How Kids Learn When They Program in Scratch

Learning Needs of the New Century

It is universally accepted that in the 21st century, facts and information will matter less. Our children will need the skills to select and process the abundance of information around them. They will require critical reasoning and system-oriented thinking skills to solve real-life problems. They will need intellectual curiosity and a life-long interest in learning. They will need the ability to learn on their own. They will need to apply creativity and multi-disciplinary skills to solve the multi-dimensional real-life problems. They will need to collaborate by sharing and exchanging ideas and building on top of work done by others. They will often be required to employ persistence and self-direction (be entrepreneurial) in achieving their goals. And they will certainly benefit from scientific methodology and mathematical skills.

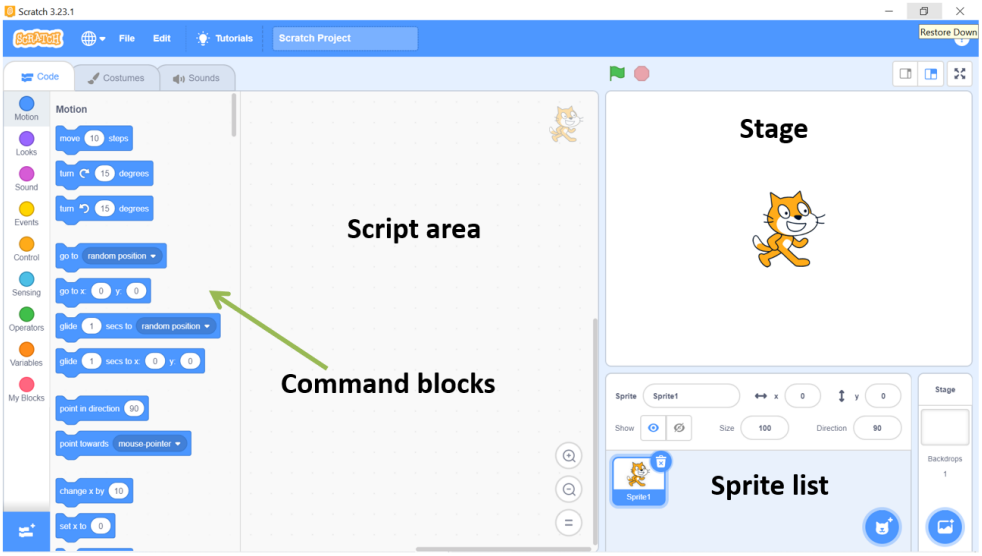
In this article, we will explore an innovative and beautiful idea that is being explored and tried around the world.

The Idea: Computer Programming for Children

To state in a sentence, this new idea involves students undertaking creative design projects on the computer using exciting and entertaining programming environments like Scratch which are specifically designed for the purpose of learning.

The Scratch Programming Environment

Scratch allows you to design your own interactive games, stories, animations, simulations, and other types of dynamic, interactive products on your computer. The following screenshot shows the main Scratch interface.

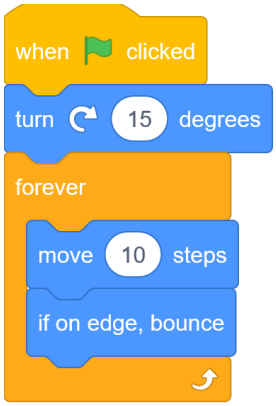


In Scratch, you can combine graphics, photos, artwork, music, and sound to create your product. You can create characters that dance, sing, and interact with one another. Or create images that whirl, spin, and animate in response to movements of the mouse. Or integrate images with sound effects and music clips to create an interactive birthday card for a friend, or an interactive report for school.

The Scratch environment looks very much like a Movie Director's studio. The white screen is where the action happens. The characters that take part in your interactive animation are called sprites and are lined up below the stage. Each sprite behaves according to its own Scratch programs (called scripts) which you build using the Scratch commands in the left-most command palette.

Scripts are built by simply dragging and dropping the command blocks into the Script window. At the core of Scratch is a graphical programming language that lets you control the actions and interactions among different characters.

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. For example, the following Scratch script will make your sprite roam around the screen forever.



