# Activity 2: Avoiding obstacles

Being an astronaut requires you to respond quickly to changes in your environment. It's all about **visual perception** and **reaction times**, and we want you to put your skills to the test in your classrooms.

Test your own reaction times by programming and playing this meteorite game on your BBC micro:bit. Record your score in each game and send us your best times at live.lessons@bbc.co.uk or using the hashtag **#bbclivelessons**. They could be featured in the Live Lesson on the 7th of June.

## Step 1: Import the code

Click on the hex file link on the Live Lessons website to view the code on the BBC micro:bit website.

The script for your meteorite game should now appear in your code window.

Hit 'run' to see it in action on the simulator, or plug in your BBC micro:bit, hit 'compile' and drag the hex file onto your micro:bit.

## Step 2: Understanding the code

```
script Meteor Game
function main ()
    initialize game
    basic → forever do
        move meteor 1
        move meteor 2
        pause difficulty :=
        basic → pause(pa
        add code here
end
```

### Initialising the game

Here a **function** called **initialize game** is introduced.

```
script Meteor Game
function initialize game ()
    meteorite 1 x := math → random(5)
    meteorite 1 y := - 4
    meteorite 2 x := math → random(5)
    meteorite 2 y := - 1
    ship left x := 0
    pause difficulty := 1000
    score := 0
    led → plot(ship left x, 4)
    led → plot(ship left x + 1, 4)
end function
```

This sets out the starting conditions for the game – the X and Y coordinates for the meteorites, the position of the 'ship', the difficulty level and the beginning score – 0.

```
script Meteor Game
function main ()
    initialize game
    basic → forever do
        move meteor 1
        move meteor 2
        pause difficulty := 
        basic → pause(paus
```

This determines that the meteorite is 'falling'. It states that the y-coordinate of the meteorite increases by one pixel with each loop.

This condition states that if your ship's x and y coordinates is equals to the x and y coordinates of the meteorite (the meteorite collides with your ship), the game is over.

## The forever loop

A loop which allows the game to continue 'forever' until the game ends is introduced. Within that, you'll find two more **functions**, which determine how the meteorites move.

```
script Meteor Game
function move meteor 1 ()
    if meteorite 1 y ≥ 0 then
        led → unplot(meteorite 1 x, me
    else add code here end if
    meteorite 1 y := meteorite 1 y + 1
    if meteorite 1 y ≥ 5 then
        score := score + 1
        meteorite 1 x := math → random(
        meteorite 1 y := - 1
    else add code here end if
    if meteorite 1 y = 4 and (ship left
     = meteorite 1 x) then
        game over
    else add code here end if
    if meteorite 1 y ≥ 0 then
        led → plot(meteorite 1 x, mete
    else add code here end if
    add code here
end function
```

At the start of each loop, we unplot the LEDs from the previous meteorite and get ready to set the coordinates for the next meteorite.

This condition determines if you have successfully navigated past a meteorite. It states that if the y-coordinate of the meteorite is more than or equals to 5 (it reaches the bottom of the screen), it is then re-positioned to a random x-coordinate and a y-coordinate of -1 (the top of the screen).

We then need to plot the LEDs with the new x and y coordinates for the meteorite, so they display on the screen.

We also increase the difficulty of the game (the variable **pause difficulty**) with each loop.

```
input → on button pressed(A) do
    if ship left x ≥ 1 then
        move ship left
        detect collision
        add code here
    else add code here end if
end
input → on button pressed(B) do
    if ship left x ≤ 2 then
        move ship right
        detect collision
    else add code here end if
end
```

## What do the buttons do?

Here we determine what each button does in the game. It states that, when button A is pressed, the ship 'moves left' if the ship's X-coordinate is more than or equals to 1.

This is done using a **function** called **move ship left**, which unplots the X-coordinate for the ship and plots it one pixel to the left. You can see the code for the function below:

```
function move ship left ()
    led → unplot(ship left x + 1, 4)
    ship left x := ship left x - 1
    led → plot(ship left x, 4)
end function
```

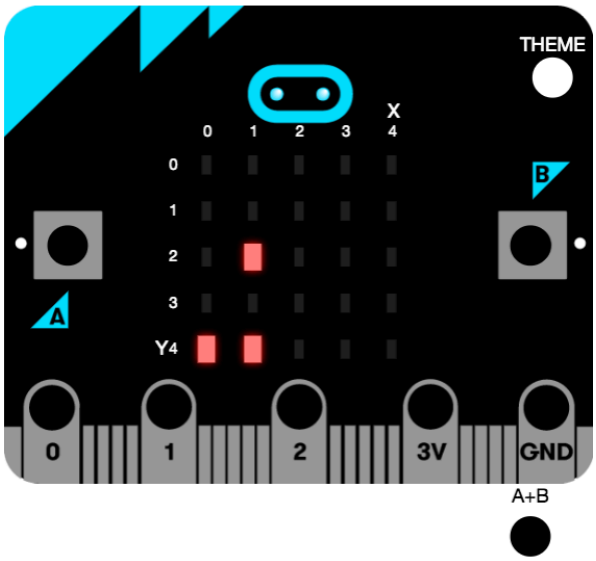
We then detect if there has been a collision, by using a **function** called **detect collision** which checks if the coordinates of the ship match the coordinates of the meteorites.

The game is over when a collision is detected, and your score is displayed on screen, as per the **function game over** below:

```
function game over ()
    led → plot all
    for 0 ≤ i < 3 do
        led → fade out(400)
        led → fade in(400)
        add code here
    end for
    basic → show string("SCORE", 150)
    basic → show number(score, 150)
    basic → pause(99999999)
end function
```

## Step 3: Test, play and show us what you've done

Have a go at playing the meteorite game on your BBC micro:bit and send in your best scores for a chance to be featured on our leaderboard during the Live Lesson.




Simply get your teachers to send your score along with your name and the name of your school to us at live.lessons@bbc.co.uk, or by using the hashtag **#bbclivelessons** before and during the Live Lesson on Tuesday 7th June.