

LEARNING WITH SCRATCH

What do students **learn** as they create interactive stories, animations, games, music, and art with Scratch?

For one thing, they learn **mathematical and computational ideas** that are built into the Scratch experience. As students create programs in Scratch, they learn core computational concepts such as iteration and conditionals. They also gain an understanding of important mathematical concepts such as coordinates, variables, and random numbers.

Significantly, students learn these concepts in a **meaningful** and **motivating** context. When students learn about variables in traditional algebra classes, they usually feel little personal connection to the concept. But when they learn about variables in the context of Scratch, they can use variables immediately in very meaningful ways: to control the speed of an animation, or to keep track of the score in a game they are creating.

As students work on Scratch projects, they also learn about the **process of design**. Typically, a student will start with an idea, create a working prototype, experiment with it, debug it when things go wrong, get feedback from others, then revise and redesign it. It's a continuous spiral: get an idea, create a project, which leads to new ideas, which lead to new projects, and on and on.

This project-design process combines many of the **21st century learning skills** that will be critical to success in the future: thinking creatively, communicating clearly, analyzing systematically, collaborating effectively, designing iteratively, learning continuously.

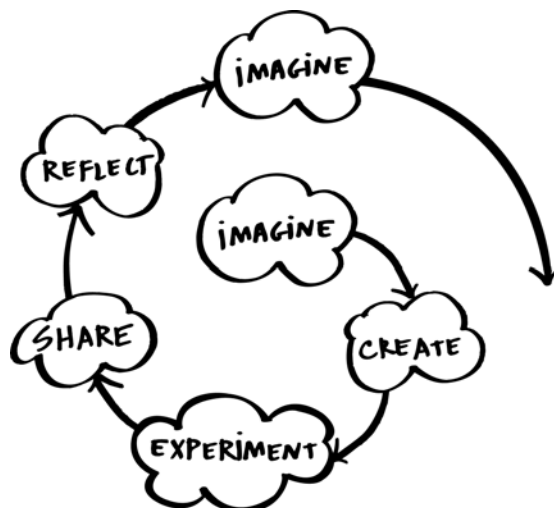
Creating projects in Scratch also helps students develop a deeper level of **fluency** with digital technology. What do we mean by fluency? To be considered fluent in English, Spanish, or other language, you must learn not only how to read but also to write – that is, how to express yourself with the language. Similarly, to be fluent with digital technology, you must learn not only how to interact with the computer but also to create with it.

Of course, most students will not grow up to become professional programmers, just as most will not become professional writers. But **learning to program** offers benefits for everyone: it enables students to express themselves more fully and creatively, helps them develop as logical thinkers, and helps them understand the workings of the new technologies that they encounter everywhere in their everyday lives.

References

Rethinking Learning in the Digital Age
<http://www.media.mit.edu/~mres/papers/wef.pdf>

Learning for the 21st Century (<http://www.21stcenturyskills.org/>)



Lifelong Kindergarten Group, MIT Media Lab

CREATING WITH SCRATCH

People have access to an incredible variety of interactive games, stories, animations, simulations, and other types of **dynamic, interactive media** on their computers today. But, for the most part, these programs are a one-way street: you can only browse and click what others have created; you can't design and create your own.

Scratch changes that, broadening the range of what you can design and create on the computer, making it easier to combine graphics, photos, music, and sound into **interactive creations**. With Scratch, you can create characters that dance, sing, and interact with one another. Or create images that whirl, spin, and animate in response to movements of the mouse. Or integrate images with sound effects and music clips to create an interactive birthday card for a friend, or an interactive report for school.

The name **Scratch** comes from the **scratching** technique used by hip-hop disc jockeys, who spin vinyl records back and forth with their hands to mix music clips together in creative ways. You can do something similar with Scratch, mixing different types of media clips (graphics, photos, music, sounds) in creative ways.

At the core of Scratch is a **graphical programming language** that lets you control the actions and interactions among different

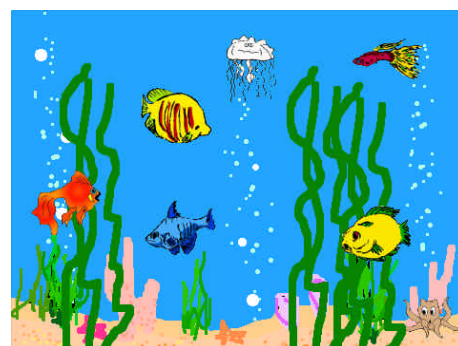
media. Coding in Scratch is much easier than in traditional programming languages: to create a script, you simply snap together graphical blocks, much like LEGO bricks or puzzle pieces.



Once you've created a Scratch project, you can **share** it on the Scratch website, the same way you might share videos on YouTube or photos on Flickr. Or you can embed your Scratch project in any other webpage – for example, embedding an interactive Scratch animation on your MySpace or Facebook homepage.