How Children Perform Computer Programming

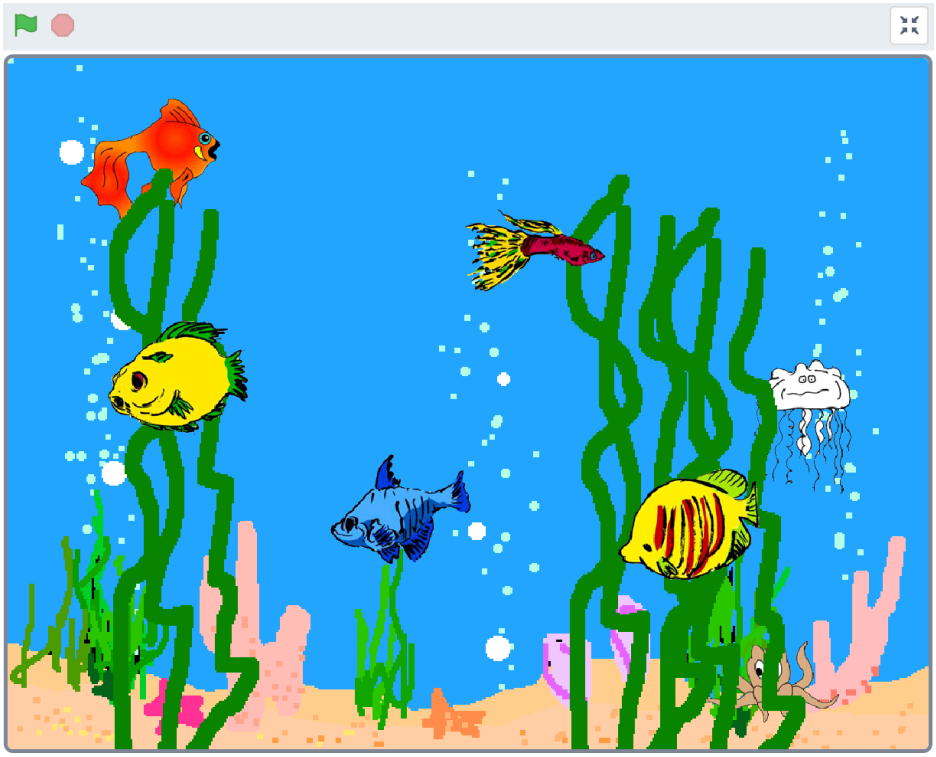
Programming means tapping into the computer's immense power by talking with it directly. Through programming, children use the computer's terrific power to draw artwork, design interactive animation and games, solve mathematical or word puzzles, and even build robots. Such a close friendship with the computer unleashes the children's intellectual ability and creativity. It also allows them to apply concepts of Math and Physics to solve interesting problems.

The choice of the programming language is critical. It is essential to use programming environments like Scratch that have been specially designed with "learning" in mind. The Scratch environment is simple and entertaining, and yet very powerful. Children get immediate feedback when they program. Scratch is called a "low floor and high ceiling" language, because anyone can start programming in it with minimal effort, but its power to deal with complexity is unlimited. Scratch allows the learner to build his/her vocabulary without getting mired in the complexities of syntax and grammar.

So, children do programming projects; and in this activity, learning to program isn't the ultimate goal; the goal is to apply principles of math and logic, to learn critical thinking, and to unleash individual creativity. The focus is on fun, exploration, and challenging projects.

When children do programming, the teacher's role is that of a facilitator or coach. His job is to introduce concepts as needed for the project, and help remove the errors in children's thinking, designs, or programs. Children spend most of their time in hands-on activity and group collaboration, and they enjoy the overall experience immensely.

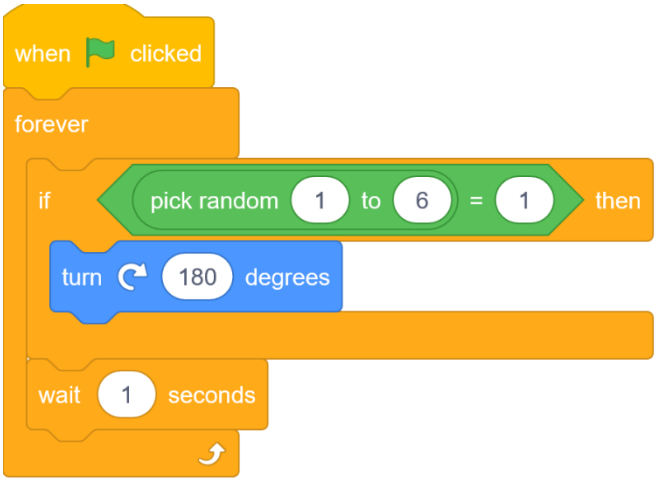
Here is an example of a Scratch project in which using Artwork and Animation an aquarium is created by designing the background, the creatures and their costumes, and the logic of their movements.



