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.
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.
**ACTIVITY DESCRIPTION**

- Divide students into small groups of 2-3 people.
- In their small groups, ask students to generate a list of games that they enjoy playing. They can compose the list using their design journals or a sheet of paper. We suggest facilitating the brain dump brainstorming activity: give students a short time period (1-2 minutes) to write down as many games as they can. Then, have students narrow down their favorites from the brain dump list.
- After a few minutes, ask groups about their list of games:
  - What do the games have in common?
  - What features of their design make them a game?
- Facilitate a class discussion about what characteristics make up a game and generate a class list of common game mechanics. Next, ask students to imagine their dream game and write a list of design elements for that game.
- Invite students to share their dream game lists in their small groups or critique groups (see Unit 0 Critique Group activity) to get feedback and suggestions.

**OBJECTIVES**

By completing this activity, students will:
+ identify common design elements of games

**RESOURCES**

- paper to write down game design elements
- things to sketch with (pencils, pens, markers, etc.)

**REFLECTION PROMPTS**

+ Make a list of your favorite games.
+ What do the games have in common?
+ What features of their design make them a game?
+ Create a list of design elements for your dream game.

**REVIEWING STUDENT WORK**

+ Do the dream game lists include features of games?
+ What design elements are similar or different from the class group list?
+ What do the lists tell you about the kinds of games and the types of play your students enjoy?

**NOTES**

+ Invite students to refer back to this dream game list while programming games in other Unit 4 activities.

**NOTES TO SELF**

- __________________________________
- __________________________________
- __________________________________
- __________________________________
Chess, monopoly, Mario, Clue, football, Candyland, Pac Man, Jump Rope, Baseball, Tennis, Flappy Bird, Wheel of Fortune, Four Square
RESPOND TO THE FOLLOWING REFLECTION PROMPTS USING THE SPACE PROVIDED BELOW OR IN YOUR DESIGN JOURNAL.

NAME:

+ Make a list of your favorite games.

+ What do the games have in common?

+ What features of their design make them a game?

+ Create a list of design elements for your dream game.
In this activity, students will create a starter game project that can be revisited and extended during the Score, Extensions, and Interactions activities. Optionally, show the Maze, Pong, and Scrolling example starter projects, and have the Maze, Pong, and Scrolling handouts available to guide students.

Choose one game project to facilitate as a class or let students choose which game they want to create: maze, pong, or scrolling. Give students time to start building their games or let them remix one of the starter projects.

Encourage students to get feedback on their games-in-progress. We suggest the feedback fair activity: half of the students stay in their seats with their projects open while the other half walks around exploring projects, asking questions, and giving feedback, then switch sides. Optionally, have students add their final game projects to the Games studio or a class studio.

Ask students to respond to the reflection prompts in their design journals or in a group discussion.

To celebrate and share final game creations, we recommend hosting an Arcade Day. Final game projects are placed in presentation mode; students walk around and play each other’s games.

The Scrolling game option introduces cloning. Help students learn more about the cloning blocks with the Cloning handout from Unit 5 Advanced Features.

Objectives
By completing this activity, students will:

- develop greater fluency with computational concepts (conditionals, operators, data) and practices (experimenting and iterating, testing and debugging, reusing and remixing, abstracting and modularizing) by working on a self-directed game project.

Resources
- Maze handout
- Maze example starter project: http://scratch.mit.edu/projects/11414041
- Pong handout
- Pong example starter project: http://scratch.mit.edu/projects/10128515
- Scrolling handout
- Scrolling example starter project: http://scratch.mit.edu/projects/22162012
- Games studio: http://scratch.mit.edu/studios/487504

Reflection Prompts
- What was challenging about designing your game?
- What are you proud of?

Reviewing Student Work
- Do games include conditionals, operators, and data?

Notes to Self
In this project, you will create a game. This game includes interactions between sprites, score, and levels. You move a sprite from the start of a maze to the end without touching the walls.

**MAZE**

**HOW CAN YOU USE SCRATCH TO BUILD AN INTERACTIVE GAME?**

- Draw a maze-like background and use different colors for the walls and end-of-maze marker.
- Add a sprite.
- Make your game interactive!

**START HERE**

**THINGS TO TRY**

- Add multiple levels to your game! This can be done through the use of different backdrops and using broadcast blocks to trigger the next level.
- Use the make a variable block to keep score!
- Experiment with timer blocks to add new challenges to your maze!

**BLOCKS TO PLAY WITH**

- **my variable**
- **color is touching**
- **>**
- **<**
- **=**
- **+**
- **-**
- **timer**
- **reset timer**
- **> 50**
- **< 50**
- **to 10**
- **by 10**
- **and**
- **or**
- **pick random**
- **1 to 10**
- **touching**
- **mouse-pointer**
- **say**
- **you win!**

**FINISHED?**

- Add your project to the Games Studio: [http://scratch.mit.edu/studios/487504](http://scratch.mit.edu/studios/487504)
- Swap games with a partner and walk each other through your creations.
HOW CAN YOU USE SCRATCH TO BUILD AN INTERACTIVE GAME?

