**SDL2 Project**

1. **Introduction**

Simple DirectMedia Layer (SDL):là một thư viện đa nền tảng, bao gồm các API để thao tác với âm thanh, bàn phím, chuột, joystick, graphics hardware thông qua OpenGL và Direct3D.

Thư viện SDL được viết bằng C chuẩn, nhưng hỗ trợ tốt với C++ và nhiều ngôn ngữ lập trình khác như:C#,Java, Pascal, PHP, Python,...

SDL hỗ trợ trên các hệ điều hành như: Windows, Mac OS X, Linux, iOS và Android.

(Simple DirectMedia Layer is a cross-platform development library designed to provide low level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D.

SDL officially supports Windows, macOS, Linux, iOS, and Android. Support for other platforms may be found in the source code.

SDL is written in C, works natively with C++, and there are bindings available for several other languages, including C# and Python.

)

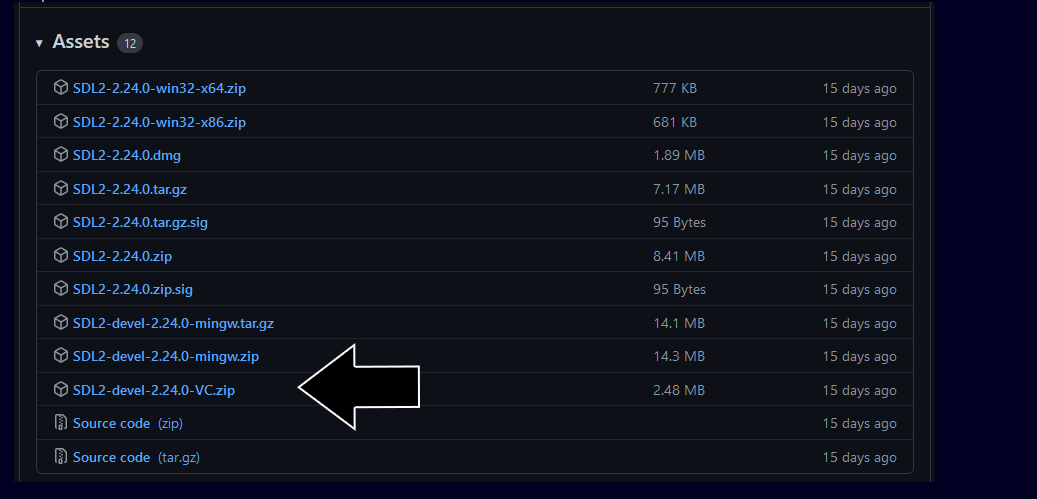
Document helpul:

<https://lazyfoo.net/tutorials/SDL/>

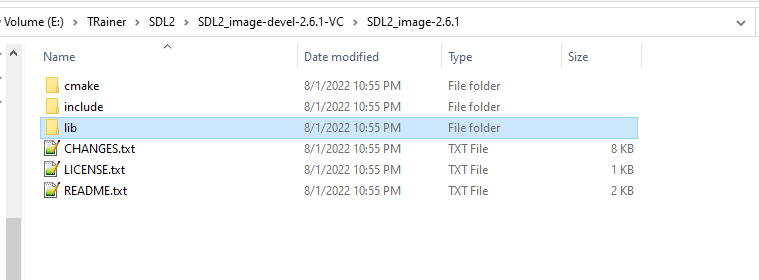
<https://wiki.libsdl.org/SDL2/Introduction>

1. **Import SDL2 engine to our project**

First, we need to download SDL2 header s and biranies. You will find them on the SDL GitHub: <https://github.com/libsdl-org/SDL/releases>

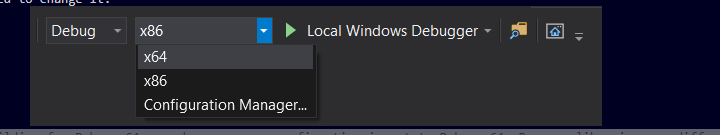


Open the zip archive and there should be a folder called SDL2-2.something.something. Copy the contents of the folder and put it anywhere you'd like. Something like that:



We just focus on **lib** and **include** folders. We will use them for your project.

**2.1 . Setup build configuration**



In folder lib we can see it contains x64 and x86 lib. We will be assuming you are building for Debug x64 so make sure your configuation is set to Debug x64. Because libraries are different per configuation, you will need to add SDL to every configuation you plan on using. So if you want to build for Release x64 or Debug x86, you will need to add SDL2 to each configuation.

