

Part 2



Chapter 1



Introducing Scratch

Learn how to create animations, interactive stories, and games through drag-and-drop way using the computer programming language Scratch. Scratch provides an intuitive interface that makes learning to program fun, easy, and well-suited as an educational tool.

• After completing this chapter you will be able to know:

- about Scratch & its origin;
- how to identify the scratch use or purpose;
- what are the 21st Century learning skills;
- how to enumerate the programming concepts and limitation of Scratch;
- the Scratch interface; and
- the Paint Editor features.

A. INTRODUCING Scratch

Scratch provides everything needed to begin developing computer games, multimedia presentations, interactive stories, graphic artwork, and computer animation. Scratch can be used to play digital music and sound effects. Scratch's building block approach to programming sets it apart from other programming languages. This makes Scratch easier to learn. And yet Scratch provides plenty of programming power, allowing you to build very powerful application projects.



We don't need to artificially restrict Scratch to the classroom though it makes a fantastic teaching tool. Anyone with a desire to learn a programming language can use Scratch as an introductory language. Perhaps you've tried other languages, such as Ruby, PHP, Java, or Python and had trouble getting started for one reason or another. Even if you can barely create a presentation using PowerPoint, you'll find comfort in Scratch's building-block approach to programming.

After using Scratch, programming will make sense. It will seem easy. It will bring a smile to your face. Whether you want to improve your digital literacy skills by learning to program or you want to learn a new tool to help you in the future, here's to happy Scratchin'.

B. SCRATCH'S ORIGIN

Scratch is developed by the Lifelong Kindergarten group at the MIT Media Lab. See <http://scratch.mit.edu> for more information. The Lifelong Kindergarten group at the MIT Media Lab developed Scratch as a teaching language specifically for 8 to 16 years old learners, but there's nothing stopping the rest of us from enjoying the Scratch experience and sharpening our 21st century learning skills.

Scratch Home: MIT Laboratory



Scratch Website



C. Training the 21st Century Learning Skills

According to Confucius “He who learns but does not think, is lost! He who thinks but does not learn is in great danger”. We should not stop learning so that we can develop the skills that would help us to keep from the increasing demands of digital environment.

Using Scratch, we learn how to design, think, collaborate, communicate, analyze, and program in a computer language. Many of these ideas incorporate 21st century learning skills. If you'd like more information about 21st century learning skills, visit the Partnership for 21st Century Skills web site at <http://www.21stcenturyskills.org>.

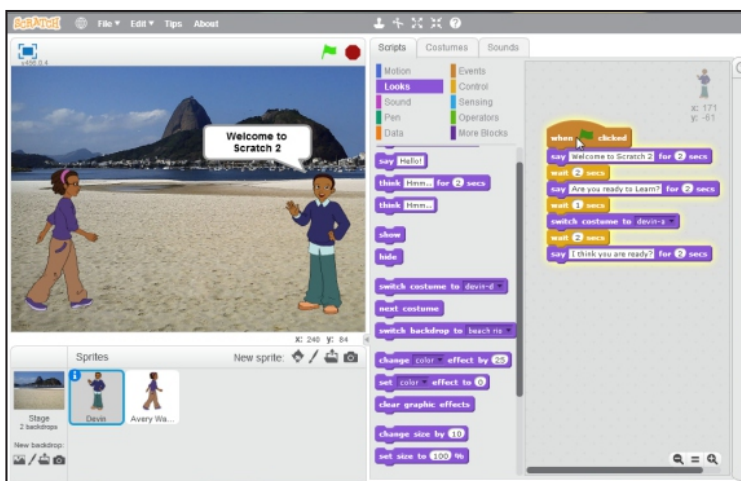


By the time we make our cat dance for the first time, we'll forget all about the academic research and theories behind Scratch. Instead, we'll be focused on discovering the next idea.

D. The Scratch Use

There are lots of possible ways for you to use Scratch. You can use your imagination and creativity to create amazing programs. Here are few ideas to get you started.

Use Scratch to teach yourself how to program. That's the obvious one. Use Scratch to demonstrate math concepts. For example, when it's time to teach variables, set up an interactive game that uses a variable to keep score or moves based on the variable data. Scratch can also demonstrate the X and Y coordinates.



I want to inspire you to read and write. Find a story and animate each scene, or encourage them to animate the story. Turn their persuasive essays into a Scratch project.

Do you love to play video games? Try to create your own game with Scratch.

I'm sure you've got a lot of ideas flowing in your mind by now. Keep writing them down no matter how hard, easy, obvious, or silly they seem to be. The next one might be your best idea yet.



E. Programming Concepts

With Scratch, we will learn how to turn our imaginations into games, stories, and animations, and in the process, we will learn some common programming concepts.

