# TTS Scratch Controller

Teacher Guide

www.tts-shopping.com

Product Code: EL00530

# TTS Scratch Controller

## Using the Scratch Controller in the classroom

Contents

# TTS Scratch Controller Teacher Guide

## Previous Experiences

Prior to using the TTS Scratch Controller, children should have some experience of using Scratch.

## Progression

The activities listed below are in a suggested order of progression. There is no specific amount of time to be spent on each as this will vary from one situation to another. It may also be necessary to break some of the activities down further to suit children's needs.

## National Curriculum

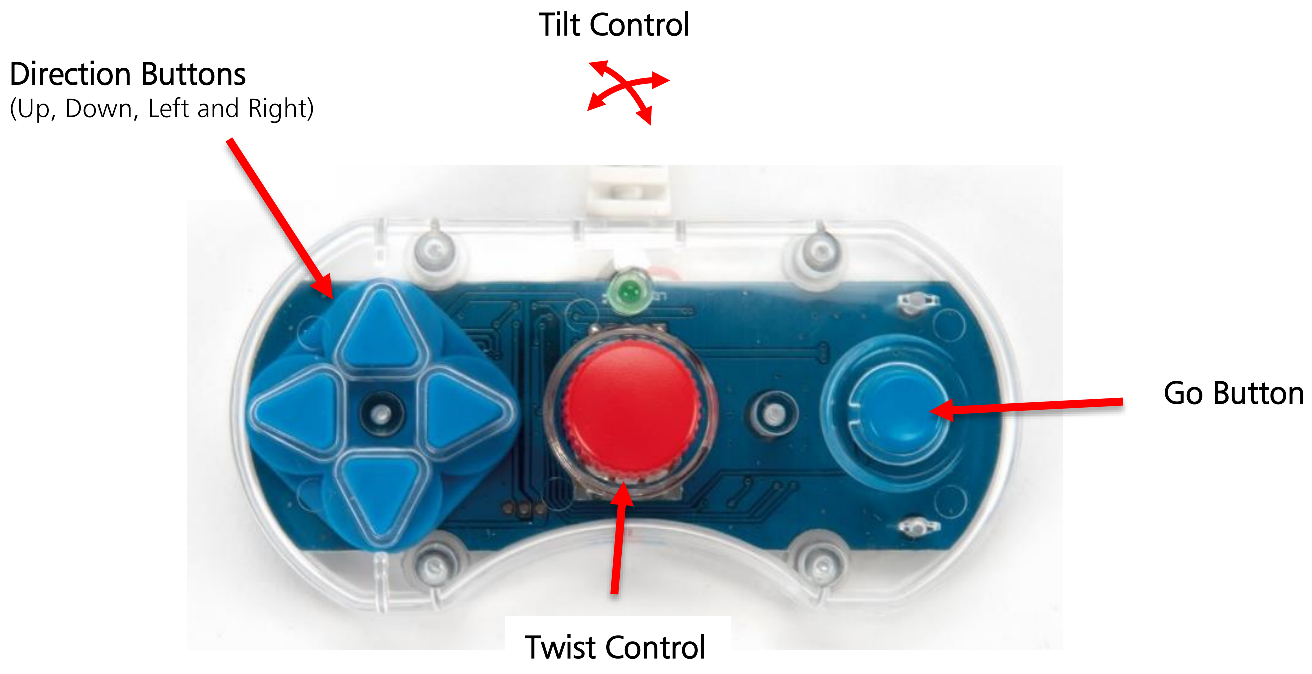
The National Curriculum for Computing references listed below are indicative of some aspects of the curriculum the activities cover. They are not an exhaustive list nor do they indicate that one activity fully covers that curriculum area. The activities support children in learning computing skills and applying computational thinking.

### Attainment Targets

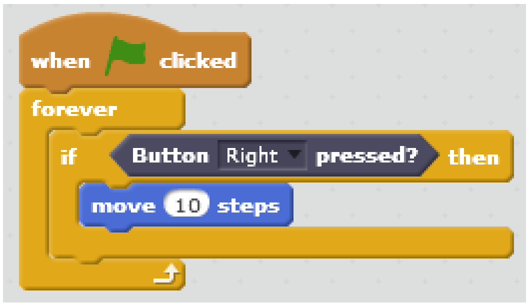
### Key Stage 2

- Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
- Use sequence, selection, and repetition in programs; work with variables and various forms of input and output
- Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs

## Key Focus 1 - Getting to know the TTS Scratch Controller



The easiest place to start with the Scratch Controller is to explore what just one button can do. For example write a Scratch program that makes a sprite move right when the right button is pressed.
(See example)



```
when [flag] clicked
forever
    if <Button Right pressed?> then
        move 10 steps
```

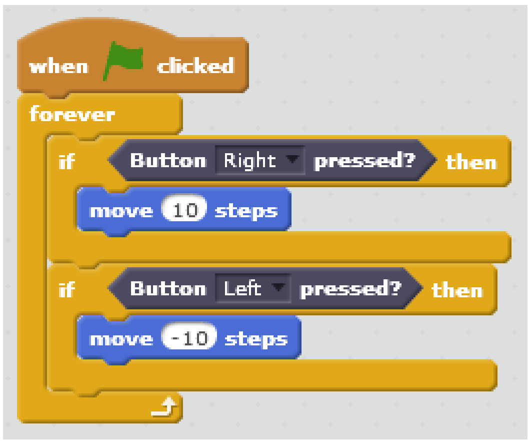
N.B. A forever loop needs to be used so the button is continually checked rather than checked once only.

Once the above example is working, try altering the number of move steps.

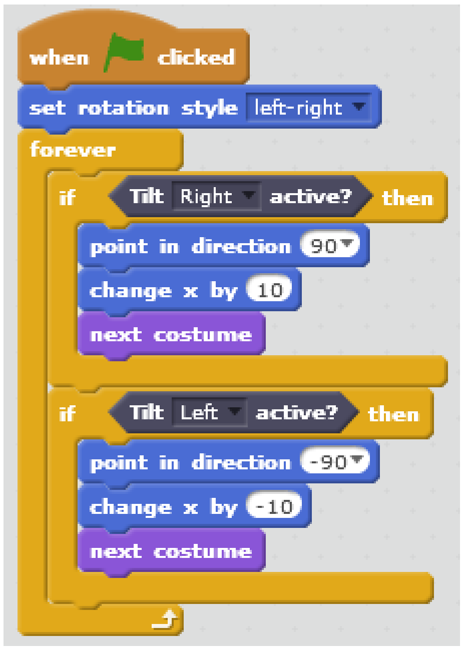
- What is the effect of reducing the number of steps?
- What is the effect of increasing the number of steps?

Next try adding another ‘if … then’ statement to see if the left button has been pressed and if so make the sprite move the other way.

The example (right) will make the sprite move left and right but not change the direction it faces. Try modifying the program to make the sprite change the direction it faces as it moves. (See example below)



```
when [flag] clicked
forever
    if <Button [Right] pressed?> then
        move (10) steps
    if <Button [Left] pressed?> then
        move (-10) steps
```



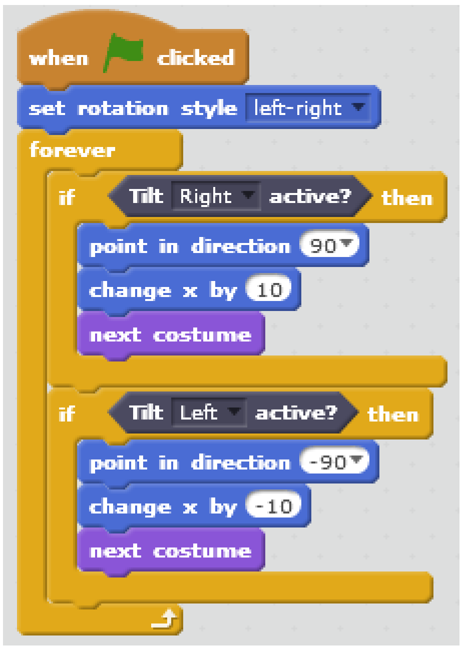
```
when [flag] clicked
set rotation style [left-right]
forever
    if <Tilt [Right] active?> then
        point in direction (90)
        change x by (10)
        next costume
    if <Tilt [Left] active?> then
        point in direction (-90)
        change x by (-10)
        next costume
```

This example could be extended even further to make the sprite move up and down the screen too.
Another way to achieve a similar effect would be to use ‘change x by ( )’ and ‘change y by ( )’ blocks.
Also explore adding ‘Wait ( ) secs’ blocks into each ‘if … then’ jaw to see what impact that has. (Try tenths of seconds.)

## Tilt

The examples above can all be easily modified to use the tilt control instead (as shown).

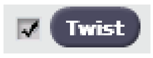
Note the addition of the ‘next costume’ block into the movement. This could be used in the earlier example to improve the look of the animation.



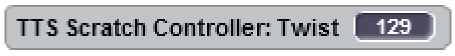
```
when [flag] clicked
set rotation style [left-right]
forever
    if <Tilt [Right] active?> then
        point in direction (90)
        change x by (10)
        next costume
    if <Tilt [Left] active?> then
        point in direction (-90)
        change x by (-10)
        next costume
```

## Key Focus 2 - Twist Variables

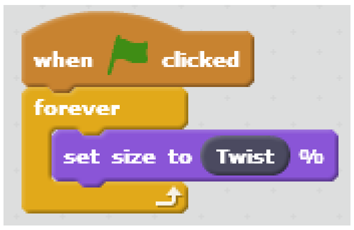
The Scratch Controller has a twist control input. This input reports values from 0 to 255 as a variable. Clicking in the tick box by the Twist variable will display the variable on the stage.



```
✓ Twist
```

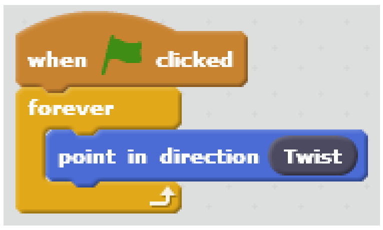


```
TTS Scratch Controller: Twist 129
```



```
when 🏴 clicked
forever
    set size to Twist %
```

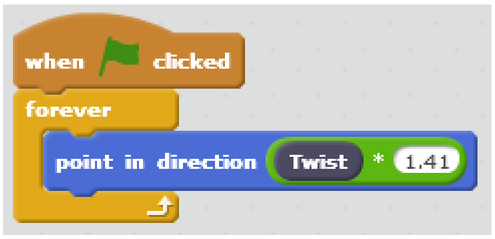
A simple way to explore the twist variable is to use it to change the size of a sprite (as shown on the left).



```
when 🏴 clicked
forever
    point in direction Twist
```

The twist control can be used to change the direction a sprite points.

The twist control can provide 256 values (0 – 255).
To make wider use of the control a little maths needs to be applied. For example, to make the controller rotate a sprite through 360°, divide 360° by 256 positions. This equals 1.41 (rounded down). This can then be used in Scratch.

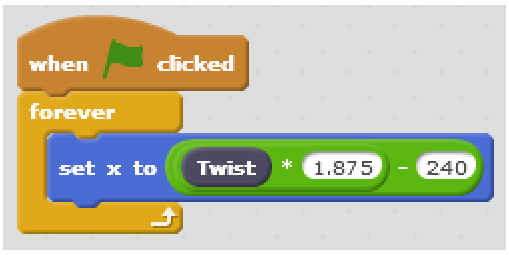


```
when 🏴 clicked
forever
    point in direction Twist * 1.41
```

If you apply the same principle, then a sprite could be made to only rotate through 180° instead i.e. 256 / 180 = 0.7 (rounded). So Twist x 0.7 will now make a sprite rotate through 180° rather than 360°.

### Converting Twist to left / right movement.

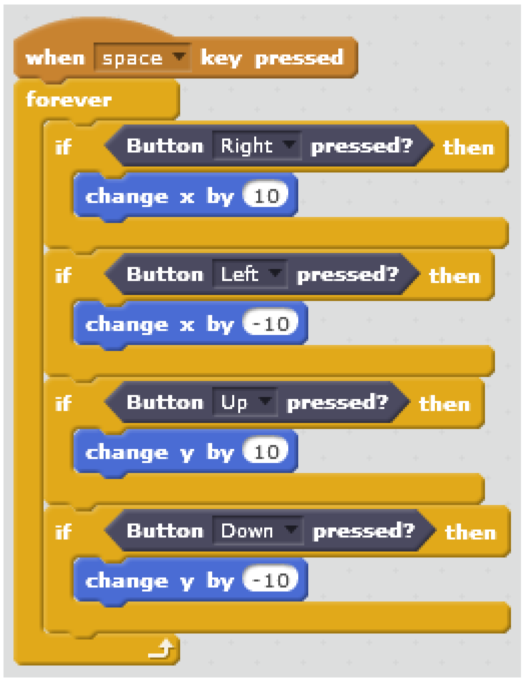
The Scratch Stage is 480 wide (-240 to 240). Divide the positions on the twist controller by the positions on the stage i.e. 256 / 480 = 1.875. Lastly, the x screen position 0 is in the centre of the screen. To make the twist controller move the sprite from one side to the other, 240 needs to be subtracted i.e. (Twist x 1.875) – 240.



```
when 🏴 clicked
forever
    set x to Twist * 1.875 - 240
```

The same principles can be applied to make the sprite move up and down the screen instead. The screen height is 360 (-180 to 180).

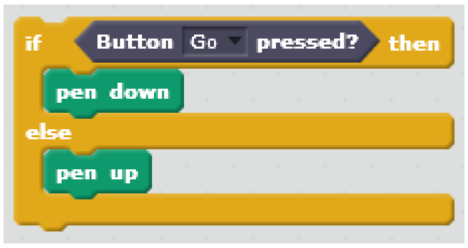
## Key Focus 3 - Using multiple controls 1 - Drawing



```
when space key pressed
forever
    if Button Right pressed? then
        change x by 10
    if Button Left pressed? then
        change x by -10
    if Button Up pressed? then
        change y by 10
    if Button Down pressed? then
        change y by -10
```

This is an example of using a number of controls. Firstly replace the cat sprite with a pencil (this can be found in the 'things' category). As shown in the earlier example, the program below will set the arrow keys to move the sprite around.

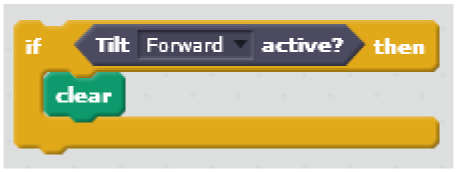
Simply adding the 'pen down' block (from the pen category) into the forever loop will make the program draw. This however will make the program draw all the time. The 'Go' button could be used to select when the pen is down.



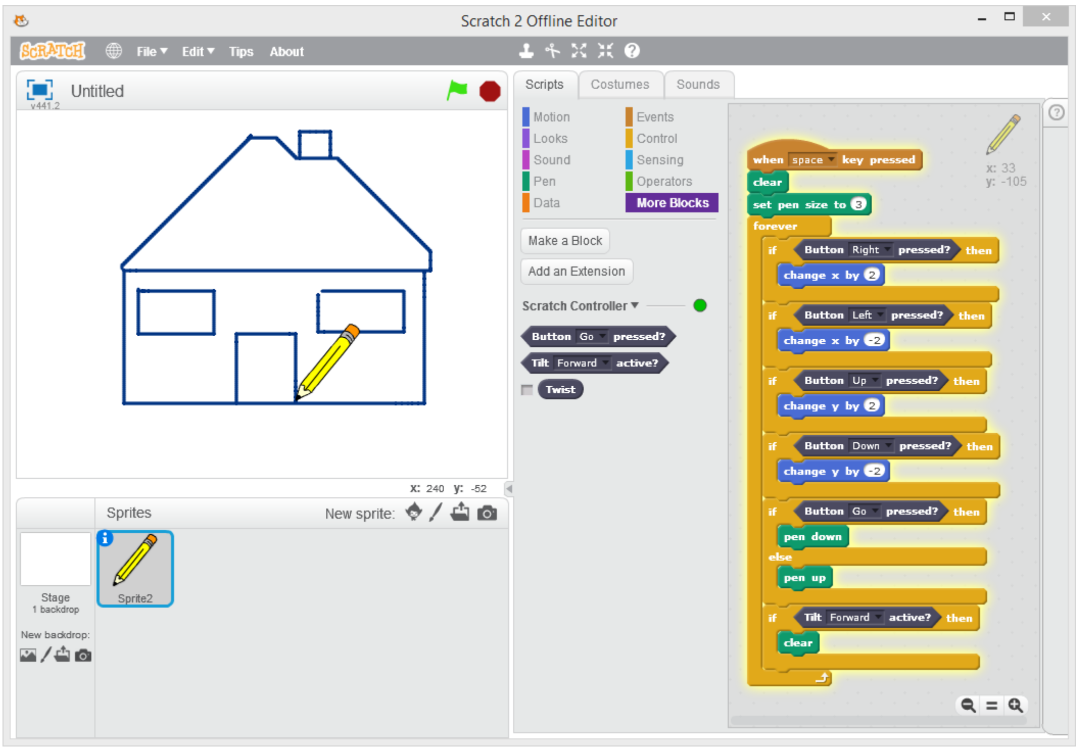
```
if Button Go pressed? then
    pen down
else
    pen up
```

This needs to be placed inside the forever loop. When the 'Go' button is held down the pen will draw.

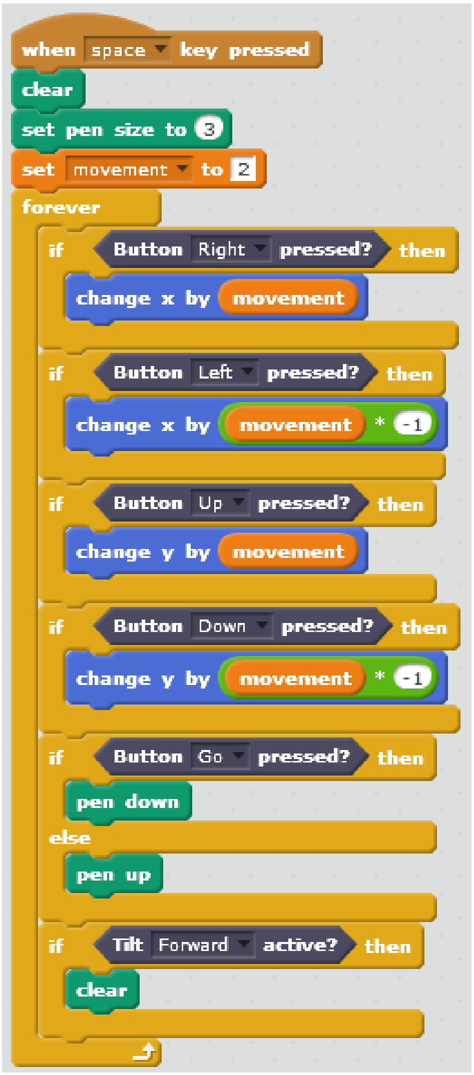
Finally a mechanism to clear the screen can be added.
Again this needs to be placed inside the forever loop.



```
if Tilt Forward active? then
    clear
```



```
when space key pressed
clear
set pen size to 3
forever
    if Button Right pressed? then
        change x by 2
    if Button Left pressed? then
        change x by -2
    if Button Up pressed? then
        change y by 2
    if Button Down pressed? then
        change y by -2
    if Button Go pressed? then
        pen down
    else
        pen up
    if Tilt Forward active? then
        clear
```



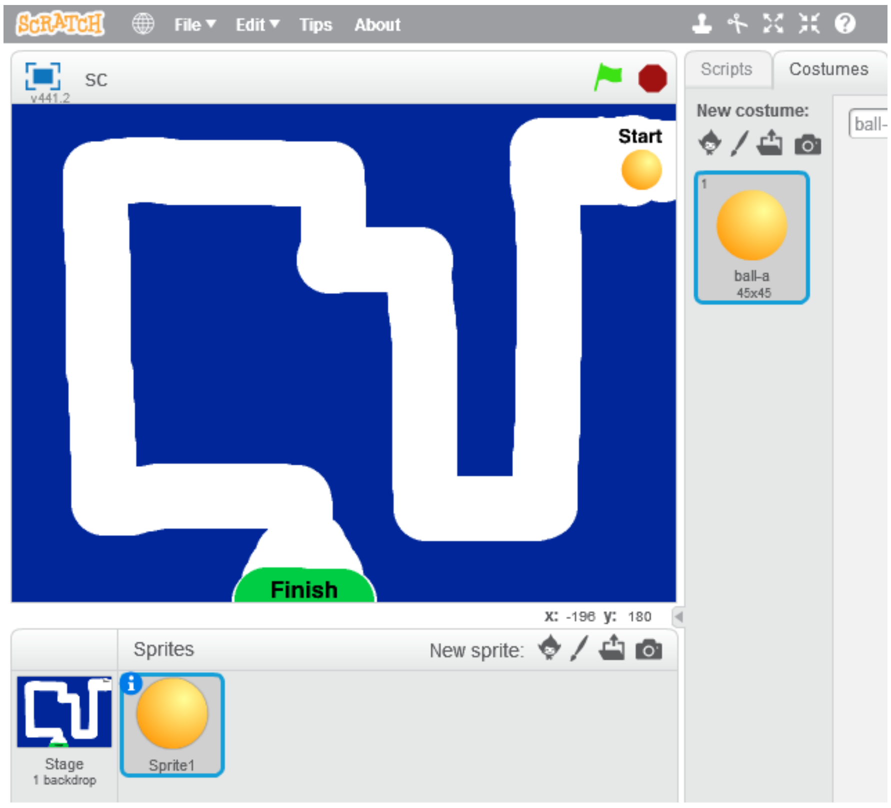
```
when space key pressed
clear
set pen size to 3
set movement to 2
forever
    if Button Right pressed? then
        change x by movement
    if Button Left pressed? then
        change x by movement * -1
    if Button Up pressed? then
        change y by movement
    if Button Down pressed? then
        change y by movement * -1
    if Button Go pressed? then
        pen down
    else
        pen up
    if Tilt Forward active? then
        clear
```

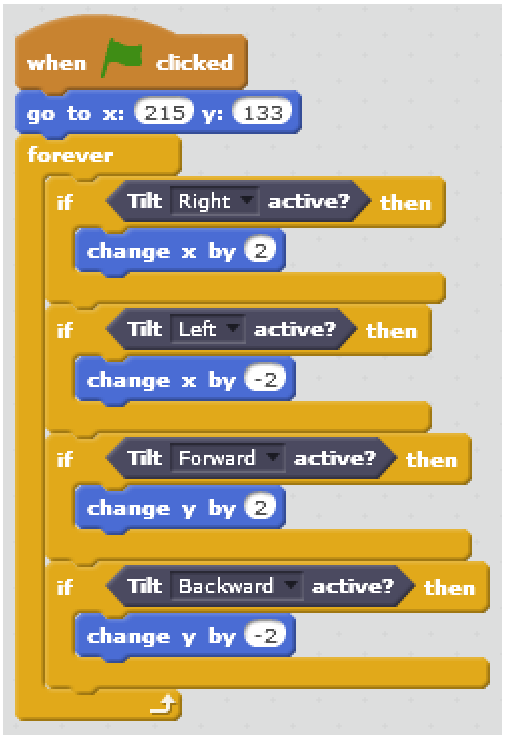
There are plenty of opportunities to extend this idea further, for example, to change the pen thickness or colour. The control of the pen can also be refined. In the example x and y is set to change by +/- 2 rather than +/-10. This movement could be set as a variable and then altered once rather than 4 times.

N.B. 'movement' x -1 turns movement into a negative number.

## Key Focus 4 - Using multiple controls 2 – Tilt Maze

The first task in this activity is to create a maze on the stage, as well as an object (sprite) to move through it. The maze needs a start and an end point, marked with a different colour. The maze must only be two colours (path and walls).





```
when [green flag] clicked
go to x: (215) y: (133)
forever
    if <Tilt [Right] active?> then
        change x by (2)
    if <Tilt [Left] active?> then
        change x by (-2)
    if <Tilt [Forward] active?> then
        change y by (2)
    if <Tilt [Backward] active?> then
        change y by (-2)
```

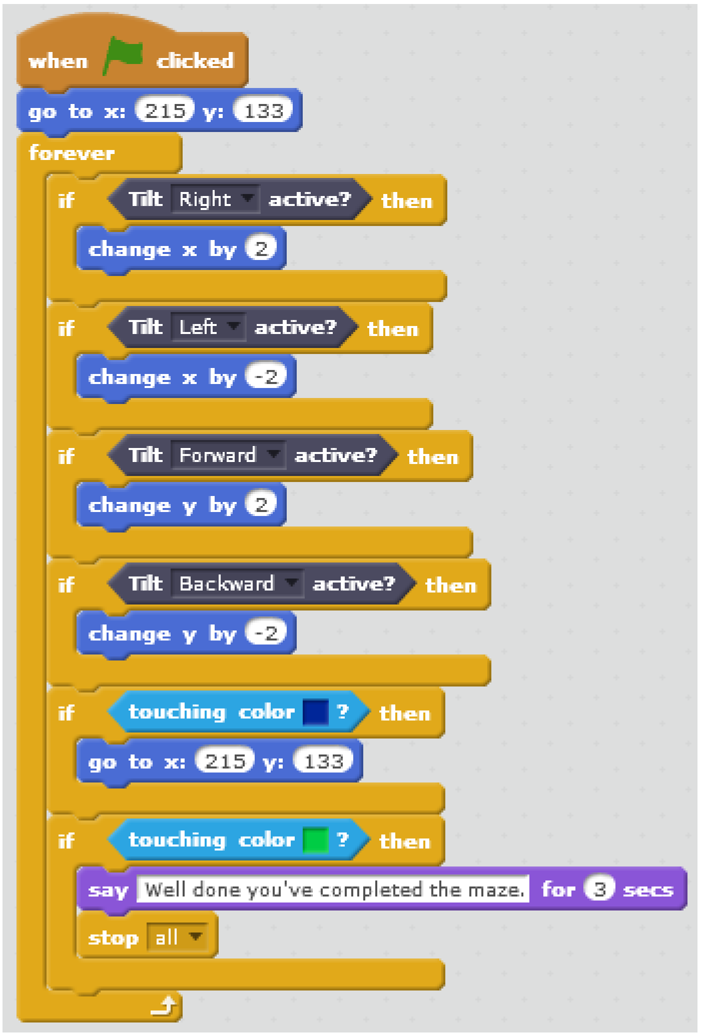
Once the stage has been set up with the sprite, blocks need adding to the sprite so it will move when the controller is tilted.

Two checks then need to be added. One check is to see if the user has reached the finish point and another is to see if they've bumped into a wall. These checks could be 'if … then' and Sensing blocks.

The program will now check to see if a wall has been bumped. If it has, then the ball returns to the start. When a player reaches the finish line a message will come up to say they have completed it.

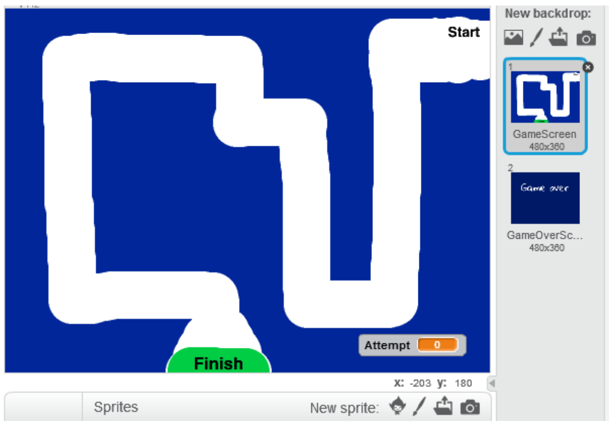
There are several ways the game could be adapted/developed:

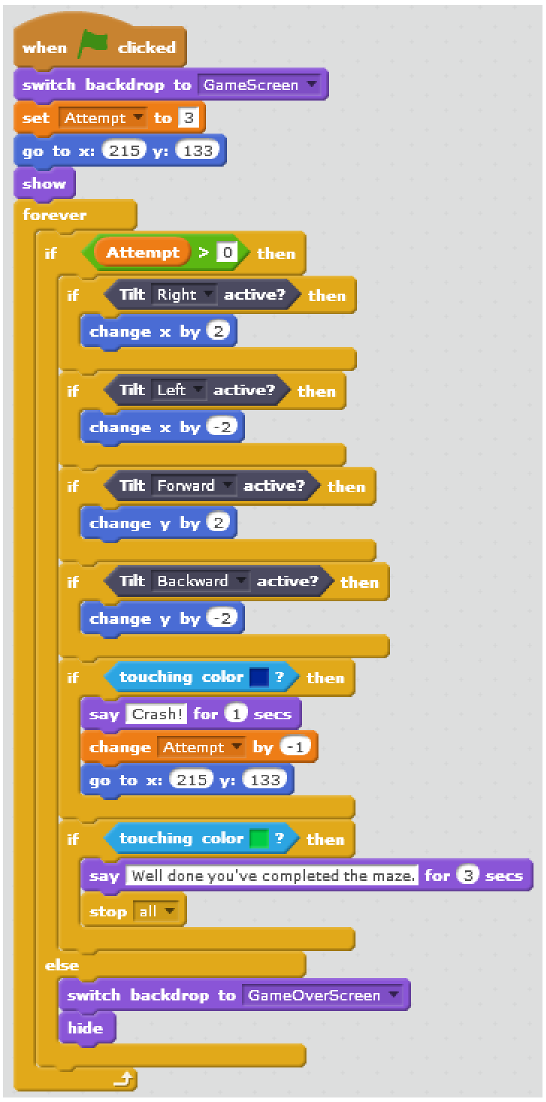
• Change the maze to make it harder or easier
• Make the ball (sprite) larger or smaller to make the challenge easier or harder
• Limit the number of attempts a player has
• Change how far/fast the ball moves when the control is tilted
• Allow the player to choose a difficulty level and set the ball speed from that choice
• Add a start trigger, i.e. the user has to press the 'Go' button to start each attempt



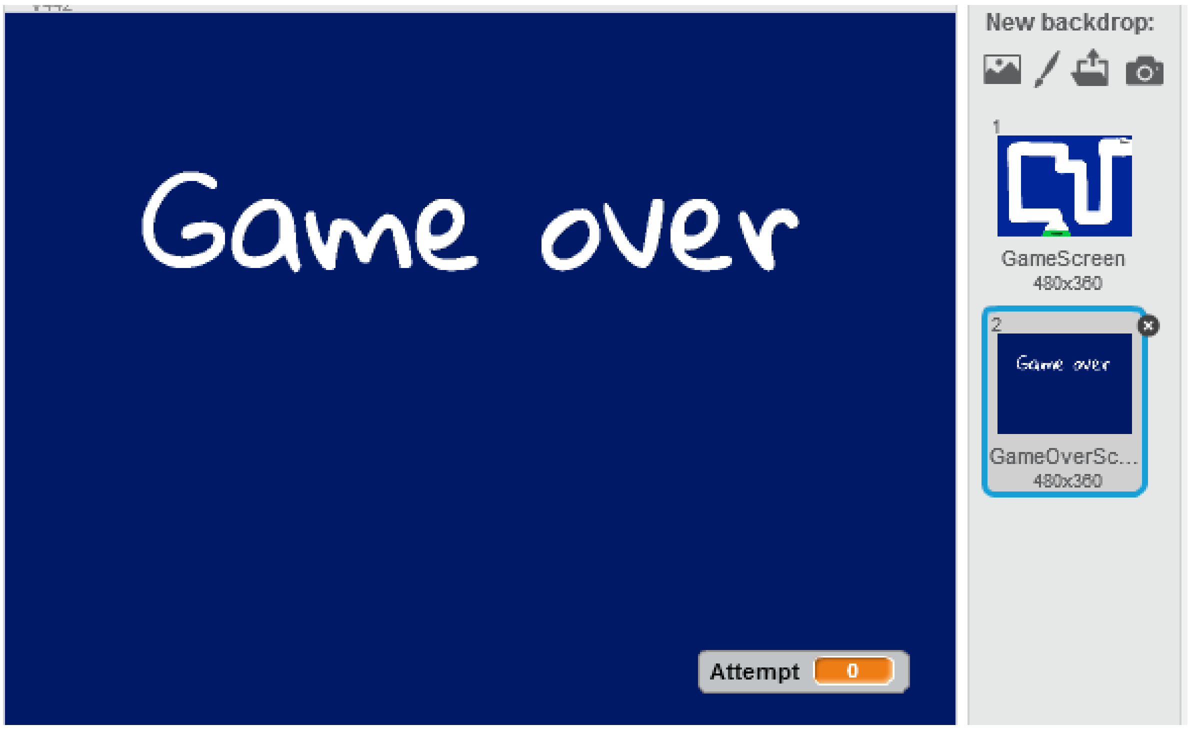
```
when [flag] clicked
go to x: 215 y: 133
forever
    if <Tilt Right active?> then
        change x by 2
    if <Tilt Left active?> then
        change x by -2
    if <Tilt Forward active?> then
        change y by 2
    if <Tilt Backward active?> then
        change y by -2
    if <touching color 🟦 ?> then
        go to x: 215 y: 133
    if <touching color 🟩 ?> then
        say Well done you've completed the maze. for 3 secs
        stop all
```

The example below limits the player to 3 attempts.





```
when 🏁 clicked
switch backdrop to GameScreen
set Attempt to 3
go to x: 215 y: 133
show
forever
    if Attempt > 0 then
        if Tilt Right active? then
            change x by 2
        if Tilt Left active? then
            change x by -2
        if Tilt Forward active? then
            change y by 2
        if Tilt Backward active? then
            change y by -2
        if touching color ? then
            say Crash! for 1 secs
            change Attempt by -1
            go to x: 215 y: 133
        if touching color ? then
            say Well done you've completed the maze. for 3 secs
            stop all
    else
        switch backdrop to GameOverScreen
        hide
```

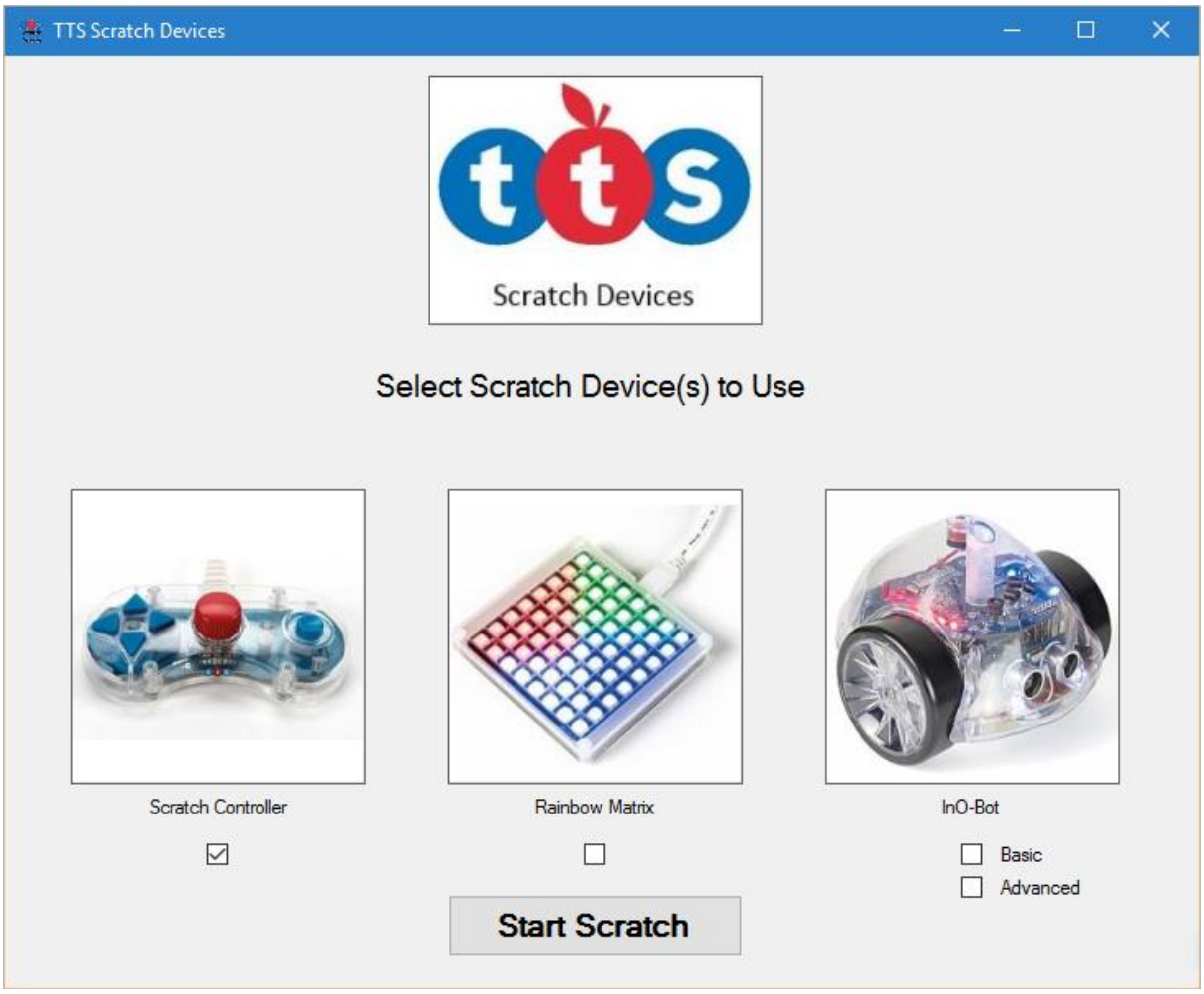


## Getting Started with the TTS Scratch Controller

The TTS Scratch Controller is an ideal tool to help develop children's understanding of computer input.

