


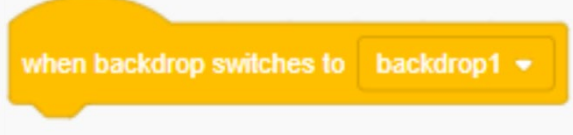
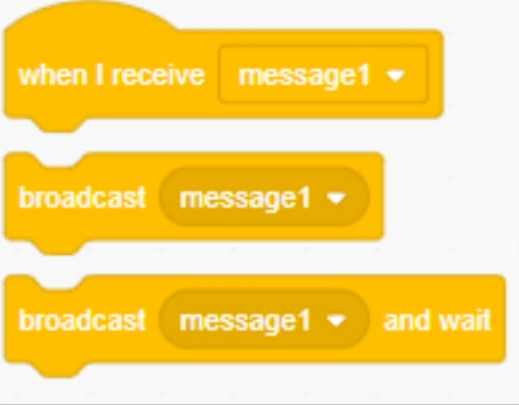
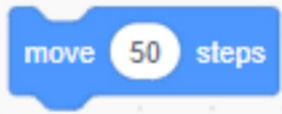
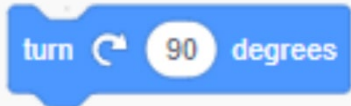
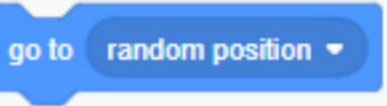
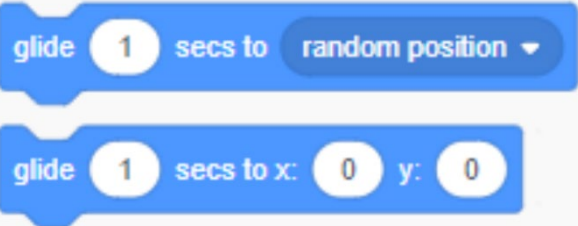
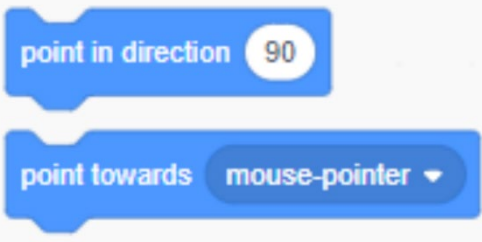


Summary: Scratch code blocks by category

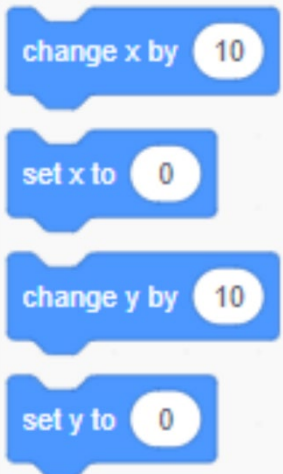
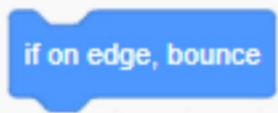
Event blocks

	<p>When the green flag is clicked, the code connected to this event block will run.</p>
	<p>This event block allows you to choose from a list of the different keys on the keyboard that can be used to trigger an event.</p> <p>Included in the list are the arrow keys that are often used when creating a game.</p>
	<p>It is useful to have a block of code run when the sprite itself is clicked when you are coding a game.</p>
	<p>Code can also be added to the various backdrops on the stage. This event block is used to run code when the backdrop changes to a specific one.</p>
	<p>The broadcast and receive blocks are used to create interaction between sprites.</p> <p>Use one of the broadcast blocks to send a signal to another sprite. You can create new messages in the drop down list and give them descriptive names.</p> <p>Use the “When I receive” block on the sprite that will react to the specified message.</p>