We have to familiarize the programming concepts below in order to get started.

<u>Concept</u>	<u>Description</u>
Program Design	When we design a program, we turn our imagination into something that can be shared with others. We create the flow of the program, the interface, and the actions each sprite takes to tell our story.
Loops (Iteration)	Loops iterate through a series of steps for as long as we tell the program to run the loop. We can use other programming concepts, such as conditional statements, to control the loop.
Conditional Statements	Check to see if a statement is true. For example, if $4 > 0$ is a conditional statement.
Boolean Logic	Boolean logic operators include and, or, and not. If $4 > 0$ and $4 > 1$ is one example.
Variables	Variables store text or numbers for reuse in the program. They come in global and local types. For example, if $x > 0$ creates a conditional statement where x is 0, 1, 2, or anything else we define.
Arrays (Lists)	Arrays are similar to variables in that they store dynamic data. However, a list stores multiple values in the same way a grocery list stores a group of items.
Events	An action in the program prompts another part of the program to take an action. For example, when the Space bar is pressed, the sprite hides.
Synchronization and Coordination	Programming a sprite to receive a broadcast message from another sprite coordinates a cause and effect. Broadcasting a message and waiting for all the other sprites to act on the broadcast synchronizes the action.
Threads	Creating two scripts to run on the same control enables parallel execution. For example, programming four different sprites to pixelate when the flag (green in color) is clicked creates four threads.
Dynamic Interaction	Dynamic interaction provides real-time input into the Scratch program in order to manipulate the sprite in some way. For example, the position of the mouse is always known, so we can create a sprite that always follows the mouse position.
Random Numbers	Random numbers are picked from the range we specify.

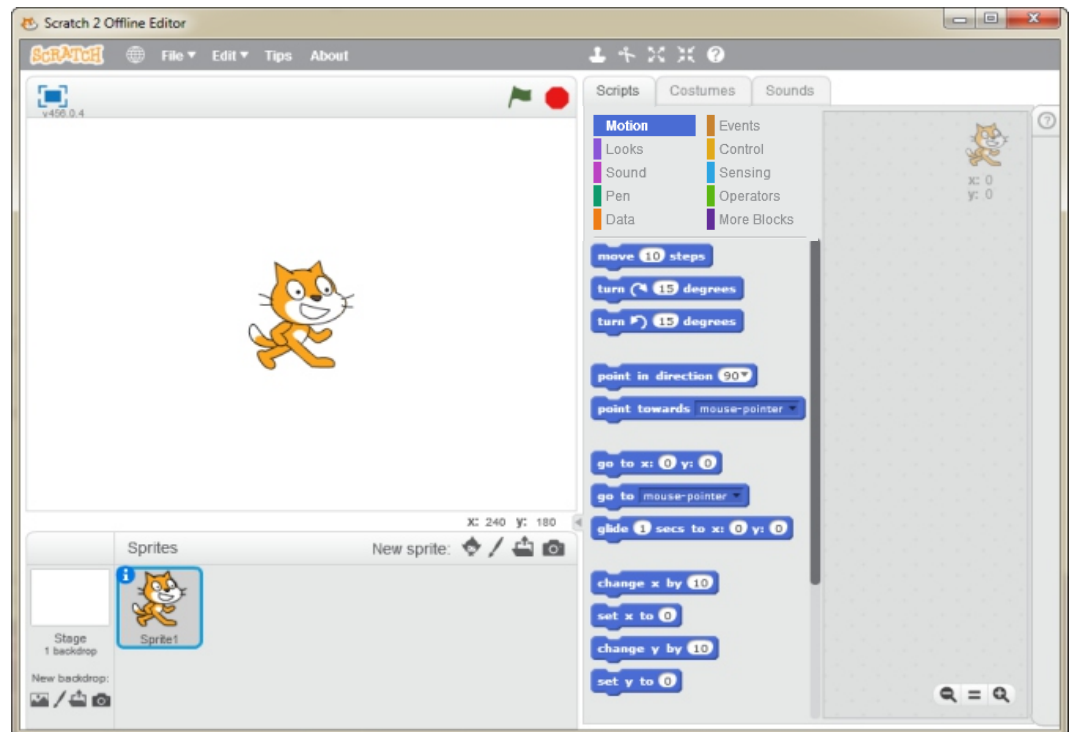
F. Scratch Programming Limitations

As of Scratch version 2.0, there are a few limitations with the language. As taken from Scratch's Programming Concepts guide, here are the concepts Scratch does not cover: functions, recursion, exception handling, file input/output, inheritance, parameter and return values, and defining classes of objects.

