVEEVENT )
    {
        //If the window was inconified or restored
        if( event.active.state & SDL_APPACTIVE )
        {
            //If the application is no longer active
            if( event.active.gain == 0 )
            {
                SDL_WM_SetCaption( "Window Event Test: Iconified", NULL );
            }
            else
            {
                SDL_WM_SetCaption( "Window Event Test", NULL );
            }
        }
```

Another type of event is SDL_ACTIVEEVENT. A SDL_ACTIVEEVENT happens whenever the screen, keyboard, or mouse becomes active or inactive.

Here we check if the screen became active or inactive by being iconified/minimized or restored. When the window is iconified/minimized, "gain" in our SDL_ActiveEvent structure is 0. When the window is restored, "gain" is 1.

In this program we change the caption to notify the user of change.

```
        //If something happened to the keyboard focus
        else if( event.active.state & SDL_APPINPUTFOCUS )
        {
            //If the application lost keyboard focus
            if( event.active.gain == 0 )
            {
                SDL_WM_SetCaption( "Window Event Test: Keyboard focus lost", NULL );
            }
            else
            {
                SDL_WM_SetCaption( "Window Event Test", NULL );
            }
        }
        //If something happened to the mouse focus
        else if( event.active.state & SDL_APPMOUSEFOCUS )
        {
            //If the application lost mouse focus
            if( event.active.gain == 0 )
            {
                SDL_WM_SetCaption( "Window Event Test: Mouse Focus Lost", NULL );
            }
            else
            {
                SDL_WM_SetCaption( "Window Event Test", NULL );
            }
        }
    }
```

This where keyboard and mouse focus is handled.

SDL_APPINPUTFOCUS is change in keyboard focus. SDL_APPMOUSEFOCUS is change in mouse focus. As with SDL_APPACTIVE when focus is lost "gain" is 0, when focus is restored "gain" is 1.

Remember to check for SDL_APPACTIVE first because bitwise AND comparisons with "state" will report other active events.

```
    //If the window's screen has been altered
    else if( event.type == SDL_VIDEOEXPOSE )
    {
        //Update the screen
        if( SDL_Flip( screen ) == -1 )
        {
            //If there's an error
            windowOK = false;
            return;
        }
    }
}
```

Lastly, we handle the SDL_VIDEOEXPOSE event.

A SDL_VIDEOEXPOSE event happens whenever the screen is altered by something outside the program. We handle it by updating the screen.

```
bool Window::error()
{
    return !windowOK;
}
```

Here's the function we use to check if there's errors with our window.

We negate "windowOK" because when the window is ok there are no errors and when the window is not ok there is an error.

```
    //Quit flag
    bool quit = false;

    //Initialize
    if( init() == false )
    {
        return 1;
    }

    //Create a window
    Window myWindow;

    //If the window failed
    if( myWindow.error() == true )
    {
        return 1;
    }

    //Load the files
    if( load_files() == false )
    {
        return 1;
    }
```

At the top of our main function we create a Window object after we initialize. Then we check for errors in the window and continue normally.

```
    //While the user hasn't quit
    while( quit == false )
    {
        //While there's events to handle
        while( SDL_PollEvent( &event ) )
        {
            //Handle window events
            myWindow.handle_events();

            //If escape was pressed
            if( ( event.type == SDL_KEYDOWN ) && ( event.key.keysym.sym == SDLK_ESCAPE ) )
            {
                //Quit the program
                quit = true;
            }

            //If the user has Xed out the window
            if( event.type == SDL_QUIT )
            {
                //Quit the program
                quit = true;
            }
        }

        //If the window failed
        if( myWindow.error() == true )
        {
            return 1;
        }

        //Fill the screen white
        SDL_FillRect( screen, &screen->clip_rect, SDL_MapRGB( screen->format, 0xFF, 0xFF, 0xFF ) );

        //Show the image centered on the screen
        apply_surface( ( screen->w - testing->w ) / 2, ( screen->h - testing->h ) / 2, testing, screen );

        //Update the screen
        if( SDL_Flip( screen ) == -1 )
        {
            return 1;
        }
    }
```

Here's our main loop. Nothing much to point out other than we have to remember check for errors in the window before we render. There's no point to render to a screen that's non-operational.

Download the media and source code for this tutorial here.

News FAQs Games Tutorials Articles Contact Donations