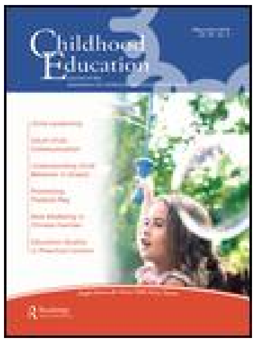

Childhood Education

ISSN: 0009-4056 (Print) 2162-0725 (Online) Journal homepage: http://www.tandfonline.com/loi/uced20

Robots, Programming and Coding, Oh My!

Eugene Geist

To cite this article: Eugene Geist (2016) Robots, Programming and Coding, Oh My!, Childhood Education, 92:4, 298-304, DOI: 10.1080/00094056.2016.1208008

To link to this article: http://dx.doi.org/10.1080/00094056.2016.1208008

Published online: 11 Jul 2016.

Submit your article to this journal

Article views: 67

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
http://www.tandfonline.com/action/journalInformation?journalCode=uced20

Download by: [Ohio University Libraries] **Date:** 23 September 2016, At: 07:53

Robots, Programming and Coding, Oh My!

© ISchmidt/Shutterstock

Today's children need learning opportunities from cradle to career that build the knowledge and skills necessary to thrive in our interconnected and constantly changing world. This means putting 21st century skills, including creativity, innovation, critical thinking, and problem solving, at the center of learning, both in and out of school. As every parent who has seen their young child quickly and adeptly learn to operate a smartphone would undoubtedly agree, no age is too early to begin fostering these skills. Eugene Geist explores how developmentally appropriate coding and programming activities can serve as a vehicle for the exploration, discovery, and innovation that are central to 21st century skill building and essential when preparing children for the jobs of the future.

By Eugene Geist

Eugene Geist is Associate Professor, Early Childhood Education, Ohio University.

The constructive process of programming can be compared to building with blocks. Young children playing with blocks have access to a finite number of different shapes, but endless possibilities for what to make with those shapes. They may experiment with the different shapes, learning about their properties and how they can interact. A parent or teacher can ask questions, but should not interfere with the child's creative process. With older children, a parent or teacher could encourage the child to solve a specific problem using the blocks, such as making a bridge or building the tallest tower possible, but the child should be allowed to use their own creative problem solving to achieve the task and overcome obstacles. When failures occur, the adult would refrain from telling them how to fix the problem and instead engage the child with questions. These same practices can make computer programming developmentally appropriate for young children, keeping it a creative, open-ended, and constructive process.

The block comparison is not a perfect analogy, however. Materials like blocks will give instant feedback. If a block is placed wrong, the tower falls. Programming languages do not directly follow this pattern. Children must plan ahead and predict what their program will do when finally executed. If an error occurs when the program is run, they must go back and find where the problem originated. This is where the unique nature of learning to code becomes especially evident. Computational thinking requires children to predict, plan, and think about sequencing. This practice supports both literacy and mathematics learning.

Computers, smartphones, and tablets have become a part of everyday life over the past 10 years. More and more consumer goods incorporate some type of electronic computer that makes it run or enhances its operations. Refrigerators, coffee pots, and even clothing are becoming "wired." Learning the language of these devices may become as important as learning mathematics, science, or reading. Already, many children know more about how to use computers, smartphones, and tablet devices than adults do (Lucas, Bridgers, Griffiths, & Gopnik, 2014). My son has been able to operate my smartphone since he was 2 years old (Geist, 2014). There may be great benefit in going beyond usage of these devices and actually starting to teach children to "code" or write computer programs for themselves.

However, is it developmentally appropriate to introduce complex coding languages to young children? The developmentally appropriate practice (DAP) framework of principles and guidelines for best practice in the care and education of young children (National Association for the Education of Young Children, n.d.) is grounded in research on how young children develop and learn and in what is known about educational effectiveness. DAP practices promote young children's optimal learning and development (NAEYC, 2014).

Recently, a number of programming languages and systems designed for young children have become available. These languages can be called "educational languages" because they do not have the traditional command prompt of programming languages such as Java and C++ and they do not require textual based coding (Lewis et al., 2014). They are also more accessible, less frustrating, and more rewarding for young students. Because educational languages, such as Scratch, ScratchJr, Code Spells, Tynker, and Lego Mindstorms, are more like games than traditional programming languages, they can open up the world of coding and programming to young children.