When children do programming, they talk with the brain of the computer and thus develop a life-long friendship with the "real" computer.

Projects foster creativity and active learning. They allow conceptual learning to happen gradually and indirectly.

Below see code snippets of this project.



First Impressions

Scratch allows children to design their own games instead of playing readymade games (such as the one shown below).

So, their perspective changes from being just "software users" to becoming "software builders" and it opens up whole new creative horizons. Students create mathematical games, spelling bees, and geography quizzes.

Children gain valuable insights through exploration and through the mistakes they make.

Programming projects are open-ended, and so, every child gets to taste the joy of achievement.

Children get introduced to computational thinking:

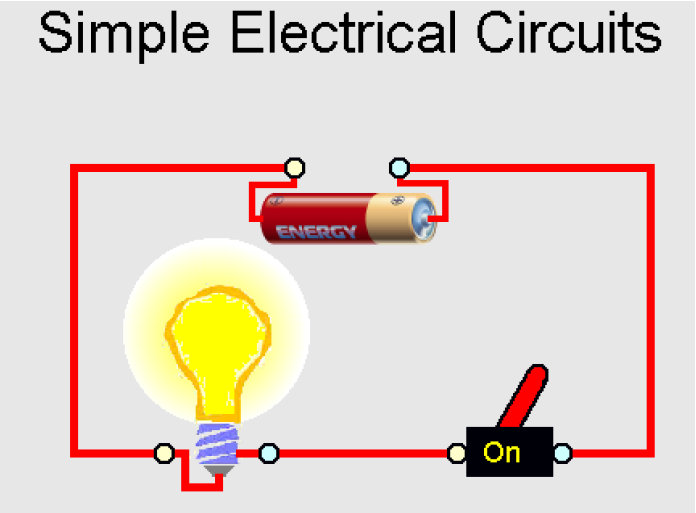
What is Computational Thinking? It is a problem-solving process that includes (but is not limited to) the following characteristics:

- formulating problems in a way that enables us to use a computer and other tools to help solve them;
- logically organizing and analyzing data;
- representing data through abstractions such as models and simulations;
- automating solutions through algorithmic thinking (a series of ordered steps);
- identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; and
- generalizing and transferring this problem-solving process to a wide variety of problems.

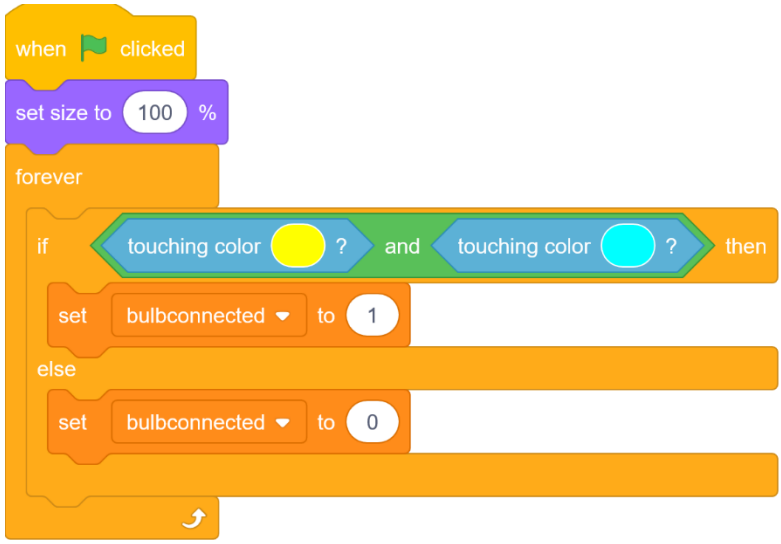
How Learning Happens through Scratch Projects

When children do programming projects, their learning has a sense of purpose.