In this project, you will create a game. This game includes interactions between sprites, score, and levels. The game is similar to the classic game of pong, where the goal is to keep the sprite from getting past you.

START HERE

- Create two sprites: a paddle for the user to control and a ball the user will be playing with.
- Make your paddle sprite interactive.
- Bring your game to life!

THINGS TO TRY

- How do you add difficulty to your game? Creating different levels, using a timer, or keeping score are a few examples of things you could do.
- Experiment with changing the look of your game by editing the backdrops!
- Explore using different key presses to control your sprites!

BLOCKS TO PLAY WITH

FINISHED?

- Add your project to the Games Studio: http://scratch.mit.edu/studios/487504
- Swap games with a partner and walk each other through your creations.
SCROLLING

HOW CAN YOU USE SCRATCH TO BUILD AN INTERACTIVE GAME?

In this project, you will create a game. This game includes interactions between sprites, score, and levels. The game is similar to Flappy Bird, where the goal is to keep an object from falling to the ground or touching certain objects.

START HERE

❑ Create two sprites: one for the player to control (helicopter) and one to avoid (gliding bars).
❑ Make the helicopter interactive.
❑ Bring your game to life by adding scripts to make the gliding bars scroll across the stage!

THINGS TO TRY

❑ How do you add difficulty to your game? Creating different levels, using a timer, or keeping score are a few examples of things you could do.
❑ Experiment with changing the look of your game by editing the backdrops!
❑ Explore using different key presses to control your sprites!

FINISHED?

+ Add your project to the Games Studio:
http://scratch.mit.edu/studios/487504

+ Swap games with a partner and walk each other through your creations.
STARTER GAMES REFLECTIONS

+ What was challenging about designing your game?

+ What are you proud of?
**ACTIVITY DESCRIPTION**

- Optionally, explore the Fish Chomp starter project as a group and have the Score handout available to guide students.
- Help students open the Fish Chomp starter project. Give students time to explore variables by remixing the Fish Chomp Starter Project to add score to the game. Optionally, give students time to incorporate score into their previously started maze, pong, or scrolling game projects.
- Allow students to share their Fish Chomp remixes or game projects with added score. We suggest the Design Demo activity: invite a few students to present their projects to the group and demonstrate how they implemented score using variables. Optionally, have students add their remixes to the Fish Chomp Remix studio or a class studio.
- Ask students to think back on the design process by responding to the reflection prompts in their design journals or in a group discussion.

**OBJECTIVES**

By completing this activity, students will:

- be able to describe what a variable is and why variables are useful
- be introduced to the computational concept of data
- experience remixing and reusing a project or part of a project

**RESOURCES**

- Score handout
- Score examples studio [http://scratch.mit.edu/studios/218313](http://scratch.mit.edu/studios/218313)
- Fish Chomp starter project [http://scratch.mit.edu/projects/10859244](http://scratch.mit.edu/projects/10859244)
- Fish Chomp remix studio [http://scratch.mit.edu/studios/475615](http://scratch.mit.edu/studios/475615)

**REFLECTION PROMPTS**

- How would you explain variables to someone else?
- What are variables good for?

**REVIEWING STUDENT WORK**

- Can students explain what a variable is and what variables are good for?

**NOTES**

- Encourage students to clarify their understanding of variables by exploring code from sample projects in the Score examples studio.
- Variables are an important mathematical and computational concept. Students are taught about variables in their math and science classes, but many students have a difficult time learning them. Games are one way to make the usefulness of variables more concrete.

**NOTES TO SELF**

- 
- 
- 
- 

**SUGGESTED TIME**

30–45 MINUTES
Fish Chomp is a game where players try to catch as many fish as they can by guiding a sprite with the mouse. In this activity, you will remix Fish Chomp by adding a score with variables.

### START HERE

- Go to the Fish Chomp project page: [http://scratch.mit.edu/projects/10859244](http://scratch.mit.edu/projects/10859244)
- Click on the Make a Variable button in the Data category to create and name a variable for score.
- Experiment with your new variable blocks to incorporate score into your project!

### FEELING STUCK?

- Not sure how to work with variables? Check out this project for more information: [http://scratch.mit.edu/projects/2042755](http://scratch.mit.edu/projects/2042755)
- Or take a look at this video: [http://youtu.be/uXq379XkhVw](http://youtu.be/uXq379XkhVw)
- Explore and study code in games that use score to learn more about creating variables and incorporating score into a project.

### FINISHED?

- Add your project to the Fish Chomp Remix studio: [http://scratch.mit.edu/studios/475615](http://scratch.mit.edu/studios/475615)
- Challenge yourself to do more! How can you use score to add difficulty to your game design?
- Find a game you are inspired by and remix it!
How would you explain variables to someone else?