Esper, Wood, Foster, Lerner, and Griswold (2014) showed how the game CodeSpells, where children use coding principles to create "spells" to make things happen, engaged elementary school children in computer programming rather than merely memorizing the rules of the language. Programs like Tynker make learning basic programming concepts into a game by requiring users to create a "program" to help a screen character achieve tasks. Other games, like ScratchJr, encourage children to be creative in developing programs to move characters around the screen and even interact with each other. This type of learning to code through game play taps into what many cognitive theorists have suggested about how children naturally learn through active interaction and engagement with their environment.

Some important questions should be considered when designing developmentally appropriate coding activities for young children. With young children, we need to ask "what" and "why," but also the questions "when" and "how."

What Is Coding and Programming?

In the late 1980s, Seymour Papert applied Piaget's theory of constructivism to computer-based learning, creating a programming language called "Logo" (Papert, 1993). Many of the current programming languages today are based on at least the basic principles that he used to design Logo if not the actual language itself (Resnick & Ocko, 1990).

Papert coined the term "constructionism," blending Piaget's constructivism, in which children construct knowledge through active engagement with their environment (Elkind & Piaget, 1991; Piaget, 1973), and the process of building and constructing with physical objects, such as blocks and Legos, and then writing computer programs (coding) to make them "go" (Papert, 1993). Currently, a number of new programming languages and systems offer graphically based coding and programming experiences that are accessible and usable for children as young as 5 years old (Code.org, 2013; Kafai & Burke, 2013).

Why Coding and Programming?

In the adult world, different variations on programming languages (e.g., HTML, Java, Java Script, Perl, C++) require knowledge of a specific syntax. This can be daunting for many novice adults, as well as children. However, with graphical programming systems and languages designed for children (i.e., Scratch, ScratchJr, Lego's WeDo and Mindstorms, Alice, CodeSpells, Play-i, TangibleK), the focus is on the logic and underlying concepts conveyed by the language rather than the language itself (Kafai, 2013). By converting programming to a system of drag and drop blocks, children as young as 5 can understand. Games such as Robot Turtles teach the same concepts without need for a computer. This way, the logical process aspect of programming and coding can be introduced to children at an early age.

Any activity for young children, whether based on technology or not, should be developmentally appropriate. The first step is to ensure that the materials being used are developmentally appropriate for the target age. Teachers and parents need to know if the child understands what is asked of them, knows how to interface with the device in question, and can reasonably be expected to engage in the action needed to produce a response from the device (Couse & Chen, 2010).

The key to keeping coding and programming developmentally appropriate for young children is to ensure the process is creative and constructive, as in the model of Papert's constructionism.

> *Constructionism* also has the connotation of "construction set," starting with sets in the literal sense, such as Lego, and extending to include programming languages considered as "sets" from which programs can be made, and kitchens as "sets" from which not only cakes but recipes and forms of mathematics-in-use are constructed. (Papert, 1993, pp. 142-143)

Just as when playing with blocks and creating towers, children are creating when using codes. The medium may be different, but the creative process is the same.

When to Begin Coding and Programming

To answer the "when" question, we should first consider language and cognitive development at various stages. For preschoolers, exposure to programming may mean open-ended exploration of the link between the digital world of the program and the physical world of the robot. Older children, in kindergarten and primary grades, may begin using the digital program to solve a problem in physical reality by programming procedural systematic steps.

Cognitive development goes through a critical transition between the ages of 5-7, according to Piaget (1973), to a more logical and less intuitive way of thinking. Giving children active experiences in order to stimulate their thought processes supports the development of this shift. Children are moving from pre-operational to concrete operational patterns of thought. Flannery and Bers (2013) found this transitional stage to be a critical time in the development of children's ability to code and program. In their study, they asked young children to program a Lego WeDo robot to "do the Hokey Pokey."

