Loops in AgentCubes online

AgentCubes introduces the new **Repeat loop commands** to comply with ISTE/CSTA K-12 Programming requirements.

**Using Loops**

The repeat loop is an action that you find in the bottom of the Actions palette. Use a repeat loop by dragging it into the THEN part of a rule. Add actions to the repeat loop by dragging any kind of action, including repeat actions, into the inside part of the repeat loop.

![Actions palette](image)

The repeat loop has a *count expression*, defaulting to “2”, indicating how many times the loop will be run. This count expression may assume a number of forms (sorted from basic to more sophisticated):

- **Constant**: A single constant positive integer value, e.g., “3”, “678” or “10000”
- **Expression with functions but without variables**: e.g., “random (3)” will repeat 0, 1, or 2 times.
- **Expression with agent attributes**: e.g., “age”
- **Expression with simulation property**: e.g., “@score”
- **Expressions with functions, agent attributes and simulation properties**, e.g., “age + @score + random (10) + 5”

**Examples of Loops**


1. **Loops as syntactic shortcut**: A bug is supposed to move onto the gold coin. The bug would have to move 5 times which could be achieved with 5 separate move actions. Instead of duplicating the move action 5 times a single repeat 5 times action containing a move action will do the trick:
2. **Loops to animate.** Loops can be used to make iterative animations. In the same project, when the bug does reach the gold coin we want it to flash between two different shapes (red bug, blue bug). This could be done using the While-Running method but because we are not interested in the parallel animation of multiple bugs, there is only one in this world, the use of a repeat loop is simpler.

Notice the use of the wait action. It will delay the switching from one shape to the next. Without this delay the animation would not be visible to the user. There are two actions that can be used inside a loop to make intermediate results such as changing to a new shape visible:

1. **Finish-animations:** All the pending animations will be finished. This will sequentialize the execution of agents.
2. **Wait**: Wait includes a finish-animations and then waits a user specified number of seconds before the next action continues.

### Further Experiments

- Experiment with using finish-animations and wait in the “Loops as Syntactic shortcut” example.
- Explore the use of different expressions for the repeat count. For instance, try the use of agent attributes.

### Philosophy

You may just want to know how to use loops and not care so much why one should, or should not, use them. In that case you may safely skip to the next section of this document. If, however, you are asking yourself why we did we wait with the introduction of Loops in AgentSheets/AgentCubes for about 22 years here is the philosophy for not providing a loop command earlier. First of all, this was never an issue of labor. The original drag and drop language developed in AgentSheets prototyped in 1994 actually did have a loop command but we decided to remove it from AgentSheets. Reasons to have loop in computer science are manifold but the short story is that loops and parallel programming have a difficult relationship. Simple examples of using loops, such as the many examples of loops introduced in code.org tutorials, e.g., Angry Bird getting the pig, provide somewhat plausible use cases of loops, but there are many use cases of using loops interfering with the opportunity to employ parallelism. Many algorithms, for instance the sorting of numbers through bubble sort, or operations on very data vectors such as DNA described in programming for biologist textbooks, will strongly suggest the use of a FOR loop implying sequentiality, i.e., the execution of computing one at a time, and order, e.g., comparing bubbles in bubble sort left to right. Not only is this not necessary in many cases, it will also likely ruin the opportunity for parallel execution. For instance, there is really not reason why in bubble sort bubbles should be sorted left to right nor is it necessary to do this sequentially. Modern computers have multiple cores that can execute code parallel. Using these kinds of unnecessary loops can completely ruin parallel executing wasting the power that your computer is capable of to solve problems efficiency. AgentSheets/AgentCubes core ideas go back to massively parallel computing. The original prototype of AgentSheets was running on Connection Machines with up to 64,000 CPUs in 1990. The ideas developed back then carry over to modern architectures including multi core CPUs, which these days can even be found on low end smart phones, or GPUs (Graphical Processing Units used for computer graphics). For many applications loops are considered harmful for parallel operations.

The AgentSheets/AgentCubes model makes it simple to express a massive parallelism precisely because it tries to avoid forms of unnecessary looping. For instance, create a large set of agents and make them move around randomly. They will do so in parallel, figuring out how to properly stack up without the need of complex user code. This is not to say that AgentSheets/AgentCubes did not support loops before the introduction of the repeat action but it did it carefully to minimize potential harm to parallel programming.

The two mechanisms of looping in AgentSheets/AgentCubes are: 1) Implicit Loops: all the rules executed in the While-Running method are, conceptually speaking, part of a simulation loop. 2) Recursion: A method may, directly, or indirectly through some additional methods, call itself again.