Implementing the text engine and why it didn’t work

I needed some way to implement a text engine inside the game Galaxian. The supplied framework supposedly didn’t support one. This didn’t seem as such a daunting task at first. I have inserted some text in my games before. I’m used to using SFML. If you want to write a text inside of the game, just make a text object.

## Let’s research

So I started to research and soon arrived on this [article](https://learnopengl.com/In-Practice/Text-Rendering). Apparently FreeType was the way to go because it is modern. So I read the tutorial and set up the dependencies. After I have read the tutorial, I remembered why I loved using SFML so much. The tutorial was full of practices that I haven’t ever really used. I thought that it was an easy task to implement a text engine inside my game, but the task was quite hard. Very hard actually.

I spend quite a few hours just getting my head around how the engine works. I gotta say, while I still don’t understand the engine, it is something that I am planning to use one day, but with the deadline nearing it postponed it as a less important priority.

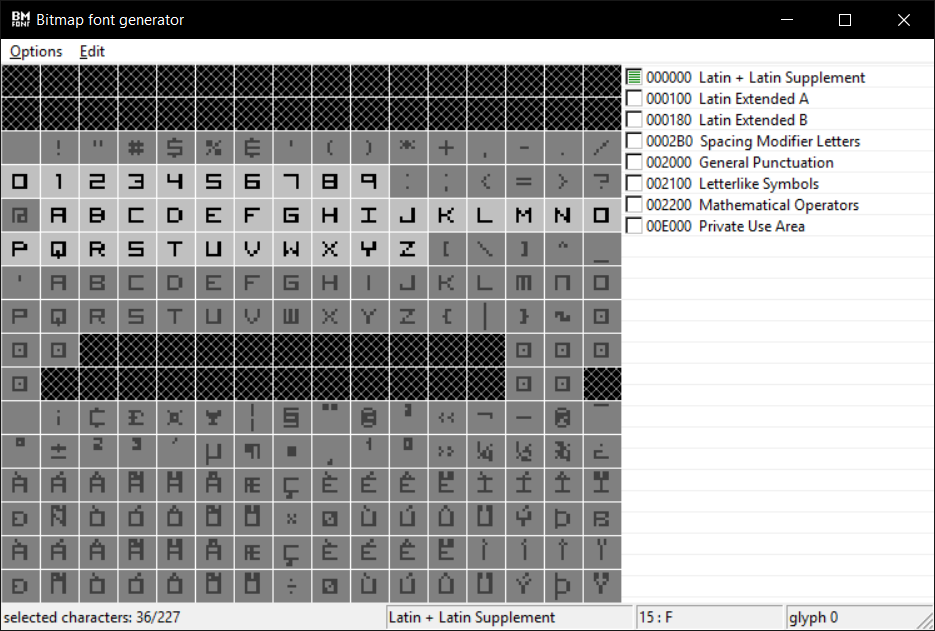
## But why didn’t I understood a tutorial?

I think it boils down to a type of stack problem, what do I mean by this. It’s another way of saying that I miss the foundation. The engine goes quite low level with fonts. Apparently it maps a whole lot of things. I don’t really know anything about fonts other than that they contain the shapes of the symbols by which we communicate with. I even learned that .ttf symbols are not really symbols but equations.

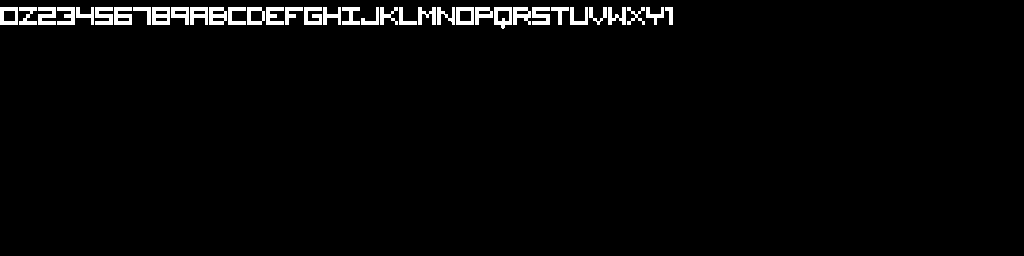
This was not the only problem. The other problem is that I am really dependent of the supplied framework. I don’t know how to directly use the libraries to modify the window. All my drawing operations are dependent upon the framework, so how will I draw the text? I don’t have a image file which I can make a sprite for. So I was quite at a loss.

