SCRATCH:
SELECTION / BRANCHING / IF / IF...ELSE / COMPOUND CONDITIONALS / ERROR TRAPPING

BY
MR. CLAUSEN
SELECTION (BRANCHING) – IF COMMAND

• Selection or Branching is computer programming jargon for choosing whether a line of code or a group of lines of code should be executed (run).

• It is often combined with repetition (repeat commands) as shown in the example below.
CONDITIONS FOR CONDITIONALS

• We can find hexagonal shaped conditions under the Sensing (light blue) group and the Operators (green) group.

• One of the easiest tasks we can do is to ask for some input and make a choice depending on what was entered.
CONDITIONALS: ONE WAY BRANCH - IF

- Sometimes you wish to execute a section of code only when a condition is true.
- The code inside the “if” block is executed only when the condition is true.
- When the condition is false, the program continues executing any code below the “if” block.
- This is called a “one way branch”, as shown in the example below.
CONDITIONALS: TWO WAY BRANCH – IF ELSE

• Sometimes you wish to execute one section of code when a condition is true, and a different section of code when the condition is false.

• The code following the “if” is executed when the condition is true, while the code following the “else” is executed when the condition is false.

• This is called a “two way branch”, as shown in the example below.
Sometimes there are more than two selections of code that you wish to choose to execute depending on certain conditions.

This is called a “multi way branch”, as shown in the example to the right.

It is constructed by inserting “if” or “if else” blocks inside an “else” block.

These are called “nested” or “extended” if statements.
SEQUENTIAL SELECTION

• Sequential selection is a list of “if” statements without “else if” statements.

• It is a good idea to avoid sequential selection whenever possible.

• Let’s compare a multi-way branch with sequential selection.

• Sequential selection is shown on the left of the two code samples.

• The logic is easier to follow, but the code is not as efficient, because the computer has to evaluate all of the if statement conditions every time the code is run.
LOGICAL OPERATORS & COMPOUND CONDITIONS

• There are 3 logical operators in Scratch, “and”, “or”, “not”.

• “Not” means the opposite of the condition or Boolean expression that follows.

• In an “or” statement, if either condition is true, or both conditions are true than the entire Boolean expression (compound condition) is true.

• In an “and” statement, both conditions have to be true in order for the compound condition to be true.

• Logical operators (compound conditions) can be used in ‘if” statements AND in Repeat Until statements.
USING COMPOUND CONDITIONALS

• Compound conditionals can make your code shorter and more efficient.
• Consider the following equivalent sections of code. Both tell the sprite to rotate 180 degrees if the sprite’s “x position” is greater than 150 OR if the sprite’s “x position” is less than -150.
COMPOUND CONDITIONALS: ERROR TRAPPING

• This example uses a compound conditional in a Repeat Until statement to “error trap” input from the user (check for valid input within an acceptable range). Remember, “and” means BOTH conditions must be met.

• Ask the user for input first. Then check if the input is within the expected range in the condition of a Repeat Until. Give the user feedback if the input is not within the range, and ask the user for the information again in the “body” of the repeat until loop.