SDL

Dr. Yihsiang Liow (October 11, 2016)

Contents

1 Installation

I assume you're using a fedora machine.

In a terminal window do the following (you have to enter your root password after `su`; if you did not change the root password, the password should be `root`).

```
su
yum -y install SDL SDL-devel
```

2 Test

To test your SDL installation, write this file (name it `a.cpp`):

```
#include "SDL.h"

const int W = 640; // Width of surface
const int H = 480; // Height of surface

int main(int argc, char* argv[])
{

    if (SDL_Init(SDL_INIT_VIDEO) != 0)
    {
        printf("ERROR");
    }
    else
    {
        SDL_Surface * surface = SDL_SetVideoMode(W, H, 0, SDL_ANYFORMAT);
        SDL_Event event;

        while (1)
        {
            if (SDL_PollEvent(&event) && event.type == SDL_QUIT)
                break;
        }
    }

    return 0;
}
```

To compile:

```
g++ a.cpp `sdl-config --cflags --libs`
```

WARNING: The ` is the backtick, not the single-quote. Look at the key with ~, i.e., to the left of 1, above the tab key.

If you installed the SDL library correctly, there shouldn't be any errors when compiling your program.

The above produces an executable `a.out`. To run it do the following in your terminal window

```
./a.out
```

and you will get a black window of 640 pixels by 480 pixels.

3 Additional Libraries

To install the additional libraries open a terminal and do the following (you need to enter the root password `su`):

```
su
yum -y install SDL_image-devel SDL_ttf-devel SDL_mixer-devel
```

To compile with the additional library you do the following

```
g++ *.cpp `sdl-config --cflags --libs` -lSDL_image -lSDL_mixer -lSDL_ttf
```

4 What's Next?

Now move on to my SDL tutorial code. You will receive a `demo` folder containing demo code on how to open a windows, draw pixels, lines, rectangles, circles, load and blit an image, play sound and music, etc.

The code examples in `demo` (together with your C++ programming skills in CISS240 and the first half of CISS245) is enough tools for you to write pretty much any 2D game. Of course some games are more complex than others.

In the `demo` folder there's a `makefile`. After updating/writing the `main.cpp` in `demo`, execute the following command in your shell:

```
make
```

to compile your program and then

```
make run
```

to run the program.

Like I said in class, the eye candy stuff like drawing and playing sounds is not the main difficulty in writing software such as games. That's just a matter of knowing what function or method to call.

Ultimately, it's really about problem-solving skills, the ability to understand what you need to do and then to translate that into algorithms or pseudocode. At this point in your programming career, the translation from algorithm/pseudocode to actual C++ code should also be pretty easy.