## How do we fix this problem

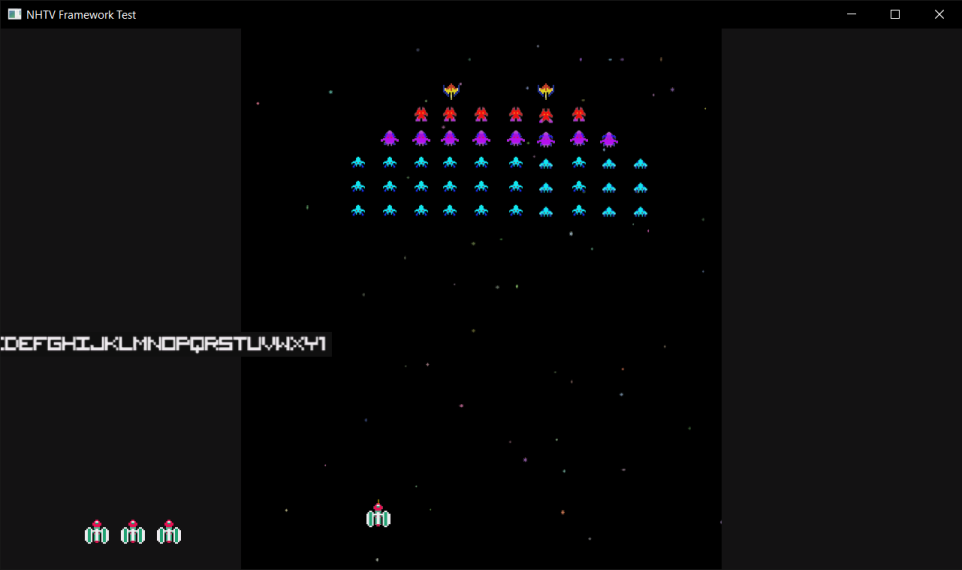
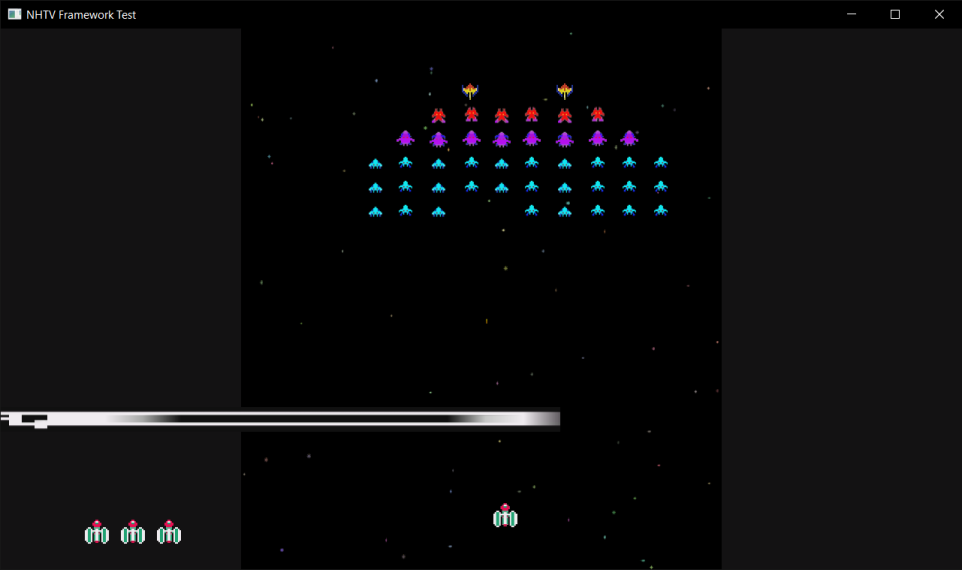
Problems arise much more often than you think and for those problems there are much more solutions that you can make up. So I started thinking. If drawing by sprite is the only way I know, I just have to make the whole font a sprite (or I should learn to use opengl directly, I will, but I really wanted to finish the problem and I don’t think I can learn a rendering engine during a toilet break).

