Lesson Overview

During this lesson, students will gain an understanding of what a compound conditional is and how to use it effectively in programming. They will integrate and exhibit learning by helping Sam create a program that simulates a game of ‘Rock, Paper, Scissors’.

Learn

Demonstrate what a compound conditional is.
Identify how compound conditionals are used.

Do

Plan, code and extend a program that simulates a game of ‘Rock, Paper, Scissors’ using the micro:bit buttons as inputs.

Reflect

Reflect and evidence learning in the Mission Journal.

Standards Focus (Refer to Standards Alignment Map)

Grade 7 Computer Science: Compound Conditionals (2-AP-12)

Materials Required

SAM Labs Learn to Code kit, including micro:bit;
packs of playing cards
Learn

Warm-Up
Demonstrate what a compound conditional is.

“What is a compound conditional?”
● A compound conditional is where two or more conditions are tested within one conditional statement, using AND/OR.
● Conditional statements have one condition, for example ‘if it is raining, do take an umbrella’.
● Compound conditionals offer a way to combine more than one condition, for example, if it is raining AND if it is cold, do take a coat with a hood.

- Divide students into small groups and give each group a pack of cards. Ask them to divide the cards equally between themselves.
- Students should turn over one card with each turn, using the following scoring rules:
  ○ If the overturned card is red AND above ‘5’ = do: win a point
  ○ If the overturned card is black AND above ‘5’ = do: win 2 points
  ○ If the overturned card is (red OR black) AND below ‘4’ = do: lose a point.
- Explain that this game demonstrates the use of conditional statements and compound conditionals.

Quick Reflection
Can you think of an example of when you use a compound conditional in making a decision?

Link Forward
Students look at how to use a compound conditionals.
Mini-lesson
Identify how compound conditionals are used.

“How can compound conditionals be used in game development?”
- In an online game there are a multitude of pathways within the game code, all activated by the user input.
- The game output is defined within the code and can be activated by one or more inputs.
- The game 'Mario Kart™' has collisions built in to the game. The compound conditional for this could read like this:
  
  ```
  If the kart collides with the edge of the track OR with another kart
  do car crash animation.
  ```
- The use of a compound conditional is more efficient as it allows two or more conditions to be represented in one line of code.

Unplugged Activity
“Can you break down a game into a compound conditional?”
- Ask students to think about their favorite game and how it is controlled (what the instructions are).
- Students then write down all the ways that the game can be controlled and consider if any of them are compound conditionals.

Quick Reflection
Why are compound conditionals more efficient than using ‘else if’ in programming?

Checks for Understanding
- What can a compound conditional include?
- Which block could the compound conditional feature in?
Keywords

**Compound conditional**
Where two or more conditions are tested within one conditional statement with the use of AND/OR/NOT.

**Conditional statement**
Enables choice of pathway through a program based on criteria.

**AND**
Both inputs must be True for the output to be True.

**OR**
Either input or both must be True for the output to be True.

---

**Link Forward**
Students create a program that will use a compound conditional to use the A and B button of the micro:bit combined to determine an output.

---

**Do**

**Let’s Build**

Code a program that uses conditional statements and compound conditionals to define an output of Rock, Paper or Scissors.

**Instructions**

**Step 1**
Click ‘ADD DEVICE’ and select:
- ‘Micro:bit’
- ‘Buzzer’

Connect the micro:bit battery, click ‘CONNECT’ and ‘Pair’.

Turn it on the Buzzer, click ‘CONNECT’ and ‘Pair’.

**Workspace & Notes**

The micro:bit will display the output (Rock, Paper or Scissors) depending on the input chosen by the user. The Buzzer sound will represent the start of the game.
Step 2
From 'Functions', drag onto the workspace:
- 1 'to (do something)' block.
Set to 'to (testRockPaperScissors)'.

This function will store the conditional statements and compound conditionals to test the input and activate the corresponding output.

Step 3
From 'Logic', drag onto the workspace:
- 1 'if do else if do else' block.
- 1 '() [and] ()' block.
Snap the 'if do else if do else' block into the 'to (testRockPaperScissors)' block.
Snap the '() [and] ()' block into the 'if' section of the 'if do else if do else' block.

These blocks will be used to develop two conditional statements to create a compound conditional.

Step 4
In the settings of the 'if do else if do else' block, drag 1 'else' block left and 1 'else if' block right.

This will remove one 'else' block and add one 'else if' section to the block on the workspace.
Step 5
From 'micro:bit' 'Values', drag onto the workspace:
● 2 'is micro:bit A pressed' blocks.