* 1. **Build your solution**

You should get error like this when you # include <SDL.h> : Cannot open include file: 'SDL.h': No such file or directory.

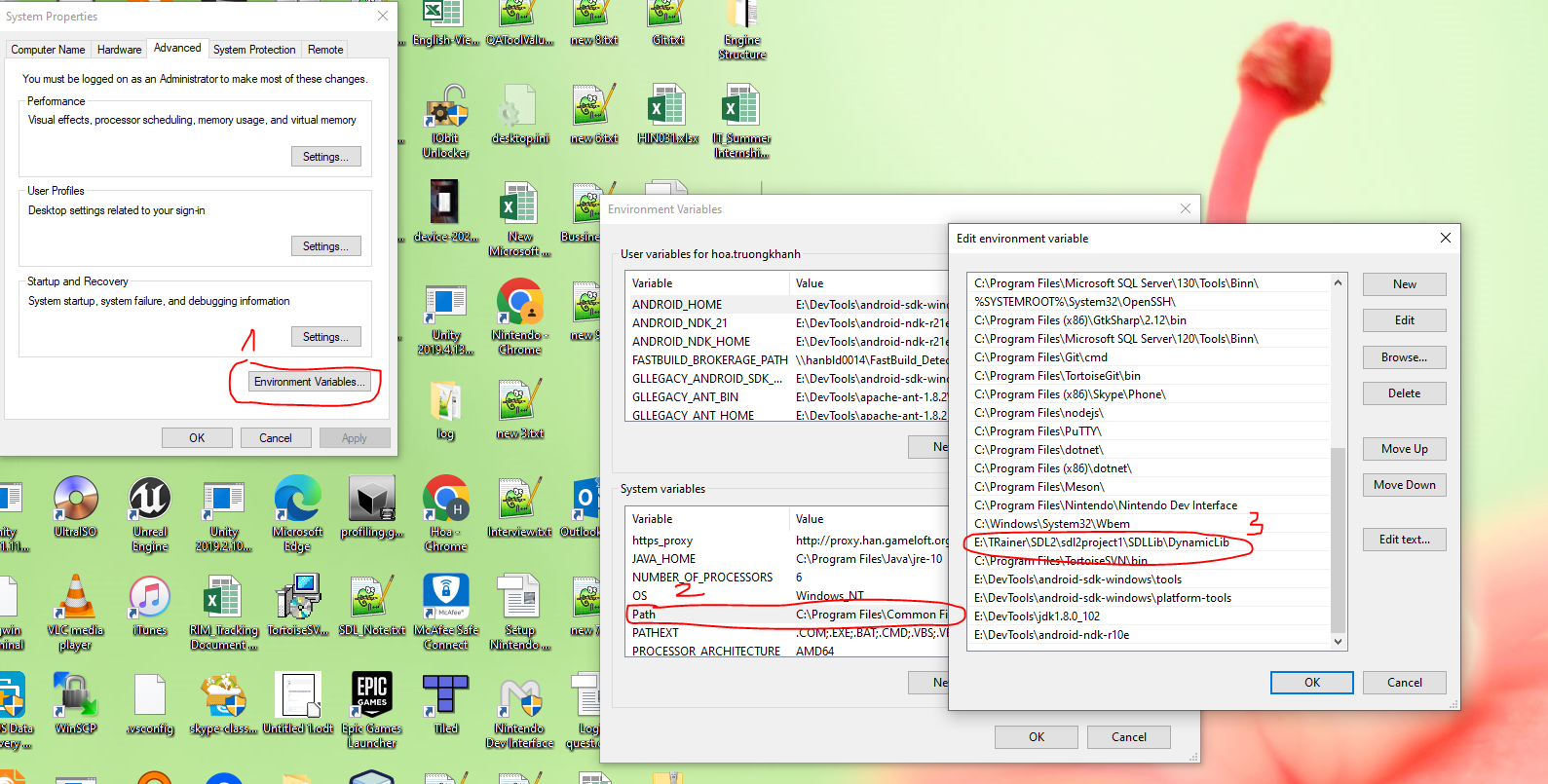
This means Visual C++ cannot find the SDL header files and you need to add the SDL include folder to the Visual C++ include directories.

