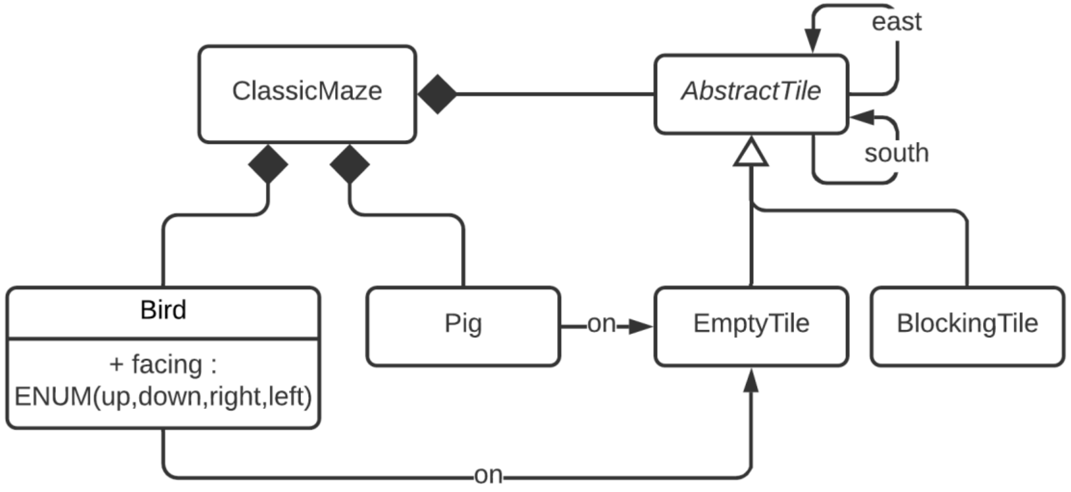
Calm Birds HoC Solutions

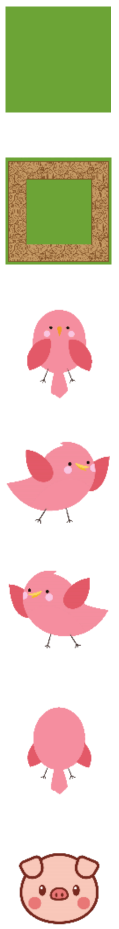
(Traditional and Model-driven)

https://studio.code.org/hoc/1

Calm Birds

Metamodel and Graphical Concrete Syntax





Empty Tile

Wall Tile

Bird (down)

Bird (right)

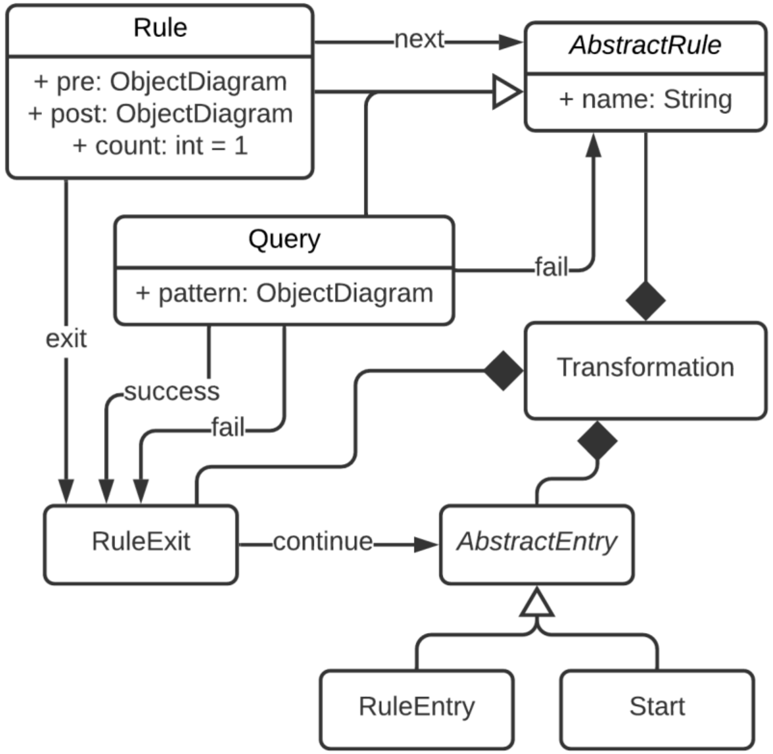
Bird (left)

Bird (up)

