```cpp
#include <iostream>
#include <random>
#include <SDL.h>

/*  Task:   17 - Spike
*   Title:  Sprites and Graphics
*   Author: Thomas Horsley
*   Date:   20/09/23
*
*   I wrote this functionally as when we're concerned with demonstration I find
*   the functional stuff easier to extrapolate into other projects.
*   Additionally I love overengineering tf out of classes.

*
*   Full truth though, this code is pretty grott...
      */

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);

```

```
44          if (window == nullptr || renderer == nullptr) { is_success = false; }
45          else { screen_surface = SDL_GetWindowSurface(window); }
46      }
47
48      return is_success;
49  }
50
51  bool loadBG(const char* bg_bmp_filepath) {
52      bool is_success = true;
53      bg_surface = SDL_LoadBMP(bg_bmp_filepath);
54
55      if (bg_surface == nullptr) { return !is_success; }
56      bg_texture = SDL_CreateTextureFromSurface(renderer, bg_surface);
57
58      bg_rect = new SDL_Rect();
59      bg_rect->x = 0;
60      bg_rect->y = 0;
61      bg_rect->w = screen_width;
62      bg_rect->h = screen_height;
63
64      // We like our memory here
65      SDL_FreeSurface(bg_surface);
66      bg_surface = nullptr;
67
68
69      return is_success;
70  }
71
72  /* Math is done for one 256x256 set of 4 128x128 tiles ignoring the bot - right tile.
73     It is also assumed the tiles are square. */
74  bool loadTileMap(const char* tm_bmp_filepath) {
75      bool is_success = true;
76      int tile_perimeter_px = 128;
77
78      tilemap_surface = SDL_LoadBMP(tm_bmp_filepath);
79      if (tilemap_surface == nullptr) { return !is_success; }
80      tilemap_texture = SDL_CreateTextureFromSurface(renderer, tilemap_surface);
81
82      // Don't need the surface anymore as the data is copied in the texture
83      SDL_FreeSurface(tilemap_surface);
84      tilemap_surface = nullptr;
85
86      // Chop the texture into rectangles which can be used to create stuff
87      tile_0_mask->x = 0;
88      tile_0_mask->y = 0;
89      tile_0_mask->w = tile_perimeter_px;
```

```
 90      tile_0_mask->h = tile_perimeter_px;
 91
 92      tile_1_mask->x = 128;
 93      tile_1_mask->y = 0;
 94      tile_1_mask->w = tile_perimeter_px;
 95      tile_1_mask->h = tile_perimeter_px;
 96
 97      tile_2_mask->x = 0;
 98      tile_2_mask->y = 128;
 99      tile_2_mask->w = tile_perimeter_px;
100      tile_2_mask->h = tile_perimeter_px;
101
102      return is_success;
103  }
104
105  void randomizeRectLoc(int tile_size_px) {
106      int rect_perim_px = tile_size_px;
107
108      tile_rect->x = std::rand() % screen_width;
109      tile_rect->y = std::rand() & screen_height;
110      tile_rect->w = rect_perim_px;
111      tile_rect->h = rect_perim_px;
112  }
113
114  void close() {
115      // Destroy heap objects using SDL methods
116      SDL_DestroyTexture(bg_texture);
117      SDL_DestroyTexture(tilemap_texture);
118      SDL_DestroyWindow(window);
119
120      // No sdl destroy rect method?
121      if (bg_mask != nullptr) { delete bg_mask; }
122      if (bg_rect != nullptr) { delete bg_rect; }
123      if (tile_rect != nullptr) { delete tile_rect; }
124      if (tile_0_mask != nullptr){ delete tile_0_mask; }
125      if (tile_1_mask != nullptr){ delete tile_1_mask; }
126      if (tile_2_mask != nullptr){ delete tile_2_mask;}
127
128      // Set our global garbage to null
129      if (bg_surface != nullptr) { bg_surface = nullptr; }
130      if (tilemap_texture != nullptr) { tilemap_texture = nullptr; };
131      if (window != nullptr) { window = nullptr; }
132
133      SDL_Quit();
134  }
135
136  //  I LOVE THIS CODE IT LOOKS VERY GOOD!
137  /*  Built the render / update / input handler function as there's really ↵
       not a whole
```

```
138 *   lot going on here       */
139 bool Rendate() {
140     if (SDL_PollEvent(&event) != 0) {
141         if (event.type == SDL_KEYUP) {
142             if (event.key.keysym.sym == SDLK_0){
143                 if (!display_bg) {
144                     // get rid of old drawings which would be lost anyways
145                     SDL_RenderClear(renderer);
146                     SDL_RenderCopy(renderer, bg_texture, bg_mask,
                       bg_rect);
147                     display_bg = !display_bg;
148                 }
149                 else { SDL_RenderClear(renderer);
150                        display_bg = !display_bg; }
151 
152                 SDL_UpdateWindowSurface(window);
153             }
154 
155             if (event.key.keysym.sym == SDLK_1) {
156                 SDL_RenderCopy(renderer, tilemap_texture, tile_0_mask,
                    tile_rect);
157                 randomizeRectLoc(128);
158             }
159             if (event.key.keysym.sym == SDLK_2) {
160                 SDL_RenderCopy(renderer, tilemap_texture, tile_1_mask,
                    tile_rect);
161                 randomizeRectLoc(128);
162             }
163             if (event.key.keysym.sym == SDLK_3) {
164                 SDL_RenderCopy(renderer, tilemap_texture, tile_2_mask,
                    tile_rect);
165                 randomizeRectLoc(128);
166             }
167             if (event.key.keysym.sym == SDLK_c) { SDL_RenderClear
                  (renderer); }
168             if (event.key.keysym.sym == SDLK_q) { return false; }
169         }
170     }
171 
172     SDL_RenderPresent(renderer);
173     return true;
174 }
175 
176 
177 /*  1. Load an image that contains 3 seperate tiles
178 *   2. Define a rectangle surrounding each of the 3 tiles
179 *   3. On keypress, display the data contained within that rectangle to a
      random
180 *      location using 1, 2 & 3.     */
```

```
181 int main(int argc, char* argv[]) {
182     // Init the first frame and it's data
183     if (init()) {
184         if (loadBG("img/helloworld.bmp") && loadTileMap("img/
              tilemap.bmp")) {
185             bool is_running = true;
186             while (is_running) { is_running = Rendate();    }
187             close();
188         }
189     }
190     return 0;
191 }
```