Researchers at the DevTech group at Tufts University and the Lifelong Kindergarten Group at the Massachusetts Institute of Technology developed a programmnig language called ScratchJr, designed especially for children between the ages of 4 and 7 (Flannery et al., 2013; Reilly, 2013). In their lab, they demonstrated that children using their software and accompanying curriculum materials engaged in age-appropriate computer programming and problem-solving while also building on such traditional early childhood experiences as storytelling, numerical and spatial reasoning, creative thinking, and self-expression (Flannery et al., 2013).

How to Teach Coding to Young Children

> How effective and appropriate technology is depends on the selection, use, integration and evaluation of technology tools and interactive media that are responsive to the age and developmental level of the child, individual readiness and interest, and what is appropriate within the context of the families, cultures and community. (National Association for the Education of Young Children & The Fred Rogers Center for Early Learning and Children's Media at Saint Vincent College, 2012)

The key to keeping coding and programming developmentally appropriate for young children is to ensure the process is a creative and constructive one. Keeping in mind the analogy of "construction," like using building blocks to build a city on the carpet, coding seems like a natural outlet for children's creativity. Children need to be engaged in three levels of interaction, according to their developmental levels.

1. *Experimentation and Exploration.* In this phase, children can become familiar with the programming language and the coding environment. Teachers can help children learn the basic functions of different components and how to perform such basic tasks as "dragging and dropping" and entering text. Once children have these basic skills, they should be allowed to experiment with the components to see what they can create. As children may lose interest or become frustrated if their efforts do not produce tangible results, a teacher or more experienced peer may need to offer support and suggestions during this phase.

2. *Construction and Creation.* In this phase, children have learned the basic functions of the programming components and are becoming skilled at linking and chaining commands together to create something or produce a pleasurable action. For example, a child may have used the programming language to draw a picture or a geometric design using repeating elements. The goal at this stage is still one of expanding a child's understanding of the many things that can be done with the programming language and how it can be fully utilized. Children are exploring the power they have to make things happen in the digital environment, either by making things happen on a screen or making a robot move in interesting ways. Children are excited about being able to make the program or robot "do something," even if it is something simple.

3. *Problem Solving.* In this phase, children begin to move beyond simply making the program or robot do "something" and begin to think about solving problems or doing something specific. They think of how to apply their programming skills to a particular goal or task, such as making a robot push a ball into a box or follow a line drawn on the floor. They can begin to design interactive games that can be played by other students. During this phase, teachers can offer support by helping the student in the planning process and by asking questions, such as, "How can you make that happen" or "Where do you think things went wrong?" A teacher could also issue challenges to the students to help them to focus on a specific task or problem to be solved, such as designing a robot that can roll a ball the farthest.

Introducing Coding to Young Children

There are many different ways to introduce coding to children. In this section, I will present three examples from different media types. I have used these systems in classrooms to great effect. The test subject I refer to here, however, is my 6-year-old son, with whom I have used these materials in both formal and informal situations.

• *Robot Turtles.* Robot Turtles does not require a computer or electronic device of any kind. It is a board game developed by a Google engineer to teach preschool children how to think like a computer programmer. The goal of the game is to move a game piece across the game board toward a gem, using cards to make the game piece do various actions. Cards include "forward [walk]," "turn," and "fire laser." The game board presents obstacles made of ice (which can be melted with the laser), boxes (which can be pushed out of the way by the turtle), and stone towers (which are immovable). The goal is to be the first to retrieve the designated gem. On each turn, a child can play one of the cards, moving their turtle according to the card's instruction. After the turtle reaches the gem, the child "reads the program" by looking at the cards played. For example, the child may say, "walk, walk, walk, turn, walk, walk, laser, walk, turn, walk, walk" or "walk 3, turn, walk 2, laser, walk, and then turn, walk twice." One of the most important things children learn from this game is to recognize the repeating patterns in the "programs" and learn how to group them into what programmers call "subroutines." The child I played this game with simply calls them "patterns" and revels in finding them when reading the program after completing the game's goal. He would even stack the cards into their different subroutines while reading them off.

Robot Turtles is appropriate for young children and requires no computer skill to play. Little fingers and hands do not have to manipulate a computer mouse or direct a cursor and there is no textual coding. To the child, this is simply a fun game that can be played with adults or peers. The author of the game even encourages children to make silly sounds when moving their turtle or shooting the laser.