Pig

Model-driven Alternative

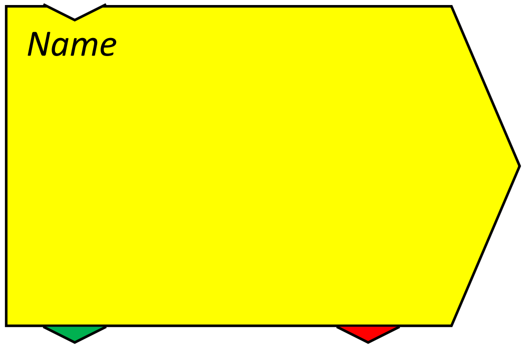
Metamodel



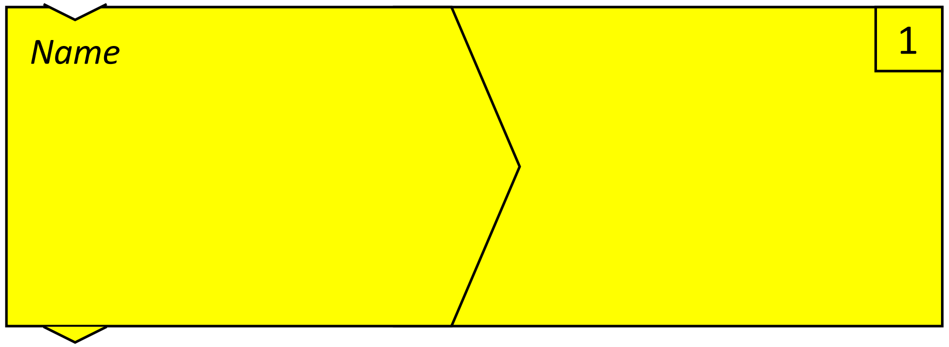
Model-driven Alternative

Graphical Concrete Syntax

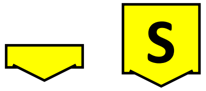
Query



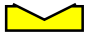
Regular rule



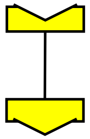
Rule entry (start)



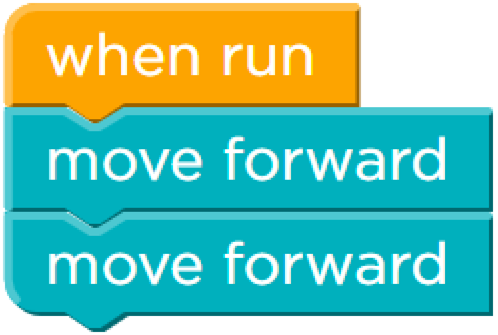
Rule exit (finish)