G. The Scratch Interface

For those of us with a desire to use geek terms, Scratch provides an Integrated Development Environment (IDE) that enables us to design, program, and run our projects. Don't worry; we will just call it the Scratch interface from this point forward. You can see it in the following screen shot:

In the following units, we will become familiar with the parts of the Scratch interface, so we don't need to spend a lot of time reviewing what each button does. Let's instead stick with the big concepts.

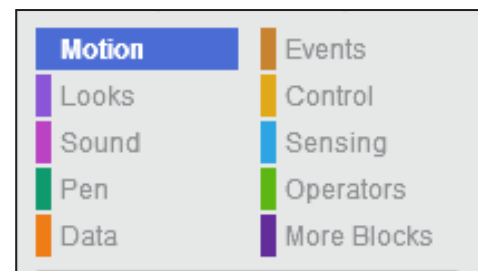


G.1 Palette of Blocks

We will review the Scratch interface from left to right, everything we need to create a project is readily accessible. To the left side of the interface, we have categories of blocks that are grouped by the kinds of tasks they perform. They are Motion, Looks, Sound, Pen, Control, Sensing, Operators, and Variables.

Throughout the book, we'll refer to these categories of **blocks** as **palettes**.

The palette of blocks available to us as Scratch programmers is analogous to the palette of colors an artist mixes when creating a painting. Each type of block is color-coded so that we can easily identify them in our scripts.



G.2 Writing Scratch Script

When we create our Scratch programs, we build a group of scripts that tells our story. Instead of using words as you're used to reading them on this page, we'll build our scripts from the palette of blocks.

We'll drag, drop, and snap them into place in the Scripts area to create our story. The following screen shot (right) shows a script that was taken from one of the sample projects included with Scratch:

If we read the blocks shown in the screen shot from top to bottom, we should have a good idea of the story this sprite tells. Who doesn't love a knock-knock joke? To see our story play out, we watch the stage.



G.3 Watch the Stage

When it comes time to review the script, we watch it on the stage. It's here that we get to see our ideas turn into reality.

And just like the stage at the theater, we can see from our screen shot that we can have a cast of characters to entertain us.

Scratch even provides a built-in image editor to help us create and modify our characters, which we call sprites.

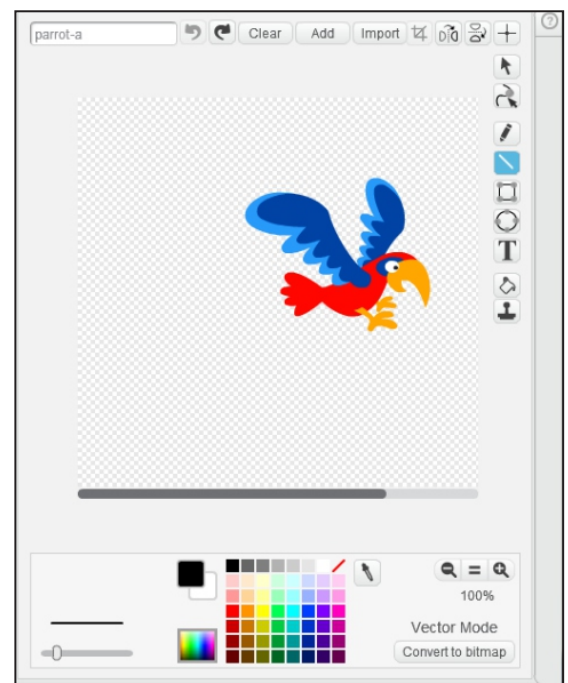
G.4 The Paint Editor

Scratch includes a simple image editing environment called the Paint Editor that allows us to apply text, color, and shape to our sprites and backgrounds.

The Paint Editor allows us to do the following:

- Create shapes and text
- Import and edit images
- Apply color treatments
- Resize, rotate, and flip an image

