

PART 2: “The Template”

Introduction

As you noticed in the first article, setting up a project in Visual Studio can be quite a task. And we didn't (nearly) touch all the settings that you can adjust for a project either. To make your life a bit easier, we will use a project template from now on. This template is simply a directory that contains all the files that you need, with all the settings tuned *just right* for the kind of programs that we will be building in these series. The template also contains a bit of code that you need for most projects, so that you don't have to type it yourself. This code aims to take away the 'platform specific' things from you; i.e., it opens a window, lets you draw to it, and updates it for you. Sounds simple, but really it isn't: Windows can be quite a nightmare to deal with properly, and since that's just not the core of game development, we felt it's best to take care of that once and for all. The result is the template.

Getting the stuff you need

Get the latest template package from: <https://jbikker.github.io/fasttrack.html>. Extract the package to c:\my_projects\template. Then, open the project by double-clicking the .sln file in the directory.

You should see the project in the 'Solution Explorer now:

Hit the '▶' next to 'Template' to see the files in the project: game.cpp and .h and precomp.h. There are some more files in 'template code': surface.cpp and .h, template.cpp and .h, and threads.cpp and .h.

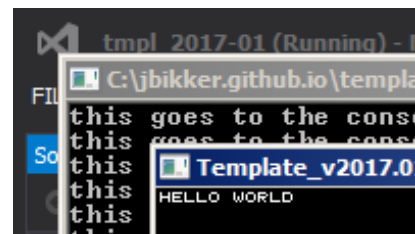
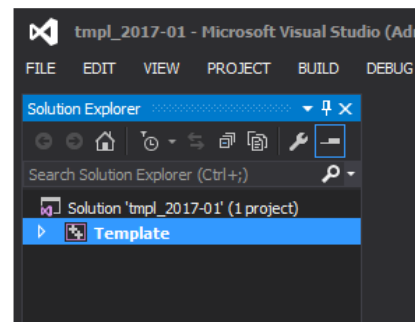
Finally, hit F5 to start the application. This should get you a nice window (as promised), with the famous words 'hello world' in it. To emphasize the magnitude of this accomplishment, there is a spinning gun turret in the center of the screen.

The code that you should be interested in right now is in game.cpp. Here are its contents:

```
#include "precomp.h" // include (only) this in every .cpp file

// -----
// Initialize the application
// -----
void Game::Init()
{
}

// -----
// Close down application
```



```

// -----
void Game::Shutdown()
{
}

static Sprite rotatingGun( new Surface( "assets/aagun.tga" ), 36 );
static int frame = 0;

// -----
// Main application tick function
// -----
void Game::Tick( float deltaTime )
{
    // clear the graphics window
    screen->Clear( 0 );
    // print something in the graphics window
    screen->Print( "hello world", 2, 2, 0xffffffff );
    // print something to the text window
    printf( "this goes to the console window.\n" );
    // draw a sprite
    rotatingGun.SetFrame( frame );
    rotatingGun.Draw( screen, 100, 100 );
    if ( ++frame == 36 ) frame = 0;
}

```

This code will be the starting point for all episodes of the Fasttrack tutorial.

Nuts and bolts

Even though this is just a small program, there's a lot to say about it. Most (all, actually) of it will be explained in this tutorial series, but for now, let's limit ourselves to some observations:

- ✓ The `#include` line at the beginning of the program basically pastes some source code from another file, `precomp.h`, in the current file. In `precomp.h`, you will find more `#include` lines. So, the 'real' `game.cpp` is quite a bit larger.
- ✓ Whenever something needs to be drawn, a line starts with `screen->`. The things we can make 'screen' do are specified in 'surface.h', starting at line 37 ('class Surface').
- ✓ There are some things that happen just once. We specify those in the `void Game::Init()` function.
- ✓ This particular program basically does nothing in the `Init` function.
- ✓ Other things are done for each frame: These are listed in `void Game::Tick(float deltaTime)`.
- ✓ There's also a `Shutdown` function, which gets executed only once: when you terminate the program (e.g., by pressing ESC).
- ✓ Lines that start with `'//'` are merely comments. The program still works if you remove or alter them.

There's quite a bit more, and even more when you explore the other source files, such as `template.cpp`. Feel free to do so; if you mess anything up, just download the clean template again.

Experiments

Let's try some random things with this program:

1. Change the '0' on line 26 to some other number. Make it a big number.
2. Change the '0xffff' on line 28 to 0xff0000. Then, change it to 0x00ff00, and finally to 0x0000ff. What happens?
3. Change the '2,2' on the same line to '100, 20'. Also change it to '640, 20'.
4. Change 'Clear' into 'clear' (lower case). Is there a problem?
5. Append line 33 to line 32 (don't remove the semi-colon at the end of line 32). Is there a problem?
6. Move the first six lines in the `Tick` function to the `Init` function (they do the same thing for each frame anyway, right?).

An important concept is the screen coordinate system. Every pixel on the screen has a position: the horizontal position is called 'X', the vertical position is called 'Y'. By default, the template opens a 800 by 512 pixel window, so if you draw something at X = 820, it doesn't fit, and wraps to the left side of the screen (one pixel lower though!).

And finally, there are numbers: 0xffff (or capitals: 0xFFFF) is a number too. In fact, 0x0000FF equals 255. We will discuss this in more detail at a later time.

Text Window

You probably noticed that you still have a text window. This seems quite useless now that we can produce proper graphics, but it may come in handy anyway: we will use it for printing information that is not supposed to be seen by the end user. This is useful for debugging.

Assignment

Take a single sheet of A4, and draw an 8x8 grid on it. Label the vertical lines: 0, 80, 160, 240, 320, 400, 480, 560, 640. Label the horizontal lines: 0, 60, 120, 180, 240, 300, 360, 420, 480. Now draw 'CODE' on this sheet, using straight lines only. Use as few lines as you can; you'll be translating each line to a line in your program, so don't make it too difficult for yourself. ☺

Now, for each line:

- Determine the X and Y coordinate of the start of the line (X1, Y1).
- Determine the X and Y coordinate of the end of the line (X2, Y2).
- Add a line to the `Tick` function: `screen->Line(X1, Y1, X2, Y2, 0xffff);`

And finally:

- Make the 'C' red;
- the 'O' green;
- the 'D' blue, and
- the 'E' yellow.

Once you have completed this assignment, you may continue with the next part.

END OF PART 2

Next part: "Variables"