AN INTRODUCTION TO SCRATCH (2) PROGRAMMING

Document Version 2 (04/10/2014)

INTRODUCTION

SCRATCH is a visual programming environment and language. It was launched by the MIT Media Lab in 2007 in an effort to make coding accessible and appealing to everyone. SCRATCH is perfect for making animations, games, interactive stories and other visually rich programs. It provides a great introduction to programming for people of all ages.

Scratch is good because:

- it is a visually-engaging programming language
- the commands (known as blocks) are colour-coded according to their function
- you don't have to remember the commands – they are all on the screen in sections
- the commands fit together like jigsaw pieces
- the commands in Scratch simplify common activities in games (e.g. collision detection)
- there are hundreds of examples of SCRATCH projects online
- people can share their projects and, of course, use each other's ideas and code

VERSIONS of SCRATCH

Version 2 is only available for use online. An earlier (but not as comprehensive) version is available to download to your computer. There is also a simplified version for iPad known as *Scratch Junior.* You can access Scratch via:

http://scratch.mit.edu/

PROJECTS

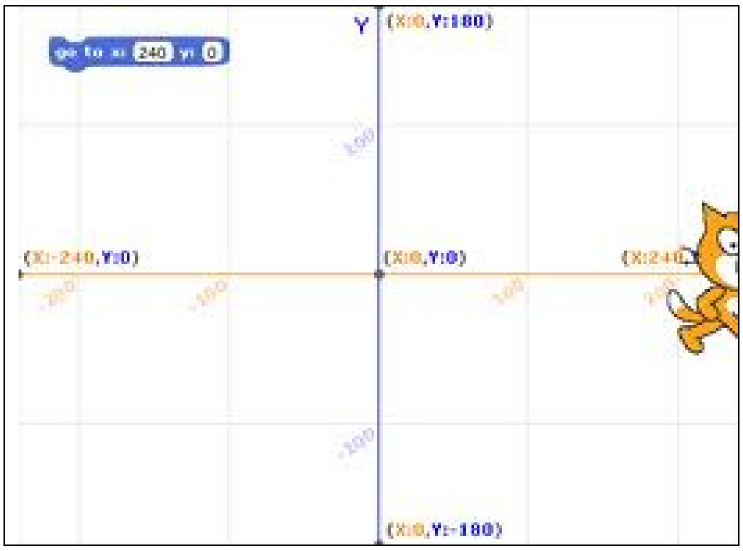
Your SCRATCH creations are called PROJECTS. You can save your projects to your local computer or online if you create a SCRATCH account. Once created, your programming environment has a number of areas:

- the STAGE (where all the action happens)
- the SPRITE LIST (where all your SPRITES are listed)
- the BLOCKS PALLETTE (where all the commands are grouped and listed)
- the SCRIPTS AREA (where you assemble your blocks) and create your programs

THE STAGE

The stage is the area of the screen where all the action happens and you see the effects you have created in the scripts. The STAGE has ...

- 360 pixels vertically - the 'y' axis has co-ordinates y=180 (top) to y=-180 (bottom)
- 480 pixels horizontally - the 'x' axis has co-ordinates x=-240 (left) to x=240 (right)
- A centre position of x=0, y=0



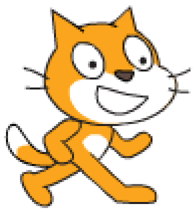
The STAGE therefore has a width of 480 pixels and a height on 360 pixels.

Directions

90° is FACING RIGHT
-90° is FACING LEFT
0° is FACING UP
180° is FACING down

SPRITES

SCRATCH works on the basis that there are a number of objects or characters (SPRITES) on the screen. These SPRITES can be moved around the STAGE, be programmed to sense things (e.g. when touching another SPRITE) or have a number of 'looks' or costumes (you can change how a SPRITE appears during the running of script). You can create your own sprites but there are many already available within SCRATCH.



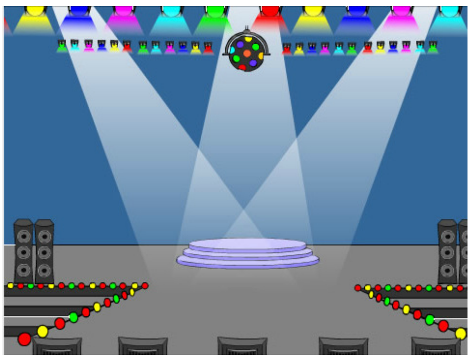
The orange cat is the default (and now famous) SPRITE. You can add SPRITES to your project by selecting from the long list of SPRITES available across a number of themes (sports, transport, people, etc.)

COSTUMES

Once you have created or selected a SPRITE, you can create different versions of that sprite to be used for different effects. These different versions of a SPRITE are called COSTUMES. A COSTUME of a sprite may simply be of a different colour or face the opposite direction.

BACKDROPS

BACKDROPS are simply background environments for your SCRATCH project. You do not have to use a background but they are useful if you want to create a particular scene for your animation or game, e.g. a haunted house or a dance floor.

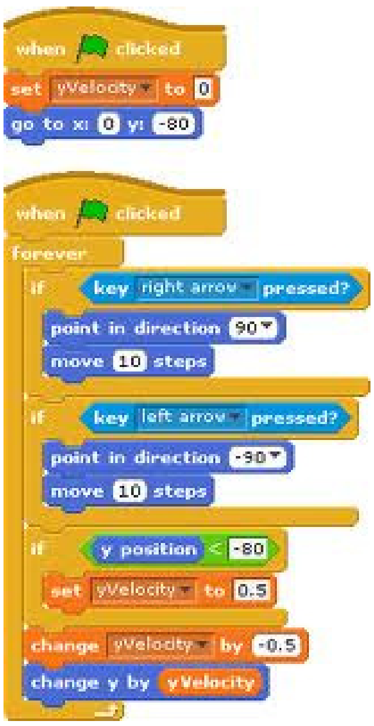


You can select a number of BACKDROPS for your animation (or interactive story) and switch between them during your program.

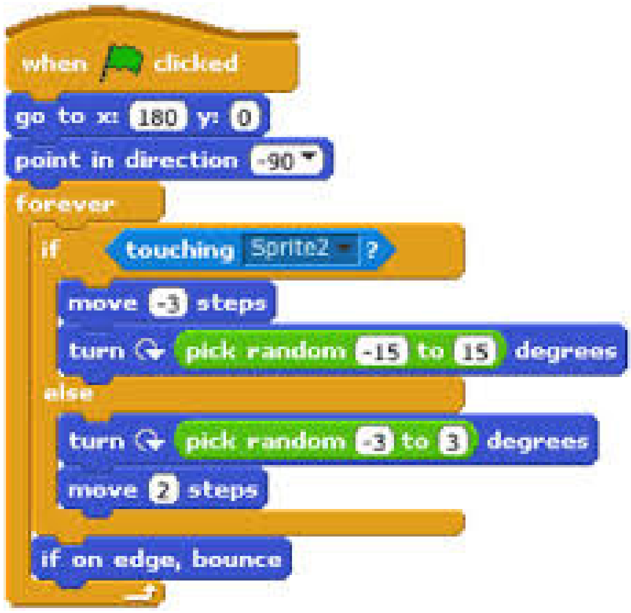
SCRIPTS

A SCRIPT is essentially the set of instructions (the program) that you write for each SPRITE in your project. SCRIPTS control the movement and actions of your SPRITES and all of the other functions of your programming project. Each SPRITE has its own SCRIPT.

Here are some SCRATCH SCRIPTs.

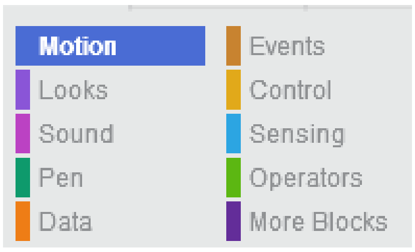


```
when  clicked
set yVelocity to 0
go to x: 0 y: -80

when  clicked
forever
    if key right arrow pressed?
        point in direction 90
        move 10 steps
    if key left arrow pressed?
        point in direction -90
        move 10 steps
    if y position < -80
        set yVelocity to 0.5
    change yVelocity by -0.5
    change y by yVelocity
```



```
when  clicked
go to x: 180 y: 0
point in direction -90
forever
    if touching Sprite2 ?
        move -3 steps
        turn pick random -15 to 15 degrees
    else
        turn pick random -3 to 3 degrees
        move 2 steps
    if on edge, bounce
```

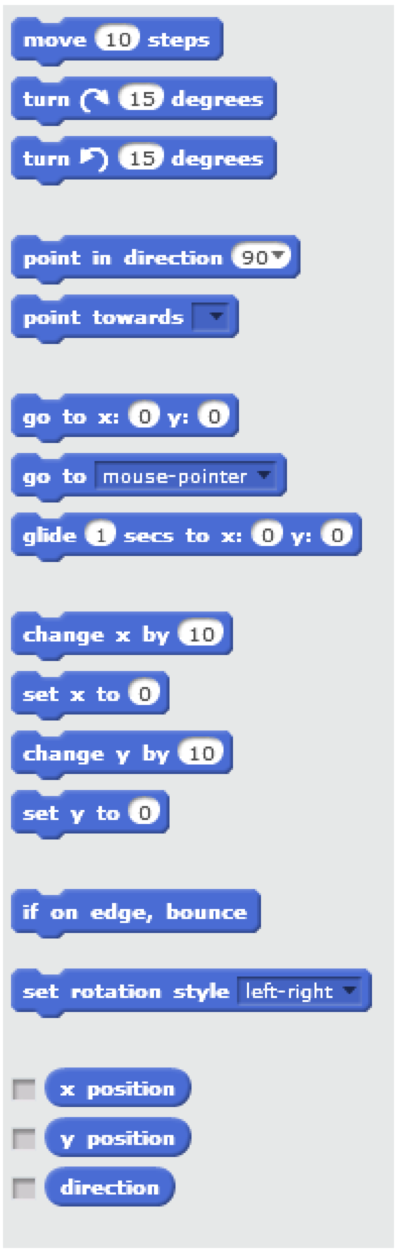
COMMAND GROUPS



There are many COMMANDS (blocks) in SCRATCH. They all have different functions. They are grouped into 10 different functional areas, all of which their own colour. The DARK BLUE blocks, for example, control the MOVEMENT and DIRECTION of sprites on the screen.

SUMMARY OF COMMANDS / BLOCKS

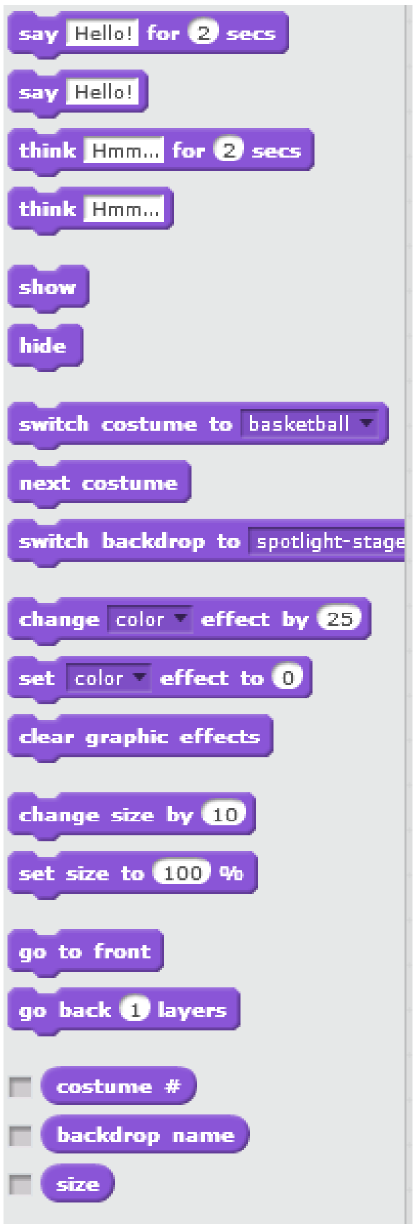
MOTION



```
move (10) steps
turn ↻ (15) degrees
turn ↺ (15) degrees

point in direction (90 ▾)
point towards [ ▾]

go to x: (0) y: (0)
go to [mouse-pointer ▾]
glide (1) secs to x: (0) y: (0)

change x by (10)
set x to (0)
change y by (10)
set y to (0)

if on edge, bounce

set rotation style [left-right ▾]

☐ (x position)
☐ (y position)
☐ (direction)
```

These commands are used to change the position of sprites and move sprites around the STAGE area. The MOVE command will move a sprite instantaneously. The GLIDE command is useful for animations as it can determine the speed at which a sprite moves (gives the impression of floating) across the screen.

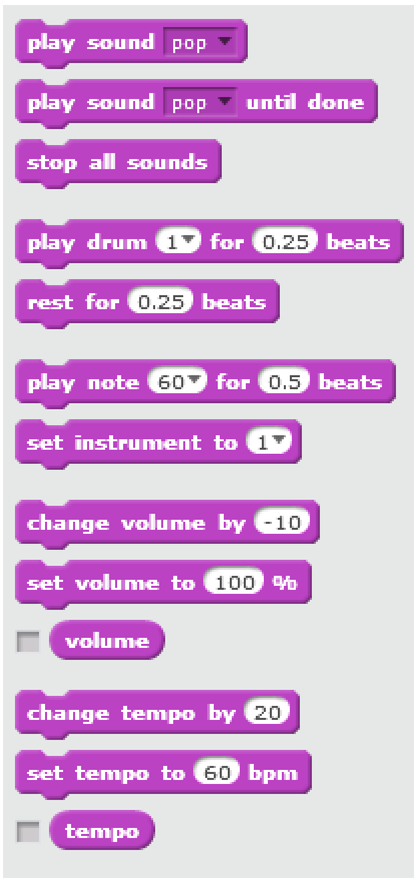
LOOKS



```
say [Hello!] for (2) secs
say [Hello!]
think [Hmm...] for (2) secs
think [Hmm...]

show
hide

switch costume to [basketball ▾]
next costume
switch backdrop to [spotlight-stage

change [color ▾] effect by (25)
set [color ▾] effect to (0)
clear graphic effects

change size by (10)
set size to (100) %

go to front
go back (1) layers

☐ (costume #)
☐ (backdrop name)
☐ (size)
```

These commands are used to change the appearance of sprites on the stage area. You can make a sprite disappear (**hide**) and reappear (**unhide**). You can also give sprites speech and thought bubbles. You can switch between the various versions (costumes) of a sprite, if you have created them. You can change the size of a sprite and bring it to the foreground if it is positioned behind another sprite.

SOUND



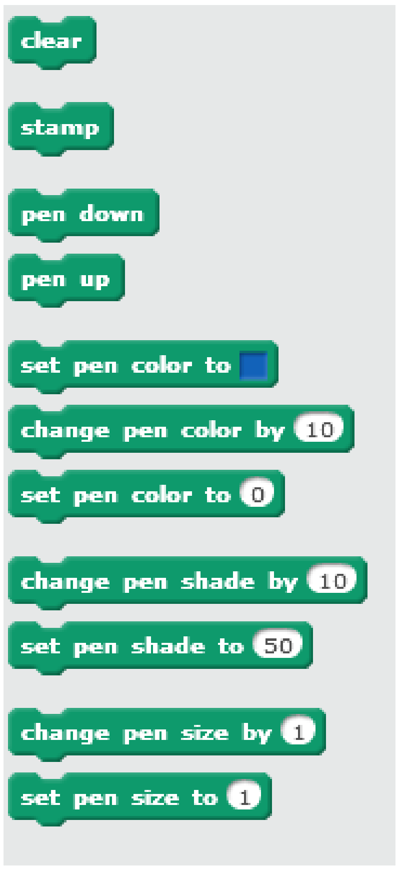
```
play sound [pop ▾]
play sound [pop ▾] until done
stop all sounds

play drum (1 ▾) for (0.25) beats
rest for (0.25) beats

play note (60 ▾) for (0.5) beats
set instrument to (1 ▾)

change volume by (-10)
set volume to (100) %
☐ (volume)

change tempo by (20)
set tempo to (60) bpm
☐ (tempo)
```

These commands are used to add sound and music to your scripts. You can choose from a range of instrumental sounds and sound effects. You can also use the PLAY NOTE command to play different musical notes (chosen from a keyboard).

You can RECORD a sound using the microphone and use it in your script. This could the sound of your own voice asking a question or a sound effect for an interactive story.

SUMMARY OF COMMANDS / BLOCKS

PEN

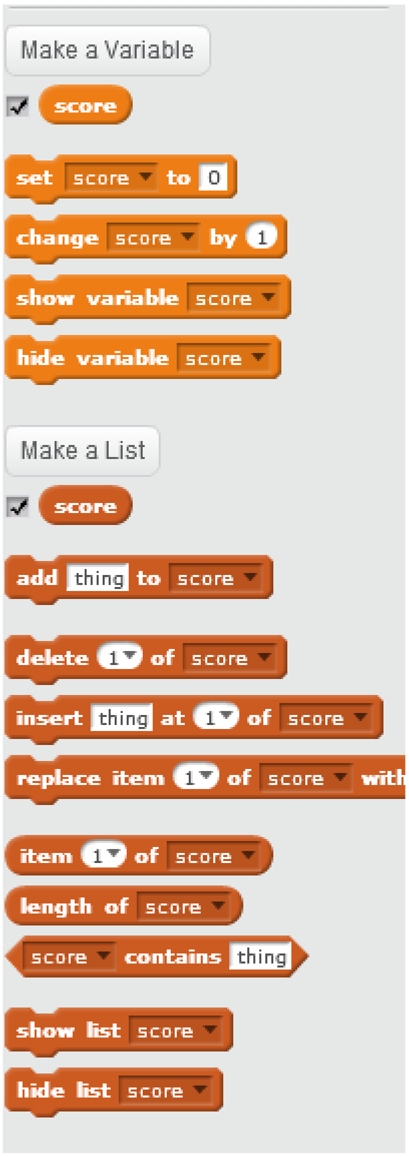


```
clear
stamp
pen down
pen up
set pen color to
change pen color by 10
set pen color to 0
change pen shade by 10
set pen shade to 50
change pen size by 1
set pen size to 1
```

These commands are used to draw lines using what is effectively a PEN. The pen in SCRATCH enables a sprite to draw a line as it moves around the stage. You can set the size and colour of the pen.

The STAMP command prints a copy of the sprite on the stage so that when the sprite moves on, it will leave a picture of itself behind. A STAMP is just a picture of the sprite, not a copy of the sprite which you can then control.

DATA

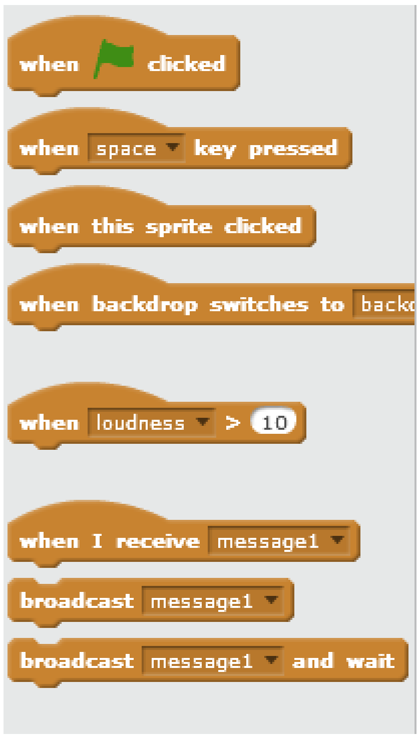


```
Make a Variable
score
set score to 0
change score by 1
show variable score
hide variable score
Make a List
score
add thing to score
delete 1 of score
insert thing at 1 of score
replace item 1 of score with
item 1 of score
length of score
score contains thing
show list score
hide list score
```

These commands are used to establish and change **variables** (values) in your script. These are useful when you want to count something (like a score in a game) or hold or do something with data entered by the user.

You can also create lists which can be used in lots of ways. For example, you might create a list of possible answers to a quiz.

EVENTS



```
when clicked
when space key pressed
when this sprite clicked
when backdrop switches to backd
when loudness > 10
when I receive message1
broadcast message1
broadcast message1 and wait
```

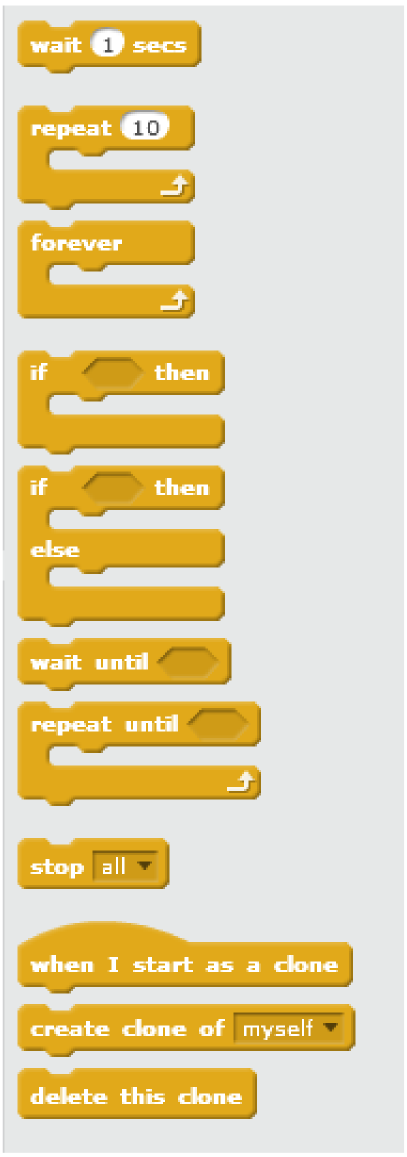
These commands are used to control when a script or part of script starts to run.

Most scripts should start with the GREEN FLAG. However, you can also instruct a script to start when you press a certain key. You could use this, for example, to move a SPRITE using the arrow keys.

SPRITES can also send or broadcast messages (simple words) to other sprites. If you use the ***when I receive*** block, a sprite might wait do run part of its script until it receives a message (a trigger) from another sprite. These triggers are important – it is the only way that one script can start another script or the action of one sprite influence the others in the stage area.

SUMMARY OF COMMANDS / BLOCKS

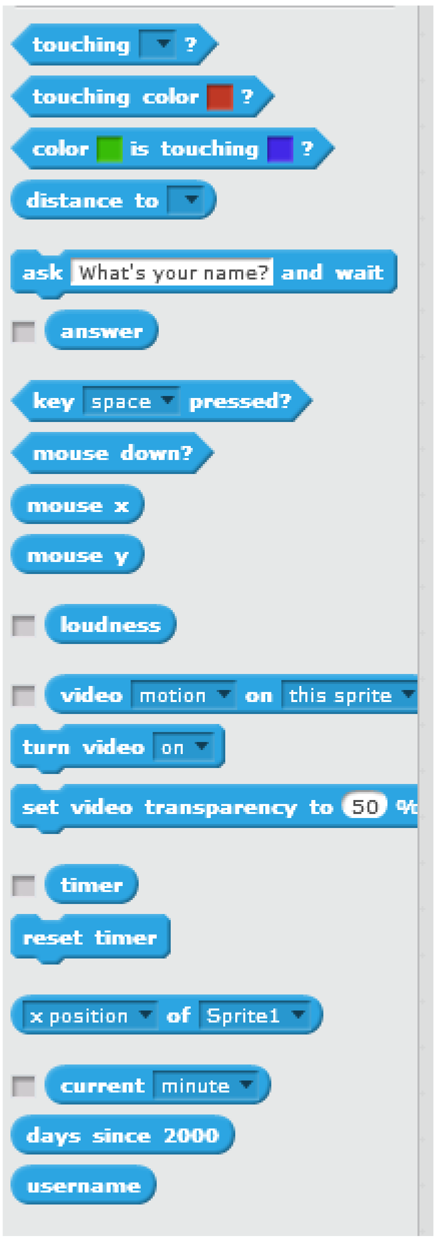
CONTROL



Control commands are really important because they enable parts of a script to be **repeated** over and over again, for a set number of times or until a certain condition is met before the script moves on. The *if … then … else* commands are used for selection so that parts of a script run only if certain conditions are met. The WAIT command is useful in building in pauses into an animation.

It is these blocks that really teach use how to use **repetition** and **selection** in our programs.

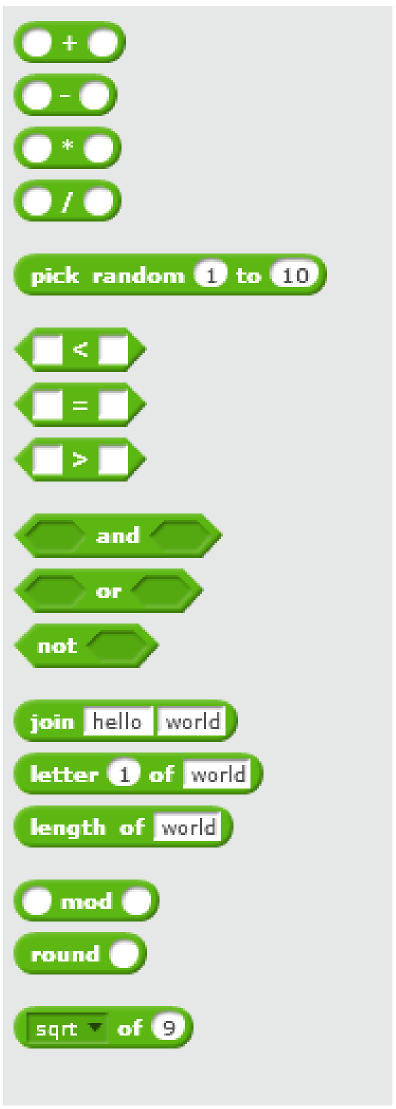
SENSING



These commands are useful in creating games and scripts where the collision of a sprite against a colour or another sprite triggers an event. For example, you would use the *touching* block to determine if a sprite (a ball, for example) was touching a wall or a paddle in a game. This might trigger a bounce or a loss of life.

You can use the ***ask*** command to promote input from the user. You can also make use of a webcam in your scripts.

OPERATORS



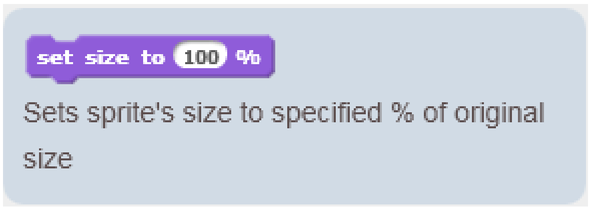
These commands are used to manipulate numbers and perform basic number operations.

The ***Pick Random*** block is useful when you want the computer to generate a random number, for example, the roll of a dice or the picking of a card.

These blocks are used in combination with the CONTROL blocks (**repeat** and ***If … then … else***) blocks to test and compare the values and conditions which trigger parts of the script to run or end a repeat.

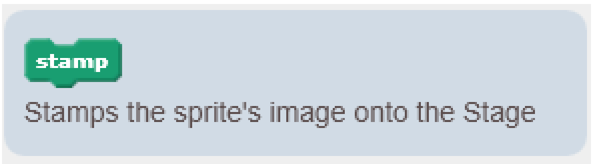
HELP FOR BLOCKS

The SCRATCH interface is wonderful is that it provides HELP for every building block and explains how it might be used. Here are some examples:



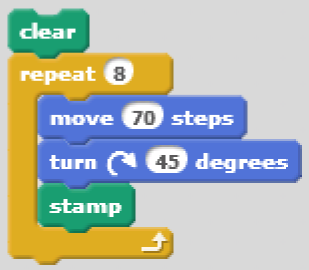
```
set size to (100) %
```

Sets sprite's size to specified % of original size

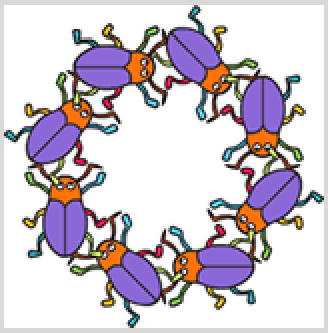


```
stamp
```

Stamps the sprite's image onto the Stage

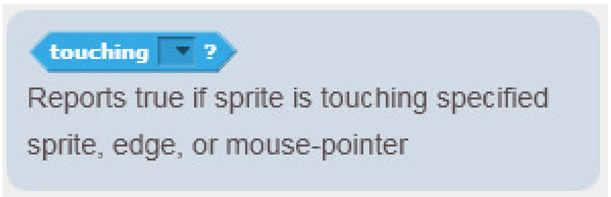


```
clear
repeat (8)
    move (70) steps
    turn ↻ (45) degrees
    stamp
```



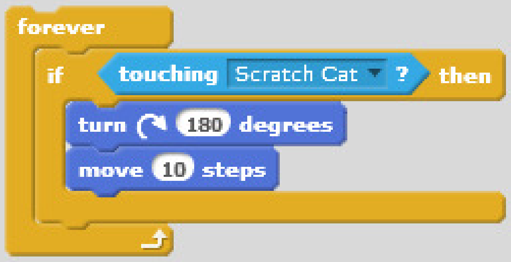
A stamp is only a temporary image of the sprite that is drawn on the Stage. It can be drawn over or cleared using the clear block.

A stamp cannot move and cannot hold scripts.

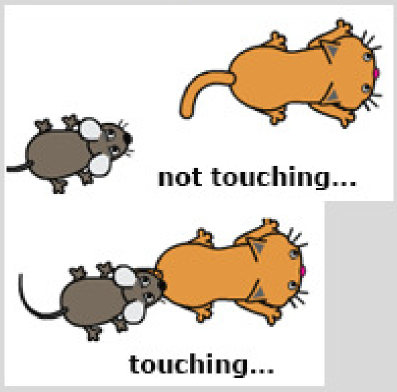


```
<touching [ ▾] ?>
```

Reports true if sprite is touching specified sprite, edge, or mouse-pointer

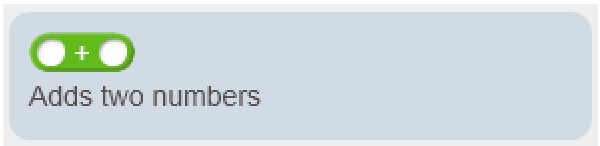


```
forever
    if <touching [Scratch Cat ▾] ?> then
        turn ↻ (180) degrees
        move (10) steps
```



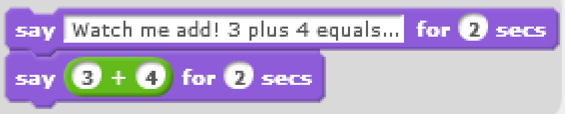
You can use the `<touching [ ▾] ?>` block in three different ways. Select from the pull-down menu to choose. You can check if the sprite is:

```
<touching [mouse-pointer ▾] ?>
<touching [edge ▾] ?>
<touching [Sprite17 ▾] ?>
```

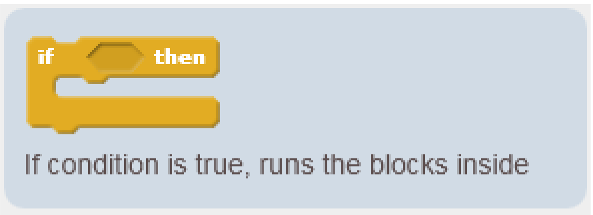


```
(( ) + ( ))
```

Adds two numbers

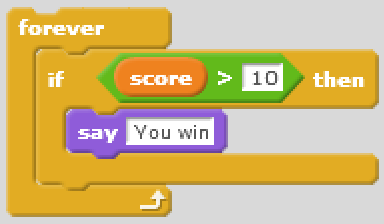


```
say [Watch me add! 3 plus 4 equals...] for (2) secs
say ((3) + (4)) for (2) secs
```



```
if < > then
```

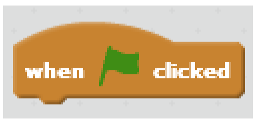
If condition is true, runs the blocks inside



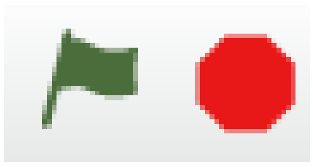
```
forever
    if <(score) > [10]> then
        say [You win]
```

STARTING YOUR SCRIPTS

You can write a number of scripts to control sprites and events in your project. If you want to start all of your scripts at the same time, you will need to place the follow BLOCK at the beginning of your scripts.



You START and STOP your programs by using the following buttons, placed at the top right of the STAGE area. When you press the GREEN FLAG to start, all of your scripts which have the above EVENT block will start and will interact (if they have been told to) with each other. When you press the RED HEXAGON, all scripts will stop and your program will end.



COMMON PITFALLS

It is important to note that every time you START your SCRIPTs, SCRATCH always uses the settings, positions, colours, directions and other values that were in place at the end of the previous running of the script. You may need to build in the initial values you want (starting position of a sprite for example) for your script right at the beginning of the program. For example …

HIDDEN SPRITES

If at the END of a script you HIDE the SPRITE, then the SPRITE will remain HIDDEN when the script is run again unless you include a command to UNHIDE it.

PEN UP / PEN DOWN

If the PEN is DOWN at the end of the script run, the PEN will also be down when you start it again (and will draw every move) unless you include the command PEN UP.

COLOUR

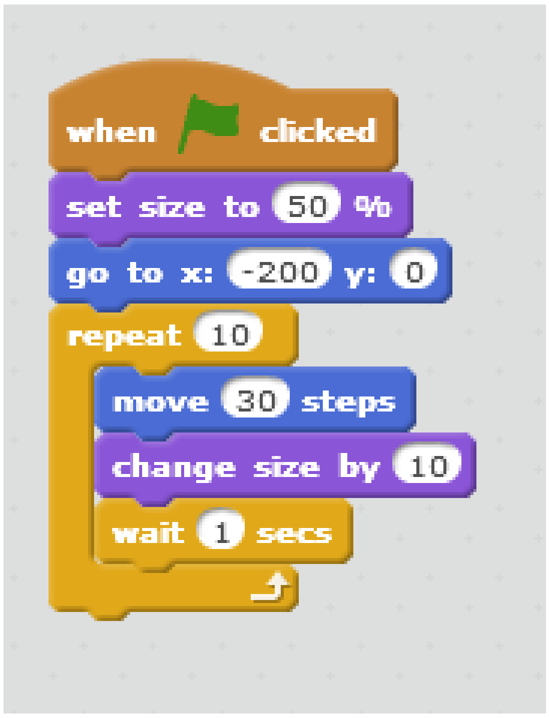
Any colours that were set at the end of the previous script run will remain at the beginning of the next script run unless you define the colours you want at the beginning of your script.

POSITION / DIRECTION

The POSITION or DIRECTION (which way a sprite is facing) at the end of a script will be the starting position or direction for the sprite the next time the script is run. If you want to start the sprite at a certain position and facing a certain way, you need to use the appropriate commands to establish this at the beginning of the script.

EXAMPLE SCRIPTS

EXAMPLE 1



```
when [flag] clicked
set size to 50 %
go to x: -200 y: 0
repeat 10
    move 30 steps
    change size by 10
    wait 1 secs
```

The script performs the following actions when the GREEN FLAG is CLICKED.

1 Reduces the size of the sprite to 50%

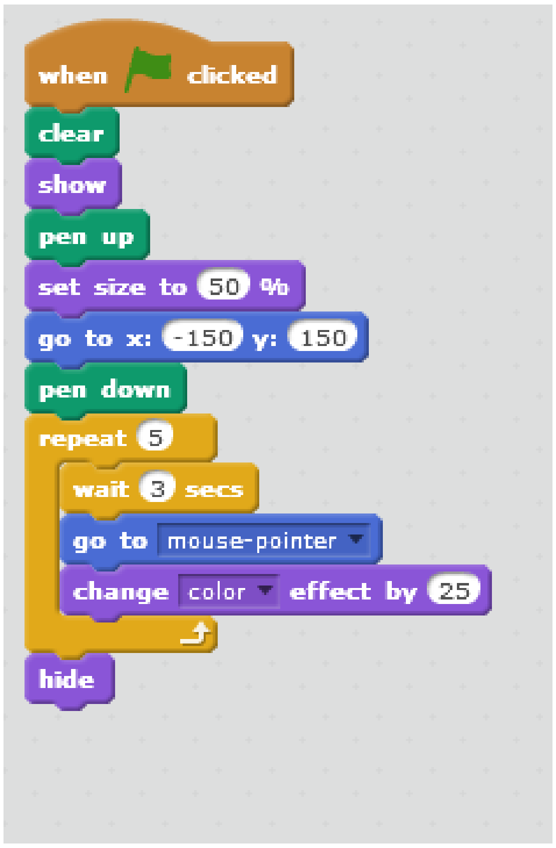
2 Moves the sprite to position -200,0

3 Repeat the following 5 times

Move 30 steps to the right
Increase size of sprite by 10
Wait 1 second

EXAMPLE 2

Place a SPRITE at the top left of the screen. After pauses of 3 seconds, move the sprite to the position of the MOUSE POINTER and change the colour effect of the SPRITE each time. After 5 lines drawn, stop and hide the SPRITE.



```
when [flag] clicked
clear
show
pen up
set size to 50 %
go to x: -150 y: 150
pen down
repeat 5
    wait 3 secs
    go to mouse-pointer
    change color effect by 25
hide
```

The script performs the following actions when the GREEN FLAG is CLICKED.