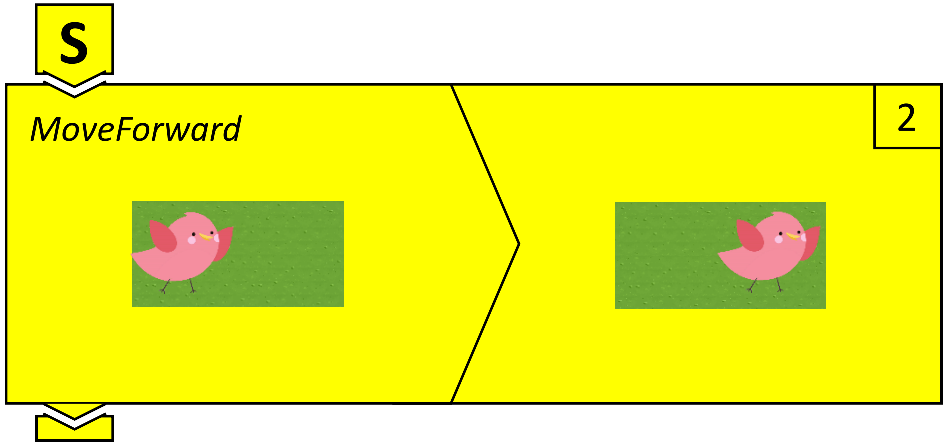
Rule connector

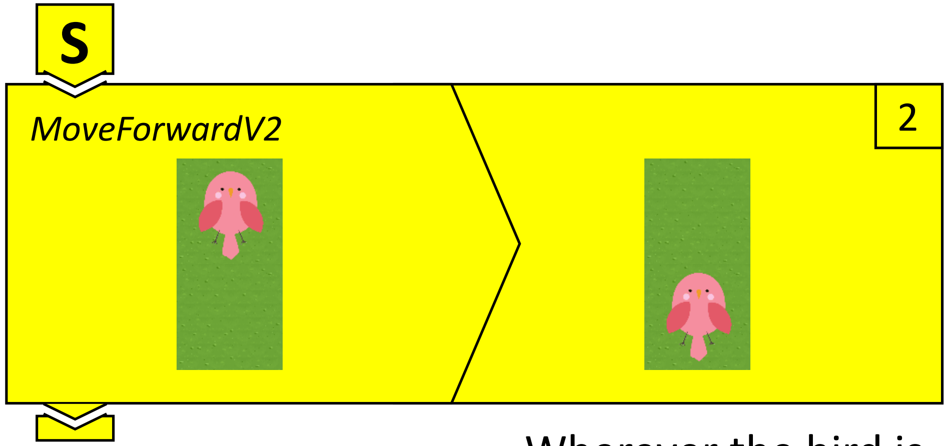


Maze 1

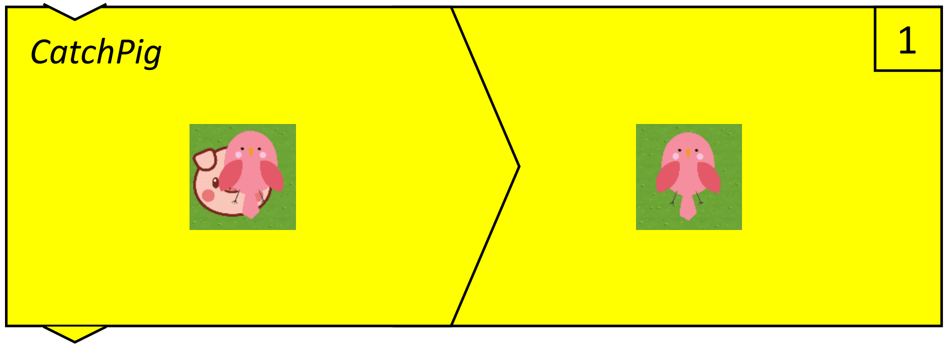


```
when run
move forward
move forward
```