What are variables good for?
EXTENSIONS

OBJECTIVES
By completing this activity, students will:
+ become more familiar with the concepts of conditionals, operators, and data by exploring programs that illustrate common game mechanics

ACTIVITY DESCRIPTION

❑ Optionally, show example projects from the Extensions studio and have the Extensions handout available to guide students.

❑ Give students time to explore the code of programs in the Extensions studio to investigate different ways games can be increased in difficulty or extended. Ask students to select one or more extensions to add to their previously started maze, pong, or scrolling game projects. Give students time to experiment and incorporate the extension(s) into their games.

❑ Allow students to share their extended game projects with one another. We suggest facilitating the pair-share or design demo activity to let students share their games and demonstrate what they learned.

❑ Ask students to think back on the design process by responding to the reflection prompts in their design journals or in a group discussion.

RESOURCES

❑ Extensions handout
❑ Extensions studio
   http://scratch.mit.edu/studios/475619

REFLECTION PROMPTS

+ What are different ways of increasing difficulty in a game?
+ Which extensions did you add to your game project?
+ Describe your process for including the extension(s) in your game?

REVIEWING STUDENT WORK

+ Were students able to incorporate extensions into their original game projects?

NOTES

+ To provide more scaffolding for students needing extra support, we suggest walking through one extension sample program (e.g., levels) as a class and helping students add the extension to their game projects.
+ The backpack tool is one way students can incorporate parts of the extension projects into their starter games. Learn more about backpack at http://bit.ly/scratchbackpack

NOTES TO SELF
EXTENSIONS

HOW CAN YOU EXTEND AND REIMAGINE GAMES IN SCRATCH?

Get into game design by adding extended features within your Scratch project! Choose at least one (or more!) of the following extensions and add it to your previously started maze, pong, or scrolling games.

START HERE

- Go to the Extensions studio: http://scratch.mit.edu/studios/475619
- Choose one (or more) of the extensions to explore.
- Incorporate your choice into your previously started game projects!

THINGS TO TRY

+ The backpack can be an extremely useful tool while programming in Scratch. It can store everything from lines of code, to music files, to sprites, and more. Try using it to incorporate extensions into your game projects.
+ Alternatively, sketching out ideas and bits of code in your design journal is another great method for planning how to incorporate your extensions.

FINISHED?

+ Add another extension to your maze, pong, or scrolling game.
+ Challenge yourself to do more! Continue going through each of the extensions and add them to your games.
+ Help a neighbor!
+ Share your project with a neighbor and give each other feedback on your games.

SCORE http://scratch.mit.edu/projects/1940443
Demonstrates how to set and change a score. Receive 10 points every time the Scratch cat is clicked.

LEVELS http://scratch.mit.edu/projects/1940453
Demonstrates how to change levels. Score increases by 1 every time the space bar is pressed. Level increases by 1 for every 10 points.

TIMER http://scratch.mit.edu/projects/1940445
Demonstrates how to use a timer. Use the mouse to navigate the Scratch cat to Gobo.

ENEMIES http://scratch.mit.edu/projects/1940450
Demonstrates how to add an enemy. Avoid the tennis ball by using the up and down arrow keys.

REWARDS http://scratch.mit.edu/projects/1940456
Demonstrates how to collect items. Use the arrow keys to move the Scratch cat around to collect quest items.

MOUSE http://scratch.mit.edu/projects/25192659
Demonstrates how to program the mouse to control game play. Move the mouse to move the paddle.

RESTART http://scratch.mit.edu/projects/25192935
Demonstrates how to make a button to restart the game. Click on the RESTART button to restart.

MENU http://scratch.mit.edu/projects/25192991
Demonstrates how to display a menu screen at the beginning of the game. Click START or DIRECTIONS on the menu screen.

MULTIPLAYER http://scratch.mit.edu/projects/25192711
Demonstrates how to add another player to the game. Player 1 uses the arrow keys to navigate Pico through the maze, and player 2 uses the W, A, S, D keys to navigate Nano through the maze.
What are different ways of increasing difficulty in a game?

Which extensions did you add to your game project?

Describe your process for including the extension(s) in your game?
**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.

**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

**RESOURCES**

- Interactions handout
- Interactions studio
  - [http://scratch.mit.edu/studios/487213](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**

- 
- 
- 
- 
- 
- 
- 
- 

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!

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.

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](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.
INTERACTIONS
REFLECTIONS

RESPOND TO THE FOLLOWING REFLECTION PROMPTS USING THE SPACE PROVIDED BELOW OR IN YOUR DESIGN JOURNAL.

NAME:

+ Which puzzles did you work on?

+ What was your strategy for solving the puzzles?

+ Which puzzles helped you think about your game project?
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

DEBUG IT!

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

RESOURCES

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.

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
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?

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!
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?