Constrained Game Development in HTML and Javascript

Jack Maber

Introduction

The challenge that I will be discussing in this poster will be my implementation of Snake into plain JS and HTML in under 100 lines of code.As I enjoy working within tight constraints, I thought that remaking Snake would be perfect to try and fit into under 100 lines of code, and still have a funtioning game. With very little space to work with when it came to creating the game features, I had to get quite creative with some of my code.

Solution

To ensure that the code for the game would fit within the 100 line limit that I set myself, I had to employ some pretty creative methods to get all of the functions working correctly, mainly by just boiling them down to their core purpose and using as comparison methods as possible as this makes for the most simple functions. The function that shows this off to the max is the "Wrap Around" function that can be seen in the code snippet at the bottom right of the screen; all it does is compares the Player's X&Y location values to the "GridSize" value, which, as mentioned before, is the outer dimensions of the canvas, and thus if in the case of the first part, the Player's X value is less than zero, meaning it is heading out of the left side of the screen, it is set to the GridSize -1, which is 19, and thus will and thus, the player will appear on the right side of the screen. The rest of function is much the same but just with the relevant values switched around to make sure the wrap around works correctly.

What Worked Well

Most of the game works perfectly, due to the simplicity of the code there is little room for any errors or bugs in the code, however this is a double edged sword as the lack of complexity also leads to some other issues such as if you input left and right rapidly the player backs up on itself and shrinks by a few squares (Depending on the length), this is due to the way that the snake is drawn, but correcting it adds significant complexity so it will have remain.

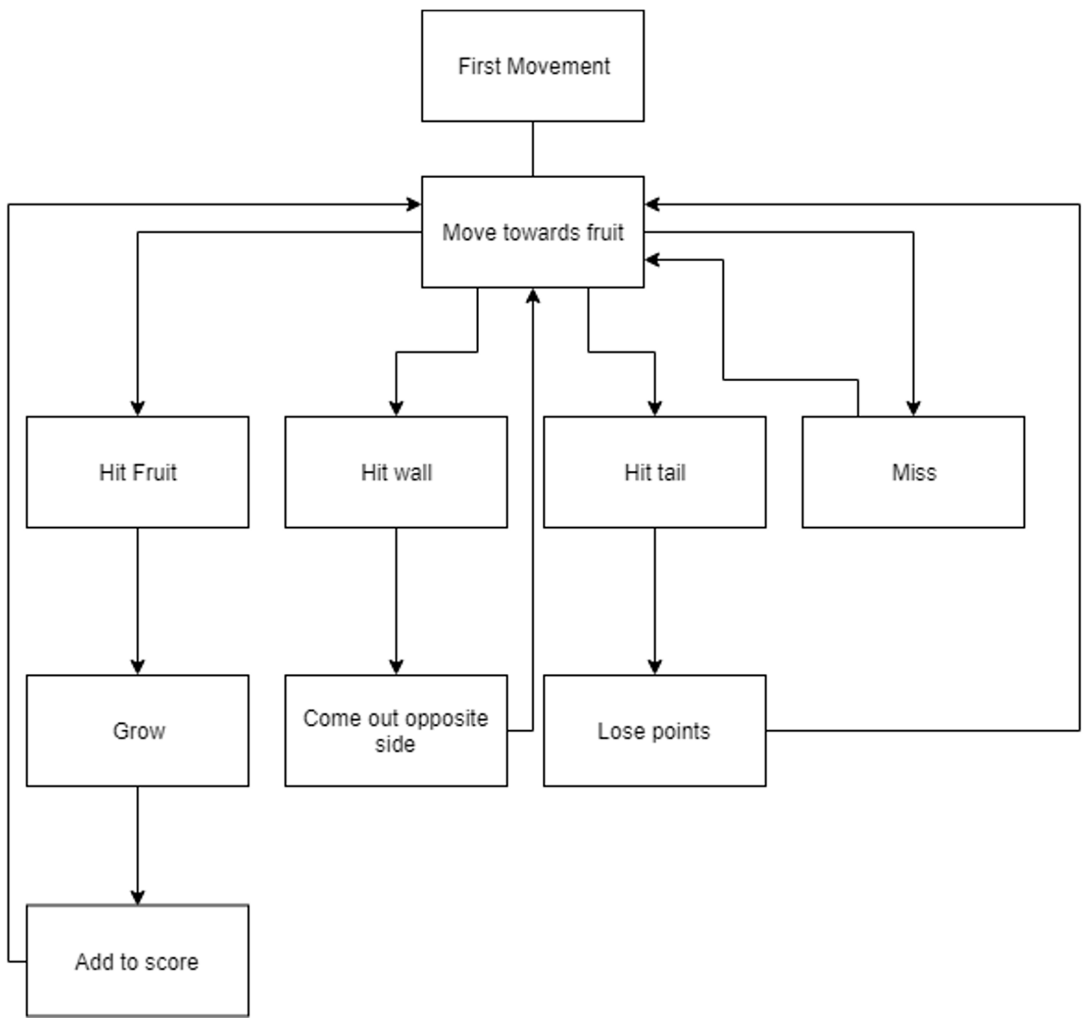
Future Improvements

If I was to go back to the project, I would add a way of tracking the score, rather than just tracking the length of the tail, as this would allow for more competitive play between players. Along with the shrinking issues I discussed in the previous section, I would probably expand the canvas, and thus gameplay area to be bigger as it's slightly hard to keep playing in such a small area when you get bigger, most of the maths is hard coded to the canvas size and thus would need refactoring.