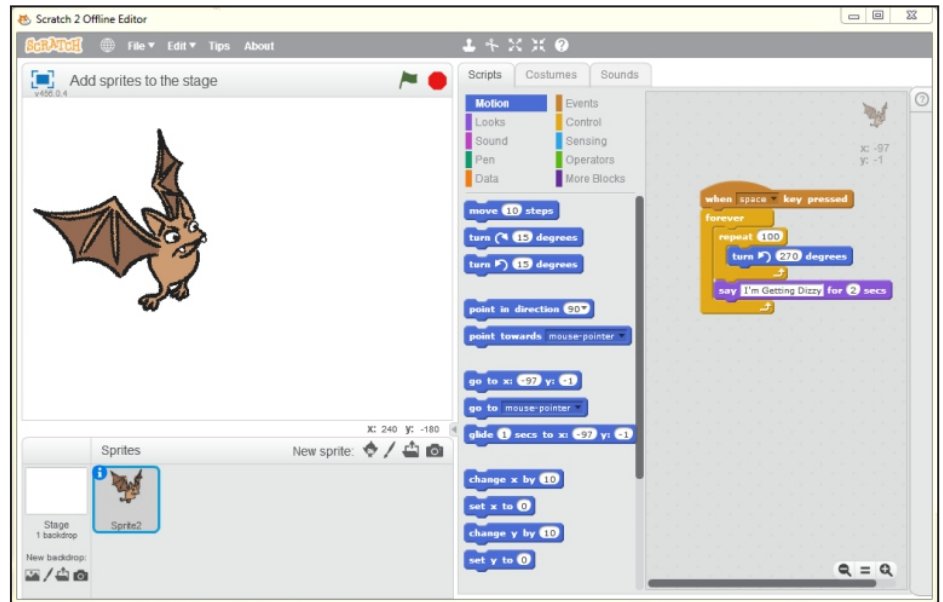
The Paint Editor is available from multiple points within the Scratch interface, as we'll see later in the book.



G.5 Play Around with the Interface

The structure of the Scratch interface makes it easy for us to tinker or play around and explore ideas. As we create, we evaluate our work and determine if the results meet our expectations. It's very easy because everything happens in one interface.

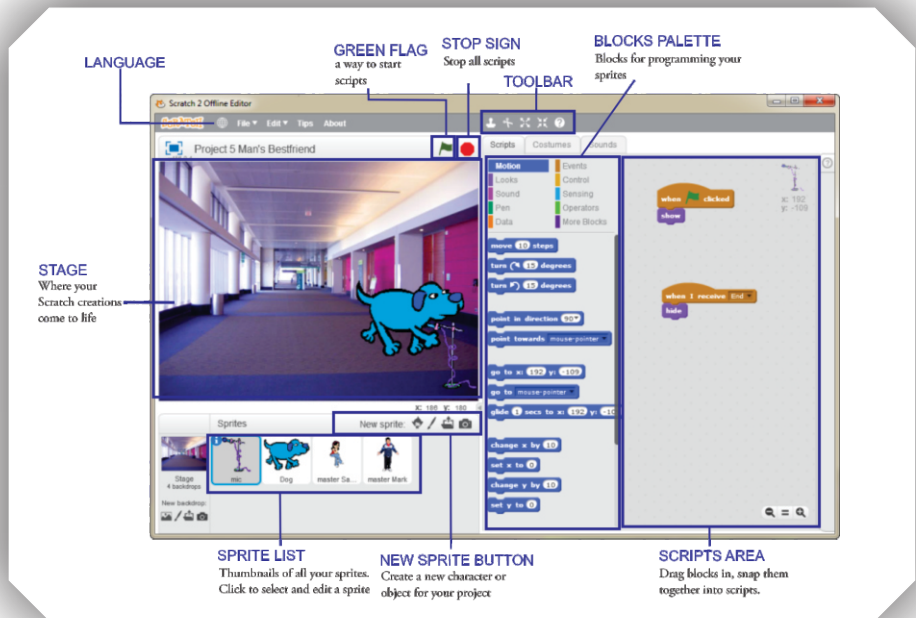
We don't have to compile code, switch windows, upload files to a server, or encounter any number of obstacles to see if our code works. Scratch enables us to modify the program as it runs and sees the results in real time.



H. Scratch Essentials

This section will discuss comprehensively the essential concepts that you need to know in making a program. The Scratch interface will be discussed in detail.

You will know in detail the function of sprite, stage, script and blocks. You will start using Scratch language in making a program. At the end of the chapter, you will be able to demonstrate the following:

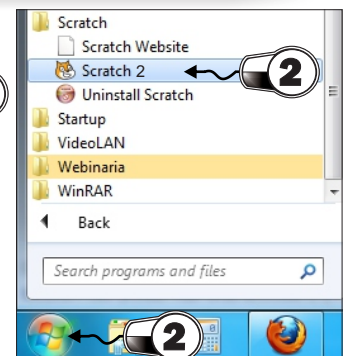
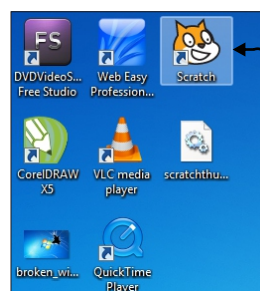


H1. Scratch Usual Routine

Loading Scratch

To start working with Scratch, it must be opened first for you to enjoy making projects.