Let’s setup right now:

* + 1. Configuration header file
* Add folders include header file of sdl lib to your project (Add all file (.h) into include folder to your project.)
* Go to Configuration Properties -> VC++ Directories -> Include Directories -> Edit. : Add IncludePath
  + 1. Configuration static lib ( file .lib)
* Declare path to .lib: Configuration Properties -> Linker -> General -> Additional Library Directories.
* Declare .lib file: Configuration Properties -> Linker -> Input -> Additional Dependencies -> Edit ( Add Sdl.lib here)
  + 1. Configuration dynamic lib ( file .dll)

**Option 1:**

Windows use environment variables to define where to look for dll files. We need to add PATH environment variable which include all .dll files something like that:

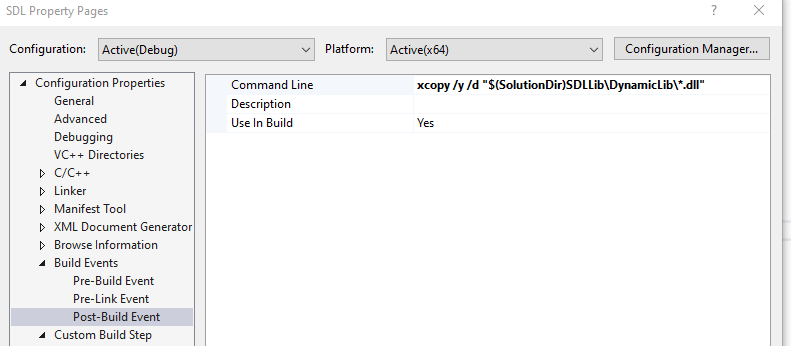


Restart Visual Studio so Visual C++ can get the updated path variable, start your program and it should run.

You just need to setup only one time, you can use it for other project if you want.

**Option 2:**

+ Add config like this:



All file .dll are in DynamicLib folder.

Your are done for setup configuration. The error shoudle be removed.

Example for this: <https://lazyfoo.net/tutorials/SDL/01_hello_SDL/windows/msvc2019/index.php>

1. **Create “SDL Tutorial” window**

//The window we'll be rendering to

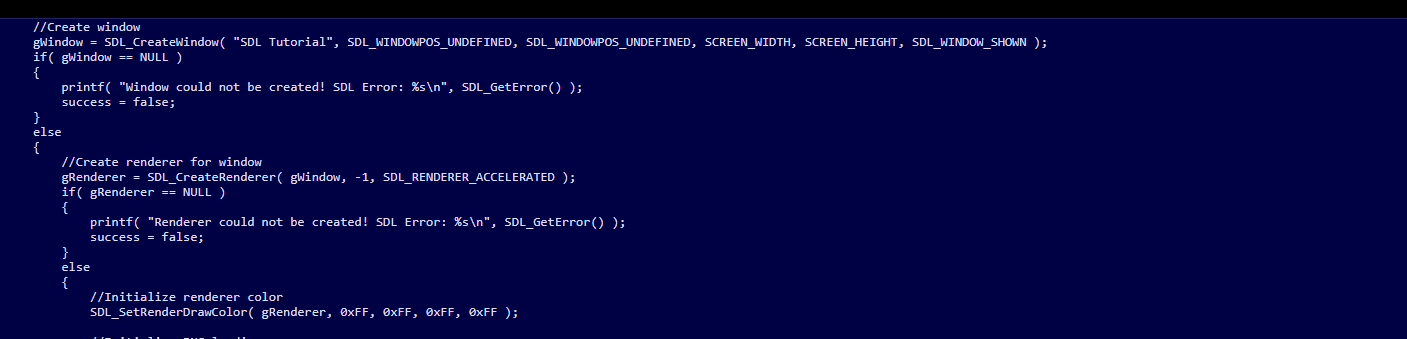
SDL\_Window\* gWindow = NULL;

//The window renderer

SDL\_Renderer\* gRenderer = NULL;

SDL\_Renderer : struct xử lý tất cả các công việc rendering lên cửa sổ.