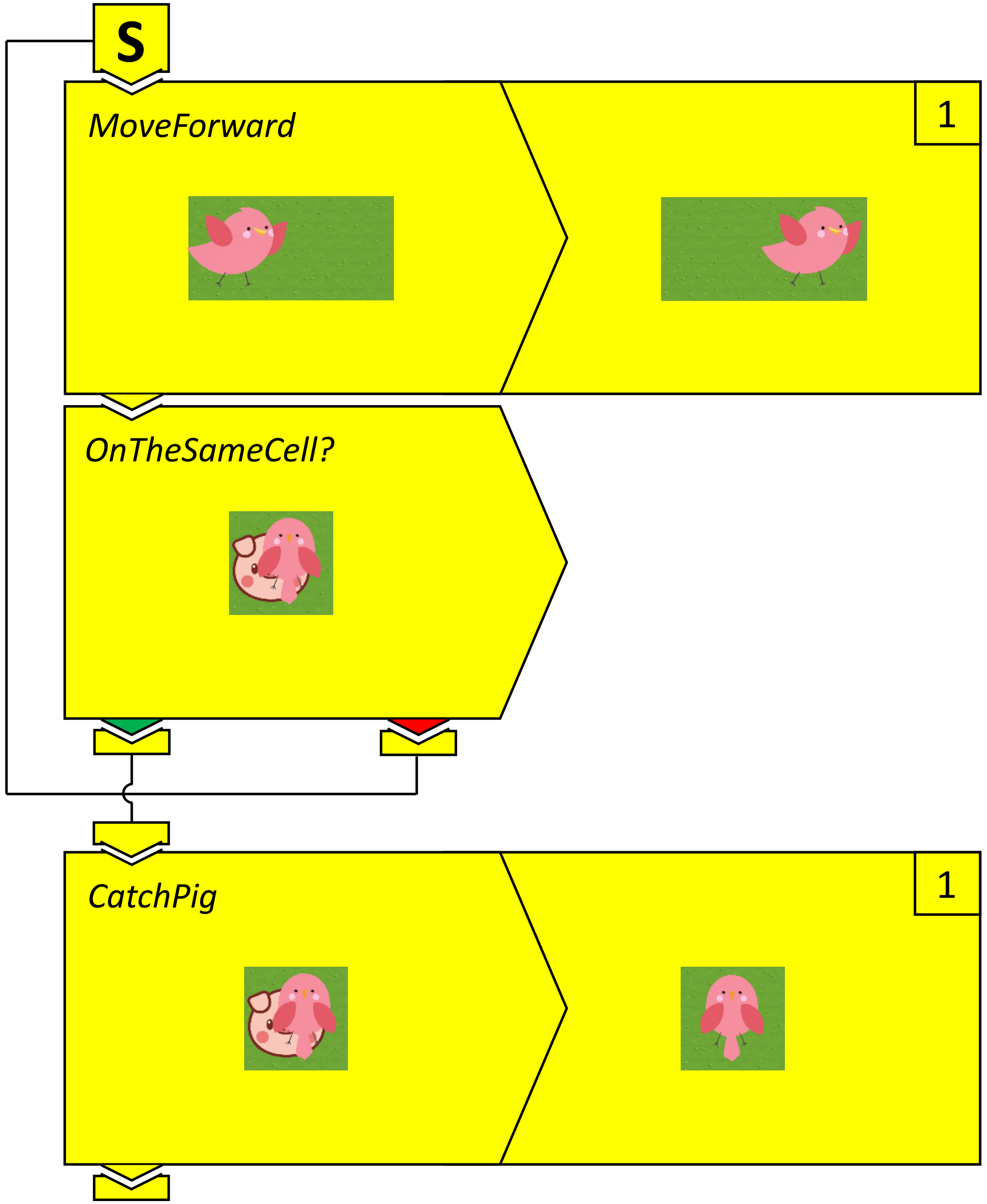


Wherever the bird is facing, move forward should work regardless.



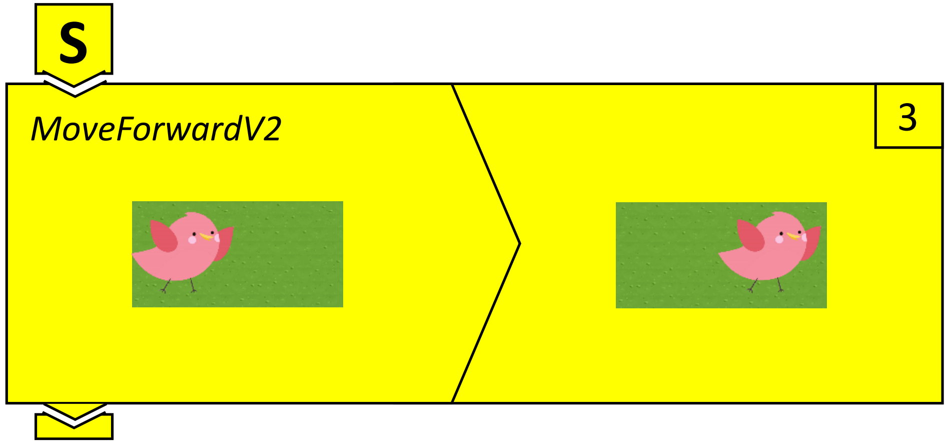
This rule is auto provided in the hour of code when the bird and pig is on the same cell, but we can also provide it like this.

General



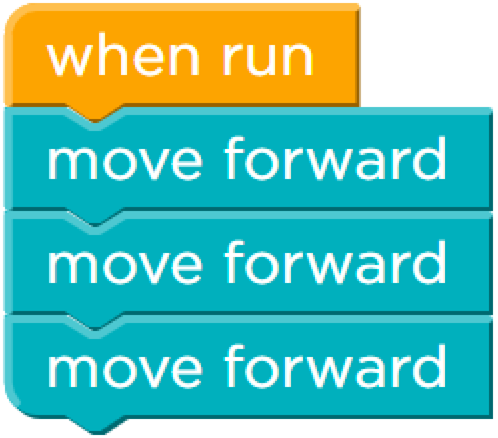
Eventually, we will give the user freedom. So they can write as flexible solutions as they can. For example, Maze 1 can be solved like this if we assume the bird won't catch pig automatically when they are on the same cell.