- 1 Double click the scratch icon in your desktop.
or
 - 2 Click Start button and search scratch.
The scratch menu pop-up in the screen ready for your program project.
- Part 2 Scratch Programming



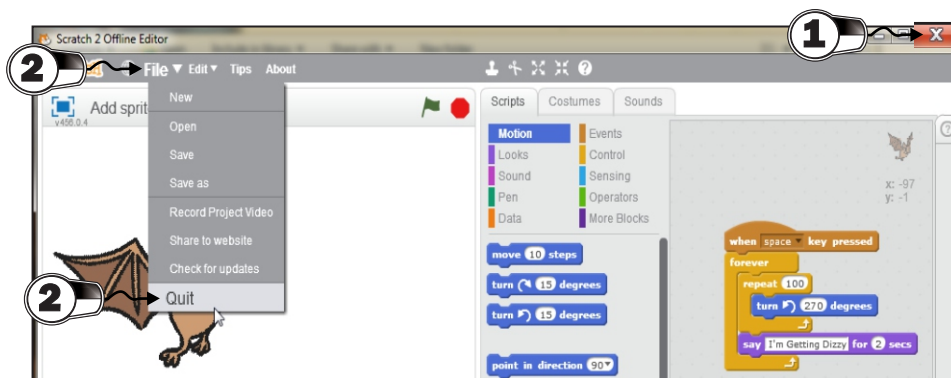
Closing Scratch

1 Click the close button (x) if you are done making your program.

or

1 Click the File menu

