Snake AI: Project 1

By Pierre Turle (pturl001)

Overview

When I started this project I had no idea how to use behaviour trees, though I was alright with some of the theory. I had a much easier time plotting it out on paper as a flowchart and so eventually ended up with my desk covered in little scraps of paper, but I couldn't test anything though so I had to figure it out eventually. As I coded I found that many structures were quite verbose so I kept and discarded pieces of code in a "code graveyard" (at the bottom of my script) to copy pieces from should I want to put them back in. As I just threw things in there when I didn't want them anymore, it's quite disorganised but if you want a laugh about how I started coding behaviour trees then the code near the bottom should be my earliest work.

The biggest revelation for me was how to make if/ elif/ else statements by nesting a Filter in a Selector: Once I could correctly manipulate conditionals I felt like I could easily code and create behaviour trees, whilst also having a clearer idea of how to navigate other nodes and filters. Eventually I ended up with a 53 line long sketch which survives until around 150 pieces of food have been eaten and was happy with my journey and learning on the way.

Process

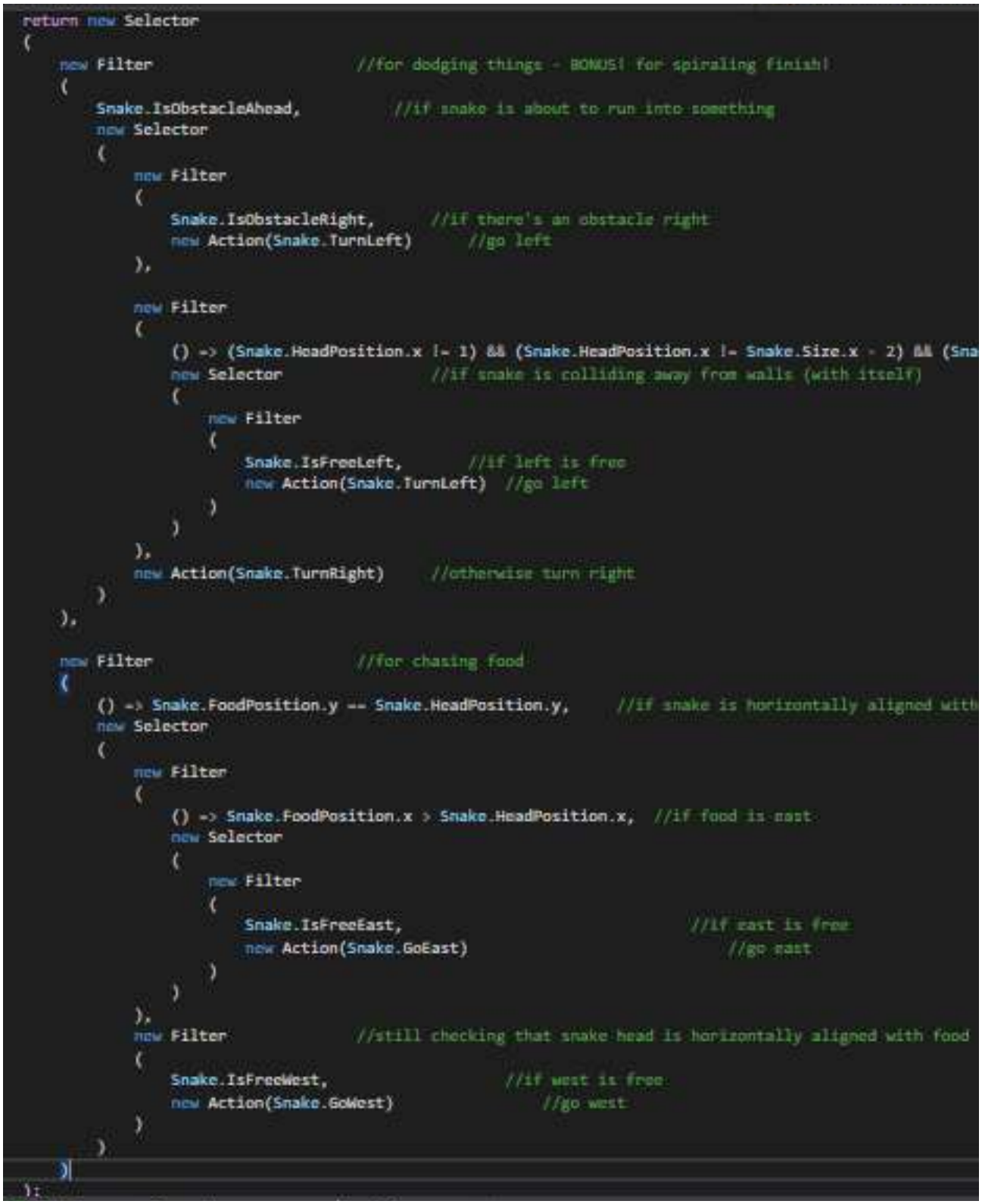
Initially my goal was to make the snake search for the food, and right up until the end of my project the collision detection was only at the very bottom of my script (IE the least important thing, as my root node was a selector). Since the snake starts going north I started checking for food that is East, and turning appropriately. I realised that unless the snake was parallel (horizontally) to the food it would turn to soon, so I added a horizontal check, nesting an East and a West check (and though it's been cleaned up, that's still the heart of my logic). When I was happy that my snake would eat 1 piece of food about half the time I copied the code and edited it for North and South.