Maze 2



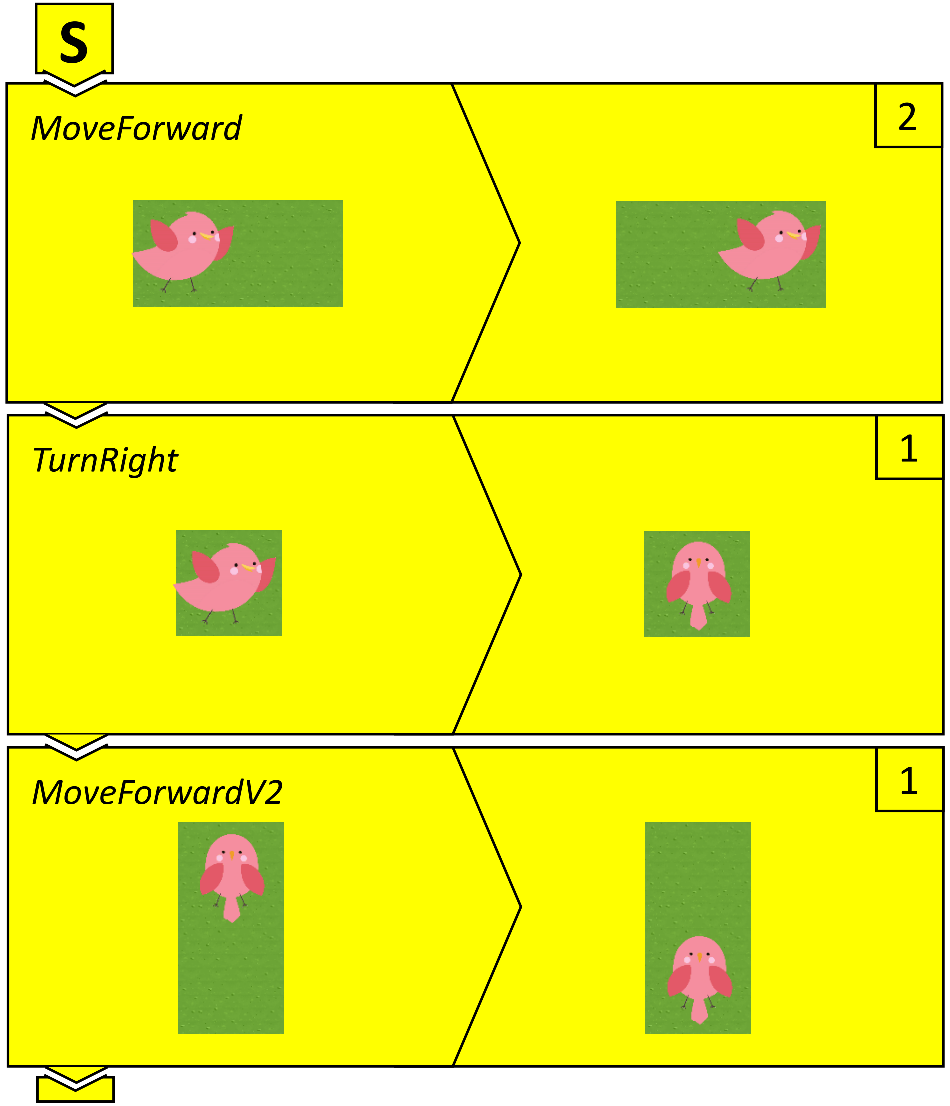
Because in the maze, the bird needs to go down, probably the users will tend to create this kind of moveForward or at least, visually this makes more sense.

Just run it 3 times basically.

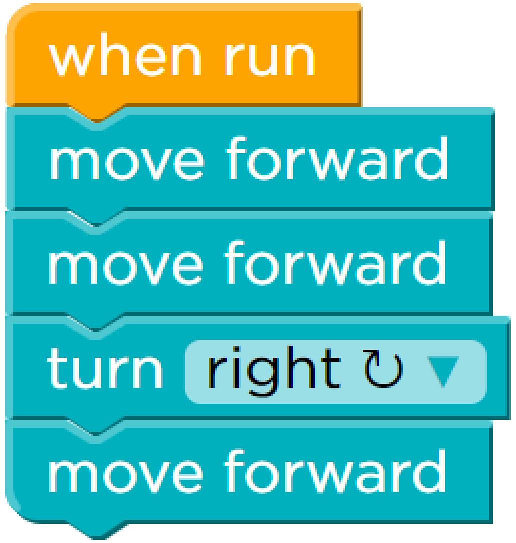


```
when run
move forward
move forward
move forward
```

Maze 3

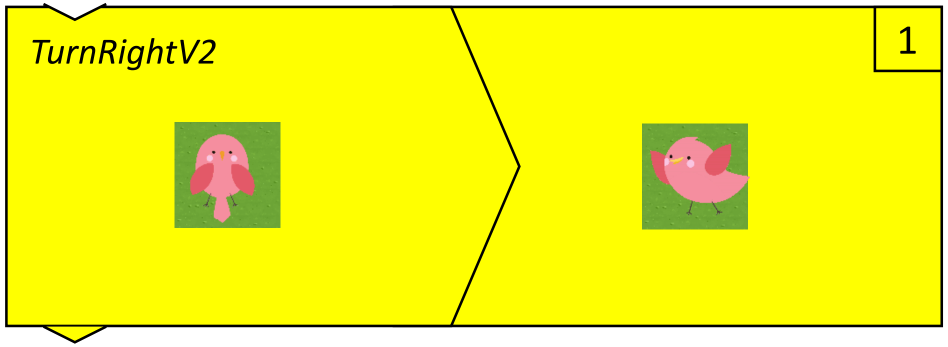


We can let the users reuse the rules they have created. The rules might be copied/pasted anyway. Technically, these moveforward rules might be same. But visually this makes more sense.