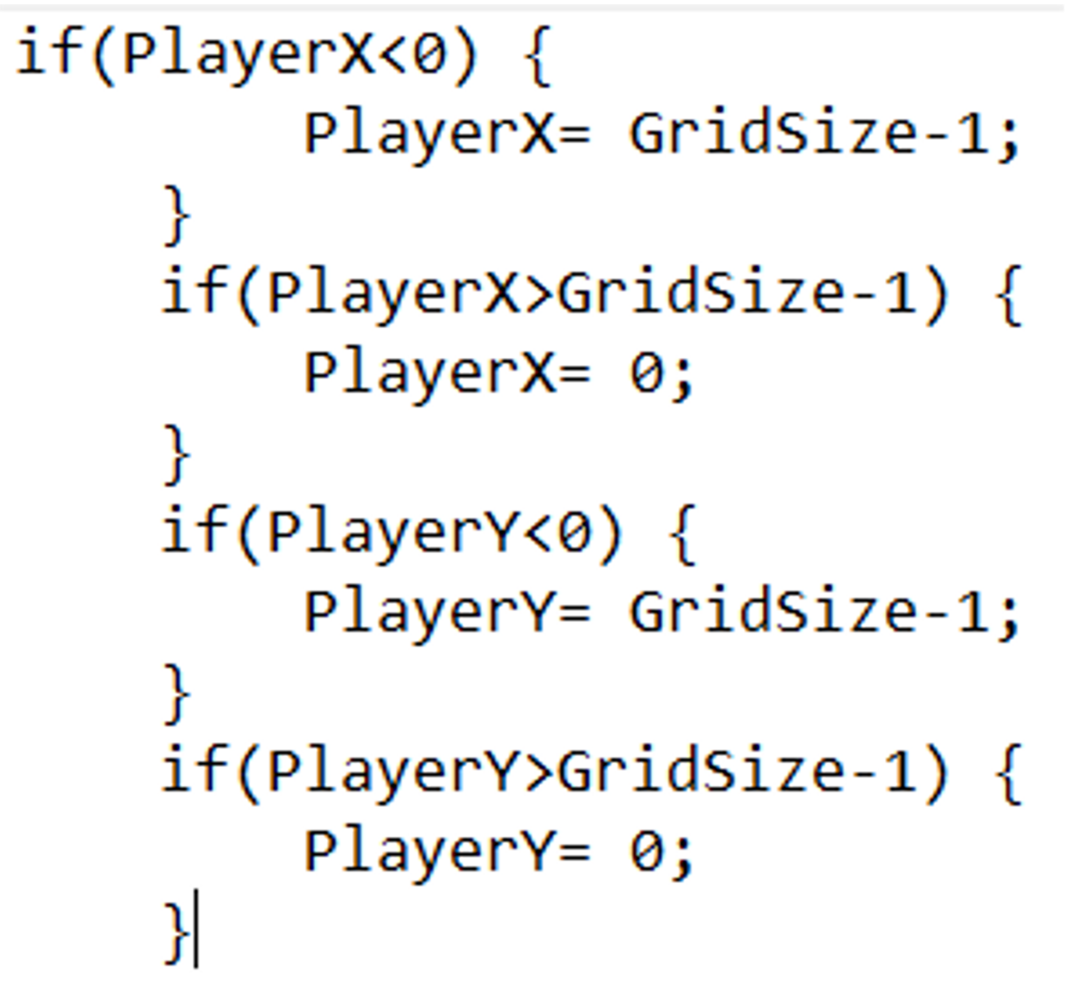
Method

To create a competent game under such tight constraints I utilised the "Canvas2D" API as this allows as it allows for the drawing and filling of simple shapes on a set canvas, along with the fact it is lightweight, and doesn't require any additional packages to be installed, the documentation is also very good, so it was relatively easy to code. To keep the code light and ensure I was under 100 lines most of the main game loop and functions revolves around the "GridSize", and such most of maths and comparisons are compared to this value, keeping most of the code very simple and short.

Game Loop Flow Chart



Wrap Around Code Snippet



```
if(PlayerX<0) {
        PlayerX= GridSize-1;
    }
    if(PlayerX>GridSize-1) {
        PlayerX= 0;
    }
    if(PlayerY<0) {
        PlayerY= GridSize-1;
    }
    if(PlayerY>GridSize-1) {
        PlayerY= 0;
    }
```