TATA CONSULTANCY SERVICES | Discovery EDUCATION

IGNITE MY FUTURE

THINK | SOLVE | CREATE | CONNECT

LESSON TITLE

Recycle and Reuse!

Guiding Question: How can we connect with each other?

Ignite Curiosity

- How often do you use shortcuts or key commands when using a computer or smartphone?
- Why can't you recycle or reuse everything you use?
- When do you recycle and reuse in the real world?

Architects, engineers, and contractors use many tools to build new structures. While each new building is unique, many elements of all structures are the same. Instead of starting from scratch, architecture professionals use blueprints and plans from similar buildings to save time and avoid error. Like building a new house or corporate office, building new computer programs takes a lot of time and trial-and-error. In both situations, the computational thinking strategy of decomposition makes it possible to break apart the building process into manageable pieces and replicate those pieces over and over again to save time and prevent mistakes. In this lesson, students will use the computational thinking strategy of decomposing to build a new, unique computer game out of preexisting blocks of code. In **THINK**, students will act as software engineers developing a new game or phone app. Students will learn how using blocks can increase speed of programming, make programming more accessible, and reduce errors in coding. In **SOLVE**, students will use the Scratch visual programming tool to explore how games are created in blocks of code. Students will identify patterns of code blocks to generalize for use in their own game or app. Then, they will design their own game and decompose it into blocks of functionality (they may draw a storyboard, flowchart, or diagram, or they may write a list). In **CREATE**, students will use Scratch to create their own game or app. In **CONNECT**, students will discuss how coding professionals like software engineers and app developers share resources through open source code databases, code libraries, and widgets.

Students will be able to:

- **Examine** how to decompose complex problems into manageable sub-problems,
- **Analyze** patterns and common features between programs, and
- **Create** an original application based off of recycled blocks of code.

SUBJECTS

Science
Engineering

COMPUTATIONAL THINKING PRACTICE

Developing and Using Abstractions

Creating Computational Artifacts

COMPUTATIONAL THINKING STRATEGY

Decompose

MATERIALS

Computers with Internet access and necessary requirements for the MIT Scratch web application

Real-Life Recycling and Reusing student capture sheet

Decomposing Code: Pros, Cons, and Solutions student capture sheet

Game Design student capture sheet

Students will act as software engineers developing code for a new game or phone app.

1 **Read** the following scenario to students:
Imagine you are a professional video game designer who has created a popular and successful game. Now, the company you work for wants you to create a follow-up that will be just as popular and successful—and you have to do it in half the time it took to create your first game! How can you meet this deadline and still produce a high-quality game? Let's see what you come up with!

Ask students if they think coders who create computer programs, video games, and phone apps write every piece of code from scratch. Ask them to imagine they are writing an email to a friend and can't copy and paste a quote, an image, or a link. They have to retype it all, character by character. That would be really annoying, right? Point out that computers have lots of functions that operate like copy and paste. They identify repetitive actions and simplify them to save us time. This is called **decomposing**. Visual programming editors allow users to drag and drop existing code blocks, create new code blocks from existing blocks, and map to a standard, text-based language that underlies the blocks.

2 **Lead** students to consider the importance of decomposing using the following guiding questions:

- What would happen if you had to create every piece of code in your game or app from scratch? (The coding would take too long, and you would miss your deadline.)
- How could decomposing help video game designers meet their deadlines? (copy simple and repetitive functions from existing games/a code block library)
- What elements do you see in video games that suggest the use of decomposing? (similar character templates, quests, health/energy/time bars, etc.)
- How would visual programming editors make it easier to decompose code? (Dragging and dropping is easier than copying and pasting.)

3 **Distribute** the Real-Life Recycling and Reusing student capture sheet to demonstrate that decomposing is not limited to the computer world. Ask students to think of other duplication tools they have encountered and add them to the list. Point out that there are many other uses for the tools listed on the handout. Have students select one of the tools and come up with at least one other example of a situation where it would be used.

