Notes on Porting ConsoleGameEngine (CGE) programs to PixelGameEngine (PGE)

Notes taken from video: https://www.youtube.com/watch?v=kRH6oJLFYxY

PGE is also a single header file solution, very similar experience for the user as CGE

Major changes PGE compared to CGE:

- All code in pixelGameEngine exists in namespace **`olc`**
- Instead of characters, the fundamental unit you operate on is the pixel (32 bit colour)
- PGE supports transparency
- Sprites need **`.png`** files (instead of **`.spr`** files)
- In the **`Construct()`** function you provide pixel width and height instead of font width and height [so there's a distinction between physical screen pixels and logical program / engine pixels]

How to port CGE based program to PGE:

[this description assumes your program to consist of one **`.cpp`** *file]*

1. At the top of your program file (**`.cpp`** file):
    a. Change the **`#include`** directive from **`olcConsoleGameEngine.h`** to **`olcPixelGameEngine.h`**
    b. Insert a line before this include directive: **`#define OLC_PGE_APPLICATION`**
    c. Make changes to your project to remove the old CGE include file, and add the new PGE include file to your project. Of course your IDE must be able to find the **`olcPixelGameEngine.h`** file, so put it either in your include directory (if you have one) or in the working directory where your program **`.cpp`** file resides
2. At the start of the engine class:
    a. Instead of publicly inheriting from **`olcConsoleGameEngine`**, you now inherit from **`olc::PixelGameEngine`**, so change this in the code.
    b. In the constructor, change **`m_sAppName`** to **`sAppName`**;
3. Throughout the code within your **`olc::PixelgameEngine`** derived class:
    a. In the CGE you passed both a glyph and a color into methods like **`Draw()`** and all variations **`DrawLine(), DrawCircle()`** etc. The glyph is gone, since the PGE works with pixels. The color can be passed using RGB values (for instance: **`olc::Pixel( r_value, g_value, b_value)`**. Each RGB value is in the range [0, 255]. Alternatively you can use one of the color constants that the engine defines. You can look these up in the PGE header file.
    b. Some function names have changed a little bit. For instance **`Fill()`** is changed into **`FillRect()`**. Check the PGE header files for the correct names if you get errors on function names.
    c. Note that in the CGE you passed two coordinates to **`Fill()`**: the upper left point and the lower right point of the rectangle to fill. In the PGE you pass the upper left point and the width and height of the rectangle to fill into **`FillRect()`**.
    d. The CGE had a key state array **`m_keys`**, that you could index directly, and ask for its state. This array is replaced by a PGE method **`GetKey()`**. You pass the key you want to query the state, for instance **`olc::Key::W`** (in stead of CGE **`L'W'`**).
4. In the main() function:
    a. **`ConstructConsole()`** [CGE] has become **`Construct()`** in the PGE – since there's no console anymore. It returns a bool, so you can check if the engine was constructed successfully before calling **`Start()`**;
5. In general:
    a. use your compiler / IDE to check where the (syntax) errors are.
    b. Unicode is no longer needed, so you can remove (find replace) the prefix **`L`** for characters and strings, and replace **`wstring()`** by **`string()`** etc.

	CGE – in your CGE program you had…	**PGE – in your PGE code this becomes…**
1a, 1b	`#include "olcConsoleGameEngine.h"`	`#define OLC_PGE_APPLICATION` `#include "olcPixelGameEngine.h"`
2a	`class YourClassName : public olcConsoleGameEngine {`	`class YourClassName : public olc::PixelGameEngine {`

2b, 5b	`YourClassName () {`
	`  m_sAppName = L"YourClassName - Title ";`
	`}`
		`YourClassName () {`
		`  sAppName = "YourClassName - Title ";`
		`}`

3a	`Draw( x, y, PIXEL_SOLID, FG_BLUE );`	`Draw( x, y, olc::BLUE );`
3b, 3c	`Fill( x1, y1, x2, y2, PIXEL_SOLID, FG_DARK_RED );`	`FillRect( x1, y1, x2 - x1, y2 - y1, olc::DARK_RED );`
3d	`m_keys[L'W'].bHeld`	`GetKey( olc::Key::W ).bHeld`

4a	`game.ConstructConsole( … );`
	`game.Start();`
		`if (game.Construct( … )) {`
		`  game.Start();`
		`}`

5b	`sMap[ny * nMapWidth + nx] == L'#'`	`sMap[ny * nMapWidth + nx] == '#'`