• *ScratchJr and Tynker.* ScratchJr and Tynker are both apps for tablet devices such as the Apple iPad. They utilize the touch-based drag and drop format to encourage children to build strings of commands to make characters on the screen move and, in the case of Tynker, complete tasks. (See Figure 1.)

Tynker is similar to games such as Angry Birds, which present a problem or situation the player needs to solve. Using the tools given, the player must make the character move and complete the task. In Angry Birds, this means figuring out the trajectory and force needed to fling a bird at a structure to knock it down. In Tynker, the player uses coding blocks to tell the onscreen astronaut when to move, jump, and perform a host of other actions. Similar to

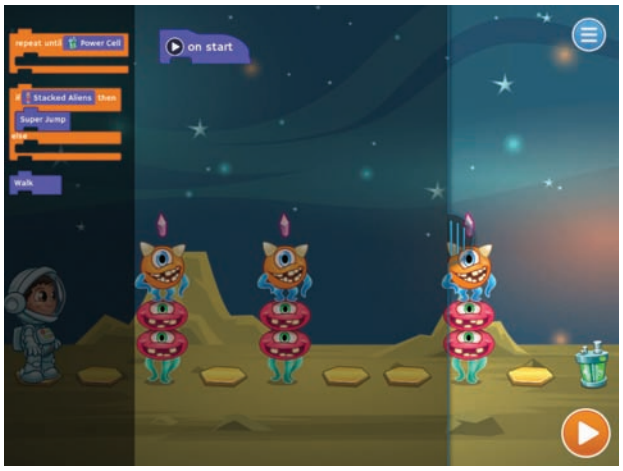

Figure 1

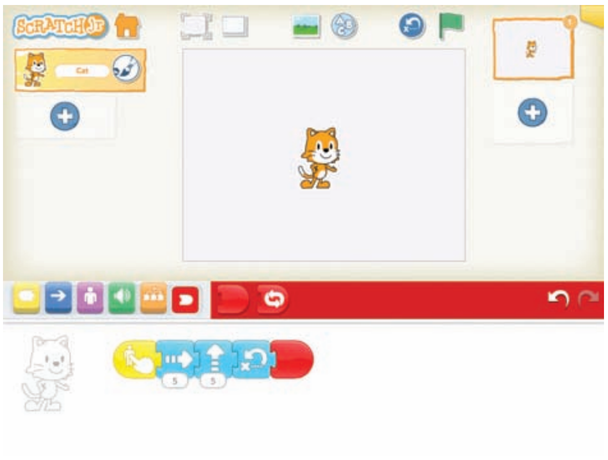

Figure 2

Robot Turtles, children begin to recognize patterns and are rewarded for creating repeating subroutines to complete the task as efficiently as possible. The program is more directive and usually has only one "correct" way of completing the level, which can be frustrating for some younger children and may require some facilitation from an adult. I found that my 1st-grade test subject was able to grasp the ideas with a little facilitation, and could independently complete many of the tasks. He found it so much fun that he stopped watching Saturday morning cartoons to play with it.

ScratchJr, developed by the MIT Media Lab, was designed for younger children. It is more open-ended than Tynker and uses a programming system that is similar to the Lego WeDo and Mindstorms systems. Children are presented with a cat and a toolbox of functions to make the cat move and interact. Children can also add other characters and items with which the cat can interact. Children can do things as simple as making the cat move around the arena or they can design interactive games that require input from the player.

The attraction of ScratchJr is its focus on creativity. Children are encouraged to design and build using the tools that they are given. Many of the skills that were learned by playing Tynker can be applied to ScratchJr. My test subject found the creative aspect to be especially enjoyable and without direction was able to add other characters and create programs to make them move around the arena. He was not timid about adding programming blocks to his program; when his program failed to do what he expected, he was quick to modify it until he achieved the outcome he wanted. At his age, this problem solving was primarily done through trial and error. Nevertheless, both of these programs helped him to recognize patterns and think about how repeating them could help him achieve his goal. (See Figure 2.)