4 **Distribute** the Decomposing Code: Pros, Cons, and Solutions student capture sheet. Point out that while the focus of this lesson has been on the advantages of decomposing tools, they also have disadvantages. Have the students write down at least five advantages and disadvantages of decomposing in coding, and encourage them to think about ways they could compensate for those shortcomings when working on their project.

5 **Challenge** students to identify and summarize the problem that needs to be solved. Remind them to consider what elements they can take from a code library or block-coding program and to consider the pros and cons of this approach.

Students will explore how games are created in blocks of code in the Scratch or Snap! visual programming tools.

Teacher Note: Now that students have explored decomposing tools in the physical and virtual worlds, ask them to consider how to use these tools to create a new video game or phone app. Tell students that in 2014, 11 high school students in North Philadelphia created the Gotcha! app as part of Temple University's second annual Urban Apps and Maps program, in which students create computer programs to help solve a real-world problem. The Gotcha! app lets users report crimes they witness to the police, which often go unreported and harm the community.

Then, explain to students how Sam Schooler used algorithms and frameworks to create Runbow, an award-winning multiplayer game.

1. **Introduce** students to the Scratch application. Explain that Scratch is a block-coding tool that makes coding easier, faster, and more accessible by providing a simple visual interface that enables coders to see blocks of code, choose the ones they want, and drag and drop them into a program. Ask students why a visual drag-and-drop interface makes coding easier and more accessible (coders do not have to learn a complex coding language). Demonstrate the program by showing examples. Teachers can access examples of Scratch via this link: https://scratch.mit.edu/starter_projects/.

2. **Encourage** students to discuss which blocks of code would be good candidates for decomposition. What blocks of code could be generalized? Remind them of the discussion about repeated elements in video games and to consider those elements as possible candidates. For example, a specific quest in a video game could be broken down into its basic story elements (for example, introduction of the protagonist, description of the quest's goal, description of the challenges the protagonist must overcome, etc.).

3. **Distribute** the Game Design student capture sheet. Have students brainstorm, either alone or in small groups, to come up with the game or app they want to build. Once they've chosen their project, have them outline its design on the worksheet, then deconstruct it into blocks of functionality.

   Outline format options to include the following:

   - Storyboard
   - Flowchart
   - Diagram
   - List

Students will use the block-coding platform Scratch to create their own game or app.

Teacher Note: Be sure to allow enough time for students to explore the Scratch program, create their game or app, and try out and assess one another's products as a way of opening discussion around what worked, what could be improved, and how decomposing affected the building process.

1. **Introduce** students to Scratch, answering any questions they may have about the platform or about the project in general.

2. **Have students build their program,** following the design they created on the worksheet and using the block-coding platform. Then, have students test one another's programs and offer comments on what did and did not work, make suggestions for improvement, and identify where they see the use of block coding in the program.

3. **Summarize** by inviting students to note where they used block coding in their own program, review the comments and suggestions made by other students, and consider what blocks of code they could use from this program should they be asked to build another app or game. Tie this into the original scenario (a programmer has been asked to create a follow-up to a popular game on a much shorter timeframe), and ask students if their perspective on the scenario has changed and why it has or has not.

Select one of the strategies listed below to help students answer these questions:

- **How do this problem and solution connect to me?**
- **How do this problem and solution connect to real-world careers?**
- **How do this problem and solution connect to our world?**

1. **Write** the three questions on PowerPoint or flip chart slides and invite students to share out responses.
2. **Display** pieces of chart paper around the room, each with one question written on it. Ask students to write down their ideas related to the questions on each sheet.
3. **Assign** one of the questions to three different student groups to brainstorm or research, and then share out responses.
4. **Invite** students to write down responses to each question on a sticky note, and collect them to create an affinity diagram of ideas.

How does this connect to students?

Students likely use more decomposing tools than they think, including physical tools such as rubber stamps and copy machines and virtual tools such as copy/paste functions in word processing programs and cloning tools. Some students might even use games such as Super Mario Maker to create custom video game levels.