1 Clears the screen

2 Show the SPRITE (it is hidden at the end of script)

3 Set the PEN UP (it is PEN DOWN at the end script)

4 Reduces the size of the sprite to 50%

5 Move the sprite to position -150,150

6 Repeat the following 5 times

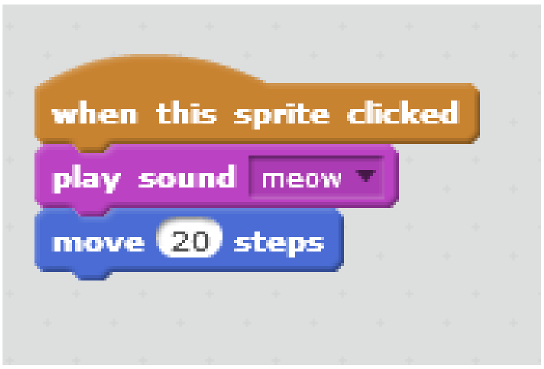
Wait for 3 seconds
Sprite moves to position of mouse-pointer
Sprite changes colour effect

7 Sprite is hidden so shape drawn is not obscured.

EXAMPLE SCRIPTS

EXAMPLE 3 Move a SPRITE 20 steps to the right and make the "MEOW" sound every time it is clicked.

VERSION 1

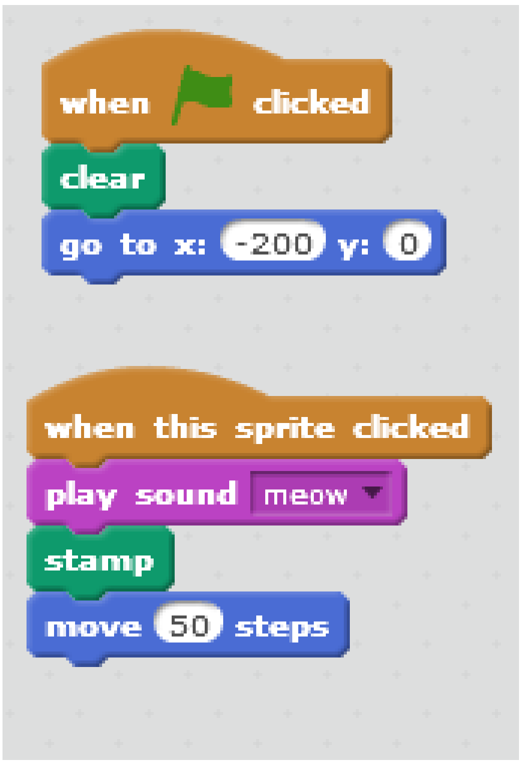


The script performs the following actions when the SPRITE is CLICKED.

1 Plays the MEOW sound.

2 Moves the SPRITE 20 steps to the right.

VERSION 2



The script performs the following actions when the GREEN FLAG is CLICKED

1 The screen is CLEARED

2 The SPRITE is positioned at -200,0

The script performs the following actions when the SPRITE is CLICKED.

1 Plays the MEOW sound.

2 Leaves a STAMP (copy) of itself behind.

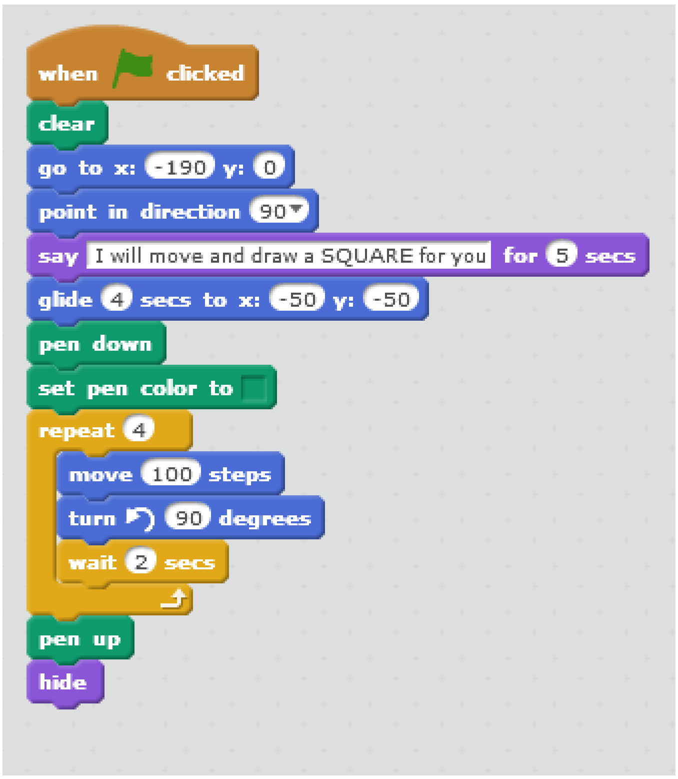
3 Moves the SPRITE 50 steps to the right.

Note that when the SPRITE is STAMPED, you cannot click on the copy of the sprite. Stamps are only copies of a picture – they are not sprites themselves and therefore have no actions.

THIS SCRIPT has TWO PARTS, one of which is actioned when the GREEN FLAG is clicked and one of which is actioned when the SPRITE is clicked.