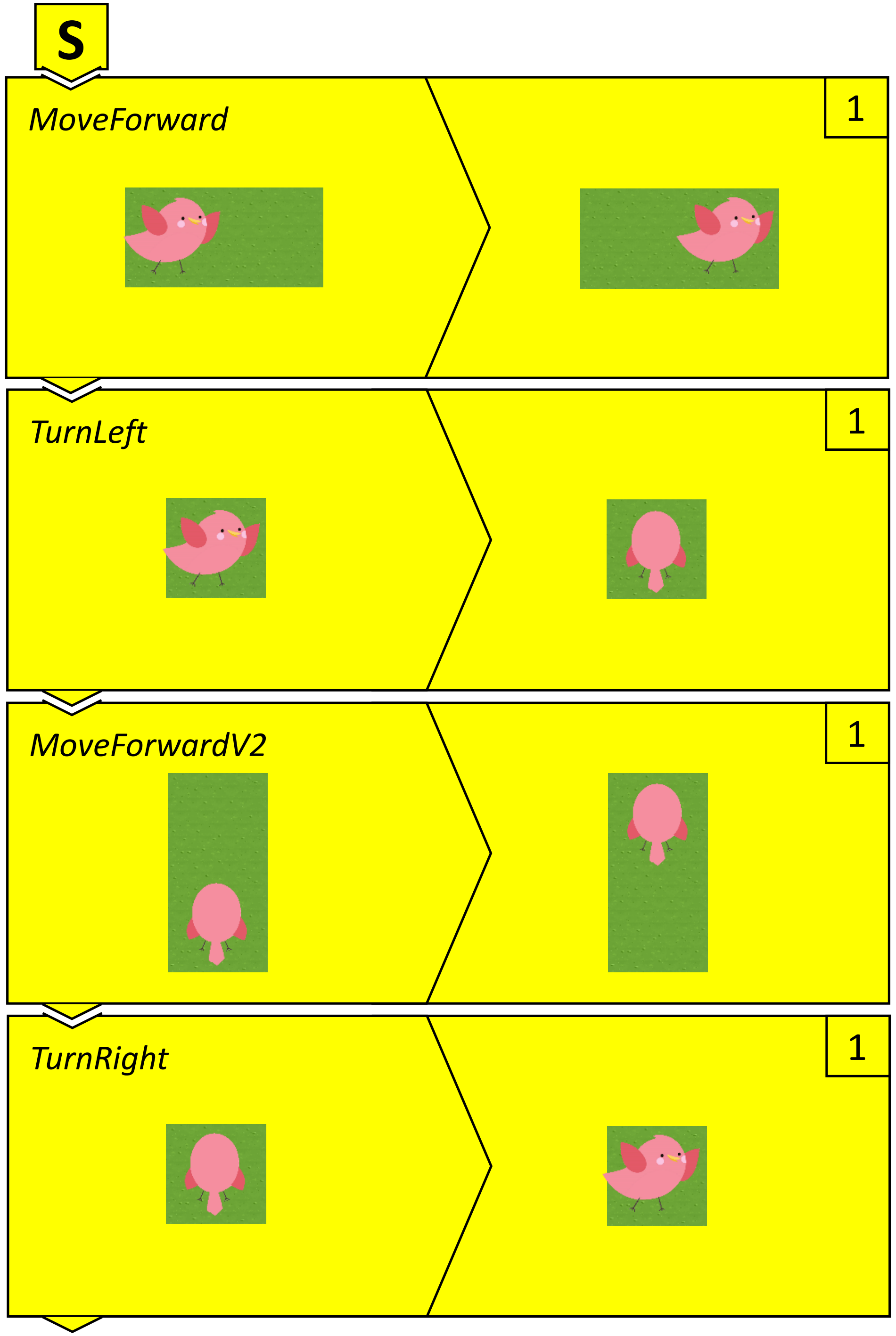
```
when run
move forward
move forward
turn right ↻ ▼
move forward
```

