Task 17 - Spike Summary Report

Spike: Task_17
Title: Sprites and Graphics
Author: Thomas Horsley, 103071494

Goals & Deliverables

Aim: Develop an understanding surrounding SDL2's graphics and rendering functionality with respect to 2D surfaces and textures.

Deliverables:

- Spike Summary Report
- Functioning code
- Git Commit Snapshot

Technology, Tools and Resources

Tech and Tools

The project was scripted in C++ 17 using Visual Studio Community 2022.

UML's and charts are made with *www.Lucidchart.com*

Source control is handled using Git.

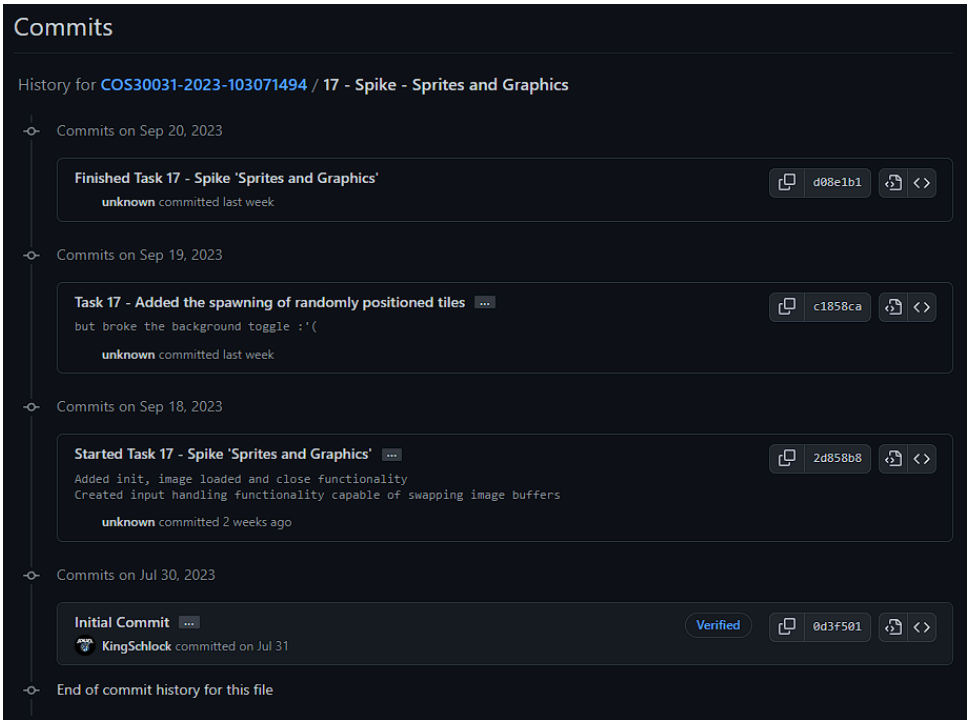
Resources

- SDL2 Handling Textures: https://www.youtube.com/watch?v=rB8N5cFCHLQ&t=537s
- SDL2 Load PNG and more Formats: https://www.youtube.com/watch?v=ERGY54efC5k&t=5s
- Lazy Foo: Texture Loading and Rendering: https://lazyfoo.net/tutorials/SDL/07_texture_loading_and_r

Tasks Undertaken

Implementation

Git Commit History



Code

```
//  I LOVE THIS CODE IT LOOKS VERY GOOD!
/*  Built the render / update / input handler function as there's really not a whole
*   lot going on here       */
bool Rendate() {
    if (SDL_PollEvent(&event) != 0) {
        if (event.type == SDL_KEYUP) {
            if (event.key.keysym.sym == SDLK_0){
                if (!display_bg) {
                    // get rid of old drawings which would be lost anyways
                    SDL_RenderClear(renderer);
                    SDL_RenderCopy(renderer, bg_texture, bg_mask, bg_rect);
                    display_bg = !display_bg;
                }
                else { SDL_RenderClear(renderer);
                       display_bg = !display_bg; }

                SDL_UpdateWindowSurface(window);
            }

            if (event.key.keysym.sym == SDLK_1) {
                SDL_RenderCopy(renderer, tilemap_texture, tile_0_mask, tile_rect);
                randomizeRectLoc(128);
            }
            if (event.key.keysym.sym == SDLK_2) {
                SDL_RenderCopy(renderer, tilemap_texture, tile_1_mask, tile_rect);
                randomizeRectLoc(128);
            }
            if (event.key.keysym.sym == SDLK_3) {
                SDL_RenderCopy(renderer, tilemap_texture, tile_2_mask, tile_rect);
                randomizeRectLoc(128);
            }
            if (event.key.keysym.sym == SDLK_c) { SDL_RenderClear(renderer); }
            if (event.key.keysym.sym == SDLK_q) { return false; }
        }
    }

    SDL_RenderPresent(renderer);
    return true;
}
```