2 Select and click Quit.



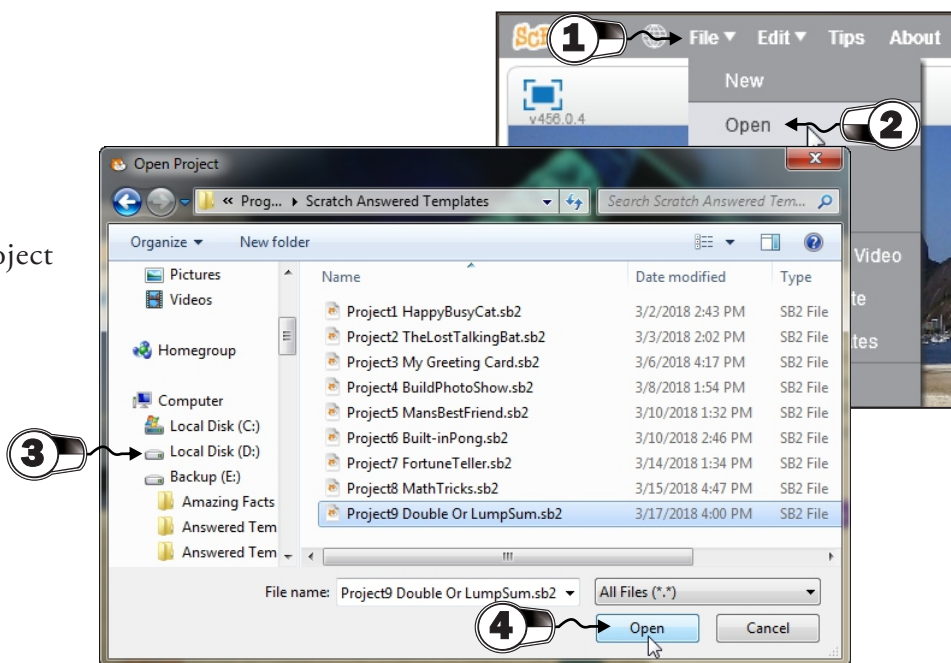
Opening a Scratch Project

1 Click the File tab.

2 Click Open.

3 Choose the location of the project in the computer.

4 Click Open.



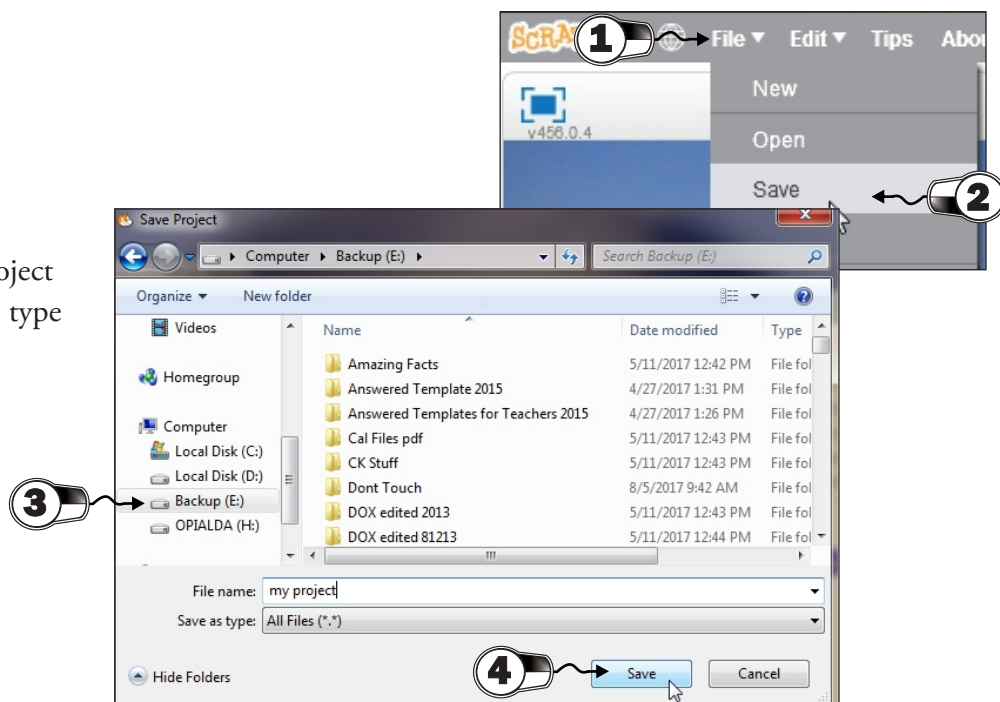
Saving a Scratch Project

1 Click the File tab.

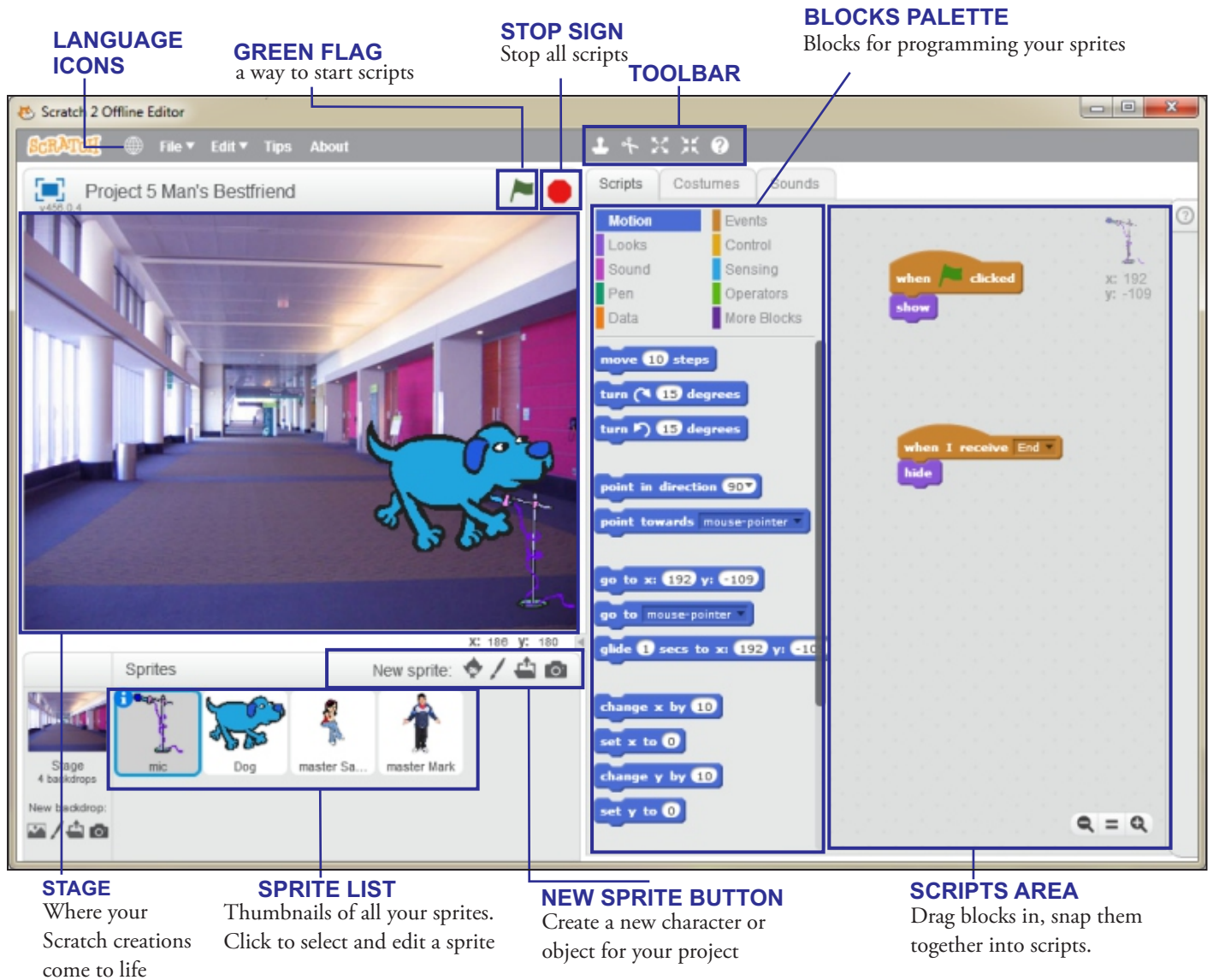
2 Click Save.