- *Lego WeDo.* Lego WeDo is a version of the Lego Mindstorms system targeted at younger children. The programming system uses graphical blocks similar to those used in ScratchJr. In the case of WeDo, however, the output is a physical structure that is built using Legos. The computer program controls motors and sensors in 3-D reality rather than on a 2-D screen. This can allow the child to think about how the program they are developing in the cyberworld is translated to movement and action in the real world.

The WeDo system comes with some sensors and motors that allow children to build simple robots. My test subject's favorite creation was the Drumming Monkey because he could use the program to trigger the monkey robot to make a sound. He relished the ability to change the duration of the program and to make it repeat indefinitely. He found the physical representation of his on-screen work to be particularly satisfying.

The one limitation to this system is that it is computer based. It does not have a touch screen tablet version at this time. Therefore, children must learn to navigate a computer keyboard and become proficient at using a trackpad or computer mouse.

Conclusion

Coding is the language of the new millennium. It is a language that will be as important for our children to learn as Latin was for scholars in the Middle Ages (Resnik et al., 2009). The Texas legislature has even decided to allow computer programming to count as a foreign language graduation requirement (www.legis.state.tx.us/tlodocs/83R/billtext/html/HB00005F.HTM). Organizations like code.org are pushing for high school students to learn how to code in order to increase their future opportunities in the STEM fields. Todd Park, Chief Technology Officer for the White House, is quoted on their website as saying,

> Technology and computers are very much at the core of our economy going forward. To be prepared for the demands of the 21st century—and to take advantage of its opportunities—it is essential that more of our students today learn basic computer programming skills, no matter what field of work they want to pursue. (code.org)

While many are understandably concerned about the amount of "screen time" young children are exposed to in the early childhood years, coding and programming are much more than passive screen time. Coding can be integrated into other aspects of children's play, explorations, and investigations. A number of resources are available to support children as they build and create animations and interactive games. There are also opportunities to use coding to animate three-dimensional creations and robots. Children can even learn coding by playing board games like Robot Turtles.

At all levels, children need the autonomy to experiment, play, and use their creativity. Whether it is building and programming robots using Lego WeDo and Mindstorms or programming games and interactive experiences using languages like Scratch, ScratchJr, Alice, or Stencyl, coding and programming can support STEM learning for young children in developmentally appropriate ways.

References and Resources

Code.org. (2013). Code.org introduces "Hour of code" campaign to inspire 10 million students to learn to code. *Business Wire (English).* Retrieved from http://www.businesswire.com/news/home/20131014006133/en/Code.org-Introduces-%E2%80%9CHour-Code%E2%80%9D-Campaign-Inspire-10

Couse, L. J., & Chen, D. W. (2010). A tablet computer for young children? Exploring its viability for early childhood education. *Journal of Research on Technology in Education, 43*(1), 75-98.

Elkind, D., & Piaget, J. (Directors). (1991). *Using what we know applying Piaget's developmental theory in primary classrooms* [Video/DVD]. Santa Barbara, CA: Quantum Video.

Esper, S., Wood, S. R., Foster, S. R., Lerner, S., & Griswold, W. G. (2014). Codespells: How to design quests to teach Java concepts. *Journal of Computing Sciences in Colleges, 29*(4), 114-122.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97.

Flannery, L. P., & Bers, M. U. (2013). Let's dance the "robot hokey-pokey!": Children's programming approaches and achievement throughout early cognitive development. *Journal of Research on Technology in Education, 46*(1), 81-101.

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 1-10). New York, NY: Association for Computing Machinery.

Geist, E. (2014). Using tablet computers with toddlers and young preschoolers. *Young Children, 69*(1), 58-63.

Kafai, Y. B., & Burke, Q. (2013). Computer programming goes back to school. *Phi Delta Kappan, 95*(1), 61-65.

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*(4), 245-255.

Lewis, C., Esper, S., Bhattacharyya, V., Fa-Kaji, N., Dominguez, N., & Schlesinger, A. (2014). Children's perceptions of what counts as a programming language. *Journal of Computing Sciences in Colleges, 29*(4), 123-133.

