CREATIVE COMPUTING

Harvard Graduate School of Education

UNIT 4 OVERVIEW

THE "BIG IDEA"

Personalization is an important guiding principle in the design of the creative computing experience. By "personalization", we mean both connecting to personal interests and acknowledging that personal interests can vary considerably. There are many ways of knowing and doing – and exploring these multiple ways can help support interest, motivation, and persistence among young learners. In this unit, learners explore some of the advanced concepts and challenging problems associated with game design. An advanced concept or challenging problem can be made more accessible if rooted in activities that are personally meaningful. As an example of the power of context, we turn to a story shared by Mitch Resnick – the director of the Scratch project at MIT.

A few years ago I was at one of our Computer Clubhouse after school centers and I saw a 13-year-old boy working on creating his own game. He was able to control a character, in this case, a fish. He wanted the game to keep track of the score, so you could see how many little fish had been eaten by the big fish, but he didn't know how.

I saw this as an opportunity to introduce the idea of variables. I showed this to him and he immediately saw how he could use this block to keep track of how many fish had been eaten in his game. He took the block and put it in the script right where the big fish eats the little fish. He quickly tried it. Sure enough, every time the big fish ate a little fish, the score goes up by 1.

I think that he really got a deep understanding of variables because he really wanted to make use of it. That's one of our overall goals of Scratch. It's not just about variables, but for all types of concepts. We see that kids get a much deeper understanding of the concepts they learn when they are making use of the concepts in a meaningful and motivating way.

LEARNING OBJECTIVES

Students will:

+ be introduced to the computational concepts of conditionals, operators, and data (variables and lists)
+ become more familiar with the computational practices of experimenting and iterating, testing and debugging, reusing and remixing, and abstracting and modularizing by building and extending a self-directed maze, pong, or scrolling game project
+ identify and understand common game mechanics

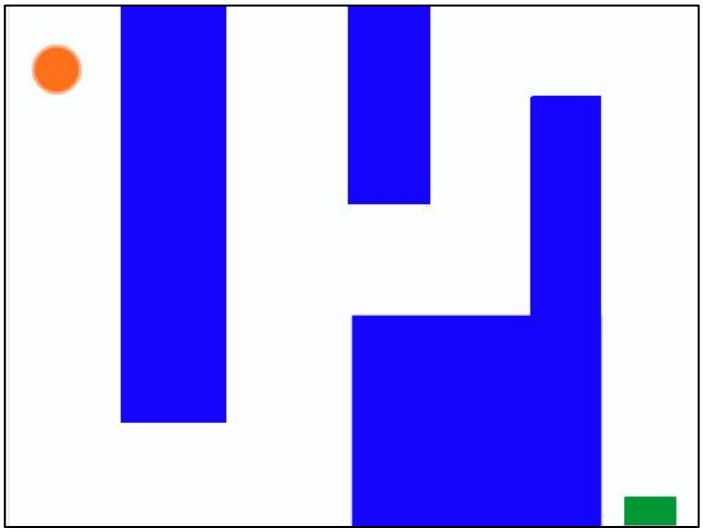
KEY WORDS, CONCEPTS, & PRACTICES

+ abstracting and modularizing
+ conditionals
+ operators
+ data
+ variables and lists
+ sensing
+ feedback fair
+ arcade day
+ puzzle jar
+ brain dump

NOTES

+ Many new concepts are explored in this unit, so we've included added support in the form of example project studios, new programming puzzles for extra practice, and starter game projects that we encourage you to remix and reuse as needed.

CHOOSE YOUR OWN ADVENTURE



In this unit, learners will become game designers and experience creating their own game project. Guided by the activities in this unit, students will be introduced to game mechanics and game development while building understandings of computational concepts (conditionals, operators, data) and computational practices (abstracting and modularizing).

You could get students started on their game projects with the Starter Games activity and then support further development through other activities. From learning common game mechanics such as keeping score and side-scrolling, to the creation of multiplayer games (e.g., Pong), Unit 4 activities offer students multiple opportunities to practice game development.

POSSIBLE PATH

SESSION 1

DREAM GAME LIST

What do all games have in common?

