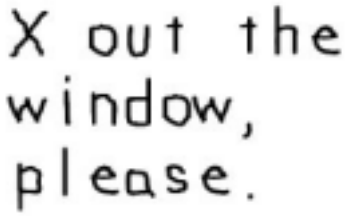
Lazy Foo' Productions

Make windows games
Learn windows pc game programming using C++, SDL / OpenGL or DirectX

SDL Real Time
Combine SDL graphical abstraction and C Language precision

Ads by Google

Event Driven Programming



Last Updated 11/15/07

Up until this point you're probably used to command driven programs using cin and cout. This tutorial will teach you how to check for events and handle events.

An event is simply something that happens. It could be a key press, movement of the mouse, resizing the window or in this case when the user wants to X out the window.

```
//The headers
#include "SDL/SDL.h"
#include "SDL/SDL_image.h"
#include <string>

//Screen attributes
const int SCREEN_WIDTH = 640;
const int SCREEN_HEIGHT = 480;
const int SCREEN_BPP = 32;

//The surfaces
SDL_Surface *image = NULL;
SDL_Surface *screen = NULL;
```

Here we have the same story as before, we have our headers, constants and surfaces.

```
//The event structure that will be used
SDL_Event event;
```

Now this is new. A SDL_Event structure stores event data for us to handle.

```
SDL_Surface *load_image( std::string filename )
{
    //The image that's loaded
    SDL_Surface* loadedImage = NULL;

    //The optimized image that will be used
    SDL_Surface* optimizedImage = NULL;

    //Load the image
```

```
    loadedImage = IMG_Load( filename.c_str() );

    //If the image loaded
    if( loadedImage != NULL )
    {
        //Create an optimized image
        optimizedImage = SDL_DisplayFormat( loadedImage );

        //Free the old image
        SDL_FreeSurface( loadedImage );
    }

    //Return the optimized image
    return optimizedImage;
}

void apply_surface( int x, int y, SDL_Surface* source, SDL_Surface* destination )
{
    //Temporary rectangle to hold the offsets
    SDL_Rect offset;

    //Get the offsets
    offset.x = x;
    offset.y = y;

    //Blit the surface
    SDL_BlitSurface( source, NULL, destination, &offset );
}
```

Here we have our surface loading and blitting functions. Nothing has changed from the previous tutorial.

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

    //Set up the screen
    screen = SDL_SetVideoMode( SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_BPP, SDL_SWSURFACE

    //If there was an error in setting up the screen
    if( screen == NULL )
    {
        return false;
    }

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

    //If everything initialized fine
    return true;
}
```

Here is the initialization function. This function starts up SDL, sets up the window, sets the caption and returns false if there are any errors.

```
bool load_files()
{
    //Load the image
    image = load_image( "x.png" );

    //If there was an error in loading the image
    if( image == NULL )
    {
        return false;
    }
```

```
    //If everything loaded fine
    return true;
}
```

Here is the file loading function. It loads the images, and returns false if there were any problems.

```
void clean_up()
{
    //Free the image
    SDL_FreeSurface( image );

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

Here we have the end of the program clean up function. It frees up the surface and quits SDL.

```
int main( int argc, char* args[] )
{
    //Make sure the program waits for a quit
    bool quit = false;
```

Now we enter the main function.

Here we have the quit variable which keeps track of whether the user wants to quit. Since the program just started we set it to false or the program will end immediately.

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

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

Now we call the intialization and file loading functions we made earlier.

```
    //Apply the surface to the screen
    apply_surface( 0, 0, image, screen );

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

Then we show the image on the screen.

```
    //While the user hasn't quit
    while( quit == false )
    {
```

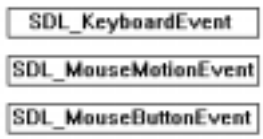
Now we start the main loop. This loop will keep going until the user sets quit to true.

```
        //While there's an event to handle
        while( SDL_PollEvent( &event ) )
        {
```

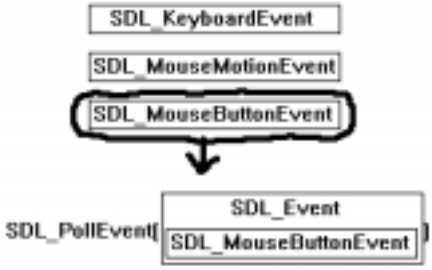
In SDL whenever an event happens, it is put on the event queue. The event queue holds the

event data for every event that happens.

So if you were to press a mouse button, move the mouse around, then press a keyboard key, the event queue would look something like this:



What SDL_PollEvent() does is take an event from the queue and sticks its data in our event structure:



What this code does is keep getting event data while there's events on the queue.

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

When the user Xs out the window, the event type will be SDL_QUIT.

But when the user does that it does not end the program, all it does inform us the user wants to exit the program.

Since we want the program to end when the user Xs the window, we set quit to true and it will break the loop we are in.

```
    //Free the surface and quit SDL
    clean_up();

    return 0;
}
```

Finally, we do the end of the program clean up.

There are other way to handle events like SDL_WaitEvent() and SDL_PeepEvents(). You can find out more about them in the SDL documentation.

Download the media and source code for this tutorial here.

Webinars on DSP Design
Free Regis. By Industry Leaders for Electronics Professionals.

Develop Smarter Game AI
Visual authoring of AI for games and simulations without programing

Ads by Google

Copyright Lazy Foo' Productions 2004-2008