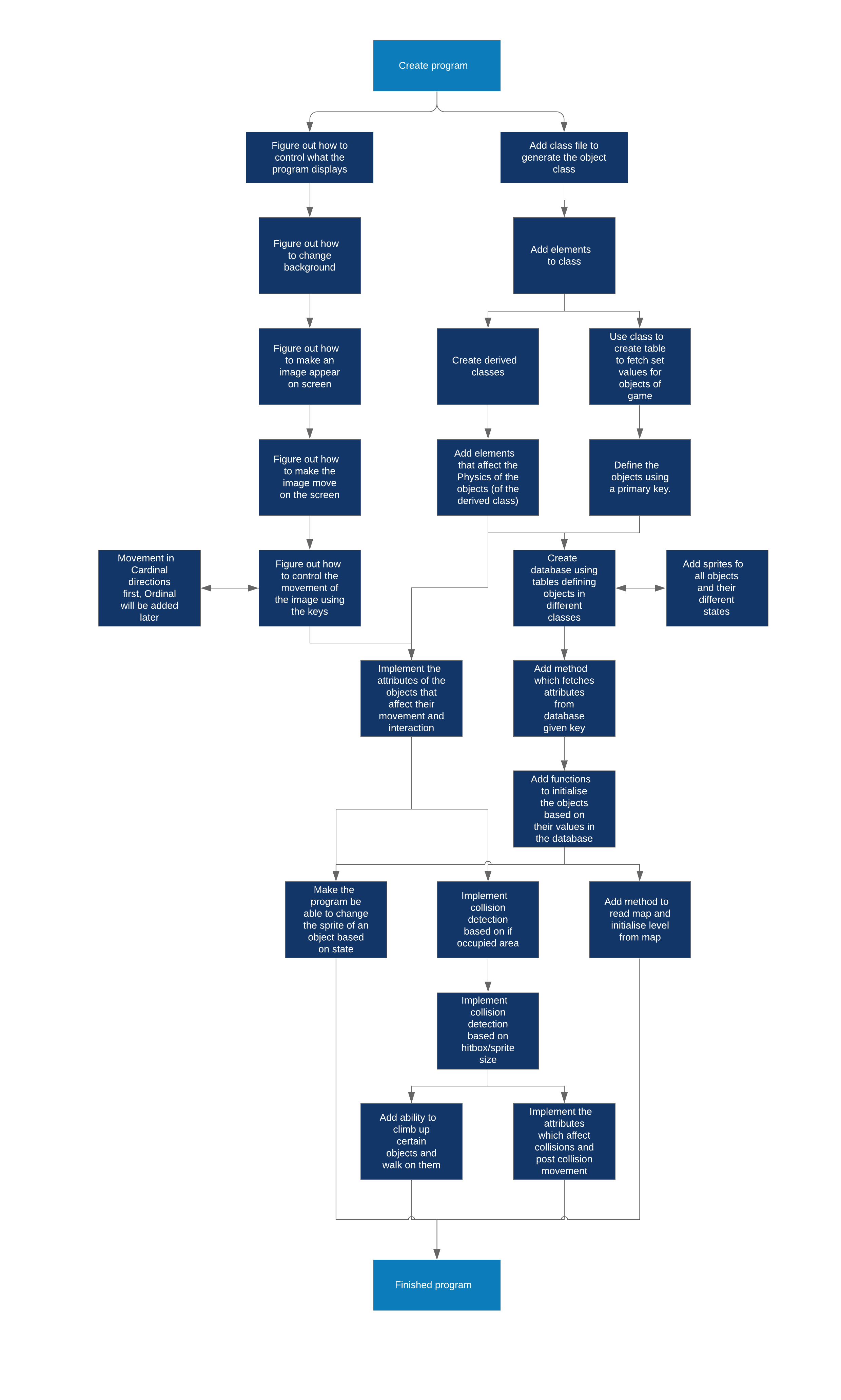
# Design

## Program development and objectives

Before I could begin making my program, I had to plan out how I was going to code it and what objectives I would aim for as the code went along to allow me to meet the planned specification. To this end I developed the Following Route map:   
As visible above, the entirety of the design process was not one single linear process, in order to be able to streamline my work, I needed to work advance on different parts of the project simultaneously, in modular implements so that I could have a clear idea of what I was doing and where I was going at any time. I could also have the ability to test and polish each individual part of the project to prevent errors from occurring later in development, where they would be more troublesome and harder to find, as opposed to coding larger parts of the program all at once. The design for my process was also itself quite flexible, I could easily switch between branches of the of the map as I advance, which would allow me to move forward if I got stuck on something to give me the time to come back to it later, since the project begins by working on the different aspects of the design and eventually bringing it all together in the final few steps.