The dubiously dubbed “Rendate” function houses the input handling and flow control of both the update and render methods. It passes a reference of the chosen texture to the renderer to be displayed.

When the background is cleared, so are the existing tiles which have been previously instantiated.

```
    int screen_width = 1000, screen_height = 1000;
    SDL_Window* window = nullptr;               // Freed in close()
    SDL_Renderer* renderer = nullptr;           // Freed in close()
    SDL_Event event;                            // Stack allocated

    bool display_bg = true;
    SDL_Surface* screen_surface = nullptr;      // Freed when window gets destroyed
    SDL_Surface* bg_surface = nullptr;          // Freed in loadBG()
    SDL_Texture* bg_texture = nullptr;          // Freed in close()
    SDL_Rect* bg_mask = nullptr;                // Freed in close()
    SDL_Rect* bg_rect = nullptr;

    SDL_Surface* tilemap_surface = nullptr;     // Freed in loadTileMap()
    SDL_Texture* tilemap_texture = nullptr;     // Freed in close()
    SDL_Rect* tile_rect = new SDL_Rect();       //  /
    SDL_Rect* tile_0_mask = new SDL_Rect();     // |   Freed in
    SDL_Rect* tile_1_mask = new SDL_Rect();     // |   close()
    SDL_Rect* tile_2_mask = new SDL_Rect();     //  \

bool init() {
    bool is_success = true;

    if (SDL_Init(SDL_INIT_EVERYTHING) < 0) { is_success = false; }
    else {  window = SDL_CreateWindow("Lab 17 - Spike 'Sprites and Graphics'",
            SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
            screen_width, screen_height, SDL_WINDOW_SHOWN);
        renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);

        if (window == nullptr || renderer == nullptr) { is_success = false; }
        else { screen_surface = SDL_GetWindowSurface(window); }
    }

    return is_success;
}
```

The SDL types used to load, splice and instantiate a texture.

```
/* Math is done for one 256x256 set of 4 128x128 tiles ignoring the bot - right tile.
   It is also assumed the tiles are square. */
bool loadTileMap(const char* tm_bmp_filepath) {
    bool is_success = true;
    int tile_perimeter_px = 128;

    tilemap_surface = SDL_LoadBMP(tm_bmp_filepath);
    if (tilemap_surface == nullptr) { return !is_success; }
    tilemap_texture = SDL_CreateTextureFromSurface(renderer, tilemap_surface);

    // Don't need the surface anymore as the data is copied in the texture
    SDL_FreeSurface(tilemap_surface);
    tilemap_surface = nullptr;

    // Chop the texture into rectangles which can be used to create stuff
    tile_0_mask->x = 0;
    tile_0_mask->y = 0;
    tile_0_mask->w = tile_perimeter_px;
    tile_0_mask->h = tile_perimeter_px;

    tile_1_mask->x = 128;
    tile_1_mask->y = 0;
    tile_1_mask->w = tile_perimeter_px;
    tile_1_mask->h = tile_perimeter_px;

    tile_2_mask->x = 0;
    tile_2_mask->y = 128;
    tile_2_mask->w = tile_perimeter_px;
    tile_2_mask->h = tile_perimeter_px;

    return is_success;
}
```

As I wrote this program Functionally and not Objectively, most of the methods return a bool type.

The loadTileMap( ) method is a simple hardcoded function which cycles clockwise through the provided 256x256 texture map and applies a spliced 64x64 tile to a mask.

What was Learned?

After completion of this spike, I have an understanding of the process surrounding texture loading and presentation. Additionally, I'm now familiar with some of the data structures used to contain relevant texturing data (such as surfaces and rects for subsurface masks).

The SDL renderer was another topic heavily researched throughout the task. With the render process of creating a frame buffer before blitting that data to the screen being something initially completely foreign to me.