// Create window



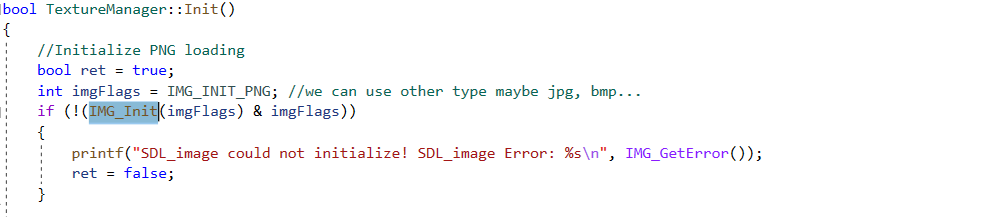
After we create our window, we have to create a renderer for our window so we can render textures on it. Fortunately this is easily done with a call to SDL\_CreateRenderer.

After creating the renderer, we want to initialize the rendering color using SDL\_SetRenderDrawColor. This controls what color is used for various rendering operations.

After that, it ready for render texture on that window which we have created.

1. **Render texture on the screen**

First, we need to init libraries and prepare them for use. In this

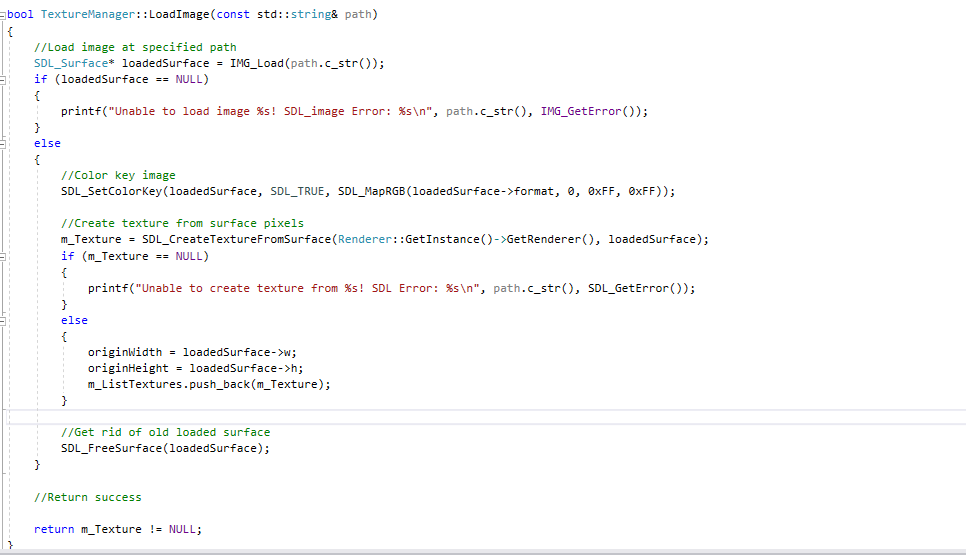


We have SDL\_Texture\* mTexture for render texture. SDL\_Texture is used in a hardware rendering, textures are stored in VRAM. The rendering operations are accelerated by GPU.

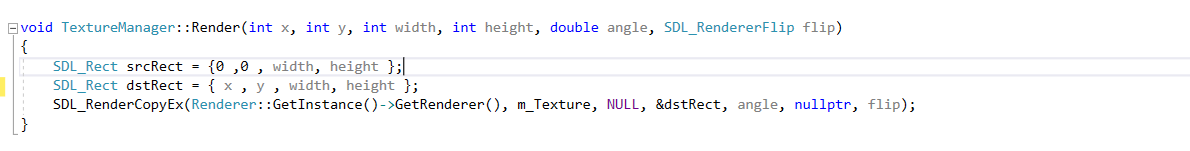
We have other way to render texture on the screen, by using SDL\_Surface. SDL\_surface is used in software rendering(CPU)

Hardware rendering is by orders of magnitude faster than software rendering. So we use SDL\_Texture for our project.

- We need to load texture : we create a texture from the loaded surface using SDL\_CreateTextureFromSurface. This function creates a new texture from an existing surface and we have to free the loaded surface and then return the loaded texture.



Second, based on position and width, height we can draw it to the screen by using SDL\_RenderCopyEx or SDL\_RenderCopy function.