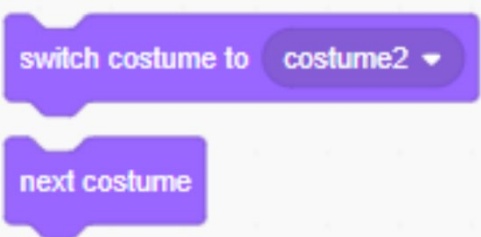
Motion blocks

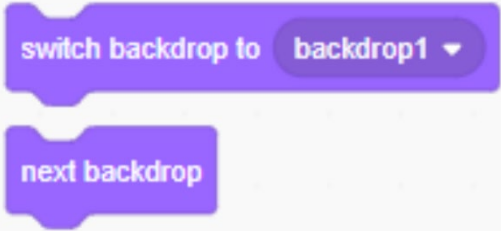
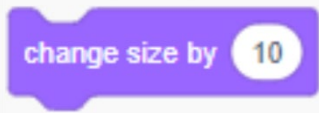
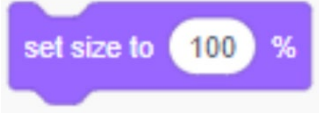
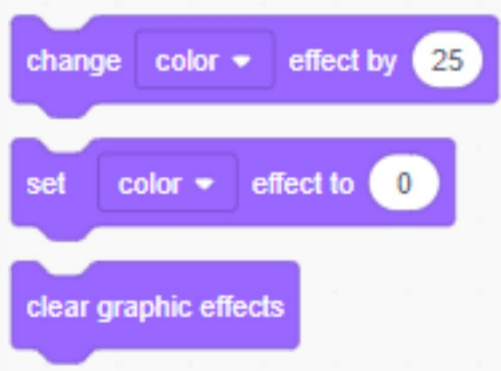
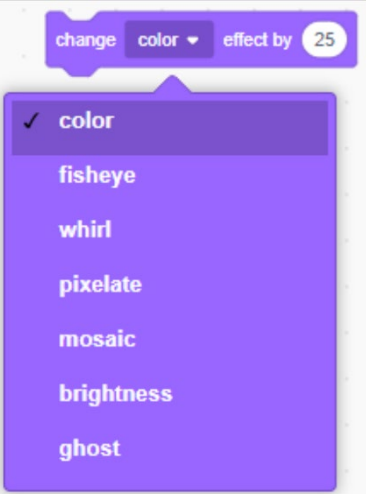
	<p>The sprite will move the number of pixels shown in the white oval across the stage. You can type in the oval and change the number to change how far the sprite moves.</p>
	<p>The sprite will turn right for the number of degrees shown in the white oval space on the block. There is also a turn left block that is used in this program. You can change the number of degrees in the oval to control how much the sprite will turn.</p>
	<p>The go to block has two options on a drop down list:</p> <ul style="list-style-type: none"> • Go to a random position on the stage • Go to where the mouse pointer is on the stage
	<p>The glide blocks allow you to enter a time in seconds to control how long it takes a sprite to move to a position. The position is either:</p> <ul style="list-style-type: none"> • A random position on the stage, or • A fixed X- and Y-coordinate that you have entered into the block
	<p>The point in direction block allows you to enter the direction in degrees:</p> <ul style="list-style-type: none"> • Right is 90 • Left is -90 • Down is 180 • Up is 0 <p>The point towards block lets the sprite face towards wherever the mouse pointer is.</p>

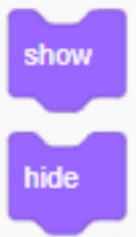
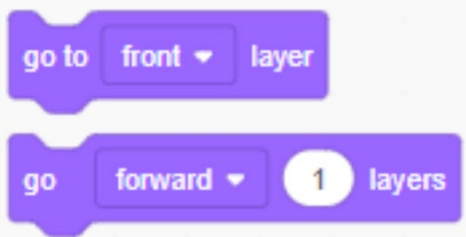


	<p>The change x or y by blocks allow you to enter by how many pixels each of the values of coordinates must change (become more or less). You will how this can be used once repetition of code is discussed.</p> <p>The set x or y to blocks allow you to change the coordinates to specific values.</p>
	<p>This block helps prevent the sprite from moving right off the stage by letting it bounce of the edge of the stage and move in another direction.</p>

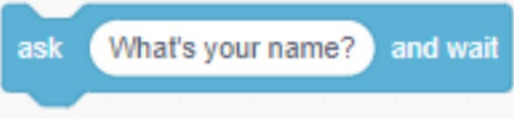
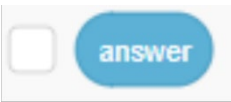
Looks blocks

	<p>The say block displays the text found on the block in a speech bubble attached to the sprite. The default text is the word Hello!, but you can type over that to enter your own words.</p> <p>The version with the time included allows you to control for how long the output is displayed before it goes away.</p> <p>The think blocks work in the same way as the say blocks, except that the output is done in a thought bubble.</p>
	<p>Sprites often have more than one look or costume. The switch costume to block allows you to specify a particular costume to change to.</p> <p>The next costume block simply change to the next costume available for the sprite.</p>

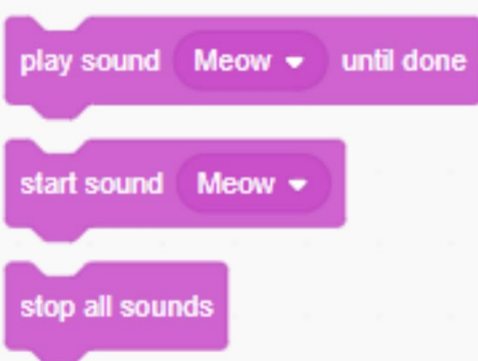

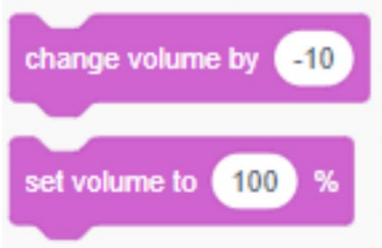
	<p>You are able to add more than one backdrop to the stage. The switch backdrop to block allows you to specify a particular backdrop to change to.</p> <p>The next backdrop block simply change to the next backdrop available for the stage.</p>
	<p>Increase the size of the sprite by the number in the block. A negative number will decrease the size of the sprite. This block is often used with repetition to grow or shrink a sprite in steps.</p>
	<p>Set the size to a percentage of the current size. Use 100% to return a sprite to its original size. Make the sprite bigger with a value greater than 100% and smaller with a value less than 100%.</p>
	<p>The change and set effect blocks are used to change the appearance of a sprite. The extent of the effect is controlled by a number value.</p> <p>The possible effects include:</p>  <p>See some examples of these effects below this table.</p> <p>The clear graphics effects block is essential to remove any of these effects as they change or distort the</p>

	sprite.
	Show or hide a sprite as needed.
	<p>Sprites can be placed over each other or can be moved over each other.</p> <p>These blocks are used to move a sprite a sprite in front of or behind another sprite.</p>

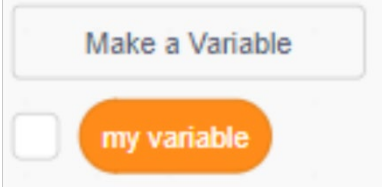
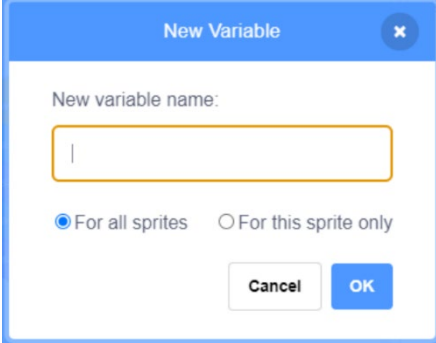
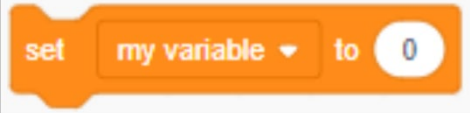
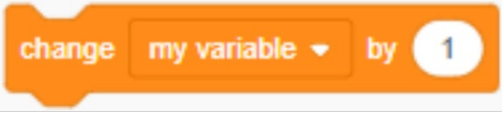
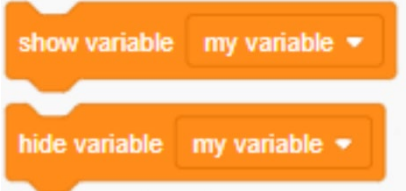
Sensing blocks

	<p>The ask block is used to ask the user a question and then it creates an input box into which the user of the program can type their answer. The user clicks on the tick button to submit their input.</p> <p>Note that this block ends with the words and wait. This means that the program will pause until the user has typed a value into the input box and clicked on the tick button.</p>
	<p>The answer variable stores the input from the ask block.</p> <p>If you tick the box next to the answer variable, it will display the value currently being held in answer on the stage.</p>

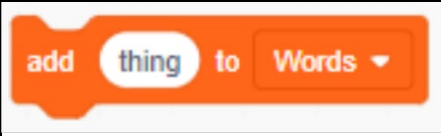

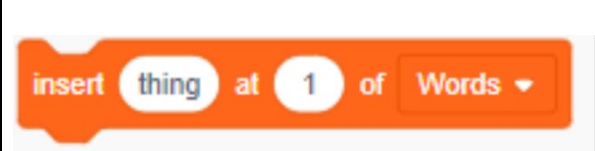
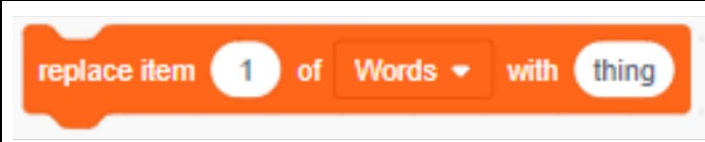

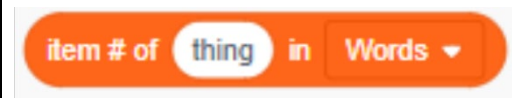

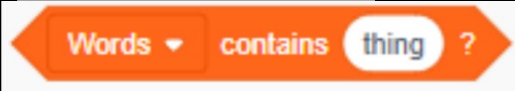
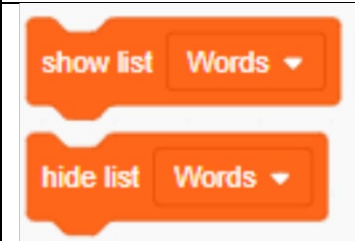
Sound blocks

	<p>There are blocks to play the entire selected sound clip to the end, start playing the selected sound, or to stop all sounds that are playing.</p> <p>In Scratch 3 default sound is the Meow sound for the Cat sprite. The default sound changes to match the sprite being used.</p> <p>All the sounds you add on the Sound tab will be added to the drop down lists on these blocks.</p>
	<p>There are a variety of effects that can be used to add interest to the chosen sounds. Just like the sprite effects the level of the effect can be set by changing the number values.</p> <p>The clear sound effects block resets the sound back to its original form.</p>
	<p>The volume of the sound can be controlled with a positive value making it louder and a negative number making it softer.</p>

Variables blocks

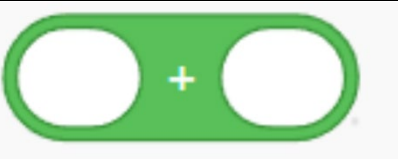



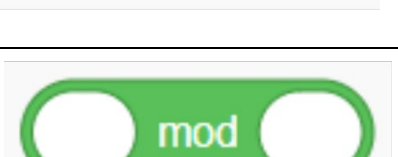
	<p>Click the Make a variable block to create a new variable.</p> <p>The following dialogue box will appear where you enter the name of the variable.</p>  <p>Note that you can choose whether all the sprites in your program can access and use the variable, or whether only the current sprite can use it.</p> <p>The new variable will appear in the blocks section. Tick the box next to the variable if you want its value to be displayed on the stage.</p>
	<p>The value stored in the variable can be typed into the white oval space. You can also drop variables such as answer into this space or any other block that has this shape to set the value in the variable.</p>
	<p>If a positive number is entered in the space, it will be added to the value currently stored in the variable. Type in a negative number to decrease the value in the variable.</p>
	<p>These blocks can be used to display (show) the value currently held in the variable on the stage, or to hide the variable on the stage.</p>

Variables blocks used for list operations

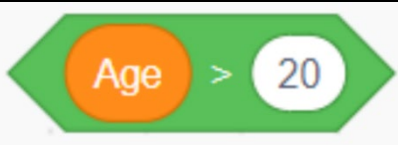
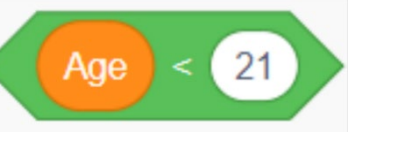

	<p>Add an item to the end of a list. The item can be typed into the white space or a variable can be placed in the space.</p>
	<p>You can delete an item at a specific index in the list by typing the index value into the white space.</p> <p>There is also an option to delete all the items in a list at once.</p>
	<p>You can insert a value into a specific index in a list. The items that follow in the list will automatically be renumbered.</p>
	<p>An item at a specific index in a list can be replaced with a new value.</p>
	<p>Access the value stored at a particular index in a list.</p>
	<p>Provides the index of a specific item in a list.</p>
	<p>Tells you how many items there are in a list.</p>
	<p>Checks whether a list contains the specified item and returns True if it does and False if it is not in the list.</p>
	<p>The list appears on the stage and these blocks are used to show and hide the list on the stage.</p>

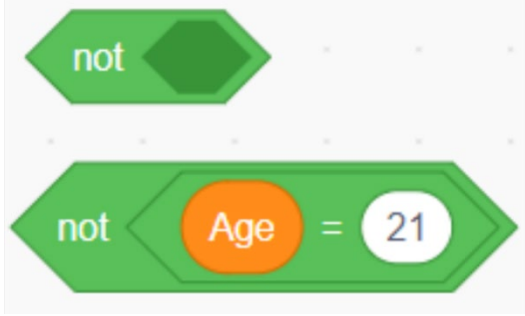
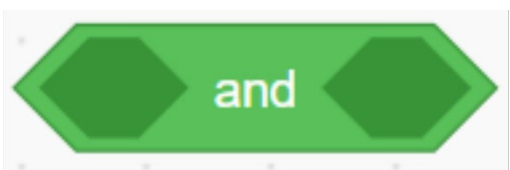



Operator blocks


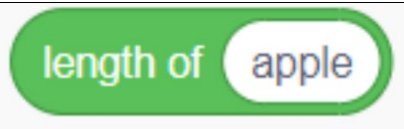
	Add two numbers together.
	Subtract the second value from the first one.
	Multiply two numbers by each other. In programming the *-symbol is used for multiplication to avoid confusion with the letter x and the multiplication sign.
	Divide the first number by the second number. In programming the /-symbol is used for division as the ÷ sign does not exist on the keyboard.
	The mod or modulus operator gives the remainder of a division sum. For example $9 \text{ mod } 2 = 1$.

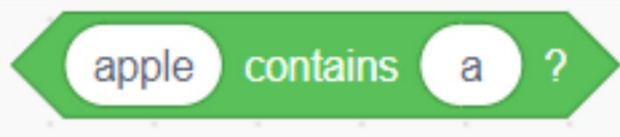
Operator blocks used for making comparisons

	This block checks whether the first value is greater than the second value. If Age = 15, then this block will return False as Age is not greater than 20.
	This block checks whether the first value is less than the second value. If Age = 15, then this block will return True as Age is greater than 20.
	This block checks whether the first value is equal to (the same as) the second value.


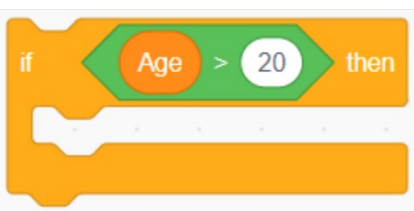
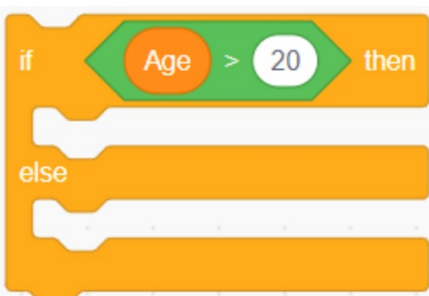
	<p>If Age = 15, then this block will return False as Age is not equal to 20.</p>
	<p>The Not block reverse the result of the comparison expression it surrounds.</p> <p>If Age = 21, then the expression given here will evaluate to false as follows: Age = 21 is True, and then the Not reverses that to become False.</p>
	<p>The AND operator will return True if both conditions are True. For example: (6 > 5) AND (9 < 11) is True True AND True is True</p> <p>(6 = 5) AND (9 < 11) is False False AND True is False</p>
	<p>The OR operator will return True if any one or both of the conditions is True. Both conditions have to be False for the OR operator to return False. For example: (6 > 5) OR (9 < 11) is True True OR True is True</p> <p>(6 = 5) OR (9 < 11) is True False OR True is True</p> <p>(6 = 5) OR (9 > 11) is False False OR False is False</p>

Operator blocks used for manipulating strings

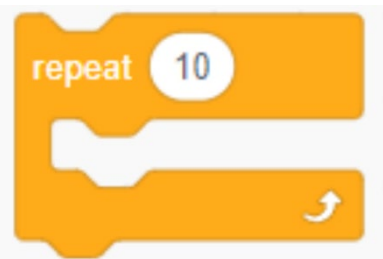
	<p>The characters in a string are numbered starting from 1 to the end of the string. Specific letters can be accessed or used if their position in the string is known.</p>
	<p>The length of a string tells you how many characters there are in the string. It also gives you the position of the last character in the string.</p>

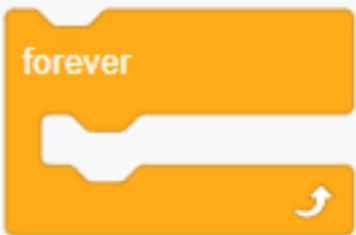

	<p>This block returns True if the character or series of characters on the right are found in the string on the left. It returns False if the character or series of characters is not found in the main string.</p>
---	--

Control blocks used in conditional statements

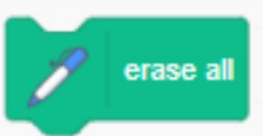
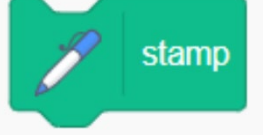

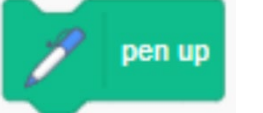
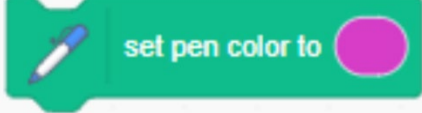
	<p>This block pauses the code for the number of seconds typed into the white oval space. It is used in this program because otherwise the sprite would move so fast that we would not actually see it moving and turning.</p>
	<p>Run the code inside the C-shape if the condition in the comparison block is met.</p> <p>If the condition is not met, the program will skip over the code inside this block.</p>
	<p>Run the code inside the top C-shape (the <i>if</i> section) if the condition in the comparison block is met (True).</p> <p>Run the code inside the bottom C-shape (the <i>e/se</i> section) if the condition in the comparison block is NOT met (False).</p>

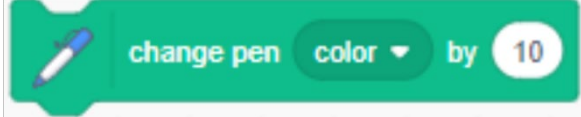
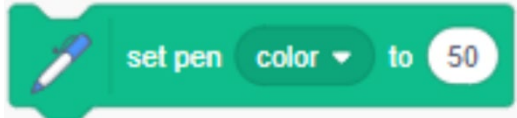
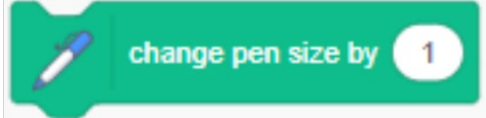
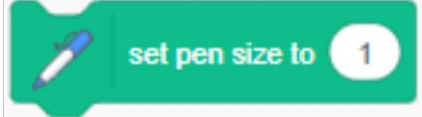
Control blocks used for repetition

	<p>Run the code placed inside the block for as many times as the number entered in the white space. A variable can also be used to control how many repetitions must take place.</p> <p>In this case you know beforehand exactly how many time the code must repeat. This is the same as the For loop mentioned in Module 2. This kind of loop is also known as an unconditional loop.</p>
---	--

	<p>Run the code inside the block over and over for as long as the program is running.</p>
	<p>The code placed in the block will keep on repeating until the condition set has been met.</p> <p>This is the same as the conditional while and repeat until loops mentioned in Module 2 as you do not necessarily know up front how many repetitions will take place.</p> <p>In this example there is a counter variable. Its value will need to be changed inside the loop or the repetition will never stop.</p>

Pen blocks

	<p>Erases all lines that have been drawn by the pen.</p>
	<p>The stamp block leaves behind an image of the sprite on the stage every time it is called.</p>
	<p>Puts the pen down so that it draws a line as the sprite moves.</p>
	<p>Lifts the pen up so that it does not draw a line as the sprite moves.</p>
	<p>Sets the colour of the line drawn by the pen to the selected colour.</p>

	<p>Changes the colour of the pen by the factor set in the white oval.</p> <p>Other colour properties can be chosen from the drop down list such as saturation, brightness and transparency.</p>
	<p>Set the colour of the pen to the colour represented by the value in the white oval.</p> <p>Other colour properties can be chosen from the drop down list such as saturation, brightness and transparency.</p>
	<p>Change the thickness of the line drawn by the pen. The higher the value in the white oval, the thicker the line will be.</p>
	<p>Set the thickness of the line drawn by the pen. The higher the value in the white oval, the thicker the line will be.</p>