**Magnetic Field Simulator Explanation 07/11/18**

Key global variables: var **strengthScaler** = 50; var **distanceScaler** = 10; var **velMultiplier** = 10;

These were used to scale the overall velocity of the compasses, primarily for aesthetic reasons however they didn’t affect the ‘heading’ (angle) of the compass so are not important for the maths.

**The North and South Poles**

There is a ‘list’ (javascript term) of northPoles and southPoles. Each pole is initialised with a vector position and a strength e.g.

northPoles[0] = new northPole(windowWidth / 4 +220, windowHeight / 2 - 100, 100);

southPoles[0] = new southPole(windowWidth / 1.47 - 100, windowHeight / 2 - 100, 100);

X coordinate (n.b. in javascript unlike a graph, (0, 0) is the top left corner and the y axis is inverted)

Y coordinate

Strength of magnet

Every frame, the poles are displayed and respond to being dragged.

**Compasses**

There is a list of compasses which increases in length as new ones are added by clicking or dragging the mouse.

Each compass is initialised with a vector position, a velocity of (0, 0), a maximum velocity of 10 (which prevents the velocity becoming incredibly large as k/r^2 tends to infinity), an initial angle of 0, and a ‘history’ list which in which it stores all its previous positions as a list of vectors (so it can draw the trail).

They also have a **velN** which is the acceleration due to the north poles and a **velS** which is the acceleration due to the south poles.

Each frame, as well as displaying itself and its trail, each compass ‘reacts’ to each north pole and then to each south pole. When it reacts to a pole, the following happens:

1) The ‘difference’ between its position and the pole’s position is calculated (pole’s vector position – compass’s vector position) and stored as a variable **dif**.

2) Then the vector distance between the pole’s position and the compass’s position is calculated and stored as a variable **dist**.

3) The magnitude is calculated and is equal to (distanceScaler x 50 x the strength of the magnet) / **dist ^** 2. This is stored as a variable **mag**. The magnitude is multiplied by distanceScaler and 50 because otherwise it is too slow the compass moves incredibly slowly.

4) The magnitude of **dif** is set to **mag** and if it is a north pole, dif is multiplied by -1(as you want a vector in the opposite direction since they are repelled)

5) **dif** is added to **velN** or **velS** (I could technically just have one vel variable but this is clearer)

This process is repeated for all of the north and south poles and you are left with a **velN**, the velocity due to the north poles, and a **velS**, the velocity due to the south poles. Then the **update ()** function is called.

In this function, **velN** and **velS** are added together to form an overall velocity which is added to the position vector to move the compass. The angle of the compass is set to the heading of the velocity. The velocity is then scaled by the velMultiplier and reduced if it is over the maximum velocity before **velN** and **velS** are reset back to (0, 0) and the process is repeated next frame. The compass’s position is added to the **history** list and each frame this is displayed with a line joining the points, so it forms a fairly smooth curve.

The black arrow at the end, shows the overall direction of the field line and lines up with the points. This was calculated with some vector math and simple trigonometry using tan.