Around this point I became fully comfortable with how filter-selector nests worked, and reduced my code a huge amount by changing multiple nested conditionals to nested if-else statements. I then added collision detection, turning randomly when hitting an object, and eventually always right. At this point my snake was eating about 30 pieces of food in one run.

My snake would often end up spiralling in on itself, and it seemed to make sense to make it turn left when running into its own tail to stop it making tiny loops anyway, so I added one ridiculously long line of code to check if its head was near a wall when it collided or not. If it collided while not near a wall it would turn left. I commented out the North/South food chasing parts to test this code in the East-West part and my scores shot up! It made sense logically too: not searching in 1 plane means that the snake will give himself more room to travel. I eventually added checks to see if the turning direction was actually free before turning and I was finished.

Behaviour Tree

(just realised that some of the comments are old and unchanged but such is life)



```
return new Selector
(
    new Filter                      //for dodging things - BONUS! for spiraling finish!
    (
        Snake.IsObstacleAhead,          //if snake is about to run into something
        new Selector
        (
            new Filter
            (
                Snake.IsObstacleRight,      //if there's an obstacle right
                new Action(Snake.TurnLeft)      //go left
            ),

            new Filter
            (
                () => (Snake.HeadPosition.x != 1) && (Snake.HeadPosition.x != Snake.Size.x - 2) && (Sna
                new Selector                //if snake is colliding away from walls (with itself)
                (
                    new Filter
                    (
                        Snake.IsFreeLeft,       //if left is free
                        new Action(Snake.TurnLeft)  //go left
                    )
                )
            ),
            new Action(Snake.TurnRight)     //otherwise turn right
        )
    ),

    new Filter                      //for chasing food
    (
        () => Snake.FoodPosition.y == Snake.HeadPosition.y,     //if snake is horizontally aligned with
        new Selector
        (
            new Filter
            (
                () => Snake.FoodPosition.x > Snake.HeadPosition.x,  //if food is east
                new Selector
                (
                    new Filter
                    (
                        Snake.IsFreeEast,                               //if east is free
                        new Action(Snake.GoEast)                            //go east
                    )
                )
            ),
            new Filter              //still checking that snake head is horizontally aligned with food
            (
                Snake.IsFreeWest,                   //if west is free
                new Action(Snake.GoWest)                //go west
            )
        )
    )
);
```

I noticed the inconsistency with IsObstacleRight and IsFreeLeft, but that is more a stylistic issue

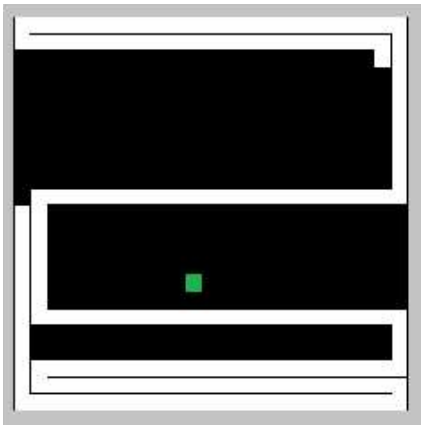
Conclusion

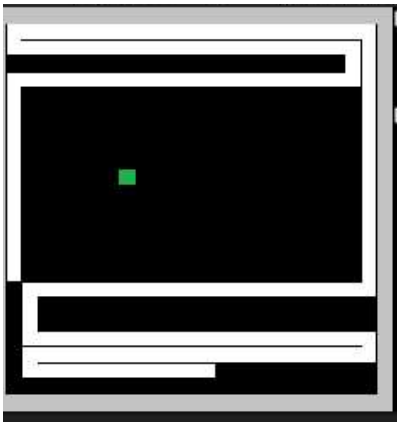
My snake survives very well to reasonably large lengths fairly consistently. Safety checks are implemented first and collisions are dealt with in such a way that the head often ends up moving along its tail which means that more space is more likely to be left unobstructed. I am pleased with the snakes overall performance, though given more time (and using variables) I would be interested in seeing the outcome of making the snake follow its body indefinitely.

I have seen similar snakes in past years and I'm sure you have seen something like this before, but hopefully my code is a bit shorter and cleaner than you often see. Though I could have spent longer on it to further extend its longevity, I am very happy with the outcome and don't feel like the amount of time required offset with the potential improvement would be worth it. I was recently told that the use of variables is allowed, which I requested early on, but considering what I have made I am very happy as it is.

Though I am by no means experienced or professional at the subject, I feel comfortable with behaviour trees. This exercise, including the way you named your functions as nodes in a tree, has helped a lot. Since trees like this make up a lot of games AI logic I feel like I have taken my first steps into the world of writing AI, so thanks :)

Long Snakes





Confused Snakes