Snap one into the first section of the ‘( ) [and] ( )’ block and the other into the first ‘else if’ section.

The first half of the compound conditional is: if micro:bit ‘A’ is pressed.

The first conditional statement is: if micro:bit ‘A’ is pressed.

The output to the conditional statements can only be True (if the button is pressed) or False (the button is not pressed).

Step 6
From 'micro:bit' 'Values', drag onto the workspace:
● 2 'is micro:bit B pressed' blocks.

Snap one into the second section of the ‘( ) [and] ( )’ block and the other into the second ‘else if’ section.

The compound conditional is now: if micro:bit ‘A’ and ‘B’ buttons are pressed together, then the condition has been met = true.

The other two conditional statements are:
if micro:bit ‘A’ is pressed then the condition has been met = true.
if micro:bit ‘B’ is pressed then the condition has been met = true.

There now needs to be code added to the ‘do’ sections.
Step 7
From 'micro:bit' 'Actions', drag onto the workspace:
● 3 'on micro:bit display ("word")' blocks.
Snap one into each 'do' section.
Set as follows:
● 1st: 'on micro:bit display ("R")'
● 2nd: 'on micro:bit display ("P")'
● 3rd: 'on micro:bit display ("S")'.
The micro:bit output will represent the three outcomes of the game Rock ("R"), Paper ("P"), Scissors ("S").

Step 8
From 'Functions', drag onto the workspace:
● 1 'to (do something)' block.
Set to 'to (startGame)'.
This function will store the code that will be activated when the game starts.

Step 9
From 'micro:bit' 'Actions', drag onto the workspace:
● 1 'on micro:bit display [heart]' block.
Snap into the 'to (startGame)' block.
Set to 'chessboard'.
This display will represent the start of a game.

Step 10
From 'General', drag onto the workspace:
● 1 'wait for (2) seconds' block.
Snap into the 'on micro:bit display [chessboard]' block.
Set to '0.5' seconds.
This block allows time for the image to be visible on the micro:bit.
Step 11
From ‘Buzzer’ ‘Actions’, drag onto the workspace:
● 1 ‘set Buzzer pitch to (0)’ block.
● 1 ‘clear Buzzer’ block.

Snap into the ‘wait for (0.5) seconds’ block.

Set Buzzer pitch to ‘50’.

Once the ‘chessboard’ image has been displayed for ‘0.5’ seconds, the Buzzer will make a quick sound representing the game start.

Step 12
From ‘General’, drag onto the workspace:
● 1 ‘program start’ block.

From ‘Functions’, drag onto the workspace:
● 1 ‘startGame’ block.

Snap into the ‘program start’ block.

This will allow the ‘startGame’ function to be called upon and activated as soon as the program is run.

Step 13
From ‘Loops’, drag onto the workspace:
● 1 ‘repeat forever do’ block.

Snap into the ‘startGame’ block.

Step 14
From ‘Functions’, drag onto the workspace:
● 1 ‘testRockPaperScissors’ block.

Snap into the ‘repeat forever do’ block.

This allows the ‘testRockPaperScissors’ function to continuously test if micro:bit ‘A’ and/or ‘B’ buttons are pressed. Without the loop, the program would stop after ‘RUN’ has been pressed and the game starts.
Test your program

Encourage students to test that when 'RUN' is pressed, the following occurs:

- the micro:bit displays a 'chessboard' image
- the Buzzer plays a short sound

Then, if they press

- micro:bit 'A' AND 'B' = outputs 'R' for Rock
- 'A' = outputs 'P' for Paper
- 'B' = outputs 'S' for Scissors.

Quick Reflection
What happens if the blocks within the compound conditional are changed from 'true' to 'false'?

Plan
Students can use the Mission Journal to complete planning tasks.

I need to defeat an evil minion bot with a game of Rock, Paper, Scissors using images not text.

Can you help me?

Building on the code on the workspace from Let’s Build, students can plan a program that outputs images for Rock, Paper or Scissors on the micro:bit.
## Challenge

Code a program that outputs images for Rock, Paper or Scissors and can be easily reset.