EXAMPLE SCRIPTS

EXAMPLE 4 MOVE A SPRITE from a START position to DRAW A SQUARE near the centre of the screen. Hide the SPRITE so that it does not obscure the square once it has been drawn.



```
when 🏁 clicked
clear
go to x: -190 y: 0
point in direction 90▾
say I will move and draw a SQUARE for you for 5 secs
glide 4 secs to x: -50 y: -50
pen down
set pen color to 🟩
repeat 4
    move 100 steps
    turn ↺ 90 degrees
    wait 2 secs
pen up
hide
```

This script looks complicated but it really isn't. The script performs the following actions when the GREEN FLAG is pressed:

1 Clears the STAGE

2 Positions the SPRITE at position -190,0

3 Points the SPRITE to the right 90° (right) so that it moves in that direction when told to do so by a MOVE command

4 Creates a speech bubble with the words *"I will move and draw and SQUARE for you"* (for 5 seconds)

5 Glides the SPRITE to position -50,-50 over a period of 4 seconds (a gentle movement)

6 Sets the PEN DOWN (so that all future movement in drawn)

7 Sets the PEN colour to GREEN

8 Repeats the following actions FOUR times:

 The SPRITE moves 100 steps
 The SPRITE turns 90° anti-clockwise
 The SPRITE waits 2 seconds

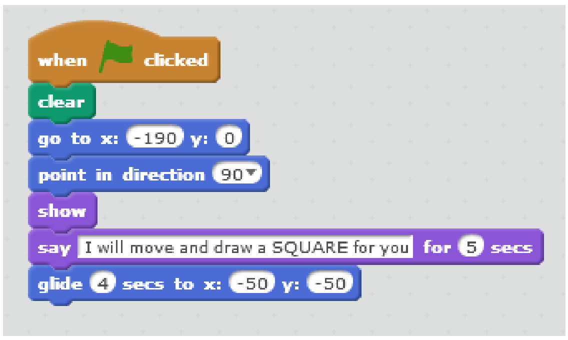
 And by doing so draws a square. The WAIT command is used so that there is a rest between each line

9 Pen is LIFTED UP so that any further movement does not draw on the stage area.

10 The SPRITE is hidden so that it does not cover the square.

Can you spot what might need changing in this script? You might not spot this until you try and run the script for a second time.

When the script ENDs, the SPRITE will be HIDDEN. When it is run again, the SPRITE will remain hidden. You therefore need to DEBUG the script by adding the UNHIDE command at the beginning so that the SPRITE is always unhidden at the start of the script. The first few lines would therefore be as shown. By including the SHOW command, you will always ensure that the SPRITE is shown every time the script is started.

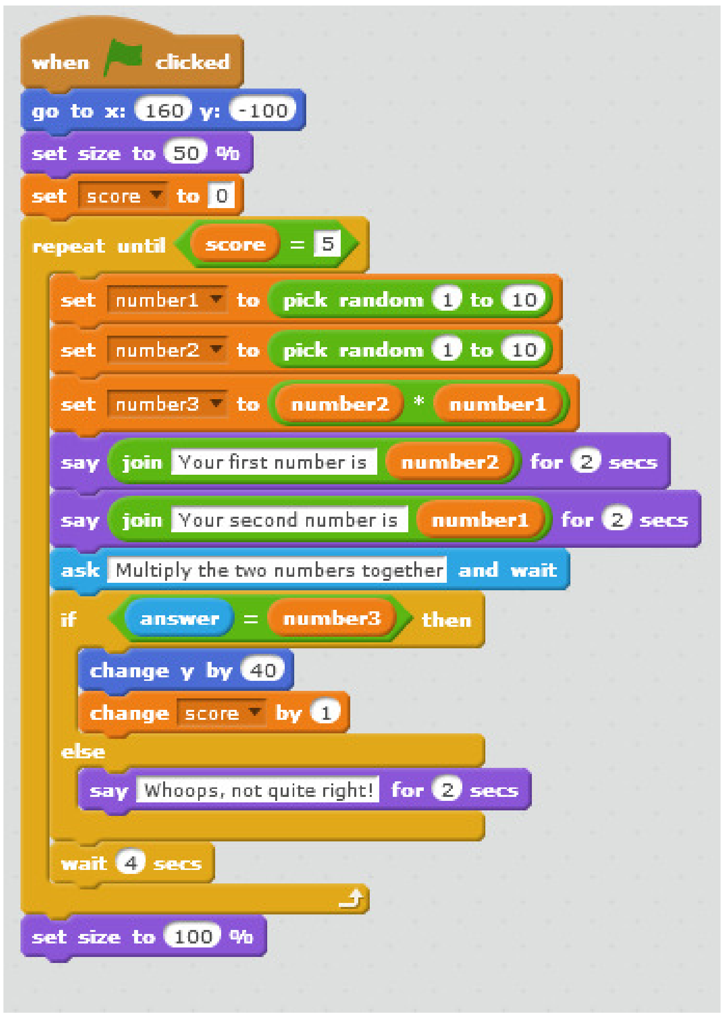


```
when 🏁 clicked
clear
go to x: -190 y: 0
point in direction 90▾
show
say I will move and draw a SQUARE for you for 5 secs
glide 4 secs to x: -50 y: -50
```

EXAMPLE SCRIPTS

EXAMPLE 5

A MULTIPLICATION GAME. The sprite is reduced in size. The player is presented with two numbers in turn and must then multiply them together and enter the answer. Every time the player gets the right answer the SPRITE moves up the screen until the player reaches 5 points at which point the sprite doubles in size and the game ends.



```
when [flag] clicked
go to x: 160 y: -100
set size to 50 %
set score to 0
repeat until score = 5
    set number1 to pick random 1 to 10
    set number2 to pick random 1 to 10
    set number3 to number2 * number1
    say join Your first number is number2 for 2 secs
    say join Your second number is number1 for 2 secs
    ask Multiply the two numbers together and wait
    if answer = number3 then
        change y by 40
        change score by 1
    else
        say Whoops, not quite right! for 2 secs
    wait 4 secs
set size to 100 %
```

SCRIPT FEATURES

Can you identify …

The part of the script which is repeated until the player reaches 5 points?

The part of the script where the numbers are generated randomly and given their variable names?

The part of the script where the user is prompted for an answer and how that answer is stored?

The part of the script where the user's response and the answer are compared and an action taken?

CHANGING THE SCRIPT

Can you think what you would have to change so that:

- The two random numbers are ADDED rather than MULTIPLIED together?
- The two random numbers are displayed for only ONE second on the stage?
- The SPRITE moves across the screen (from left to right) instead of up the screen?
- A high sound is played every time the user enters a correct answer?