## Language and graphics APIs/Libraries

When making the game, I decided to code it in C#, since it was the language I was most familiar with, and since the machine I was using to code my project was a mac which did not have Wine or Parallels Desktop, or any other software that would let me run Windows on Mac, I had to find an external graphics library, frameworks and/or APIs which would be compatible with Visual Studio Mac, and which would let me be able to have an output for the program. Through looking at online forums and what developers of recent games that I knew were compatible with OSX, I ended up with a list of possible solutions, following the most recommended and popular ones:

### Direct X

I decided to take a look at Direct X on windows since Yacht club games had used it for Shovel knight to build their own game engine. Direct X is a collection of APIs, however, since it is Windows exclusively, it was not relevant to my project.

### OpenGL-SharpGL

I decided to take a look at OpenGL for OSX and Linux since it was used by Yacht Club games for the development of shovel knight, specifically looking into the SharpGL library which made it compatible with C#, since OpenGL is coded in C. However, I learned that "OpenGL was deprecated in MacOS 10.14" whilst looking at the apple developer tutorials and instructions on the API. Since I use OSX 10.14, and Apple had deprecated OpenGL in favour of their own API: Metal. I had no choice but to look elsewhere.

### Metal

Seeing as Apple had deprecated OpenGL in favour of Metal, I decided to give a look at the API, but gave up after finding out it wasn't compatible with C#, working only in objective-C and swift, both languages in which I am unfamiliar.

### Monogame

Monogame was a strong contester and framework which I saw being recommended in multiple online forums, being cross platform and compatible with Visual studio, however I did not end up using it, because of the learning Resources that Spritekit offered and how much easier it was to use.

### SpriteKit

Since the Spritekit library came packaged with Xamarin and Visual Studio community 2017, and was very strong with a lot of online support, it was ideal for me to use, since was easily accessible and had all the features I was looking for, as well as the fact that it was compatible with my current OSX version meant it was by far the best choice to use. Although most online support is in Swift, as are most tutorials, due to the similarities between C# and Swift, it was easy to bridge the gap.

### Choosing

In the end, I decided to choose SpriteKit because of how much more available it was in comparison to the other libraries, as well as the fact that it had a lot of online support from the Xamarin website, and a multitude of helpful and useful tutorials (in swift). The fact that it was built by Apple to work using their own integrated systems in OSX and iOS, made it the obvious option, Monogame, being a close second due to its viability and cross-platform abilities.

After having decided to use SpriteKit, I had to make sure that it would be compatible with the built-in libraries I needed to detect key presses so, I changed the project framework from Xamarin.mac, to .NET 4.7. In order to learn how the library worked and its functions I turned to online swift tutorials, due to the lack of C# tutorials, since the languages were so similar.