**Focus to these parameters:**

* renderer: đối tượng được sử dụng để render.
* m\_Texture: đối tượng texture muốn render.
* srcRect: pointer to source rect(the area and position where you get the sprite on the texture) (1 phần nào đó của texture muốn render. Nếu để NULL nó sẽ render toàn bộ texture.)
* dstRect: pointer to dest rect(the area and position on the renderer you are going to draw). (Hình chữ nhật xác đinh tọa độ mà texture muốn vẽ trên cửa sổ. Nếu chiều rộng và chiều cao lớn hơn hoặc nhỏ hơn texture muốn vẽ thì texture sẽ bị scale theo hình chữ nhật này. Nếu để NULL texture sẽ được vẽ full màn hình)
* Flip: flipping texture by using Flip\_Horizontal or Flip\_Vertical.( default = None)
* Rotation angle: a point to rotate texture around.( default = 0)

For more information please see it in the document: <https://wiki.libsdl.org/SDL2/SDL_RenderCopyEx>

Finally, With the texture rendered, we still have to update the screen, but since we're not using SDL\_Surfaces to render we can't use SDL\_UpdateWindowSurface. Instead we have to use **SDL\_RenderPresent.**

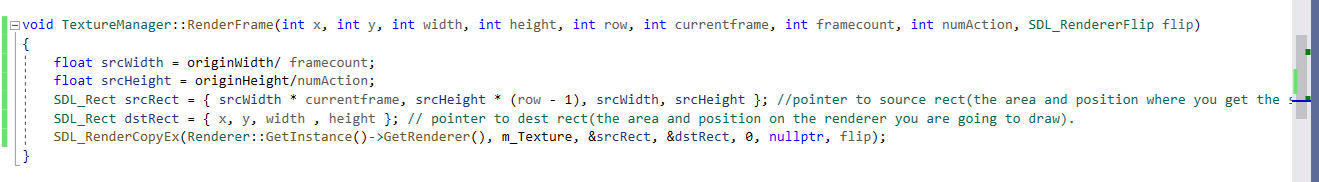


Please see more in here: https://wiki.libsdl.org/SDL2/FrontPage

Example for Render texture: <https://lazyfoo.net/tutorials/SDL/07_texture_loading_and_rendering/index.php>

1. **Load animation**

We have renderFrame function based on currentframe and numAction

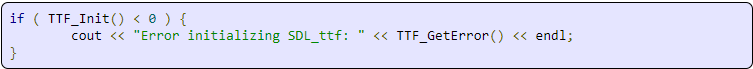


1. **Text rendering ( SDL\_TTF)**

SDL\_ttf provides an API for True Type Font (TTF) loading and rendering. It works very similarly to other extensions we've used.

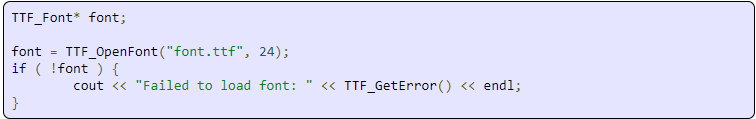
**6.1: Initialization**

Again like the other extensions, SDL\_ttf includes an initialization function, TTF\_Init(). This function does not take any parameters; simply call it in your program startup.



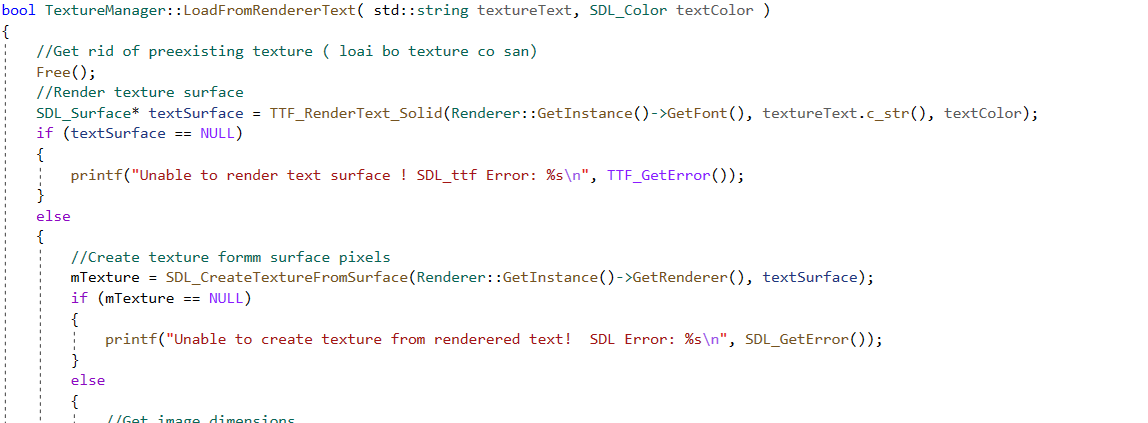
**6.2: Load Font**

We use function TTF\_OpenFont() for single font files and the function TTF\_OpenFontIndex() to load a font from a file containing multiple.



