GAM220 SDL Walkthrough

1. Download C++ SDL2 development libraries from Github and extract it to a logical place
2. Create new empty C++ project
3. Change build configuration to x64
4. Open project Properties
5. Go to Configuration Properties -> VC++ Directories -> Include Directories -> Edit
6. Go to the downloaded development libraries and find the include directory. Copy this path and paste it into the first text box and hit OK
7. Go to Configuration Properties -> Linker -> Additional Dependencies -> Edit
8. Add "SDL2.lib; SDL2main.lib;" to the first text box and hit OK
9. Go to Configuration Properties -> VC++ Directories -> Library Directories -> Edit
10. From the development libraries find the lib directory. Copy the path for the x64 architecture
11. Open the settings on your computer and search for "Environment Variables"
    a. Click environment variables and under System Variables select Path and click Edit
    b. Click New, then Browse, then navigate to the same x64 lib as the previous step, then hit OK
12. In your project's main.cpp, include SDL.h

You are ready to make some graphics!

1. In the main function, we need to create an SDL_Window pointer and an SDL_Surface pointer.
2. Initialize SDL so we can start using SDL functions. Pass the SDL video flag as an argument
3. Create the window with a name for the window, the space where the window will appear, and the window dimensions.
4. Create the window's surface, fill it with a color, and update the window
5. Use hacky line to keep window open for now
6. To free the memory and deallocate, we use SDL_DestroyWindow and then SDL_Quit before ending the main function

Nice! You've made a window pop up now! Now we're ready to start some cool game programming.

1. To replace the hacky line, we will make a main loop that responds to Events. Create an SDL_Event variable in order to handle events.
2. In nested while loops, we can check if the user is trying to quit the game. Use a regular game loop such as while(playing) for the outer loop. For the inner loop, we will call the SDL_PollEvent() function. This will process through the event queue one by one until it is empty.
    a. Inside this loop, we can check if the event is the player quitting the game. if(e.type == SDL_QUIT){//end your outer game loop here}
3. To render an image onto the window, we can create another SDL_Surface pointer that we initialize using the SDL_LoadBMP() function.
    a. Make sure your image is located in the working directory!
4. To apply the image to the screen, we need to call two functions
    a. SDL_BlitSurface applies the image surface to the screen surface of the window
    b. SDL_UpdateWindowSurface to update the window with the new information
5. When closing the game, before we call SDL_DestroyWindow, you must deallocate the surface we loaded with the image. Call SDL_FreeSurface to deallocate the memory for the surface pointer.

Now we know how to make images appear - pretty cool! Here's some other important things

- In our loop that checks for events, we can check for specific key presses
    - Check if the event type is SDL_KEYDOWN
    - Check which key by getting e.key.keysym.sym
        - SDLK_UP = up arrow
        - SDLK_DOWN = down arrow
- We can check for mouse events including mouse motion, clicks, and position in the window
    - Use the SDL_GetMouseState(&x, &y) function to get the position of the mouse in the window
        - NOTE: SDL's coordinates origin is the top left corner, like web programming
    - Mouse event types
        - SDL_MOUSEMOTION
        - SDL_MOUSEBUTTONDOWN
        - SDL_MOUSEBUTTONUP
- We can load text onto the screen by setting up the SDL ttf extension
    - Similar to images, font files need to be in the working directory
        - Font data type is a TTF_Font pointer
    - Load text to an SDL_Surface using TTF_RenderText_Solid() function

        - Pass font, text to type, and color of the type

```
//Free loaded images
gTextTexture.free();

//Free global font
TTF_CloseFont( gFont );
gFont = NULL;

//Destroy window
SDL_DestroyRenderer( gRenderer );
SDL_DestroyWindow( gWindow );
gWindow = NULL;
gRenderer = NULL;

//Quit SDL subsystems
TTF_Quit();
IMG_Quit();
SDL_Quit();
```

    - Do the above when closing to free memory from text textures
- We can use SDL_Mixer to play audio in our game by setting up the libraries
    - The data type for longer sounds (music) is Mix_Music pointer
    - The data type for short sounds (effects) is Mix_Chunk pointer
    - Load music using Mix_LoadMUS() function
        - Make sure the file type is .wav and in the working directory
    - Load sounds using the Mix_LoadWAV() function
    - To play sounds, use the Mix_PlayChannel() function
        - Use -1 for first argument to get nearest available channel, then pass the sound effect, then pass the number of times you want it to repeat
    - To play music, there are a few functions to use
        - Mix_PlayMusic() plays the music that is passed, pass -1 to loop until stopped, or pass number of times to repeat
        - Mix_PauseMusic() to pause the music
        - Mix_ResumeMusic() to resume the music
        - Mix_HaltMusic() to completely stop the music

- Do the following when closing to free memory

```
//Free loaded images
gPromptTexture.free();

//Free the sound effects
Mix_FreeChunk( gScratch );
Mix_FreeChunk( gHigh );
Mix_FreeChunk( gMedium );
Mix_FreeChunk( gLow );
gScratch = NULL;
gHigh = NULL;
gMedium = NULL;
gLow = NULL;

//Free the music
Mix_FreeMusic( gMusic );
gMusic = NULL;

//Destroy window
SDL_DestroyRenderer( gRenderer );
SDL_DestroyWindow( gWindow );
gWindow = NULL;
gRenderer = NULL;

//Quit SDL subsystems
Mix_Quit();
IMG_Quit();
SDL_Quit();
```

-