## Classes

### Sprite class

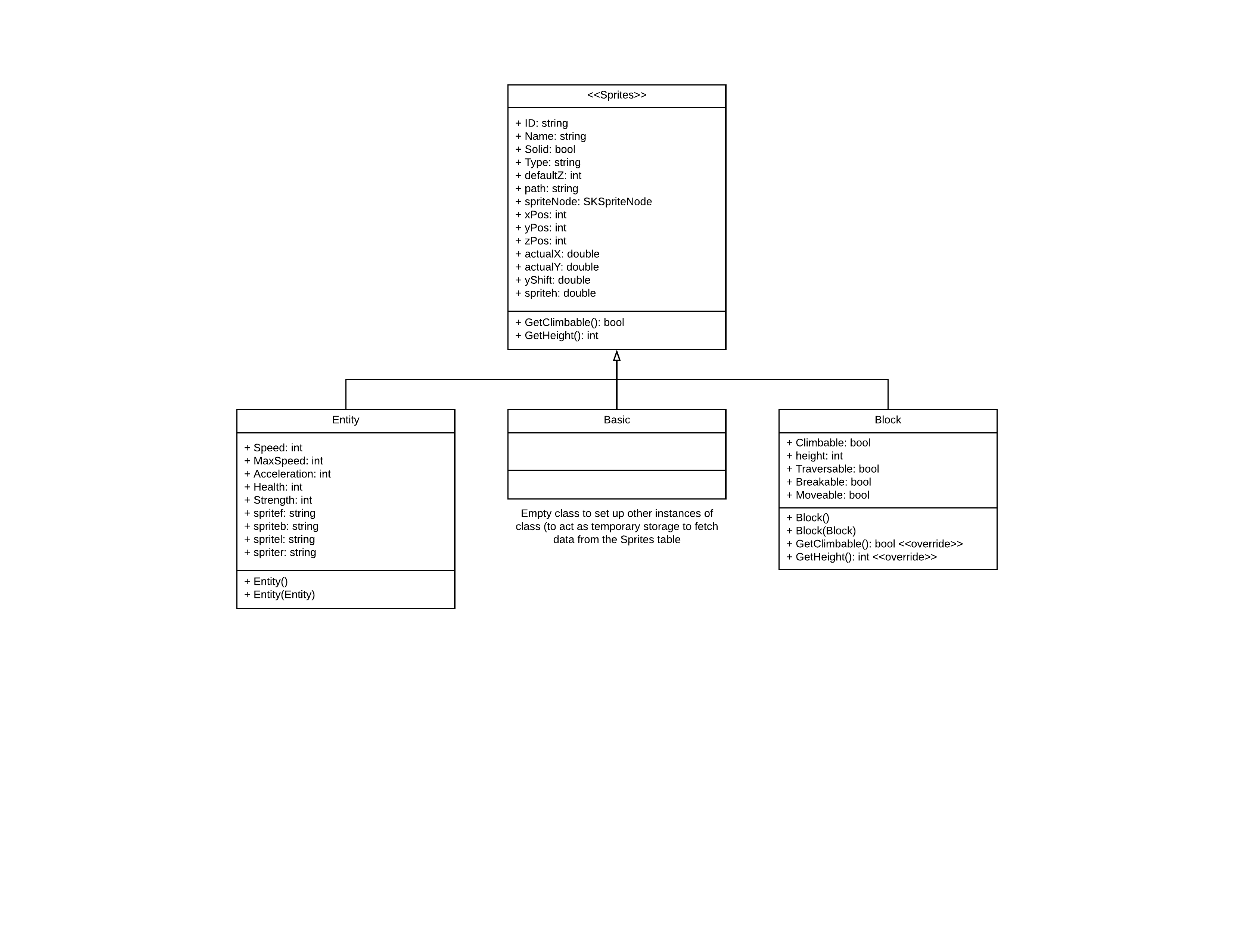
Taking inspiration from the tile systems present in the original Zelda and Metroid games, as well as the tile and height system heavily present in Pokémon, I built a class that would be able to represent the objects inside the game.

