Visual programming blocks – Loops

The **Control** category contains blocks that control whether other blocks placed in their **body** are run. (For example, in the "repeat" block below, the body contains the "print" block and its input.) There are two types of control blocks: IfElse (described on its own page) and this one, which controls how many times the body is run and, in some cases, the value of a variable used within the body. These structures are called **loops** since the body is repeated (possibly) multiple times, reminiscent of a rope containing loops. Each pass through the loop is called an **iteration**. For more information, see https://en.wikipedia.org/wiki/Control_flow#Loops.

Blocks for Loop Creation

repeat

The simplest "repeat" block runs the code in its body the specified number of times. For example, the following block will display "Hello!" ten times.

Image not found or type unknown

repeat while

Imagine a game in which a player rolls a die and adds up all of the values rolled as long as the total is less than 30. The following blocks implement this game:

1. A variable named **total** gets an initial value of 0.
2. The loop begins with a check that **total** is less than 30. If so, the blocks in the body are run.
3. A random number in the range 1 to 6 is generated (simulating a die roll) and stored in a

variable named **roll**.
4. The number rolled is displayed.
5. The variable **total** increases by **roll**.
6. At end of the loop, control returns to step 2.

Image not found or type unknown

When the loop completes, any subsequent blocks (not shown) would be run. In our example, the loop would end after random numbers in the range 1 to 6 had been displayed a certain number of times, and the variable **total** would contains the sum of these numbers, which would be guaranteed to be at least 30.

For more information, see https://en.wikipedia.org/wiki/While_loop.

repeat until

"Repeat while" loops repeat their bodies *while* a condition is true. Repeat-until loops are similar except that they repeat their bodies *until* a condition is true. The following blocks are equivalent to the previous example because the loop contains until **total** is greater than or equal to 30.

Image not found or type unknown

count with

The **count with** block (called a for loop in most programming languages) advances a variable from the first value to the second value by the increment amount (third value), running the body once for each value. For example, the following program displays the numbers 1, 3, and 5.

Image not found or type unknown

The following two loops will each display the numbers 5, 3, and 1. The first value can be larger than the second. The behavior is the same whether the increment amount (third value) is positive or negative.

Image not found or type unknown

for each

The **for each** block (see https://en.wikipedia.org/wiki/Foreach) is similar, except instead of giving the loop variable values in a numeric sequence, it uses the values from a list in turn. The following program displays each element of the list: "alpha", "beta", "gamma".

Image not found or type unknown

Loop Termination Blocks

Most loops run until the terminating condition (in the case of **repeat** blocks) is met or until all values have been taken by the loop variable (in the case of **count with** and **for each** loops). Two rarely needed but occasionally useful blocks provide additional means for controlling loop behavior. Although the examples below show **for each** loops, they can be used with any type of loop.

continue with next iteration

The **continue with next iteration** (called continue in most programming languages) causes the remaining code in the body to be skipped and for the next iteration (pass) of the loop to begin.

The following program displays "alpha" on the first iteration of the loop. On the second iteration, the **continue with next iteration** block is run, skipping the display of "beta". On the final iteration, "gamma" is displayed.

Image not found or type unknown

break out of loop

The **break out of loop** block provides an early exit from a loop. The following program displays "alpha" on the first iteration and "breaks out" of the loop on the second iteration when the loop variable is equal to "beta". The third item in the list is never reached.

Image not found or type unknown

Revision #3

vytvořené 2 roky nazpět uživatelem Admin

aktualizováno 2 roky nazpět uživatelem Jiri Krulis