Spike: Task 17.P
Title: Sprites & Graphics

Author: Mitchell Wright, 100595153

Goals / deliverables:

To demonstrate an understanding of frameworks and spite/image functionality within SDL2.

Items created during task:

- Code, see: \17 - Spike – Sprites and Graphics\SDL2

Technologies, Tools, and Resources used:

- Visual Studio 2022
- SourceTree
- GitHub
- SDL2 Development Library
- Lecture 3.2 – Data Structures

Tasks undertaken:

- Toggling Backgrounds
- Using Rectangles to Create Sprite Tiles
- Using Rectangles to Randomise Position
- Commit to Git

What we found out:

1. Toggling Backgrounds:

   In order to show something within the game window, we need to first create a surface and then use that to create a texture.

```
//Surface and Texture for background image.
SDL_Surface* imageSurface = SDL_LoadBMP("wamce.bmp");
SDL_Texture* bgTexture = SDL_CreateTextureFromSurface(renderer, imageSurface);
bool bgVisible = true;
```

   In this case, a bool is also created so that we can toggle the background.

```
case SDLK_0:
    bgVisible = !bgVisible;
    break;
```

   The renderer set up in T15 is cleared at the start of the render loop.

```
SDL_RenderClear(renderer);
```

   The texture is then copied to the renderer.

```
if (bgVisible)
{
    SDL_RenderCopy(renderer, bgTexture, nullptr, nullptr);
}
```

   And then rendered.

```
SDL_RenderPresent(renderer);
```

2. Using Rectangles to Create Sprite Tiles:

   Rectangles can be used to select portions of textures to render.

```
//Display rects
SDL_Rect tileRect = { 0, 0, 100, 100 };
```

   These are used as parameters when copying the texture to the renderer.

```
SDL_RenderCopy(renderer, tileTexture, &tileRect, &screenRect);
```

   And thus only a portion of the original BMP is shown.

3. Using Rectangles to Randomise Position:

In order to randomise the positioning of the rectangle on screen, the following function is used. Nothing too dissimilar to what has been done with randomisation in previous tasks.

```
SDL_Rect randomisePos()
{
    int x, y;
    x = rand() % 701;
    y = rand() % 501;

    SDL_Rect result = { x, y, 100, 100 };

    return result;
}
```

This function is called on keypress, same as in the previous task. Once the position has been randomised, the rectangle is passed through to the renderer.

```
SDL_RenderCopy(renderer, tileTexture, &tileRect, &screenRect);
```

4. Commit to Git: