<u>CL-118 Programming Fundamentals</u>

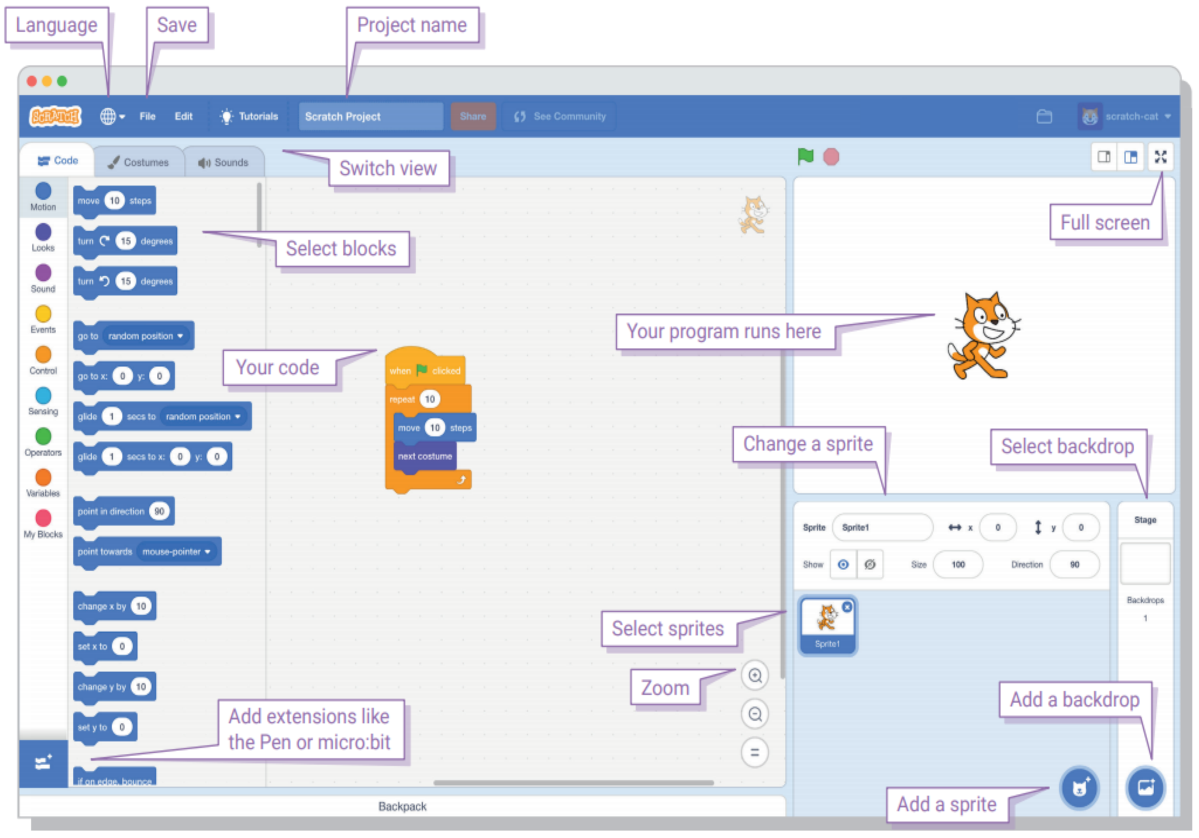
Scratch Booklet for reference

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
Fall 2021

Introduction

Visual Programming Language (VPL) is an application development environment designed on a graphical block-based programming model. Rather than series of imperative commands sequentially executed. VPL is targeted for beginner programmers with a basic understanding of concepts such as variables and logics.
Scratch is a visual programming environment from MIT that allows you to learn computer programming while working on personally meaningful projects. It can be used to create interactive stories, games, animations and images using script built with ready-to-use blocks. Scratch uses a single-window, multi-pane design to ensure that key components are always visible. Scratch programs, also called projects, are created by dragging, dropping, and snapping together different blocks.

Scratch Interface



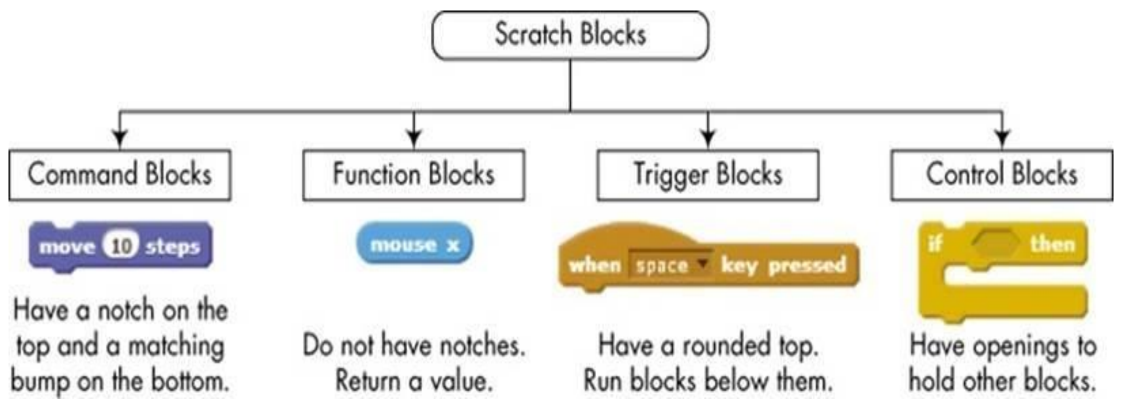
Detail of Scratch programming Environment

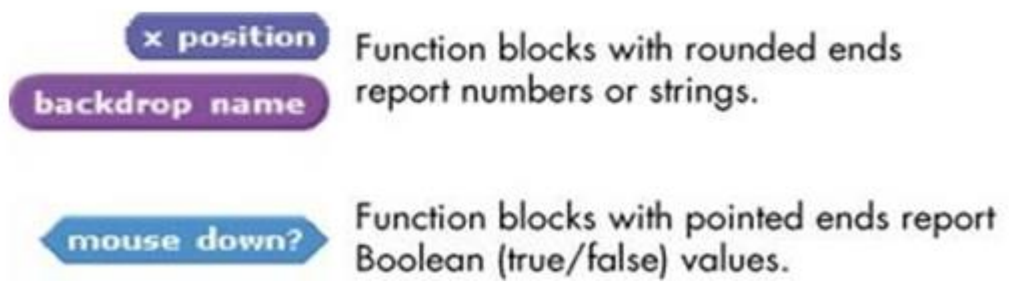
Area	Function
Script	All blocks that are joined together are called a *script.*
Sprite	An object in **Scratch** which performs functions controlled by scripts.
Scripting Area	Where you will drag and drop the blocks that make up the script(s) your sprite(s) will follow. Each sprite that you create will have its own scripting area.
Block Palette	There are ten different Blocks category that are Motion, Looks, Sound, Pen, Data, Events, Control, Sensing, Operators and more Blocks

List of Blocks	Once you click on one of the categories listed, the different blocks that are included in this category are listed.
Resource Tab	There are three tabs: • The Scripts tab is where you find all drag and drop blocks that make up the script(s). • The Costumes tab is where all of the poses or looks a sprite can have are created. When writing scripts for the staging area (referred to as the stage), this tab becomes the background tab. • The Sounds tab is where different sounds and pieces of the music a sprite can use are created.
Stage	This is the area where the sprites execute or run the script that you built in the scripting area. Clicking on the green flag allows you to start executing the script (if you set up your script to do so) and the red button will make the script stop. The Scratch stage is 480 steps wide and 360 steps tall
Sprite List	You can have as many sprites as you want in a single program. Clicking on an individual sprite shows you its scripting area.
Backdrop	A backdrop is one out of possibly many frames, or backgrounds, of the Stage. When you select backdrop the costume tab will become backdrop tab for editing.

Types of Blocks

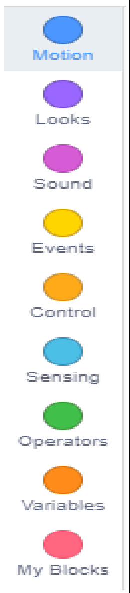
Scratch has four kinds of blocks: command blocks, function blocks, trigger blocks, and control blocks. Command blocks and control blocks (also called stack blocks) have bumps on the bottom and/or notches on the top. You can snap these blocks together into stacks. Trigger blocks, also called hats, have rounded tops because they are placed at the top of a stack. Trigger blocks connect events to scripts. They wait for an event—such as a key press or mouse click—and run the blocks underneath them when that event happens. For example, all scripts that start with the when green flag clicked block will run when the user clicks the green flag icon.


Scratch Blocks
Command Blocks
Function Blocks
Trigger Blocks
Control Blocks
move 10 steps
mouse x
when space key pressed
if then
Have a notch on the top and a matching bump on the bottom.
Do not have notches. Return a value.
Have a rounded top. Run blocks below them.
Have openings to hold other blocks.



Function blocks with rounded ends report numbers or strings.

Function blocks with pointed ends report Boolean (true/false) values.

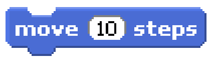
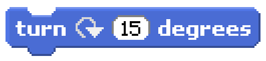
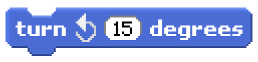
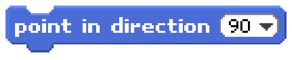
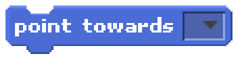
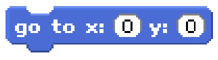


Block Palette Description

The Scratch blocks are organized into ten color-coded categories: Motion, Looks, Sound, Event, Control, Sensing, Operators, Variables and More Blocks.

Motion

The category **Motion** groups' sprite movement blocks, used for their movement, rotation and reading current data related to a sprite's location. These are very often used basic blocks. Below is a description of all the blocks in this category.

Motion	
	Moves sprite forward or backward.
	Rotates sprite clockwise.
	Rotates sprite counterclockwise.
	Points sprite in the specified direction. (0=up, 90=right, 180=down, -90=left)
	Points sprite toward mouse-pointer or another sprite.
	Moves sprite to specified x and y position on Stage.

go to ▾	Moves sprite to the location of the mouse-pointer or another sprite.
glide (1) secs to x: (0) y: (0)	Moves sprite smoothly to a specified position over specified length of time.
change x by (10)	Changes sprite’s x-position by specified amount.
set x to (0)	Sets sprite’s x-position to specified value.
change y by (10)	Changes sprite’s y-position by specified amount.
set y to (0)	Sets sprite’s y-position to specified value.
if on edge, bounce	Turns sprite in opposite direction when sprite touches edge of Stage.
☐ x position	Reports sprite’s x-position. (Ranges from -240 to 240)
☐ y position	Reports sprite’s y-position. (Ranges from -180 to 180)
☐ direction	Reports sprite’s direction. (0=up, 90=right, 180=down, -90=left)

Looks

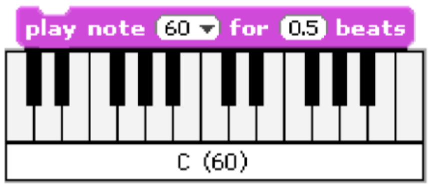
The **Looks** category is a collection of sprite appearance blocks concerning costume, size, visibility, etc. Some of these blocks are used very often. Below is a description of most blocks in this category.

Looks	
switch to costume costume1	Changes sprite's appearance by switching to different costume.
next costume	Changes sprite's costume to next costume in the cos-tume list. (If at end of the costume list, jumps back to first costume.)
costume #	Reports sprite's current costume number.
switch to background background1	Changes Stage's appearance by switching to a different background or backdrop
next background	Changes Stage's background to next backdrop in the backdrop list.
background #	Reports Stage's current background or backdrop number.
say Hello! for 2 secs	Displays sprite's speech bubble for specified amount of time.
say Hello!	Displays sprite's speech bubble. (you can remove speech bubble by running this block without any text.)
think Hmm... for 2 secs	Displays sprite's thought bubble for specified amount of time.
think Hmm...	Displays sprite's thought bubble.
change color effect by 25	Changes a visual effect on a sprite by specified amount. (Use pull-down menu to choose effect.)
set color effect to 0	Sets a visual effect to a given number. (most visual effects range from 0 to 100.)
clear graphic effects	Clears all graphic effects for a sprite.
change size by 10	Changes sprite's size by specified amount.
set size to 100 %	Sets sprite's size to specified % of original size.
size	Reports sprite's size, as % of original size.

show	Makes sprite appear on the Stage.
hide	Makes sprite disappear from the Stage. (When sprite is hidden, other sprites cannot detect it with **touching?** block.)
go to front	Moves sprite in front of all other sprites.
go back 1 layers	Moves sprite back a specified number of layers, so that it can be hidden behind other sprites.

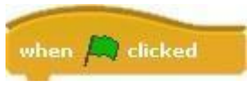
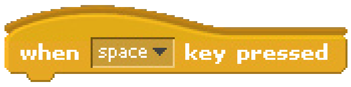
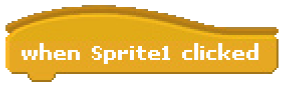
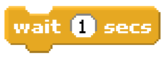
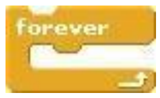
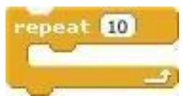
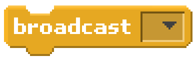
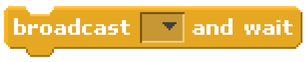
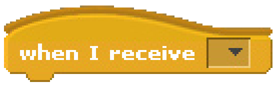
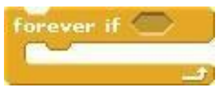
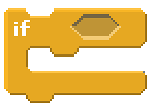
Sound

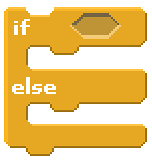
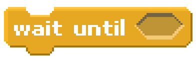
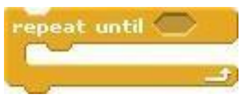
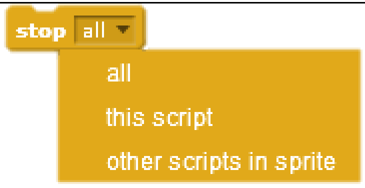
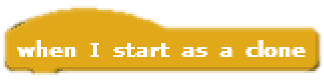
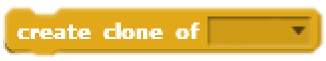
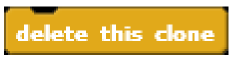
The **Sound** category is a group of blocks for sounds, instruments and musical notes.

Sound	
play sound meow ▾	Starts playing a sound, selected from pull-down menu, and immediately goes on to the next block even as sound is still playing.
play sound meow ▾ until done	
stop all sounds	Plays a sound and waits until the sound is finished playing before continuing with next block.
play drum 48 ▾ for 0.2 beats	
 play note 60 ▾ for 0.5 beats C (60)	Plays a musical note (higher numbers for higher pitches) for specified number of beats.
rest for 0.2 beats	Rests (plays nothing) for specified number of
set instrument to 1 ▾	Sets the type of instrument that the sprite uses for **play note** blocks. (Each sprite has its own instrument.)
change volume by -10	Changes sprite's sound volume by specified amount. Volume ranges from 0 to 100.
set volume to 100 %	Sets sprite's sound volume to specified value.
☐ volume	Reports sprite's sound volume.
change tempo by 20	Changes sprite's tempo by specified amount.
set tempo to 60 bpm	Sets sprite's tempo to specified value in beats per minute.
☐ tempo	Reports sprite's tempo in beats per minute.

Control

The **Control** category contains very important blocks – equivalent to instructions controlling programming languages (loops, conditional instructions). Below is a description of all the blocks in this category.

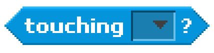
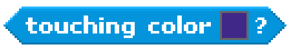
Control	
	Runs script below when green flag is clicked.
	Runs script below when specified key is pressed.
	Runs script below when sprite is clicked.
	Waits specified number of seconds, then continues with next block.
	Runs the blocks inside over and over.
	Runs the blocks inside a specified number of times.
	Sends a message to all sprites, then continues with the next block without waiting for the triggered scripts.
	Sends a message to all sprites, triggering them to do something, and waits until they all finish before continuing with next block.
	Runs script below when it receives specified broadcast message.
	Continually checks whether condition is true; whenever it is, runs the blocks inside.
	If condition is true, runs the blocks inside.

	If condition is true, runs the blocks inside the **if** portion; if not, runs the blocks inside the **else** portion.
	Waits until condition is true, then runs the blocks below.
	Checks to see if condition is false; if so, runs blocks inside and checks condition again. If condition is true, goes on to the blocks that follow.
	Stops all scripts in all sprite, current sprite or other sprite.
	Tells a clone what to do once it is created
	Creates a clone (temporary duplicate) of the specified sprite
	Deletes the current clone

Sensing

The **Sensing** category includes blocks related to the recognition of different situations that occur on the stage concerning, among others, sprites and the mouse, blocks to pull data from the keyboard, and blocks associated with the timer, date and camera. Below is a description of most blocks in this category, except for the camera operation and date blocks.

Sensing

	Reports true if sprite is touching specified sprite, edge, or mouse-pointer. (Select from pull-down menu.)
	Reports true if sprite is touching specified color. (Click on color patch, then use eyedropper to select color.)

ask [ ] and wait	Asks a question on the screen and stores keyboard input in the answer. Causes the program to wait until the Enter key is pressed or check mark is clicked.
☐ answer	Reports keyboard input from the most recent use of ask [ ] and wait. Shared by all sprites (global).
mouse x	Reports the x-position of mouse-pointer.
mouse y	Reports the y-position of mouse-pointer.
mouse down?	Reports true if mouse button is pressed.
key space ▾ pressed?	Reports true if specified key is pressed.
distance to ▾	Reports distance from the specified sprite or mouse-pointer.
reset timer	Sets the timer to zero.
☐ timer	Reports the value of the timer in seconds. (The timer is always running.)
x position ▾ of Sprite1 ▾	Reports a property or variable of another sprite.
☐ loudness	Reports the volume (from 1 to 100) of sounds detected by the computer microphone.
video motion ▾ on this sprite ▾	Senses how much **motion** or **direction** is currently in the video image
turn video on ▾	Turns the video camera on
set video transparency to 50 %	Sets the video transparency
current minute ▾	Reports the current time
days since 2000	Reports the number of days since 2000
username	Reports username of the viewer

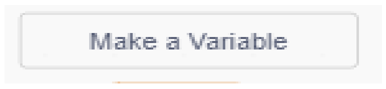
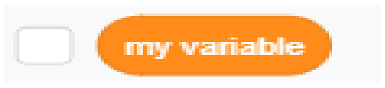
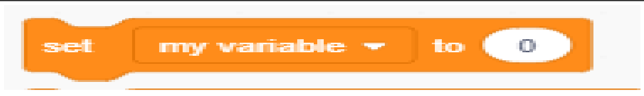
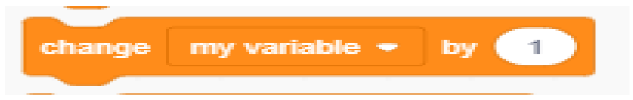
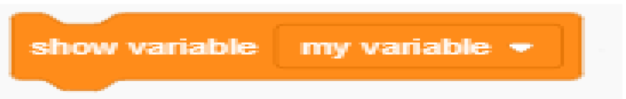
Operators

The category of **Operators** groups blocks of basic arithmetic operations, logical operations and various functions (both arithmetic and on texts). All blocks in this category are used as arguments for other blocks. Below is a description of most blocks in this category.

Operators	
+	Adds two numbers.
-	Subtracts second number from first number.
*	Multiplies two numbers.
/	Divides first number by second number.
pick random 1 to 10	Picks a random integer within the specified range.
<	Reports true if first value is less than second.
=	Reports true if two values are equal.
>	Reports true if first value is greater than second.
and	Reports true if both conditions are true.
or	Reports true if either condition is true.
not	Reports true if condition is false; reports false if condition is true.
join	Concatenates (combines) strings.
length of	Reports the number of letters in a string.
letter of	Reports the letter at the specified position in a string.
sqrt of 10	Reports result of selected function (abs, sqrt, sin, cos, tan, asin, acos, atan, ln, log, e^, 10^) applied to specified number.
mod	Reports remainder from division of first number by second number.
round	Reports closest integer to a number.

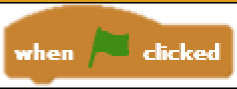
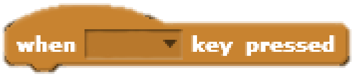
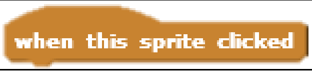
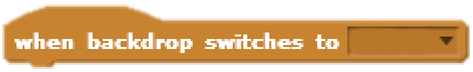
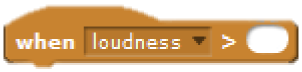
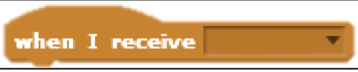
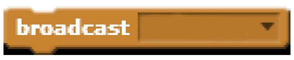
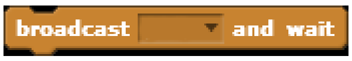
Variable

The **variable** category contains buttons for creating variables. Once we have created our own variables, blocks will appear to be modified and use values. Variables allow us to store data. We can delete a variable.

Variable	
	We can create a variable
	Set initial value to zero
	Set value of your own choice
	Change a value of a variable
	we can rename it or delete it

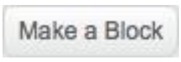
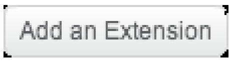
Events

The **Events** category includes blocks which start scripts reacting to specific events (e.g. clicking a sprite with the mouse, pressing a key on the keyboard) and related to message support (self-generating events in the program). Below is a description of almost all the blocks in this category.

Event	
	The basic block. It starts the majority of scripts. Scripts are run when we click the green flag on the screen – start the programme.
	Runs the script after pressing a specific key on the keyboard. We select a key from a drop-down list.
	The block runs the script when we left-click the sprite.
	The block runs the script when a change of background occurs. The background name can be selected from a drop-down list. The script will be started when the background becomes the current one (displayed on the stage)
	Runs a script when the selected attribute (loudness, timer, video motion) is greater than a specified value
	The block starts the script when a message of the specified name selected from the drop-down list is broadcast.
	This block allows us to define and broadcast a message. We can choose the message name from the drop-down list or define a new message using the **new message...** option available when we expand the list.
	Works in the same way as the block **broadcast *message name*** . The difference is that the operation of the script in which the message was broadcast is paused until the end of the operation of all the scripts receiving this message.

More Blocks

More blocks allow user to build user define procedures or functions.

More Blocks	
	Creates a custom block A **define** block will appear in Scripts. Use **define** to tell the custom block what to do
	Add blocks that extend what you can do in Scratch Like LEGO and WeDo

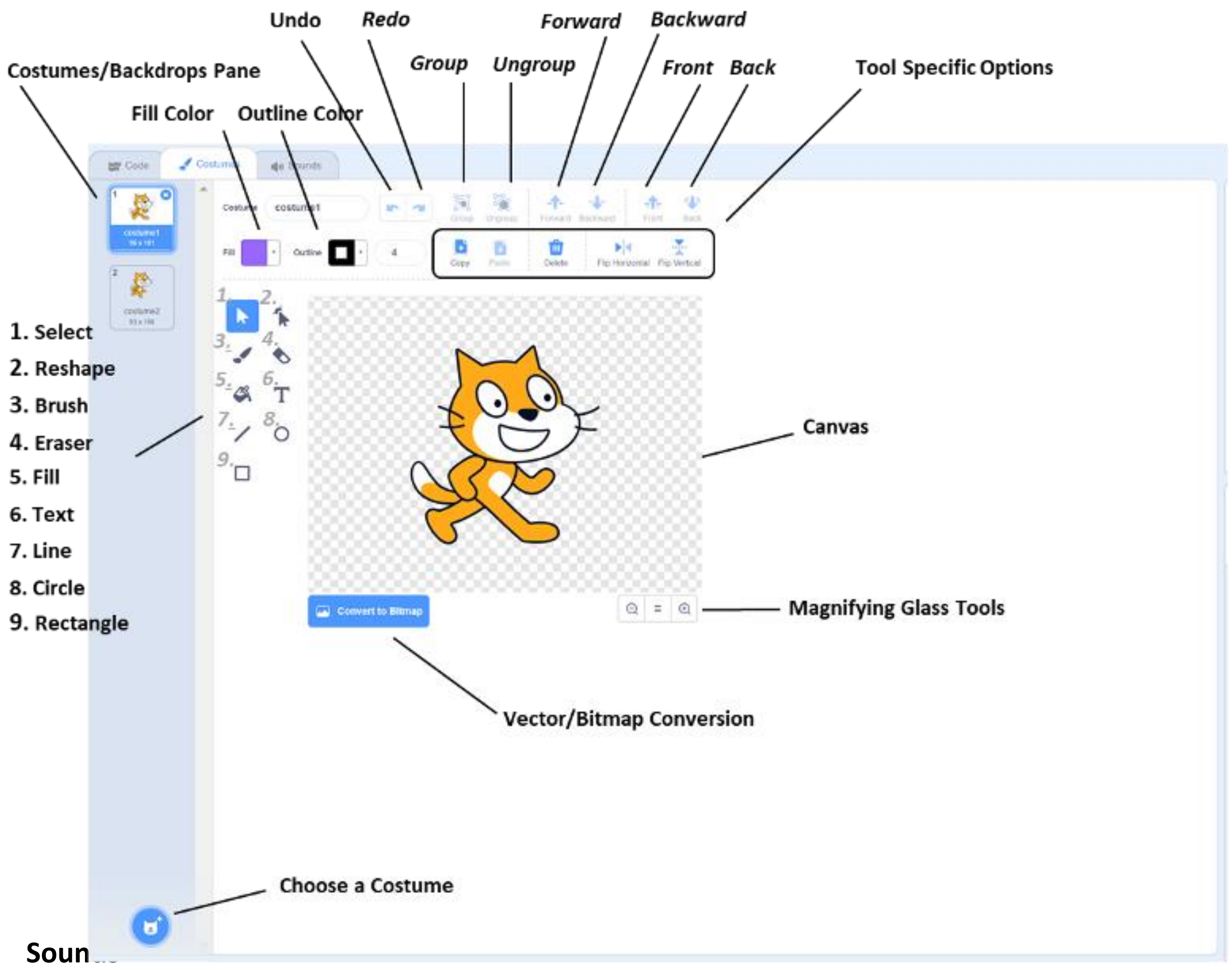
Pen

The **Pen** category is a collection of blocks connected with a sprite's pen, its status, colour and thickness. This group also includes a block used to clear created drawings and the possibility of copying the sprite's image as a stamp. Below is a description of all the blocks in this category.

Pen	
clear	Clears all pen marks and stamps from the Stage.
pen down	Puts down sprite's pen, so it will draw as it moves.
pen up	Pulls up sprite's pen, so it won't draw as it moves.
set pen color to 🟪	Sets pen's color, based on choice from color picker. Picking the color also changes the pen shade.
change pen color by 10	Changes pen's color by specified amount.
set pen color to 0	Sets pen's color to specified value. (pen_color=0 at red end of rainbow, pen_color=100 at blue end of rainbow. Ranges from 0 to 200 to go around the color wheel.)
change pen shade by 10	Changes pen's shade by specified amount.
set pen shade to 50	Sets pen's shade to specified amount. (pen_shade=0 is very dark, pen_shade=100 is very light. Default is 50, unless set with color picker.)
change pen size by 1	Changes pen's thickness.
set pen size to 1	Sets pen's thickness.
stamp	Stamps sprite's image onto the Stage.

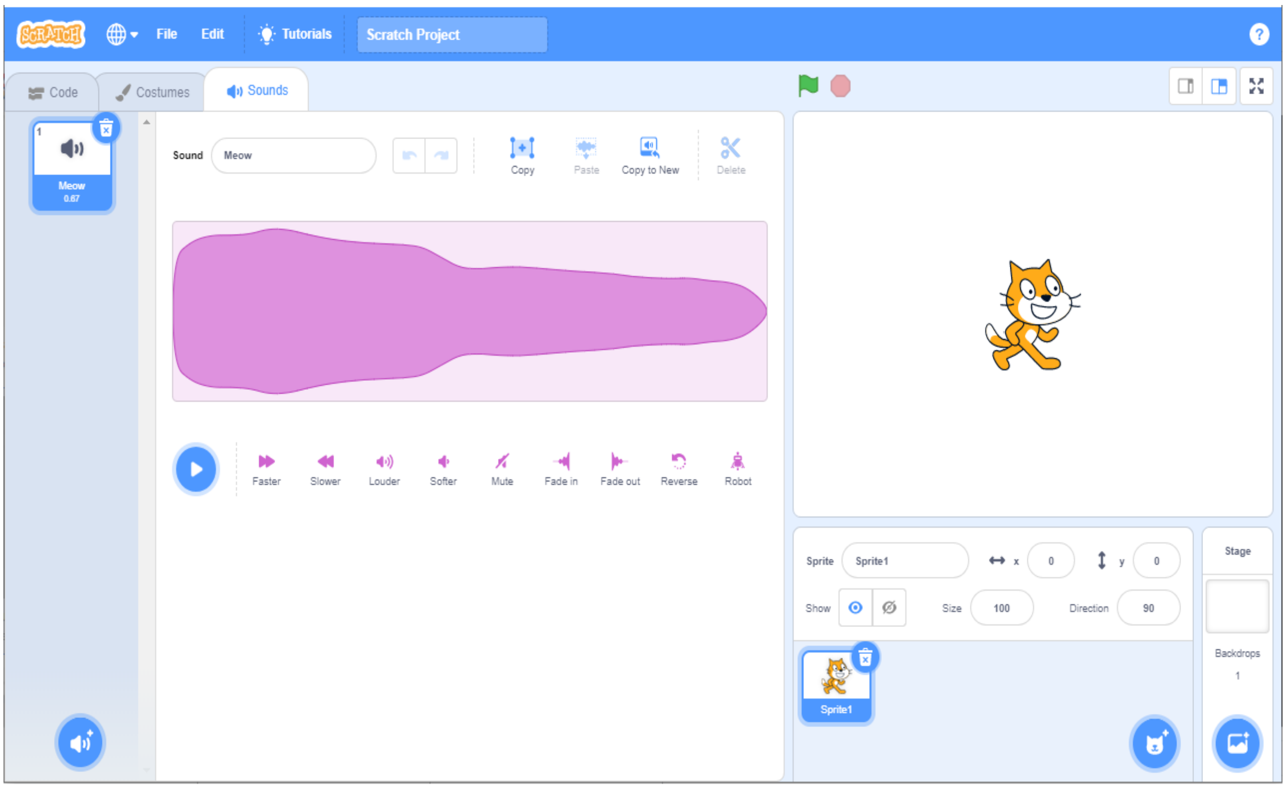
Sprite costumes

When we select the tab costume, we can change sprite costumes, i.e. sprite pictures. The tray contains all of the sprite’s costumes. The panel on the right side is now a graphics editor where we can create or change a costume.



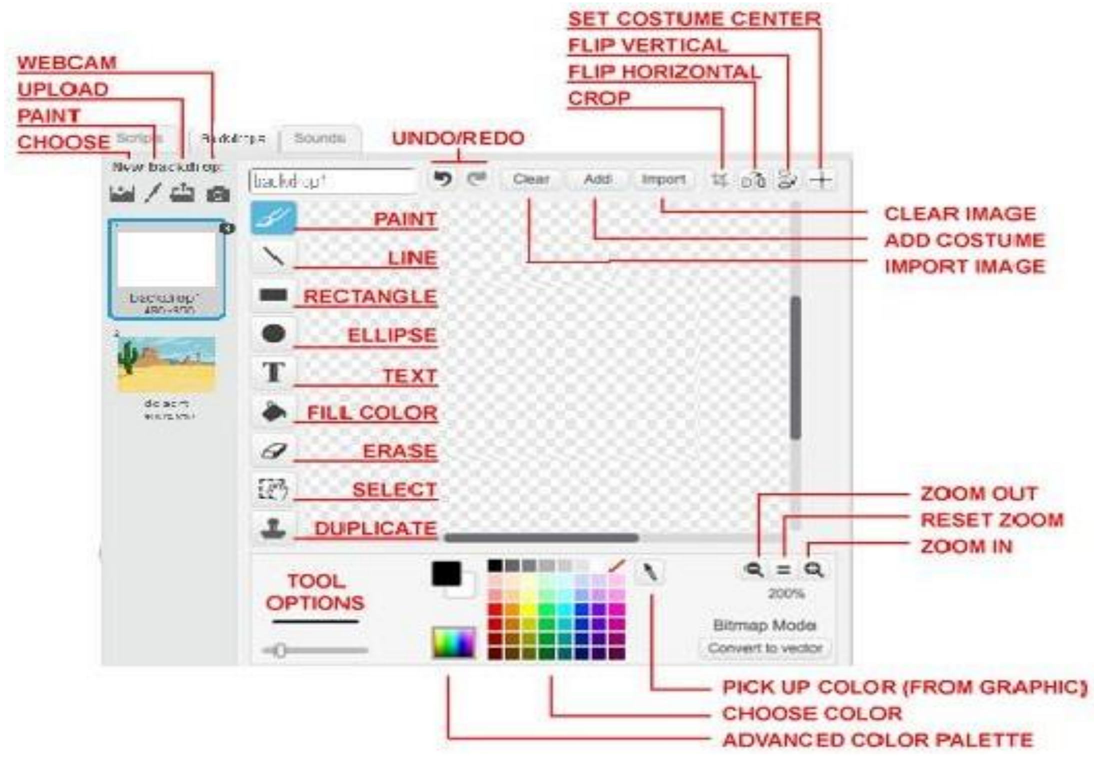
Soun

After we select the Sounds tab we can change and add sounds for a sprite. We have the option to add new sounds and process them.



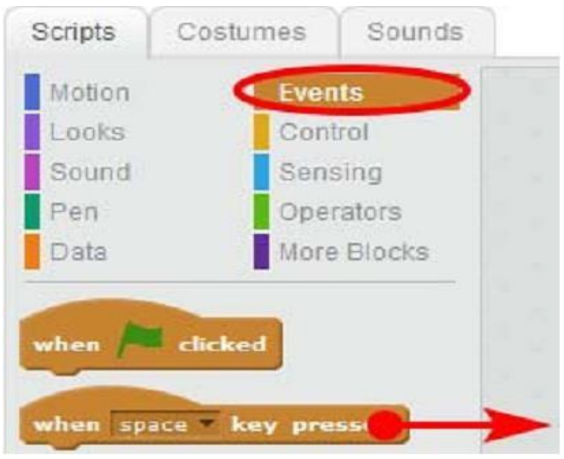
Backdrop:

When clicking to backdrop a window is open which allows users to create backdrops with the bitmap tools and techniques in Bitmap Mode. The Bitmap Paint Editor is the default editor when creating a backdrop. By clicking the Convert to Vector button, users can backdrops with the vector editing tools and techniques in Vector Mode.

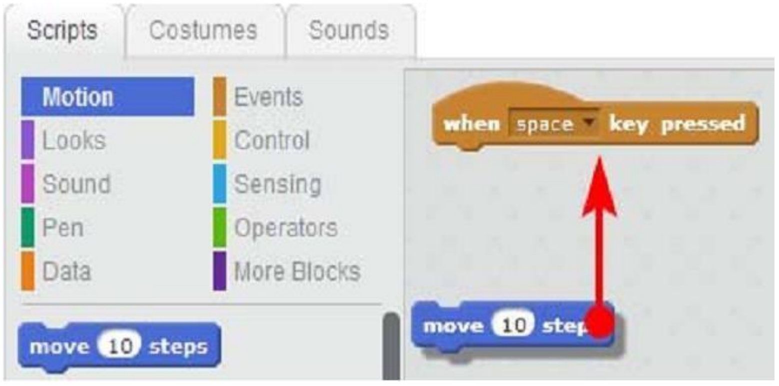


CREATING YOUR FIRST SCRIPT IN SCRATCH:

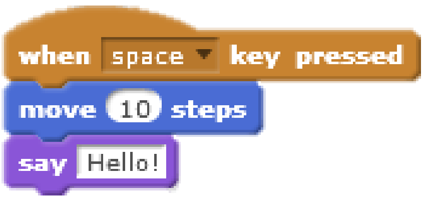
We're going to create a script for the cat sprite to make it move and say Hello. The script start with an event, First you have to select the “Events' block and drag the block “with the space key pressed” to the right and drop it by releasing the mouse button.



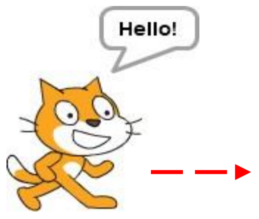
Now Select the Blue Motion Block and drag the “move 10 step” to the right. Fix it under the brown block. Release the mouse button and it will be attached.



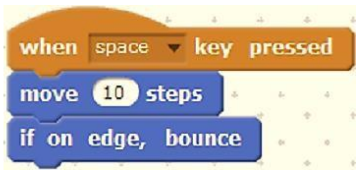
Now select “looks” and drag the block “say Hello” to the right and attached with the blue one.

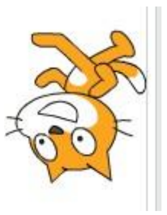


Now your script is ready to be execute press the spacebar and see how the cat is moving and saying Hello. If you keep on pressing, he will go to the edge of the screen and will hide there.



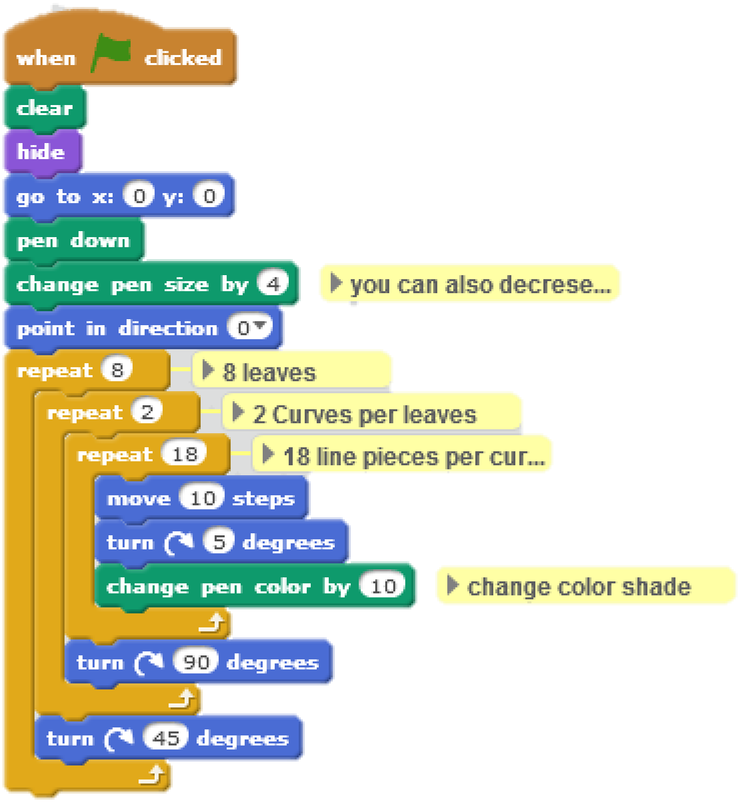
You can expand the script by removing “say Hello” block and replacing it with blue block named “if on edge bounce” .Now cat moves silently to the screen edge , pushes off, turn back and stands on his head.



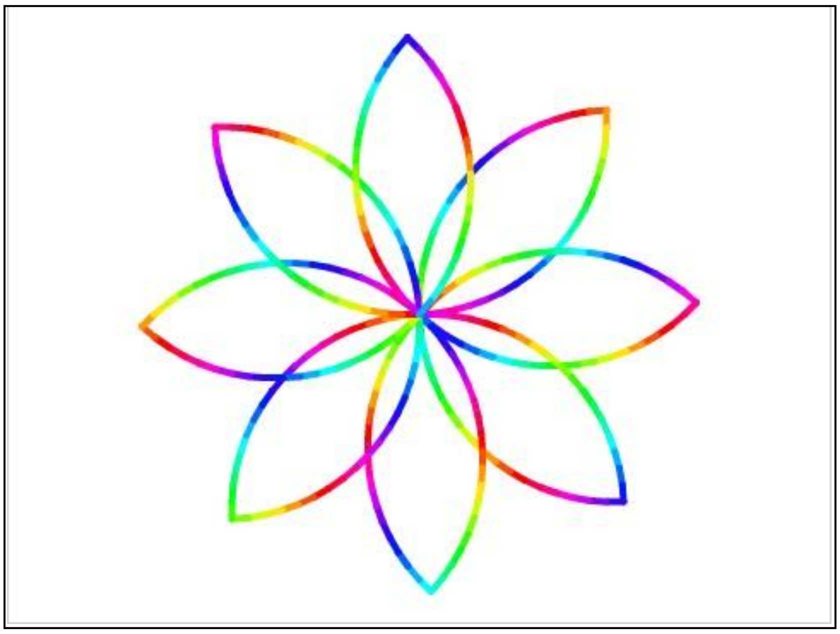


Tracing Path of Sprite to make Colorful Flower:

We can trace the path of a sprite by using the pen to draw on the stage.it can be used to draw many different things. In this example you will learn to draw flower by using only pen, loops and color.



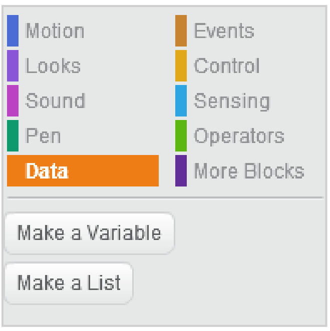
```
when [flag] clicked
clear
hide
go to x: 0 y: 0
pen down
change pen size by 4          you can also decrese...
point in direction 0
repeat 8                      8 leaves
    repeat 2                  2 Curves per leaves
        repeat 18             18 line pieces per cur...
            move 10 steps
            turn ↻ 5 degrees
            change pen color by 10     change color shade
        turn ↻ 90 degrees
    turn ↻ 45 degrees
```

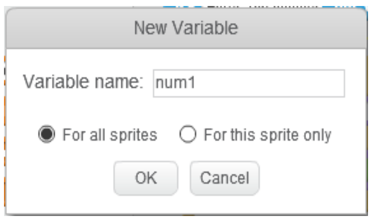


Performing arithmetic operations using variables

In this example you learn to utilize variables, control blocks and function blocks. For arithmetic operation we first need to define 2 variables work as operand. The sprite then ask user to enter numbers for division operation. Numbers are initially stored in answer block and then assigned to appropriate variable. The conditional block checks for condition if the divisor is non zero then it perform the operation over 2 operands otherwise it says “Try again”.

Similarly we can perform addition, subtraction or any other mathematical operation. Variables can also be used to store strings.

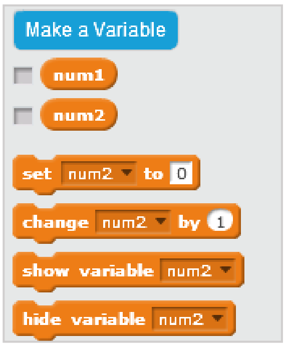


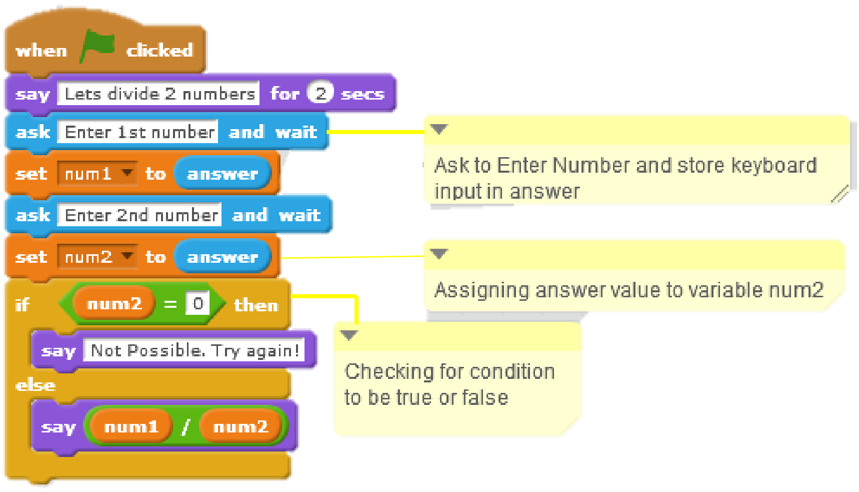

New Variable

Variable name: num1

[✓ For all sprites] For this sprite only

OK Cancel

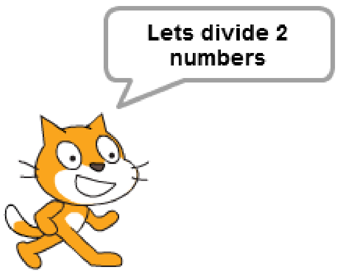


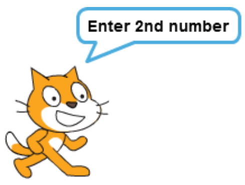

```
when [flag] clicked
say [Lets divide 2 numbers] for (2) secs
ask [Enter 1st number] and wait
set [num1] to (answer)
ask [Enter 2nd number] and wait
set [num2] to (answer)
if <(num2) = [0]> then
    say [Not Possible. Try again!]
else
    say ((num1) / (num2))
```

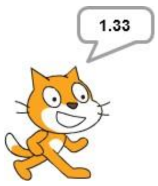
Ask to Enter Number and store keyboard input in answer

Assigning answer value to variable num2

Checking for condition to be true or false





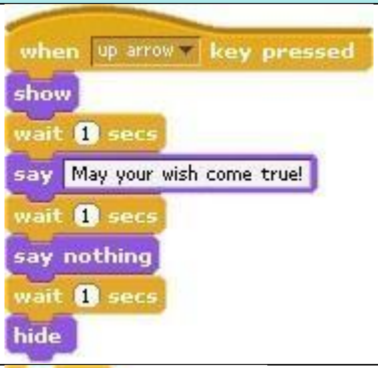
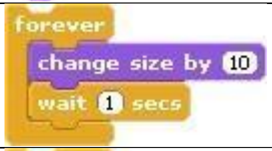
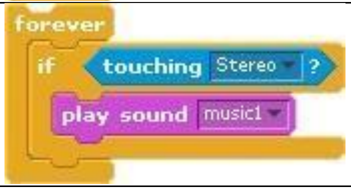
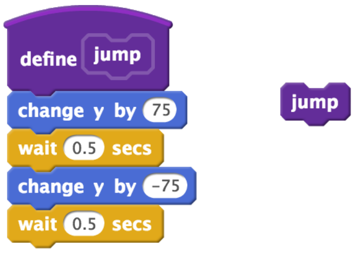
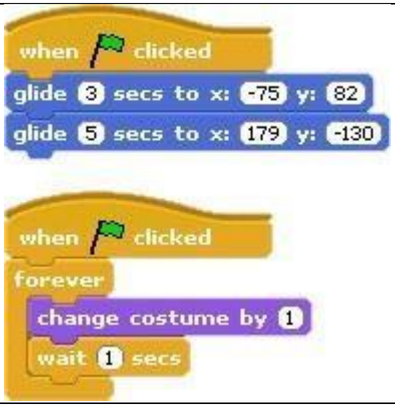


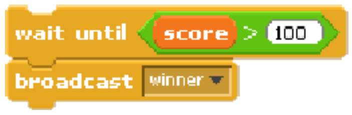
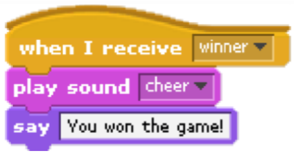
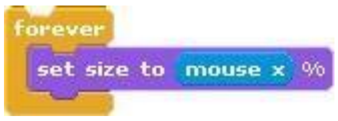
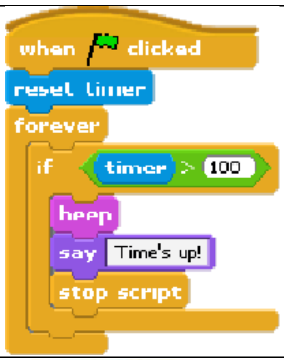
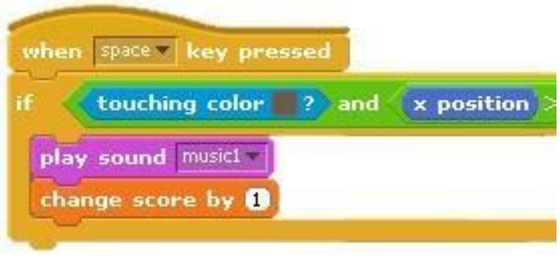
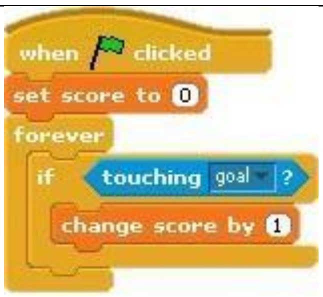
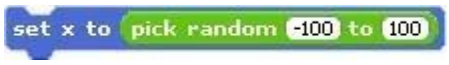
4

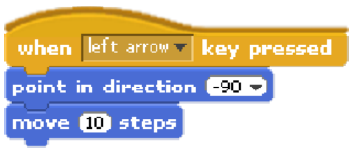
3

Fundamental concepts Support by Scratch:

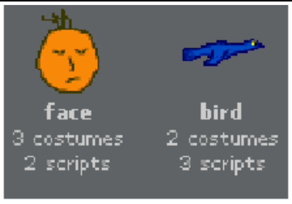
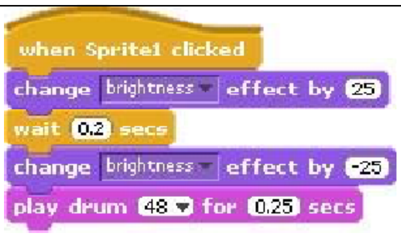
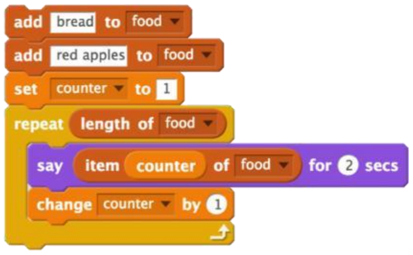
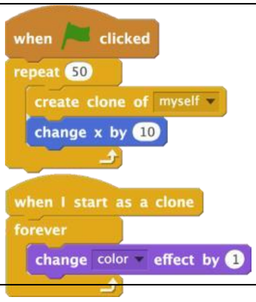
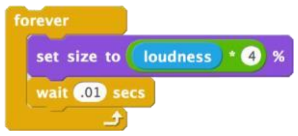
While creating project with scratch, it offers a wide number of fundamental Programing and computational concept.

Concept	Explanation	Example
Sequence	Creating just about any program in Scratch requires thinking of the order of steps	 when up arrow key pressed show wait 1 secs say May your wish come true! wait 1 secs say nothing wait 1 secs hide
Iteration (looping)	**forever** and **repeat** are examples of iteration.	
Conditional statements	**if, if-else** and **waituntil** check for a condition	
Procedures	The *Make a Block* feature lets you define a new block that you use in your scripts. Defining a block can also be called naming a procedure. The Make a Block feature allows reusing code within a sprite, and can support modularity and abstraction.	 define jump change y by 75 wait 0.5 secs change y by -75 wait 0.5 secs jump
Threads (parallel execution)	Launching two stacks at the same time creates two independent threads that execute in parallel.	 when clicked glide 3 secs to x: -75 y: 82 glide 5 secs to x: 179 y: -130 when clicked forever change costume by 1 wait 1 secs

Synchronization	**broadcast** can coordinate the actions of multiple sprites	For example, Sprite1 sends message when condition is met:  wait until score > 100 broadcast winner This script in Sprite2 is triggered when the message is received:  when I receive winner play sound cheer say You won the game!
Real-time interaction	**mouse_x**, **mouse_y,** and **loudness** can also be used as dynamic input for real-time interaction.	 forever set size to mouse x %
Time triggering	Scratch includes an internal clock that you can access with **timer**	 when clicked reset timer forever if timer > 100 beep say Time's up! stop script
Boolean logic	**and**, **or**, or **not** are examples of boolean logic	 when space key pressed if touching color ? and x position play sound music1 change score by 1
Variables	The Variables category allows creating a new variable and using it in a program Scratch supports both global and object-specific variables.	 when clicked set score to 0 forever if touching goal ? change score by 1
Random numbers	The **pick random** block selects random integers within a given range	 set x to pick random -100 to 100

Event handling	**when key pressed** and **when sprite1 clicked** are examples of event handling—responding to events triggered by the user or another part of the program	

```
when [left arrow] key pressed
point in direction (-90)
move (10) steps
```

Object-oriented programming	Each sprite can have its own scripts and data. (However, Scratch has no classes and no Inheritance.)	 face 3 costumes 2 scripts bird 2 costumes 3 scripts
User interface design	You can design interactive user interfaces in Scratch – for example, using clickable sprites to create buttons.	when Sprite1 clicked change brightness effect by 25 wait 0.2 secs change brightness effect by -25 play drum 48 for 0.25 secs
List (array)	The list blocks allow for storing and accessing a list of numbers and strings. This kind of data structure can be considered a "dynamic array."	add bread to food add red apples to food set counter to 1 repeat length of food say item counter of food for 2 secs change counter by 1
Cloning	*create clone* makes a copy of a sprite that exists until the project stops running. You can use it to dynamically create many copies of the same sprite with the same code.	when clicked repeat 50 create clone of myself change x by 10 when I start as a clone forever change color effect by 1
Physical sensing	Blocks such as loudness allow interactions with microphones and other physical interfaces.	forever set size to loudness * 4 % wait .01 secs