SESSIONS 1 - 5

STARTER GAMES

How can you use Scratch to build an interactive game?

SESSION 2

SCORE

How can you add score to a game using variables?

SESSION 3

EXTENSIONS

What are different ways of extending and increasing difficulty in a game?

SESSION 4

INTERACTIONS

Tackle nine Scratch programming puzzles.

SESSION 5

DEBUG IT!

Help!
Can you debug these five Scratch programs?

UNIT 4 ACTIVITY

DEBUG IT!

SUGGESTED TIME
15–30 MINUTES

OBJECTIVES

By completing this activity, students will:

+ investigate the problem and find a solution to five debugging challenges
+ explore a range of concepts (conditionals, operators, and data) through the practices of testing and debugging

ACTIVITY DESCRIPTION

- ❑ Optionally, have the Unit 4 Debug It! handout available to guide students during the activity.
- ❑ Help students open the Debug It! programs from the Unit 4 Debug It! studio or by following the project links listed on the Unit 4 Debug It! handout. Encourage students to click on the "Look Inside" button to investigate the buggy program, tinker with problematic code, and test possible solutions.
- ❑ Give students time to test and debug each Debug It! challenge. Optionally, have students use the remix function in Scratch to fix the bugs and save corrected programs.
- ❑ Ask students to reflect back on their testing and debugging experiences by responding to the reflection prompts in their design journals or in a group discussion.
- ❑ Create a class list of debugging strategies by collecting students' problem finding and problem solving approaches.

RESOURCES

- ❑ Unit 4 Debug It! handout
- ❑ Unit 4 Debug It! studio
  http://scratch.mit.edu/studios/475634

REFLECTION PROMPTS

+ What was the problem?
+ How did you identify the problem?
+ How did you fix the problem?
+ Did others have alternative approaches to fixing the problem?

REVIEWING STUDENT WORK

+ Were students able to solve all five bugs? If not, how might you clarify the concepts expressed in the unsolved programs?
+ What different testing and debugging strategies did students employ?

NOTES

+ This activity provides an opportunity to check in with students who might need some additional attention or support, particularly around the concepts of conditionals (e.g., if), operators (e.g., arithmetic, logical), and data (e.g., variables, lists).

NOTES TO SELF

- ❑ _____
- ❑ _____
- ❑ _____
- ❑ _____

DEBUG IT!

HELP! CAN YOU DEBUG THESE FIVE SCRATCH PROGRAMS?

In this activity, you will investigate what is going awry and find a solution for each of the five Debug It! challenges.

START HERE

- ❑ Go to the Unit 4 Debug It! Studio: http://scratch.mit.edu/studios/475634/
- ❑ Test and debug each of the five debugging challenges in the studio.
- ❑ Write down your solution or remix the buggy program with your solution.

❑ **DEBUG IT! 4.1** http://scratch.mit.edu/projects/24271192

In this project, the "Inventory" list should be updated every time Scratch Cat picks up a new item. But Scratch Cat can only pick up the laptop. How do we fix the program?

❑ **DEBUG IT! 4.2** http://scratch.mit.edu/projects/24271303

In this project, Scratch Cat gets 10 points for collecting Yellow Gobos and loses 10 points for colliding with Pink Gobos. But something isn't working. How do we fix the program?

❑ **DEBUG IT! 4.3** http://scratch.mit.edu/projects/24271446

In this project, Scratch Cat is thinking of a number between 1 and 10. But something is wrong with the guess checking -- it doesn't work consistently. How do we fix the program?

❑ **DEBUG IT! 4.4** http://scratch.mit.edu/projects/24271475

In this project, the "# of hits" display should increase by 1 every time the Scratch Cat is hit by a tennis ball. But the "# of hits" increases by more than 1 when Scratch Cat is hit. How do we fix the program?

❑ **DEBUG IT! 4.5** http://scratch.mit.edu/projects/24271560

In this project, Scratch Cat is navigating a maze to get to the yellow rectangle. But Scratch Cat can walk through walls. How do we fix the program?

FEELING STUCK?

THAT'S OKAY! TRY THESE THINGS...

