THE SUPER SCRATCH EDUCATOR'S GUIDE

Here are a few notes to help you get started with your class, group, or after-school meet up!

UNDERSTANDING THE *SUPER SCRATCH* RESOURCES

First, download the book's **Resource** files from *http://nostarch.com/scratch*, and put them in a place where students can easily find them on every computer.

There are **two** project files for each chapter in this book: One version is complete and can be played right now; the other has all the characters, sprites, and sound effects but no programming. Depending on your time, as well as your students' age and ability, you can use one set of projects or the other. All of the completed project files are in a folder called *Complete Games*.

Note: Feel free to delete the Complete Games *folder, if you want your students to build the projects from the ground up.*

Things going poorly or have limited time with your class? Try loading the completed game and experiment. Trial and error is a great way to start programming, and many programmers get their start as "plagiarists."

Have students who want to use the sprites from the book in their own projects? No problem. Just open the original project and then drag the sprite into the student's Backpack, located at the bottom of the page (see Chapter 2). Or you can right click a sprite in the Sprite list and choose **Save to Local File**.

Let Your Kids Get *Creative!*

Make sure that students get a chance to imagine and build their very own, original games or stories after they've tried re-creating a game or two from the book. You'll want to make sure kids get a chance to play with Scratch, not just duplicate the programs in this book. Scratch is all about being creative and making your ***OWN*** cool projects.

Students can check out other folks' work at *http://scratch.mit.edu* and remix their projects into something entirely new!

Where Can I Get Support?

MIT has a forum for educators at *http://scratched.media.mit.edu.* It's a great place to ask questions of your fellow teachers and share your success stories.

Please share any feedback you have on this document or *Super Scratch Programming Adventure!* with the editor of the book, tyler@nostarch.com. We welcome your comments on the projects on the Scratch website, too! Visit *http://scratch.mit.edu/users/nostarch.* Did a student create a cool project in class? Share it with us, and we'll add it to the *Super Scratch* gallery of projects.

SHOULD WE USE SCRATCH 1.4 OR SCRATCH 2.0?

If you have access only to older or "slower" computers, you should use Scratch 1.4, not the online version (Scratch 2, available at *http://scratch.mit.edu*). Download Scratch 1.4 here:

http://scratch.mit.edu/scratch_1.4/

There's also a beta offline version of Scratch 2 available here:

http://scratch.mit.edu/scratch2download/

Note: If you're concerned with your students' privacy (or your students reading blogs instead of following along with your lessons), you should also use Scratch 1.4 or the offline version of Scratch 2.

Scratch 1.4 is currently the only version of Scratch that supports the Picoboard, a hardware interface with Scratch.

The biggest practical difference between 1.4 and 2 is the position of various interface elements (the Stage, Programming Palette, and Scripts Area). A few blocks have been moved into new groups, and there are some minor new features as well, like the Backpack.

Here are links to the different project files from the book (the games), compatible with Scratch 2 and Scratch 1.4:

http://nostarch.com/download/Super-Scratch-2-Resources.zip (2 version)

http://nostarch.com/download/Super-Scratch-Resources.zip (1.4 version)

You'll be able to load the 1.4 files into 2.0, but not vice versa. Scratch 2 files are *not* backward compatible.

You and your students should be able to follow along with the programming instructions, no matter which version of the book you own and which version of Scratch you want to play with!

Note: If you'd like a free PDF of Chapters 1 and 2 from Super Scratch Programming Adventure! *(these chapters explain the user interface) for a particular version of Scratch, 1.4 or 2.0, please email info@nostarch.com. We are happy to send you those. Really!*

Stage 1

FILES

You don't need anything for this project besides a computer with an Internet connection. Just visit *http://scratch.mit.edu* and click **Create**.

If this is your first lesson, make sure that all computers for students are functioning, with Scratch ready to use. You'll want to get students' parents' permission to be online, too, and even have students register at *http://scratch.mit.edu* in advance of meeting.

Be prepared for lots of questions!

The goal of this chapter is to familiarize readers with the Scratch user interface and to introduce them to basic programming concepts.

KEY CONCEPTS

* In Scratch, you give short programs (called "Scripts") to sprites, which are the individual characters in the project.