Lucas, C. G., Bridgers, S., Griffiths, T. L., & Gopnik, A. (2014). When children are better (or at least more open-minded) learners than adults: Developmental differences in learning the forms of causal relationships. *Cognition, 131*(2), 284-299.

National Association for the Education of Young Children. (n.d.). *DAP frequently asked questions.* Retrieved September 21, 2014, from www.naeyc.org/dap/faq

National Association for the Education of Young Children and the Fred Rogers Center for Early Learning and Children's Media at Saint Vincent College. (2012). *Technology and interactive media as tools in early childhood programs serving children from birth through age 8.* Washington, DC, and Latrobe, PA: Authors. Retrieved from https://issuu.com/naeyc/docs/ps_technology_issuu_may2012?backgroundColor=

Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas / Seymour Papert* (2nd ed.). New York, NY: BasicBooks.

Piaget, J. (1973). *To understand is to invent: The future of education*. New York, NY: Grossman Publishers.

Reilly, M. (2013). The kindergarten coders. *New Scientist, 219*(2927), 21-22.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60-67.

Resnick, M., & Ocko, S. (1990). *LEGO/logo--learning through and about design.* Cambridge, MA: Epistemology and Learning Group, MIT Media Laboratory.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Play-i. The focus of Play-i is to help very young children learn programming in a way that is fun and developmentally appropriate. It is designed to help children reach their goals without frustrating them to the point where they quit. The idea is that a tangible item makes the concepts of programming accessible to children in a very concrete way. Their Robots, Bo and Yana, are fun for kids to play with, engaging children in a very different way from typical online tutorials.

www.play-i.com/

Scratch and **ScratchJr.** Designed by MIT students and staff in 2003, Scratch is a programming language designed specifically for 8- to 16-year-olds. ScratchJr, developed by Tufts University, is targeted at the younger ages. Scratch is a web-based visual programming language made up of bricks that users drag to a workspace in order to animate characters. Various types of bricks trigger loops, create variables, initiate interactivity, play sounds, and more.

Scratch – http://scratch.mit.edu/
ScratchJr - http://ase.tufts.edu/DevTech/ScratchJr/

Daisy the Dinosaur. From the makers of Hopscotch, a coding language for children 8 and older, Daisy the Dinosaur targets the youngest coders. The interface is similar to Hopscotch, so it will be easy for children to step up to Hopscotch. The Daisy interface is much simpler than Hopscotch. There is only a dinosaur to move and only basic functions to use; this is appropriate for younger students. Additionally, it is an iPad app (available for free in the app store), so the touch interface makes it easier for little fingers to manipulate the blocks.

www.daisythedinosaur.com/

Move the Turtle. This is basically a new version of the old Logo turtle programming language for the iPad and iPhone. Each new level of achievement increases in difficulty and teaches a new command that directs the turtle to reach a star, make a sound, draw a line, etc. A free play "compose" mode lets students move the turtle however they want.

http://movetheturtle.com/

Robot Turtles. Robot Turtles is a board game for children ages 3-8. Designed by a programmer at Google, Robot Turtles is a board game using the turtle metaphor. Children learn the basics of programming while they play. This game can be a good place to start for classrooms with limited technology resources.

www.robotturtles.com/

Lego WeDo. The LEGO® Education WeDo Construction Set is an easy-to-use set that introduces young students to robotics. Lego robotics is very popular in elementary and middle schools as a way to introduce robotics. The Lego Mindstorms EV3 is the standard kit used for Lego's First Lego League, a robotics competition for elementary and middle school teams. The WeDo system is designed to help younger children learn the basics of robotics. It uses a similar programming language that used with the Mindstorms robots and is simple enough for children as young as 5 to understand.

www.legoeducation.us/eng/product/lego_education_wedo_robotics_construction_set/2096

Alice. Alice is an innovative programming environment for animating stories. This free program is designed to be a student's first exposure to object-oriented programming. It allows students to construct fundamental programming concepts in a context that is familiar to them. By creating animated movies and simple video games, children learn how to think like a programmer.

www.alice.org/index.php

Other Resources: Code.org; Codeacademy.com; Lego Mindstorms (www.lego.com/en-us/mindstorms/?dom ainredir=mindstorms.lego.com)