### Getting Started

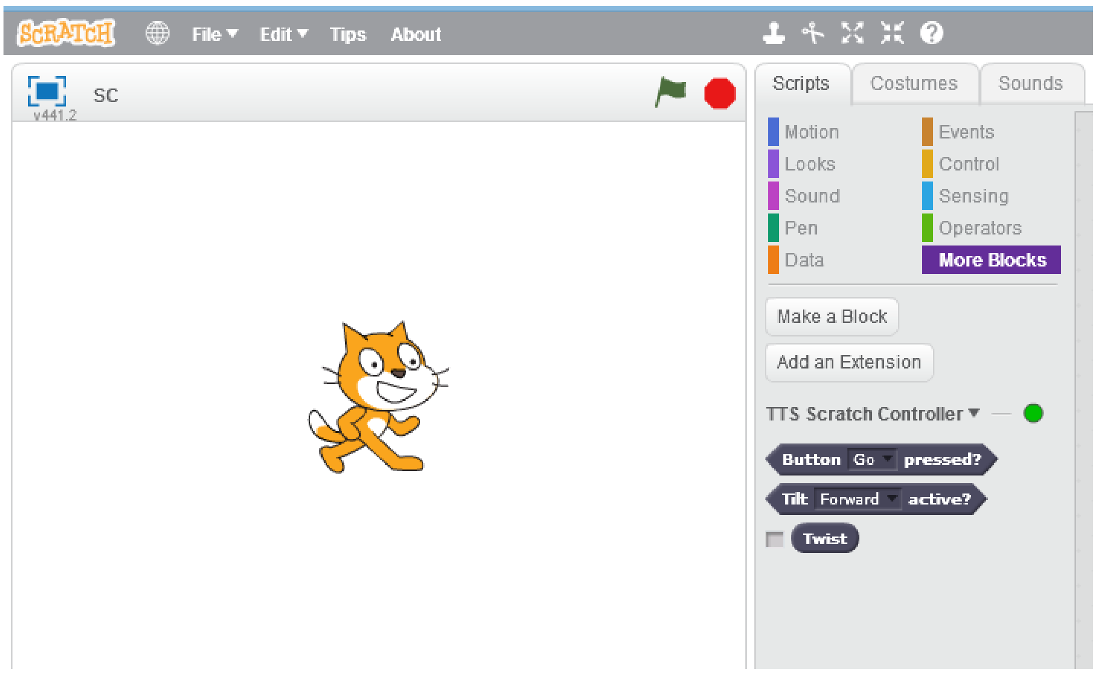
Please see the user guide supplied with the product for information on connecting the TTS Scratch Controller to a PC and installing the software.
Once installation is complete double click on the TTS Scratch Devices icon to open the Launcher.



The Launcher should automatically detect that a Scratch Controller is connected and tick the box underneath. If a controller isn't detected the box can be ticked manually. This will make the additional block appear in Scratch. NB This will not make an undetected controller work. Click on 'Start Scratch' to begin programming.

## Scratch

When Scratch opens there will be three additional blocks available in 'More Blocks'.

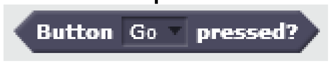


There is also an indicator (green dot) to show if the Scratch Controller is detected and available.

## Scratch Blocks

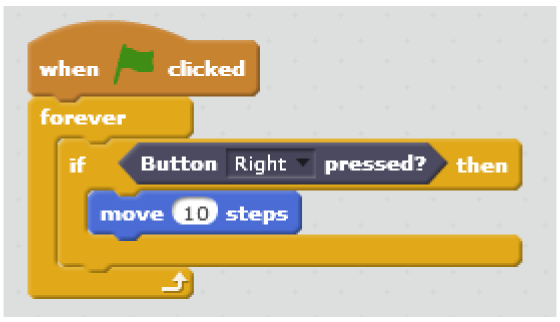
Three special blocks can be found in the 'more blocks' section of Scratch.

### Button … pressed?



```
Button [Go] pressed?
```

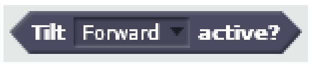
This block can be used to check if one of the five buttons (Up, Down, Left, Right or Go) on the Scratch Controller is pressed. For example:



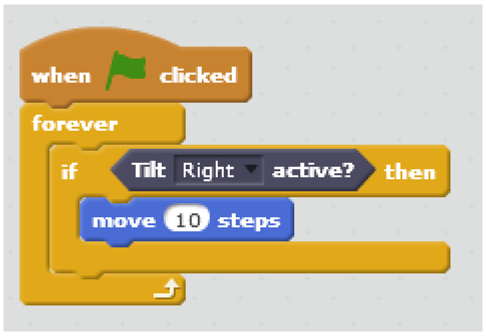
```
when [flag] clicked
forever
    if Button [Right] pressed? then
        move (10) steps
```

In the example above, pressing the right controller button will make the sprite move 10 steps. Note that the block is used in an 'if' statement and has to be in a loop to be checked more than once. The drop down option on the 'Button … pressed?' block allows one of the five different buttons to be selected.

## Tilt … active?

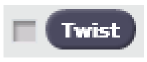


This block can be used to check if the Scratch Controller has been tilted (Forward, Backward, Left or Right). For example:

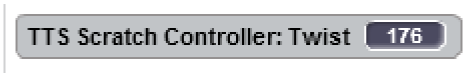

```
when [flag] clicked
forever
    if <Tilt [Right] active?> then
        move (10) steps
```

In the example above, tilting the controller right will make the sprite move 10 steps. Note that the block is used in an 'if' statement and **has to be** in a loop to be checked more than once. The drop down option on the 'Tilt … active?' block allows one of the four directions to be selected.

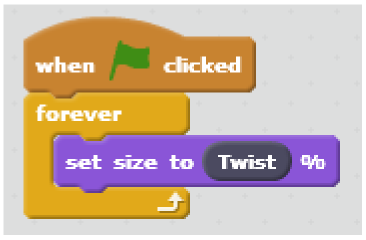
## Twist



This block is a variable and reports the position of the twist controller on the middle of the Scratch Controller (0 – 255). When the control is at its maximum anti-clockwise position, the value is 0. Its maximum clockwise position is 255. Ticking the box by the Twist variable will display its value on the Scratch stage.



An example of using the twist variable:


```
when [flag] clicked
forever
    set size to (Twist) %
```

This example will change the size of the sprite according to the position of the control. Again note the need for a forever loop to ensure the control's position is continually checked.

## Technical Support

Please visit www.tts-group.co.uk for the latest product information.

Email feedback@tts-group.co.uk for technical support.

TTS Group Ltd.
Park Lane Business Park,
Kirkby-in-Ashfield,
Nottinghamshire,
NG17 9GU, UK.

Freephone: 0800 318686
Freefax: 0800 137525