

```

x = GWIDTH / 2
y = GHEIGHT / 2
SETWAVETYPE (0, 3)
SETDISTORTION (0, 0)
SETVOLUME (0)
STARTNOTE (0,0)
LOOP
CLS2
leftRight = MBACCELX
upDown = MBACCELY
COLOUR = RND (30)
CIRCLE (x, y, 20, 0)
COLOUR = TEAL
IF leftRight > 400 THEN
SETVOLUME (100)
SETFREQUENCY (0, 660)
COLOUR = LIGHTPINK
ENDIF
IF leftRight < -400 THEN
SETVOLUME (100)
SETFREQUENCY (0, 524)
COLOUR = LIGHTYELLOW
ENDIF
IF upDown > 200 THEN
SETVOLUME (100)
SETFREQUENCY (0, 699)
COLOUR = LIGHTBROWN
ENDIF
IF upDown < -200 THEN
SETVOLUME (100)
SETFREQUENCY (0, 588)
COLOUR = RASPBERRY
ENDIF
xx = x + leftRight
IF xx > GWIDTH THEN xx = GWIDTH
IF xx < 0 THEN xx = 0
yy = y - upDown
IF yy > GHEIGHT THEN yy = GHEIGHT
IF yy < 0 THEN yy = 0
IF ABS (leftRight) < 100 AND ABS (upDown) < 100 THEN SETVOLUME (0)
LINE (x, y, xx, yy)
CIRCLE (xx, yy, 10 + ABS (leftRight + upDown) / 10, 1)
UPDATE
REPEAT
    
```

This project introduces quite a few new commands and functions.

First off we're going to set up a new sound channel by selecting the sound waveform (0=Sine, 1=Sawtooth, 2=Square, 3=Triangle and 4=Noise).

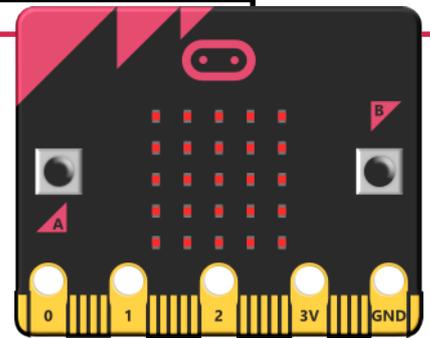
Next we set the distortion effect to 0. This can be anywhere from 0.000 to 1. Finally we set the audio channel volume to 0 (silent) and set a note playing, even though we can't hear it.

We begin a main **LOOP** and define two variables **leftRight** and **upDown** to store the values given by the BBC micro:bit's accelerometer's movement sensors.

Now we check to see if we have moved the micro:bit left, up, right or down and if so adjust the note frequency, set the volume to max and set a colour. The value of the accelerometer increases or decreases when we move the micro:bit around. We have used a value of 400 (+/-) for left and right and 200 (+/-) for up and down. This stops the program from being too sensitive - try lower values and you'll see what we mean.

The final section checks to make sure our circle is not drawn off the screen by testing the **leftRight** and **upDown** variables against the edges of the screen (**GWIDTH** & **GHEIGHT**).

The line with **ABS(leftRight) < 100** checks to see if the position of our moving circle is within 100 pixels of the centre in all directions and if so set the volume to silent (phew!). Because **leftRight** & **upDown** can be positive or negative, **ABS** (absolute) is used to convert them into a positive number only. Very clever!



### ADVANCED CHALLENGE:

How about adding more sounds if you press one of the micro:bit buttons (IF MBBUTTONA = 1 THEN...). Or what about making the LED matrix display something when a sound is played, or changing the SETWAVETYPE (0, #) value or...