First parameter is a file name, second parameter is font size in pixels.

LoadFromRenderedText :



Here is where we actually create the text texture we're going to render from the font. This function takes in the string of text we want to render and the color we want to use to render it. After that, this function pretty much works like loading from a file does, only this time we're using a SDL\_Surface created by SDL\_ttf instead of a file.

After freeing any pre-existing textures, we load a surface using TTF\_RenderText\_Solid. This creates a solid color surface from the font, text, and color given. If the surface was created successfully, we create a texture out of it just like we did before when loading a surface from a file. After the text texture is created, we can render with it just like any other texture.

Example for font: https://lazyfoo.net/tutorials/SDL/16\_true\_type\_fonts/index.php

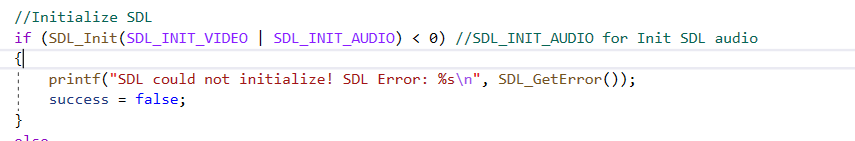
**6.3: Render Font on the scene**

Like render texture, we use function SDL\_RenderCopyEx / SDL\_RenderCopy to render text on the screen.

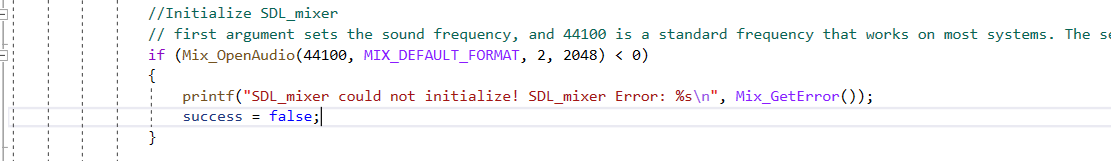
1. **Sound**

SDL\_mixer is a library we use to make audio playing. Before use it, we have to import library SDL\_mixer to our project same as other library.

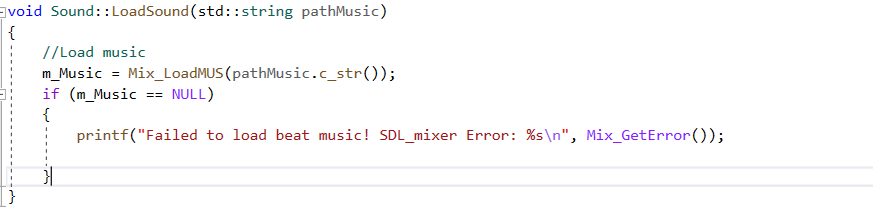
First, Initialize SDL when begin program:



After that, since we're using music and sound effects, we need to initialize audio along with video



To load music we call Mix\_LoadMUS and to load sound effect we call Mix\_LoadWAV.



To play music: use Mix\_PlayMusic

To pause: use Mix\_PauseMusic()

To Resume music: use Mix\_ResumeMusic()

To Stop music: use Mix\_HaltMusic()

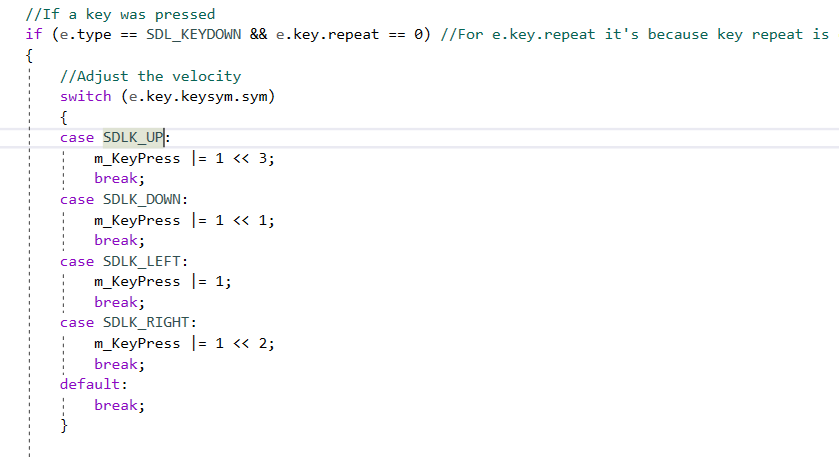
Example for sound:

https://lazyfoo.net/tutorials/SDL/21\_sound\_effects\_and\_music/index.php

1. **Key Event / Key States**

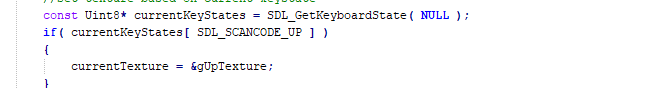
We handle an SDL\_KEYDOWN event. This event happens when ever you press a key on the keyboard.

Inside of the SDL Event is an SDL Keyboard event which contains the information for the key event. Inside of that is a SDL Keysym which contains the information about the key that was pressed. That Keysym contains the SDL Keycode which identifies the key that was pressed.



Note: e.key.repeat == 0: it's because key repeat is enabled by default and if you press and hold a key it will report multiple key presses. That means we have to check if the key press is the first one because we only care when the key was first pressed.

Other ways to get the state of the input devices( mouse, keyboard) is use key states.



Know issue:

**With handle multiple key presses at once:**

KeyState: Ví dụ trong trường hợp cần ấn nhiều phím để ra chiêu.

Vd: Combat:

CHEM CO BAN = T

CHEM THAP = T+S -> Nhả S thì sẽ chuyển về chém cơ bản tức là keystate hiện tại sẽ nhận là Keypressed là T. Nhưng trong nhiều trường hợp keystate của phím T không được nhận là pressed.

-> Lỗi này do phần cứng của phím ( keyboard ghosting) khi kết hợp các phím bấm đồng thời. Workaround bằng cách kết hợp với các phím khác.

Key States: Sdl\_GetKeyState(): SDL keeps track of the current state of all keys ( state pressed or released)

Key Event: seems only process the latest key that was pressed. Example: if holding ‘S’ to walk backwards and press ‘D’ to strafe right, then release ‘D’ to end up strafing right but not going backwards. -> Not useful for this case.

Example for Key press: https://lazyfoo.net/tutorials/SDL/04\_key\_presses/index.php

1. **Mouse event**

State mouse event happened:

SDL\_MOUSEMOTION: when the mouse moves

SDL\_MOSEBUTTONDOWN: when clicked a mouse button ( both left and right click)

SDL\_MOUSEBUTTONUP: when released a mouse click,

SDL\_BUTTON\_LEFT: left click (e->button.button == SDL\_BUTTON\_LEFT)

SDL\_BUTTON\_RIGHT: right click (e->button.button == SDL\_BUTTON\_RIGHT)

If one of these mouse events do occur, we check the mouse position using SDL\_GetMouseState

Example for mouse event: <https://lazyfoo.net/tutorials/SDL/17_mouse_events/index.php>

<https://wiki.libsdl.org/SDL2/SDL_MouseButtonEvent> that contains mouse button event information.

1. **Timer**

SDL\_GetTick(): Get time since the program started in miliseconds.

**Vài note cần phải chú ý:**

Build có chế độ build x64/x86 chú ý 2 cái này lúc include lib của SDL. Mỗi chế độ build sẽ dùng lib khác nhau.

Vị trí position của object x, y sẽ là góc trên cùng bên trái. Cơ chế vẽ dựa vào các đối tượng hình chữ nhật có vị trí (x,y), kích thước (width, height).

Xoay trái, xoay phải, trên dưới thì dùng Flip ( Horizontal, Vertical).

Rotation; Xoay đối tượng thì ta sẽ dùng góc xoay angle. Xoay đối tượng theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ.

Collision:

Draw box to check collision:

SDL\_Rect draw\_collider;

draw\_collider.x = m\_position.x

draw\_collider.y = m\_position.y

draw\_collider.w = m\_iWidth;

draw\_collider.h = m\_iHeight;

SDL\_SetRenderDrawColor(renderer, 0, 0, 0, 255); // Set color

SDL\_RenderDrawRect(renderer, &draw\_collider);