You can get new ideas for Scratch projects by browsing through projects on the Scratch website. If you like one of the characters or images or scripts in another project, simply **download** the project and use parts of it in your own Scratch project.

Below are snapshots from projects that other people created with Scratch. What do **you** want to create with Scratch?



Lifelong Kindergarten Group, MIT Media Lab

PROGRAMMING WITH SCRATCH

Most people view **computer programming** as a tedious, specialized activity, accessible only to those with advanced technical training. And, indeed, traditional programming languages like Java and C++ are very difficult for most people to learn.

Scratch, a new graphical programming language, aims to change that. Scratch takes advantage of advances in computing power and interface design to make programming more engaging and accessible for children, teens, and others who are learning to program. Key features of Scratch include:

- **Building-block programming.** To create programs in Scratch, you simply snap graphical blocks together into stacks. The blocks are designed to fit together only in ways that make syntactic sense, so there are no syntax errors. Different data types have different shapes, eliminating type mismatches. You can make changes to stacks even as programs are running, so it is easy to experiment with new ideas incrementally and iteratively.



- **Media manipulation.** With Scratch, you can create programs that control and mix graphics, animations, music, and sound. Scratch extends the media-manipulation activities that are popular in today's culture – for example, adding programmability to Photoshop-style image filtering.



- **Sharing and collaboration.** The Scratch website provides inspiration and audience: you can try out other people's projects, re-use and adapt their images and scripts, and post your own projects. The ultimate goal is to develop a shared community and culture around Scratch.



Scratch offers a **low floor** (easy to get started), **high ceiling** (ability to create complex projects), and **wide walls** (support for a wide diversity of projects). In developing Scratch, we put high priority on simplicity, sometimes even sacrificing functionality for understandability.

As students work on Scratch projects, they have opportunities to learn important **computational concepts** such as iteration, conditionals, variables, data types, events, and processes. Scratch has been used to introduce these concepts to students of many different ages, from elementary school through college. Some students transition to traditional text-based languages after getting introduced to programming with Scratch.

Scratch is built on top of the **Squeak** programming language. It was inspired by previous work on **Logo** and Squeak **Etoys**, but it aims to be simpler and more intuitive.

Scratch is an **open-source** but **closed-development** project. The source code is freely available, but the application is developed by a small team of researchers at the MIT Media Lab.

PROGRAMMING CONCEPTS AND SKILLS SUPPORTED IN SCRATCH

In the process of creating interactive stories, games, and animations with Scratch, young people can learn important computational skills and concepts.

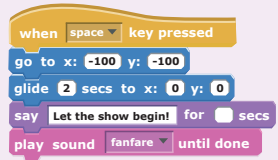

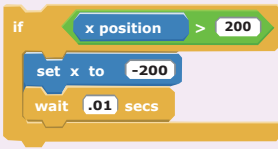


PROBLEM-SOLVING AND PROJECT-DESIGN SKILLS



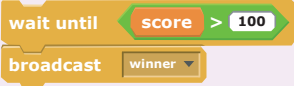




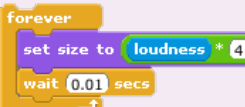

- logical reasoning
- debugging problems
- developing ideas from initial conception to completed project
- sustained focus and perseverance

FUNDAMENTAL IDEAS ABOUT COMPUTERS AND PROGRAMMING

- Computer programs tell the computer precisely what to do, step-by-step
- Writing computer programs doesn't require special expertise, just clear and careful thinking

SPECIFIC PROGRAMMING CONCEPTS

Concept	Explanation	Example
sequence	To create a program in Scratch, you need to think systematically about the order of steps.	
iteration (looping)	forever and repeat can be used for iteration (repeating a series of instructions)	
conditional statements	if and if-else check for a condition.	
variables	The variable blocks allow you to create variables and use them in a program. The variables can store numbers or strings. Scratch supports both global and object-specific variables.	
lists (arrays)	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.”	

Concept	Explanation	Example
event handling	when key pressed and when sprite clicked are examples of event handling – responding to events triggered by the user or another part of the program.	
threads (parallel execution)	Launching two stacks at the same time creates two independent threads that execute in parallel.	
coordination and synchronization	broadcast and when I receive can coordinate the actions of multiple sprites. Using broadcast and wait allows synchronization.	For example, Sprite1 sends the message <i>winner</i> when this condition is met:  This script in Sprite2 is triggered when the message is received: 
keyboard input	ask and wait prompts users to type. answer stores the keyboard input.	
random numbers	pick random selects random integers within a given range.	
boolean logic	and, or, not are examples of boolean logic.	
dynamic interaction	mouse_x, mouse_y, and loudness can be used as dynamic input for real-time interaction	
user interface design	You can design interactive user interfaces in Scratch – for example, using clickable sprites to create buttons.	

PROGRAMMING CONCEPTS NOT CURRENTLY INTRODUCED IN SCRATCH:

- procedures and functions
- recursion
- exception handling
- parameter passing and return values
- defining classes of objects
- file input/output
- inheritance