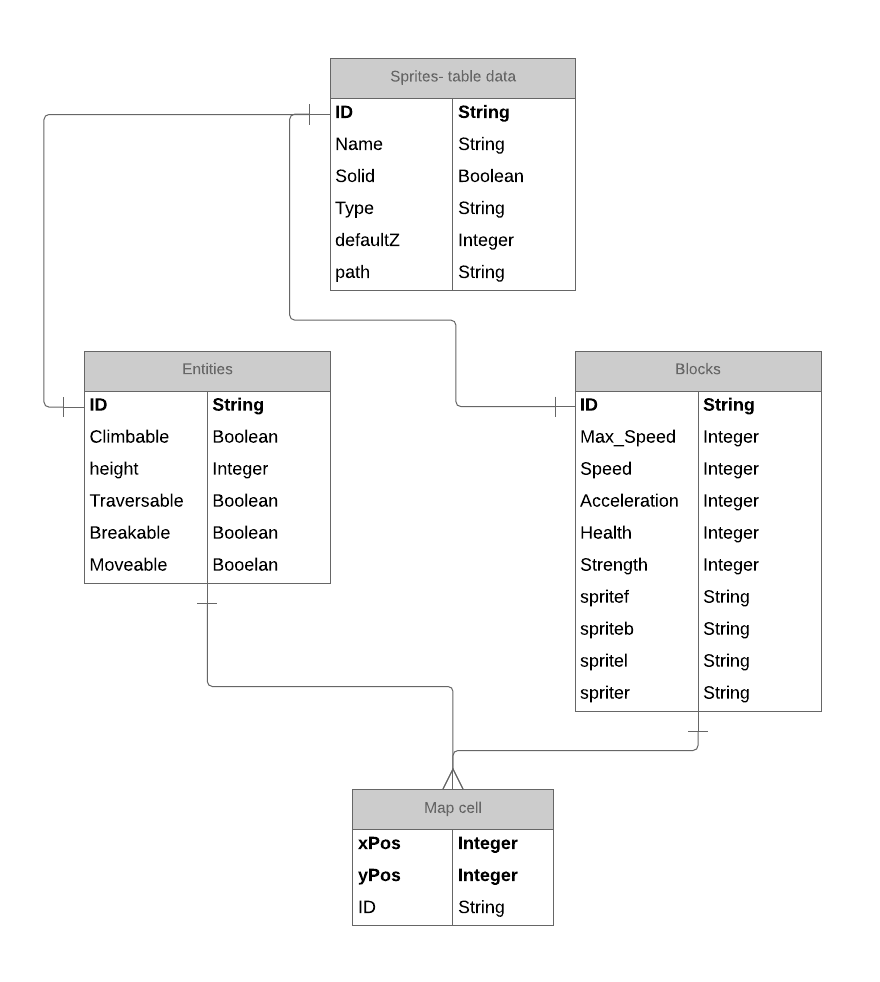
The Sprite class stores all the necessary attributes of the objects that fill out the world, such as the player, enemies, and blocks/obstacles.

The base class is made to act as an abstract class to build its derived instances. It stores the main values that allow the program to initiate and display the objects and their sprites onto the screen.

#### Entity and Block derived classes

The entity derived class represents a character in the game,





# Bibliography

Apple inc. 2018. *About OpenGL for OS X.* 4 June. Accessed March 17, 2019. https://developer.apple.com/library/archive/documentation/GraphicsImaging/Conceptual/OpenGL-MacProgGuide/opengl\_intro/opengl\_intro.html#//apple\_ref/doc/uid/TP40001987-CH207-TP9.

Apple inc. 2019. *Metal.* 11 March. Accessed March 17, 2019. https://developer.apple.com/metal/.

Kerr, David. 2014. *OpenGL in .NET - Getting Started.* 8 April. Accessed March 17, 2019. https://www.codeproject.com/Articles/749543/OpenGL-in-NET-Getting-Started.

OpenGL. 2018. *OpenGL coding resources.* 8 September. Accessed March 17, 2019. https://www.opengl.org/resources/.

Stolpe, Philip. 2019. *Monogame.* 12 March. Accessed March 17, 2019. http://www.monogame.net.

u/Just\_Farming\_DownVs. 2017. *What can I use for basic, fast 2D graphics?* 19 February. Accessed March 17, 2019. https://www.reddit.com/r/csharp/comments/5uvyw9/what\_can\_i\_use\_for\_basic\_fast\_2D\_graphics/.

Wikipedia. 2019. *DirectX.* 15 March. Accessed March 17, 2019. https://en.wikipedia.org/wiki/DirectX.

—. 2019. *OpenGL.* 5 March. Accessed March 17, 2019. https://en.wikipedia.org/wiki/OpenGL.

Zeffrit. 2013. *Looking for the best Graphic Library to use with C# (No Xna Please).* 3 December. Accessed March 17, 2019. https://www.gamedev.net/forums/topic/650755-looking-for-the-best-graphic-library-to-use-with-c-no-xna-please/.