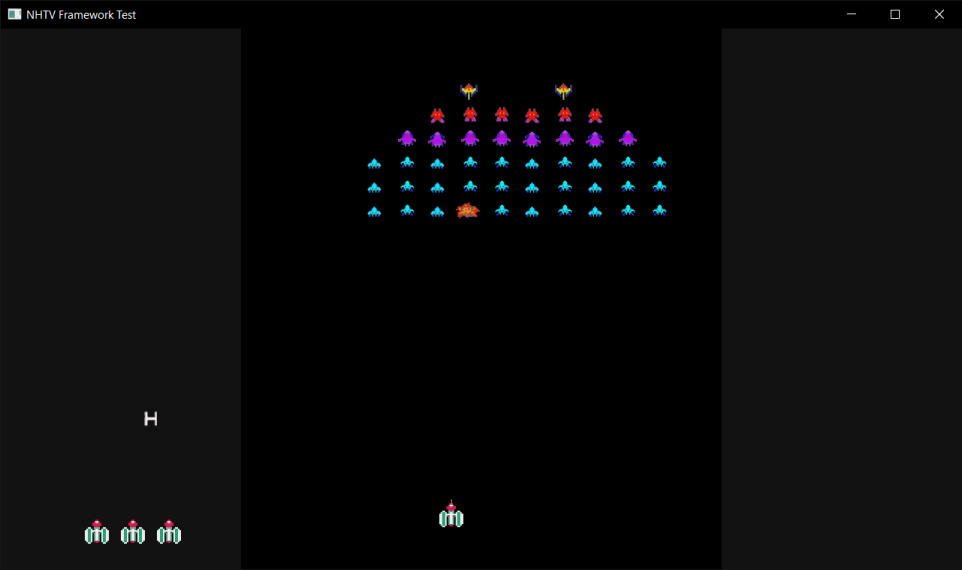
So let’s start getting our hands dirty. First of all, we needed a font, or an image file to be specific. I couldn’t really find anything handy, so I started reading online on how to convert a ttf to a png. I stumbled upon a tool called [BMFont](http://www.angelcode.com/products/bmfont/). 

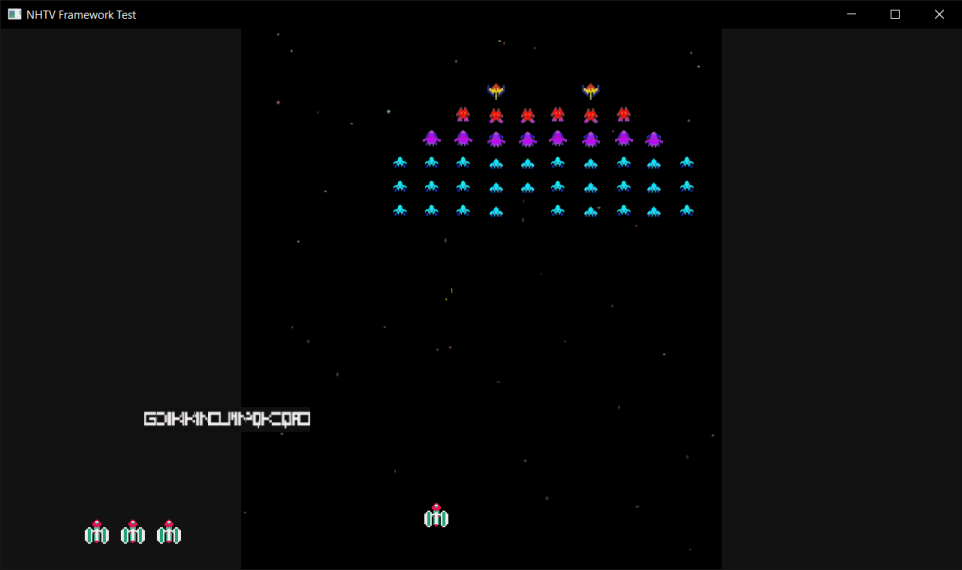
In this I exported the ttf file to an png. This got converted into this image