* Click and drag command blocks from the Palette to form stacks of commands—many blocks together make up a program. Make sure you and the students know how to separate, delete, and duplicate programming blocks. You and your students will need to be able to manipulate and improve your programs; ***the rest of the book assumes you can move, delete, nest, and rearrange blocks***

* Sprites can have more than one script attached to them.

* You can test any single command block or stack of blocks by clicking it. Scratch is great for trying things out.

* The placement of sprites on the Stage is based on a *coordinate grid*, like the one you have seen in math class.

* The computer is "dumb"—it does exactly what you tell it to and nothing else. Did the computer surprise you today?

* You will make mistakes! That's okay. Programmers make so many mistakes that they have a special name for fixing them: *debugging*.

FURTHER EXPLORATION

Wondering what each Scratch block does? Check out MIT's guide, the **Getting Started** PDF in the *Resources* file. For block information within Scratch itself, click the **Tips** button at the top of the interface.

Scratch's Pen blocks can be used like the classic "learn to program" tool called LOGO (or the Turtle module in the Python programming). You can explore many different geometry and mathematical functions using the Pen blocks.

Can your students make a "Spirograph" in Scratch? What about making a flower with many repeated petals? Can they plot a function?

FURTHER READING (FOR EDUCATORS AND MOTIVATED KIDS)

Scratch builds on a rich history of programming environments for kids first built in the 1970s at MIT.

http://en.wikipedia.org/wiki/Logo_(programming_language)

Mindstorms: Children, Computers, and Powerful Ideas by Seymour Papert, 1980. ISBN 9780465046744

Turtle Geometry: The Computer as a Medium for Exploring Mathematics by Harold Abelson and Andrea diSessa, 1986. ISBN 9780262510370

Stage 2

FILES

You'll want to upload (**File > Upload**) the blank file ***02 - A Space Odyssey.sb2*** or the complete working game.

Know that this game also doesn't require *any* special sprites in the file—it's totally okay to build the game using artwork of your own creation!

KEY CONCEPTS

* Sprites can have more than one costume—changing costumes lets you create animated effects.

* The Stage has "costumes" called backdrops.

* Scratch can receive user input from the player—this lets you make cool games!

* Help your students with nested command blocks—you need to "fit" the blocks into one another carefully in Scratch 2.

* If you don't want to, you don't have to draw *anything* to write Scratch programs—Scratch comes with built-in graphics that are ready to use.

FURTHER EXPLORATION

* Did your students notice that you can't lose this game? Point advanced students to look at a more complicated version of the same concept, written by Edmond, the author of *Super Scratch Programming Adventure* at *http://scratch.mit.edu/projects/101909/*. How did he improve the original concept? Which version of the game is more fun to play? Kids can write a simple "prototype" of their favorite iPhone or Flash game and then get ideas for improving it in the same way.

* Have some little Picassos wanting a more robust graphics editor? Check out Inkscape (http://inkscape.org/) and GIMP (http://www.gimp.org/), two free graphics-editing tools. Scratch can import sounds and graphics from other programs.

* Many of the "retro" sound effects in this book were generated using the free online tool at http://www.bfxr.net/

Stage 3

Make sure that you actually play the game before asking the students to create it. This is a two-part game—a simple quizzer, followed by a memory puzzle.

Note: This game is not the most interactive game in this book! It's a great start for building a larger puzzle game and for understanding how to control the flow of a game. It's less well-suited for impatient students expecting a dramatic, interactive game.

FILES

You'll want to upload (**File > Upload**) the blank project ***03 - Louvre Puzzle.sb2*** or the complete working game—there are complicated sprites in this game that are ***hard*** to re-create.

KEY CONCEPTS

* You can make sprites say things using the say block.

* Control the flow of your scripts using broadcasts, and learn how to split up big programs into smaller segments using this technique.

FURTHER EXPLORATION

Can your students think of a more exciting puzzle? What about re-creating their favorite board game? What about a matching memory puzzle that has lots of face-down cards?

Stage 4

FILES

You'll want to upload (**File > Upload**) the blank project ***04 - Hack Attack!.sb2*** or the complete working game**.**

This game doesn't require *any* special sprites in the file either—it's okay for your students to build the game using artwork of their own creation!

KEY CONCEPTS

* You can nest Scratch blocks within one another. A conditional (if … then… statement) is a powerful tool used by computer programmers.