3 Choose the location of the project where you want to save. Then type the name of the file.

4 Click Save.





H2. The Scratch Interface



The Current Sprite Info



To display the current sprite info, right-click on the sprite then close info or click the (i) icon. Current Sprite Info shows a sprite's name, x-y position, direction, lock state, and pen state. You can type in a new name for the sprite.

The sprite's direction indicates which direction the sprite will move when it runs a move block (0=up, 90=right, 180=down, -90=left). This icon  shows the sprite's direction. You can drag this line to change the sprite's direction. Set the direction back to its original state (direction=90) by dragging the .

ROTATION STYLE

Click the Rotation Style buttons to control how the costume appears as the sprite changes its direction.

Rotate: The costume rotates as the sprite changes direction.

No-rotate: The costume never rotates (even as the sprite changes direction).

Left-right flip: The costume faces either left or right.

Toolbar

Click on the Toolbar to select a tool, then click on other objects to perform an action.

Duplicate (). Duplicate sprites, costumes, sounds, blocks, and scripts.

Delete (). Delete sprites, costumes, sounds, blocks, and scripts.

Grow (). Make sprites bigger.

Shrink (). Make sprites smaller.


Block help (). Describe a selected block or any section of the Scratch interface.

Menu

Language icon (): Changes the default language of scratch interface.

File menu (): It allows you to create a new(**New**) project, open an existing project, and save(**Save now**) projects to the Scratch Projects folder or to other locations.


Edit menu (): It provides several features for editing the current project.

Tips icon (): It allows you to access a Help page with links to reference materials, tutorials, and frequently asked questions. You can also access a page with all the Scratch help screens.

About icon ()

Green Flag

The Green Flag provides a convenient way to start many scripts at the same time.


Click the Green Flag  (at the top-right corner of the stage) to start all scripts that have  at the top.

The Green Flag remains highlighted while the scripts are running.

New Sprite Buttons

When you start a new Scratch project, it begins with a single cat sprite.

To create new sprites, click on these buttons:

() Paint your own costume for a new sprite using the Paint Editor.

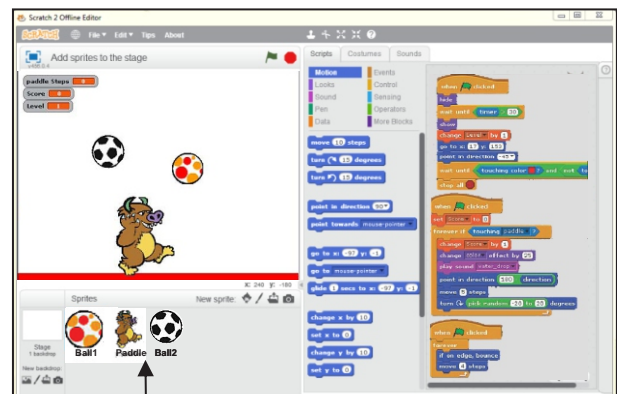
() Select a costume for a new sprite or import an entire sprite from library.

() Upload sprite from file.

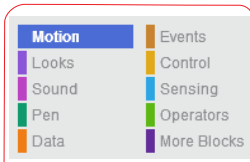
If you want to delete a sprite, select the scissors from the Toolbar and click on the sprite. Or right-click on the sprite and select delete from the pop-up menu.

Sprite List

It displays thumbnails for all sprites in a project. The name of each sprite appears below its thumbnail.



Sprite List



All Scratch Blocks with Functions

The Scratch blocks are organized into eight color-coded categories: Motion, Looks, Sound, Pen, Data, Events, Control, Sensing, Operators, and More Blocks.

MOTION

FUNCTIONS

- move 10 steps** Moves sprite forward or backward.
- turn 15 degrees** Rotates sprite clockwise.
- turn 15 degrees** Rotates sprite counterclockwise.
- point in direction 90** Points sprite in the specified direction. (0=up, 90=right, 180=down, -90=left).
- point towards** Points sprite toward mouse-pointer or another sprite.
- go to x: 0 y: 0** 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.
- set rotation style left-right** Turns sprite rotate horizontally(left-right), flip vertically (all around)
- ☐ **x position** Reports sprite's x-position. (Ranges from -240 to 240)
- ☐ **y position** Reports sprite's direction. (0=up, 90=right, 180=down, -90=left)
- ☐ **direction** Reports sprite's direction is heading to.