- ❑ Make a list of possible bugs in the program.
- ❑ Keep track of your work! This can be a useful reminder of what you have already tried and point you toward what to try next.
- ❑ Share and compare your problem finding and problem solving approaches with a neighbor until you find something that works for you!

FINISHED?

- + Add code commentary by right clicking on blocks in your scripts. This can help others understand different parts of your program!
- + Discuss your testing and debugging practices with a partner. Make note of the similarities and differences in your strategies.
- + Help a neighbor!

UNIT 4 ACTIVITY

INTERACTIONS

OBJECTIVES

By completing this activity, students will:

+ explore different approaches to making projects interactive by solving a series of nine programming puzzles
+ gain more fluency in the concepts of conditionals, operators, and data, and the practice of testing and debugging

ACTIVITY DESCRIPTION

- ❑ On their own or in small groups of 2-3 people, challenge students to further explore Scratch by creating Scratch programs that solve each of the nine Interactions programming puzzles. These Interactions puzzles explore Sensing blocks, engaging some of the more advanced concepts in Scratch related to interactivity. Optionally, have the Interactions handout available to guide students during the activity.
- ❑ Each puzzle can have several possible solutions. Invite students or groups to share different solutions and strategies. We suggest the Pair-Share or Design Demo activity to allow students to share their work and describe their process. Optionally, have students add their projects to the Interactions studio or a class studio.
- ❑ Ask students to think back on the challenge by responding to the reflection prompts in their design journals or in a group discussion.

RESOURCES

- ❑ Interactions handout
- ❑ Interactions studio
  http://scratch.mit.edu/studios/487213

REFLECTION PROMPTS

+ Which puzzles did you work on?
+ What was your strategy for solving the puzzles?
+ Which puzzles helped you think about your game project?

REVIEWING STUDENT WORK

+ Are the puzzles solved?
+ Did students explore other approaches for solving the puzzles?
+ Are there certain blocks or concepts students are still struggling with? How might you help?

NOTES

+ Choose particular challenges that highlight new blocks or concepts that you would like students to explore. Or let students invent their own interaction puzzle prompts.
+ Repurpose these puzzles as an unstructured activity for students who finish other activities early or as a warm-up challenge. Create a puzzle jar: print out, cut, fold, and place copies of each puzzle description in a jar. Then, let students pick puzzles from the jar to solve.

NOTES TO SELF

- ❑ _____
- ❑ _____
- ❑ _____
- ❑ _____

INTERACTIONS

WHAT DIFFERENTIATES A SCRATCH PROJECT FROM A STILL IMAGE OR A VIDEO?

Tackle these nine puzzles that engage some of the more advanced concepts in Scratch related to interactivity. Each of these challenges has several possible solutions.

START HERE

- ❑ Create a Scratch program for each of the nine interactivity puzzles.

- ❑ **PUZZLE 1:** Whenever you press the B key, the sprite gets a little bigger. Whenever you press the S key, the sprite gets a little smaller.
- ❑ **PUZZLE 2:** Whenever the sprite hears a loud sound, it changes color.
- ❑ **PUZZLE 3:** Whenever the sprite is in the top 25% of the screen, it says "I like it up here."
- ❑ **PUZZLE 7:** Whenever you click on the background, a flower appears at that spot.
- ❑ **PUZZLE 8:** Whenever you click on a sprite, all other sprites do a dance.
- ❑ **PUZZLE 9:** Whenever you move the mouse-pointer, the sprite follows but doesn't touch the mouse-pointer.

FEELING STUCK?

THAT'S OKAY! TRY THESE THINGS...

- ❑ Before getting started in Scratch, write down ideas in your design journal for possible ways of programming each of the interactivity puzzles.
- ❑ Work with a neighbor. Collaborating with a partner can be a great way to solve problems and gain new perspectives on ways of programming in Scratch!

FINISHED?

- + Add each of the projects you create to the Interaction Studio: http://scratch.mit.edu/studios/487213
- + Help a neighbor!
- + Discuss your strategies for approaching each puzzle with a partner. Take notes about the similarities and differences in your methods.