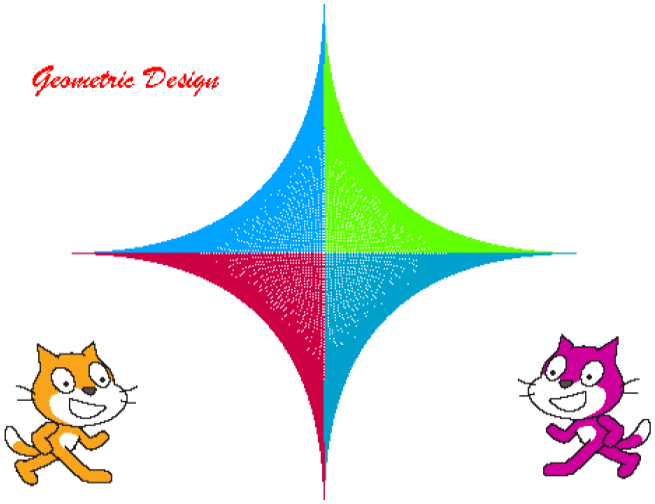
Students must internalize the content of a subject before they can "compute" the knowledge into a computer program. See this example.



In addition to designing the circuit diagrams and the logic of the animation, students get a real feel for the basic principles of electricity. The following is a code snippet.

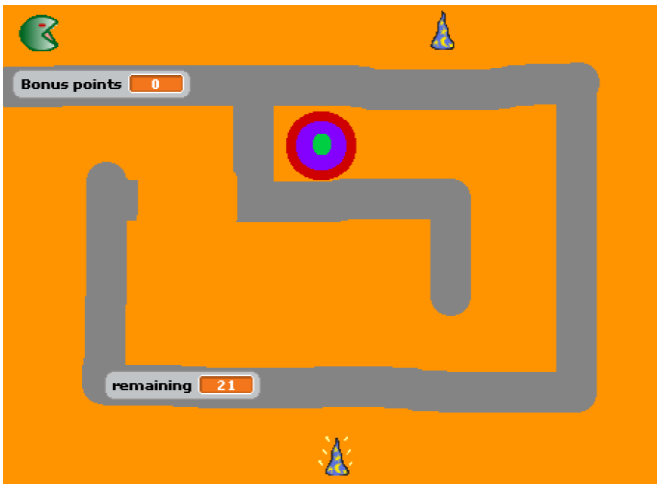


The following project shows how mathematical principles can have meaningful and exciting applications.



In programming projects, such as the one shown here, problem-solving takes center stage. Students discover that the first step in problem-solving is to state the problem clearly and unambiguously. They learn the iterative design process inherent in a computing activity: A) Start with a clearly stated challenge. B) Gather information. C) Formulate a plan. D) Create a working prototype. E) Experiment and debug. F) Gather feedback from others. G) Revise and redesign.

In this game project, there are challenges such as duration, moving obstacles, and enemy attack.

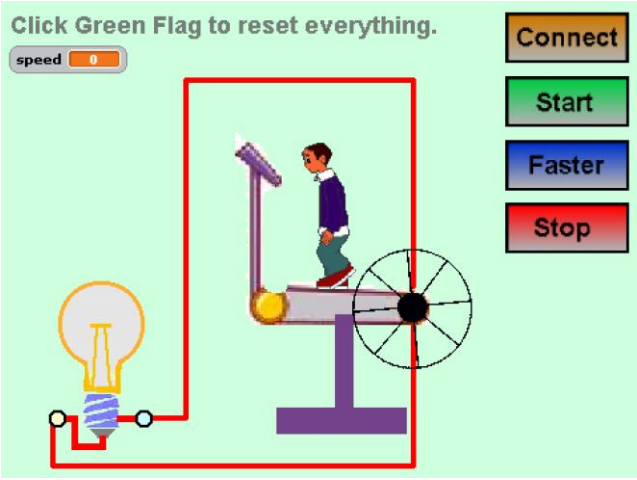


A healthy competitive spirit is seen at play when students create their own challenging games, invite their friends to play, and exchange ideas about how to increase the complexity of their games.

Game projects involve strategy, decision-making, and a whole set of challenging ideas.