* Get familiar with more complicated games, and build a game that's controlled by the player's mouse clicks.

FURTHER EXPLORATION

Add a second virus to the game to make it more difficult. You'll need to tweak some programming!

Ready for a real challenge? Try re-creating Pong, the classic arcade game: https://www.youtube.com/watch?v=XNRx5hc4gYc

Pong took several professional programmers ***months*** to create. How quickly can your students re-create it? It's amazing to think how far the tools for programmers have progressed over the past 40 years. Do we have it "easy" these days…?

Stage 5

FILES

You'll want to upload (**File > Upload**) the blank project ***05 - Rio Shootout.sb2*** or the complete working game **.**

KEY CONCEPTS

* Use variables for keeping score in the game or for anything else that's countable in your games.

* The Ball sprite contains most of the game logic in this chapter (that's a pretty arbitrary decision!), so it has most of the programs.

Stage 6

FILES

You'll want to upload (**File > Upload**) the blank project ***06 - Desert Ralley.sb2*** or the complete working game. This one has complicated sprites that would be difficult to re-create.

KEY CONCEPTS

* Changing the Stage's backdrops creates a cool animation.

* By manipulating a variable, you can make objects scroll across the screen! We needed two Road sprites in this game to make the effect

FUTHER EXPLORATION

Students can make the game go faster, add more obstacles, or even try creating a vertical, top-down *Spy Hunter*-style racing game (https://en.wikipedia.org/wiki/Spy_Hunter).

Stage 7

FILES

You'll want to upload (**File > Upload**) the blank project ***07 - The Maze.sb2*** or the complete working game. You ***absolutely*** need the graphics used in this game for the programming to work—the game logic depends on the consistent coloring of the sprites.

KEY CONCEPTS

* `if touching [color]` blocks allow you to create simple "edge detection" and the ending conditions for this game.

FURTHER EXPLORATION

How could you use `if touching [color]` blocks to create a car-racing game, where your car gets a flat tire when it touches the shoulder of the road?

Stage 8

FILES

You'll want to upload (**File > Upload**) the blank project ***08 - Wizard's Race.sb2*** or the complete working game.

KEY CONCEPTS

* Even simple games can be pretty complicated to program

* We built in logic that helps prevent cheating.

FURTHER EXPLORATION

How could players "cheat" in the other games in this book? How do game designers have to account for their players in this way?

Stage 9

This game is by far the largest and most complicated in the book! Make sure you have enough time to complete the project. It may take several hours or even several sessions, depending on your students' age and patience. Because it's long, you'll want to occasionally test your work. Good points to test your work are called out in the text.

FILES

You'll want to upload (**File > Upload**) the blank project ***09 - Final Fight.sb2***or the complete working game. There are complicated sprites in this game, including multiple sprites for the same in-game characters, animations, and more.

KEY CONCEPTS

* We're pushing Scratch to the envelope now! Kids may have trouble understanding the sprite-swapping mechanic for the Cat sprite.

FURTHER EXPLORATION

You can create unique fight moves for Scratchy or create new attacks for the Dark Wizard.

Want to go even more advanced? How about creating a whole new character and a selection screen? It's rumored that Mitch has a fierce laptop attack.

WHAT NEXT?

Ready to build more programs with Scratch? Email info@nostarch.com for free chapters of another Scratch book called *Learn to Program with Scratch*, which explores computer science concepts using the interface you already know and love.

http://nostarch.com/learnscratch

Ready for a text-based programming language? Check out Python and Jason Briggs's *Python for Kids*.

http://nostarch.com/pythonforkids

Want to try a lighthearted introduction to Lisp, suitable for motivated teens? Check out *Land of Lisp*, an illustrated guide to the classic programming language, or *Realm of Racket*, which uses the beginner-friendly DrRacket development environment.

http://nostarch.com/lisp.htm

http://nostarch.com/realmofracket

ScratchEd

Don't forget to check out ScratchEd, an information-sharing website created for teachers and other educators who use Scratch. Share your success stories, exchange Scratch resources, ask questions, and more.

http://scratched.media.mit.edu/

Please email the editor of the book with any feedback! Thanks so much for reading, and keep on Scratching!

tyler@nostarch.com