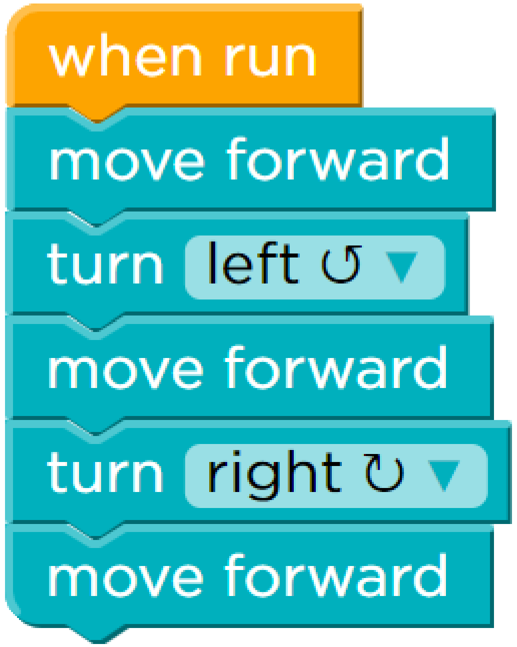
Again, there is an alternative to turning right. Both does the same thing.



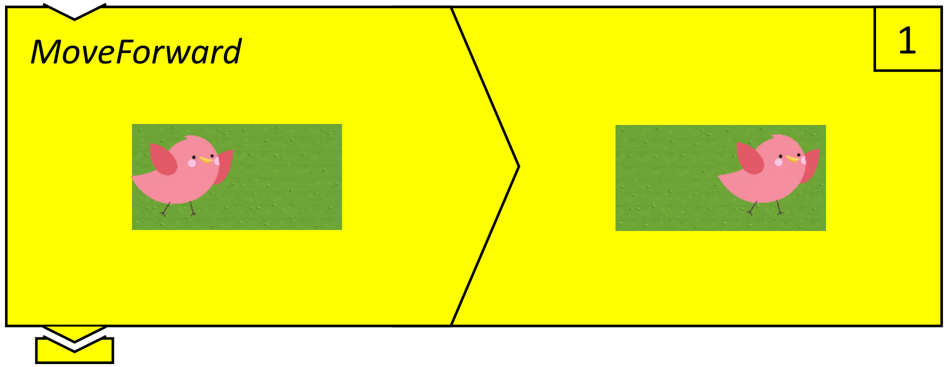
Maze 4

All 5 blocks connected.



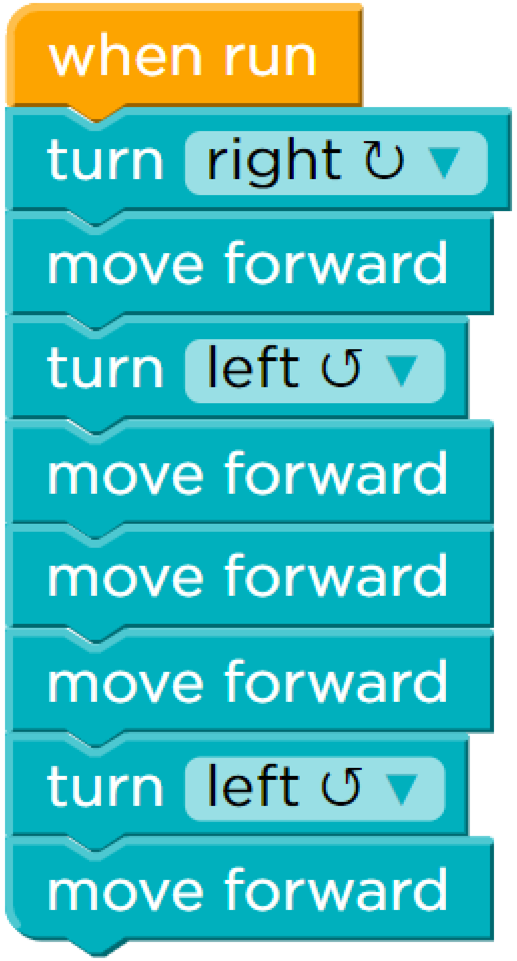


```
when run
move forward
turn left ↺ ▾
move forward
turn right ↻ ▾
move forward
```



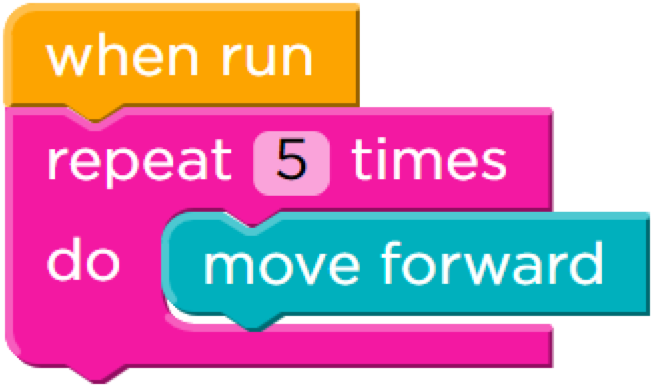
Maze 5

One to one mapping with necessary facing direction for the bird, except calling moveForward 3 times in the middle. So it maps to 6 rules in model-driven alternative (8 blocks here in traditional.)