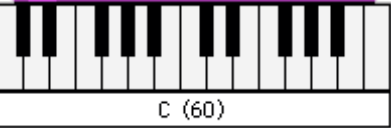
LOOKS

FUNCTIONS

- switch costume to costume2** Changes sprite's appearance by switching to different costume.
- next costume** Changes sprite's costume to next costume in the costume list.
- switch backdrop to backdrop1** Changes Stage's appearance by switching to a different background.
- ☐ **backdrop name** Reports Stage's current background name.
- ☐ **costume #** Reports sprite's current costume number.
- say Hello! for 2 secs** Displays sprite's speech bubble for specified amount of time.
- say Hello!** Displays sprite's speech bubble.
- 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.
- 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.
- 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 FUNCTIONS

- 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** Plays a sound and waits until the sound is finished playing before continuing with next block.
- stop all sounds** Stops playing all sounds.
- play drum** 48 **for** 0.2 **beats** Plays a drum sound, selected from pull-down menu, for specified number of beats.
- play note** 60 **for** 0.5 **beats** 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 beats.
- 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.

PEN FUNCTIONS

- 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.
- 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).
- 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)
- 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.

CONTROL FUNCTIONS

- wait 1 secs** Waits specified number of seconds, then continues with next block.
- repeat 10** Runs the blocks inside a specified number of times.
- forever** Runs the blocks inside over and over.
- if then** If condition is true, runs the blocks inside.
- if then else** If condition is true, runs the blocks inside the if portion; if not, runs the blocks inside the else portion.
- wait until** Waits until condition is true, then runs the blocks below.
- repeat until** 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.
- stop all** Stops all scripts in all sprites.

when I start as a clone

Tells a clone what to do once it is created.

create clone of myself

Creates a clone of the specified sprite.

delete this clone

Deletes the current clone.

SENSING FUNCTIONS

touching ?

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

touching color ?

Reports true if sprite is touching specified color.

color is touching

Reports true if first color (within sprite) is touching second color (in background or another sprite).

ask and wait

Asks a question on the screen that causes the program to wait until Enter key is pressed or check mark is clicked.

answer

Reports keyboard input from the most recent use of the ask and wait.

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.

username

Reports the username of the viewer.

reset timer

Sets the timer to zero.

timer

Reports the value of the timer in seconds. (The timer is always running.)

current minute

Reports the current time.

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.


set video transparency to 50 %

Sets the video transparency to the specified percentage.

 Turns the video camera on.

 Reports true if specified key is pressed.

 Reports an attribute of the sprite or stage.


 Reports the no. of days since the year 2000.

OPERATORS FUNCTIONS

 Adds two numbers.


 Subtracts second number from first number.

 Multiplies two numbers.

 Divides first number by second number.

 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.

 Reports true if both conditions are true.


 Reports true if either condition is true.

 Reports true if condition is false; reports false if condition is true.

 Concatenates (combines) strings.

 Reports the number of letters in a string.

 Reports the letter at the specified position in a string.

 Reports result of selected function (abs, sqrt, sin, cos, tan, asin, acos, atan, ln, log, e[^], 10[^]) applied to specified number.

 Reports remainder from division of first number by second number.

 Reports closest integer to a number.

EVENTS

FUNCTIONS



Runs script below when Green flag is clicked.



Runs script below when specified key is pressed.



Runs script below when sprite is clicked.



Runs script below when sprite is clicked.



Runs script below when it receives specified broadcast message.



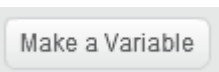
Sends a message to all sprites, then continues with the next block without waiting for the triggered scripts.



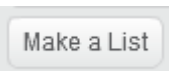
Sends a message to all sprites, triggering them to do something, and waits until they all finish before continuing with next block.

DATA

FUNCTIONS



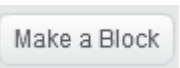
Allows you to create and name a new variable. When you create a variable, the blocks for that variable will appear. You can choose whether the variable is for all sprites (global) or just for one sprite (local).



Allows you to create and name a new list. When you create a list, the blocks for that list will appear. You can choose whether the list is for all sprites (global) or just for one sprite (local).

MORE BLOCKS

FUNCTIONS



Allows you to create, name define a block.



Allows you to add extensions such as Lego WeDo version 1.0-2.0, PicoBoard and etc. This is applicable for you are going to program external devices such as robots.