How does this connect to careers?

Computer Instructors use decomposing tools such as block code and code libraries to provide their students greater access to coding.

Computer Designers, including game designers and compiling designers who translate code from one language to another, use decomposing tools to code more rapidly and efficiently.

Robotics Engineers can use decomposing tools to repeat robotic code snippets within whole coded instructions to repeat identical actions.

How does this connect to our world?

Decomposing tools are used to increase the speed and efficiency of repetitive tasks in everything from typing emails and texts to customer service. For example, if you order the same thing every time you go to a restaurant, the staff can save time by pulling up the information from a saved order you made before and duplicating that information to create a new order. Or, if you write a lot of letters for your job, you can save time by creating a template for the letters and a database from which you can copy addresses.

One concern about decomposing tools is that they can remove the thought or effort behind repeated actions, leading to errors if the action is not identical. For example, if the restaurant duplicates your order and you want something different, the restaurant may not make the change. There is a reason "rubber stamping" refers to approving a request without reading its details. For example, if you copy the wrong address from a database, you could send an important letter to the wrong person. Additionally, AutoFill functions don't account for address or phone number changes.

Find more easy-to-implement resources to integrate computational thinking practices into your classroom by visiting **ignitemyfutureinschool.org**

National Standards

NEXT GENERATION SCIENCE STANDARDS

Science and Engineering Practices	Disciplinary Core Ideas	Crosscutting Concepts
Developing and Using Models ▪ Develop a model to generate data to test ideas about designed systems, including those representing inputs and outputs. (MS-ETS1-4).	ETS1.B: Developing Possible Solutions ▪ A solution needs to be tested, and then modified on the basis of the test results, in order to improve it. (MS-ETS1-4)	Influence of Science, Engineering, and Technology on Society and the Natural World ▪ The uses of technologies and limitations on their use are driven by individual or societal needs, desires, and values; by the findings of scientific research; and by differences in such factors as climate, natural resources, and economic conditions. (MS-ETS1-1).

K-12 COMPUTER SCIENCE FRAMEWORK

Practice 4. Developing and Using Abstractions
Abstractions are formed by identifying patterns and extracting common features from specific examples to create generalizations. Using generalized solutions and parts of solutions designed for broad reuse simplifies the development process by managing complexity.

Practice 5. Creating Computational Artifacts
The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts include programs, simulations, visualizations, digital animations, robotic systems, and apps.

Real-Life Recycling and Reusing

Tool	Medium	Purpose	Example of Use
Rubber stamp	Rubber and ink	Duplicating a simple image, word, or phrase	Marking files "Approved" or "Rejected"
Copy machine	Light-based machinery	Duplicating a printed image or page of text	Copying a page from a library book for note-taking purposes
Sculpture mold	Silicone or rubber	Duplicating a three-dimensional object	Producing a line of action figures
Copy/paste function	Text-based computer programs	Selecting and duplicating text in a text-based artifact	Inserting a hyperlink into a document
Cloning tool	Image editing programs such as Photoshop	Selecting part of an electronic image and copying it to another part of that image	Editing a digital photo to create a meme (such as face-swapping)
Visual coding interface	Block-based coding programs such as Scratch	Building a computer program from existing blocks of code	Designing a new app
Open source code	Computer program	Code that is available to anyone to use or modify without needing to ask permission	Improving the coding of an Android phone app

Decomposing Code: Pros, Cons, and Solutions

Pros	Cons	Possible Solution
Frees time from having to code simple and repetitive elements from scratch each time	End product could be boring and repetitive.	Use block coding only for the most basic elements, then elaborate so the end product stands out.
Helps prevent typos	Errors from the decomposed code could be copied over.	QC check decomposed code as carefully as original code.

Game Design Worksheet

1. **Use this space** to design your game or app.

2. **Note** opportunities to use block codes in your design.

3. **Comments:**