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Workspace &amp; Notes</th>
</tr>
</thead>
</table>
| **Step 1**   | Click ‘ADD DEVICE’ and select:  
|              | ● ‘Slider’  
|              | Turn it on, click ‘CONNECT’ and ‘Pair’.  
|              | The Slider will be used to reset the game.  
|              | Note, there is no micro:bit test in this lesson as both the micro:bit buttons are being used within the program.  
|              | The workspace will then look like this:  
| **Step 2**   | Remove from the workspace:  
|              | ● 1 ‘on micro:bit display (“R”)’ block  
|              | ● 1 ‘on micro:bit display (“P”)’ block  
|              | ● 1 ‘on micro:bit display (“S”)’ block.  
| **Step 3**   | In ‘Variables’, create two variables:  
|              | ● ‘X’  
|              | ● ‘Y’.  
|              | Create variable...
Step 4
From 'Functions', drag onto the workspace:
● 1 'to (do something)' block.
Set to 'to (drawPaper)'.

Step 5
From 'micro:bit' 'Actions', drag onto the workspace:
● 1 'clear micro:bit LEDs' block.
Snap into the 'to (drawPaper)' block.

Step 6
From 'Loops', drag onto the workspace:
● 1 'count with [i] from (1) to (10) by (1) do' blocks.
Snap into the 'clear micro:bit LEDs' block.
Set 'count with [X] from (0) to (4) by (1)'.

Step 7
From 'micro:bit' 'Actions', drag onto the workspace:
● 1 'on micro:bit plot X: (0) Y: (0)' block.
Snap into the 'count with [X] from (0) to (4) by (1) do' block.
Set to 'on micro:bit plot X: (0) Y: (2)'.

This function will store the sequence for drawing a line on the micro:bit to represent ‘paper’.

This block will clear the micro:bit display before a new one is added.

This is a ‘for’ loop and will count through the range ‘0–4’ by ‘1’.
Step 8
From 'Variables', drag onto the workspace:
● 1 '[X]' block.

Snap into '(0)' section on the 'on micro:bit plot X: (0) Y: (2)' block.

Step 9
From 'Functions', drag onto the workspace:
● 1 'to (do something)' block.

Set to 'to (drawRock)'

Step 10
From 'micro:bit' 'Actions', drag onto the workspace:
● 1 'clear micro:bit LEDs' block
● 1 'on micro:bit display [heart]' block.

Snap into the 'to (drawRock)' block.

Set to 'on micro:bit display [small square]'.

Step 11
Duplicate:
● 'to (drawRock)'.

Rename the function to 'drawScissors'.

Set the 'on micro:bit display' block to 'on micro:bit display [scissors]'.
Step 12
From 'Functions', drag onto the workspace:
- 1 'drawRock' block
- 1 'drawPaper' block
- 1 'drawScissors' block.
Snap in the above order into the 'do' sections within the 'testRockPaperScissors' function.

Step 13
From 'Slider' 'Events', drag onto the workspace:
- 1 'when Slider value changes'.
From 'Functions', drag onto the workspace:
- 1 'startGame' block.
Snap into the 'when Slider value changes' block.

Test your program

Encourage students to test that when 'RUN' is pressed, the following occurs:

- micro:bit displays chessboard image and Buzzer sounds
- If 'A' AND 'B' buttons are pressed, a square is displayed to represent 'rock'.
- If 'A' button is pressed, a straight line is displayed to represent 'paper'.
- If 'B' button is pressed, a scissors image is displayed to represent 'scissors'.
- When the Slider value changes (it is moved), the program is reset (chessboard display and Buzzer sound).

Encourage students to join with another team to compete against each other.
**Quick Reflection**
Does your program look the same as mine?
Students can follow the flowchart on the Lesson Slides.

**Debug**
The outputs may not be working as expected. How can I debug it?

**Instructions**
Test each output to ensure the program works as expected.
Use a testing table to inform Debug process.

<table>
<thead>
<tr>
<th>Test</th>
<th>Expected outcome</th>
<th>Actual outcome</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Press 'A'</td>
<td>Straight line appears</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Press 'B'</td>
<td>Scissors appear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Press 'A' AND 'B'</td>
<td>Small square appears</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program start</td>
<td>Chessboard and Buzzer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slider moves</td>
<td>Chessboard and Buzzer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Encourage students to complete a structured test plan to help identify and fix errors in their program code.
Quick Reflection
What is the purpose of the Slider in this program?

Checks for Understanding
● When is the compound conditional used within the program?
● Which is an example of where compound conditionals are used?

Chili Challenges
Students can self-select or teacher can assign an extension activity.

Experiment with adding text to the output to state what the image shows. Can you add text output for Rock, Paper, Scissors?

Experiment with displaying the Rock, Paper, Scissors images for a set time. Can you use the ‘wait’ and ‘clear’ blocks to restrict the displays?

Experiment with adding a game prompt. Can you use variables, prompts and logic to ask the user if they are ready to play?

Reflect Students can complete activities in the Mission Journal.