Benefits of Creative Design with Programming

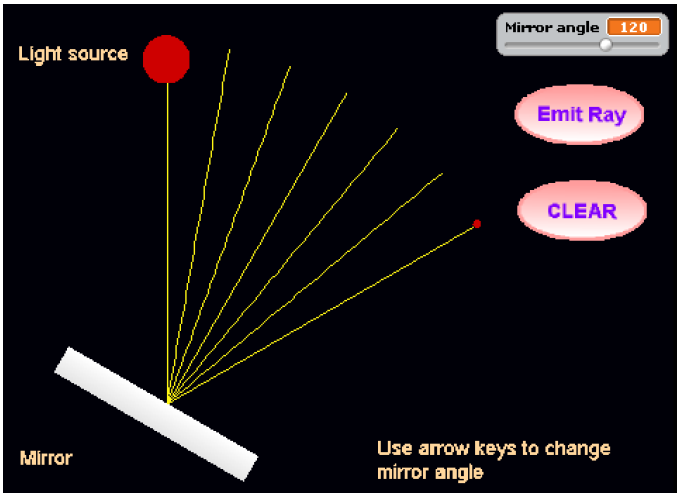
This project demonstrates through interactive animation how electricity can be generated through a spinning wheel.



Computational thinking is a new way of solving problems. In this way there is no right or wrong – there are multiple approaches to a solution, which is achieved through incremental improvement. Computational concepts create interesting new possibilities. Complexity is tackled by dividing big problems into sub-problems.

Children learn the valuable skill of "debugging" – the process of uncovering faults in their own thinking (or its implementation) and designing fixes for them.

This is another interactive animation that demonstrates the principle of reflection of light.



By letting children work on creative design work we sustain the next generation of Creators and Innovators. Programming skill gives the power to solve real-life problems (and thus help the community). It prepares students for future careers & endeavors related to science and technology.

For a future in which technology will touch almost all aspects of life, it is vital that we prepare children to not only use technology but to be reflective about how it works.

This is an example of taking an existing project "Nations and flags" and making your own contribution to it.

This is called "Re-mixing".

Human knowledge is built as a pyramid. Newton famously said, "I stand on the shoulders of giants". Students need to learn how to responsibly take others' work and enhance it to make it more interesting and useful. They do so while giving credit to the original.

Rarely are programming projects solo events; communication and collaboration are their integral components.

Computer Programming for Children: a brief history

Seymour Papert, a mathematician, educator, and computer scientist at MIT is credited to have pioneered the idea of taking computational thinking to children. Since he wrote his famous book "Mindstorms" and designed the Logo programming language, the idea of "learning through programming" has become popular all over the world. Scratch, created at the Lifelong Kindergarten Group at MIT, is believed to be the next generation avatar of Logo.

Since 2009, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) began a multi-phase project supported by the National Science Foundation, aimed at developing an operational definition of Computational Thinking (CT) for K–12 and creating resources and strategies that would support the implementation of CT concepts and skills across grade levels and subject areas.

Several researchers at Carnegie Mellon University (CMU), including the famous professor Randy Pausch, designed ALICE, a programming environment that allows creation of 3-D graphics and animation. The Robotics Institute at CMU has designed a programmable robotics kit for children.

Computer Programming: a compelling alternative to the present situation

As an exercise in comparison, let's consider what our children do with the computer today. The interaction of most children with computers consists of browsing the Internet, connecting with friends using chat and email, playing games, or using ready-made software like word processors. Even the best "computer education" programs in schools involve training students on how to use readymade applications to perform specific tasks like documentation and data formatting. After this type of interaction, the children's impression of the computer is naturally as follows:

- A set of readymade software applications meant for well-defined tasks
- A device that provides access to entertainment (games, social networking)
- A window to information (the Internet)

It is not uncommon for many children to even develop fear of the mysterious powers of the computer.

Conclusion

It is clear that teaching children computer programming is a great idea. Students discover that the computer is a powerful assistant that can help them in any of their favorite subjects. Through the interesting ideas embedded in the programming environments and their focused programming projects, students' interest and ability in "difficult" subjects like Math and Physics improve substantially.

Students learn a great deal through programming. They become active learners (they learn through their own activity and creativity). They learn that answers are not just "right" or "wrong"; real life solutions usually require gradual improvement through "debugging". They learn to deal with complex problems by starting with smaller sub-problems.

They learn to think about and analyze their own thinking, because that is the only way to program computers. Their overall learning process transforms from acquiring facts to creative thinking.

When you teach programming at your school, remember the following important points:

- Programming is for the sake of learning a new way of thinking, and not to teach routine curriculum.

- Choosing an appropriate programming environment (that is interesting, easy to learn, and rich with learning metaphors) is important. Do not use industry languages like C, Java and C#.
- Use proper teaching methodology (in which exploratory hands-on learning takes precedence over blackboard-based rote teaching).

Author: Abhay B. Joshi (abjoshi@yahoo.com)
Last modified: 29 August 2022