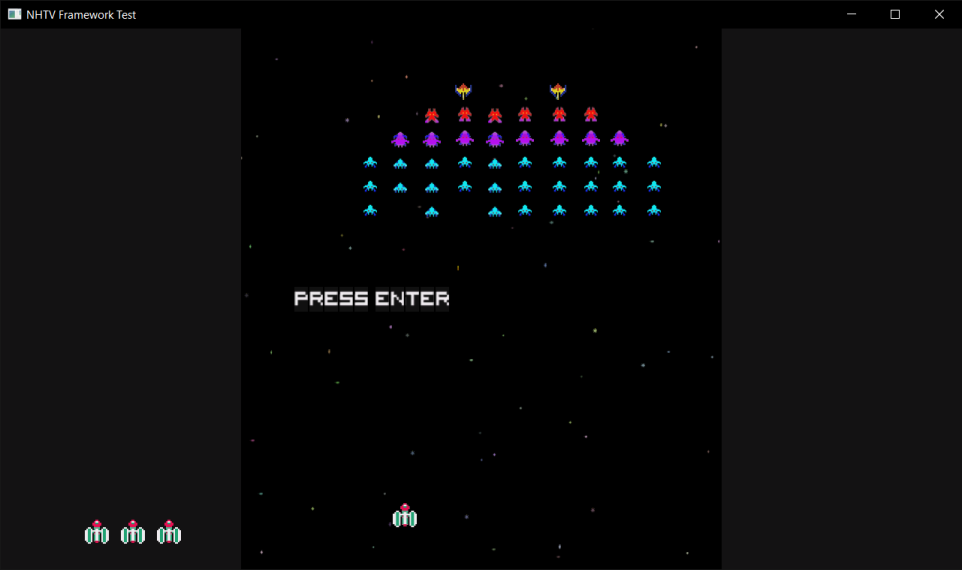


Which I changed to this. (every symbol has the same size, makes life much easier) 

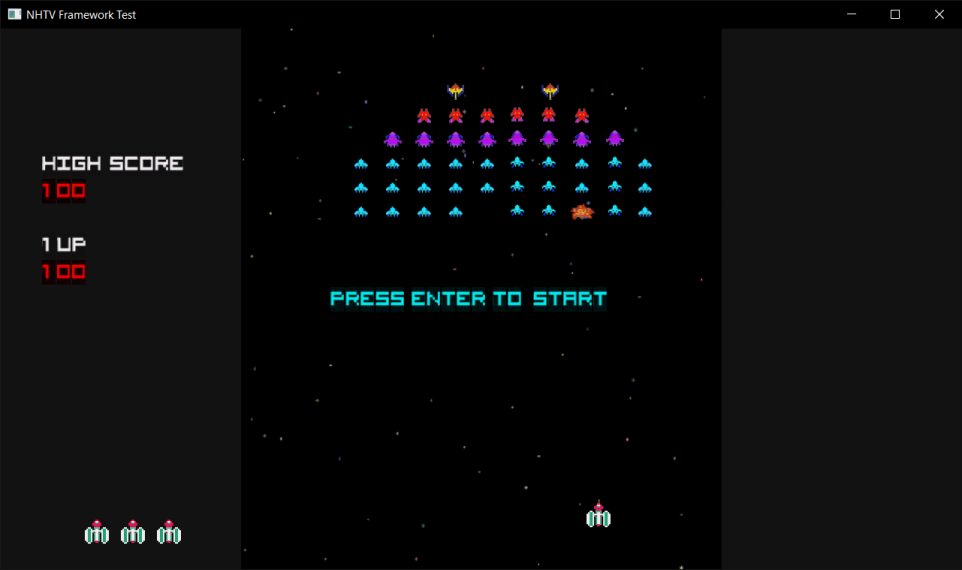
Alright we got the image file, we can work with this. So let’s print it. Doesn’t look to bad. I decided to map every letter to its index. By doing this, we can use the uv coordinate to access the right value. Just multiply the u coordinate by the index.

But it didn’t look all to good. I decided to scale the sprite to a size of the letter.

Learned something new from the engine. It’s quite funny how much SFML did for me without me knowing it. So let’s try writing a sentence now.

It looks unreadable, but why? After a bit of debugging it couldn’t handle the spacebar. I forgot to add the spacebar to the font image and to the map, so I wrote some code to split the string when a spacebar was hit and wrote some code to move the beginning position of the subsequent strings to make it look like it was one sentence.

It didn’t support multiple spaces through, which I easily fixed with a loop.

We are almost there. The last letter didn’t appear, These are unclear instructions, The reason why the last letter didn’t appear is because I didn’t track the index past the end. Only at the end, so the offset was wrong in the last string. Easy to fix, just add a plus 1 to it. And with that, the text engine is actually done. To reap the benefits of the engine, I added color support just to make it all look a bit nicer.Couldn’t be happier with it.

## So what did I learn

I applied what's called a bitmap font. I actually also converted one myself. It is quite handy, but my implementation really lacks flexibility. This however is something that I don’t mind. There is no reason for the game to support a text engine that is more flexible than this.

I’m starting to decide on what I should compromise in the code. It kinda makes me feel like a lazy programmer taking the easy way out but in the end it is the better solution because a simple solution allows you to work faster on the next problem. Being efficient is something very important because solving problems motivates you to solve more.

One day I will use TypeFree to implement my text, but for now I will roll my own version of a text engine.