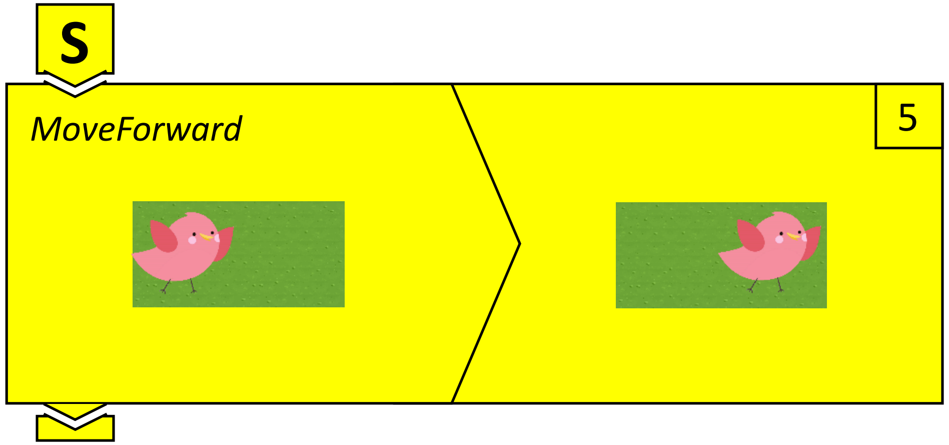
```
when run
turn right ↻ ▼
move forward
turn left ↺ ▼
move forward
move forward
move forward
turn left ↺ ▼
move forward
```

Maze 6



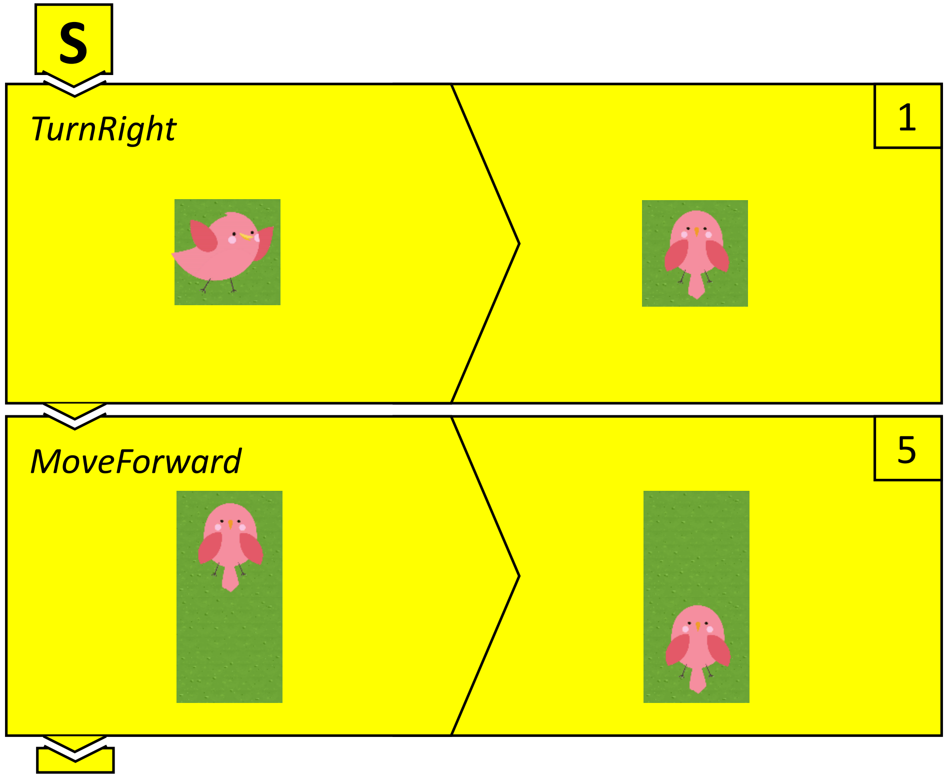
```
when run
repeat 5 times
do  move forward
```

This is already given in our system with the count attribute.



Maze 7

Again, this is already given in our system with the count attribute.



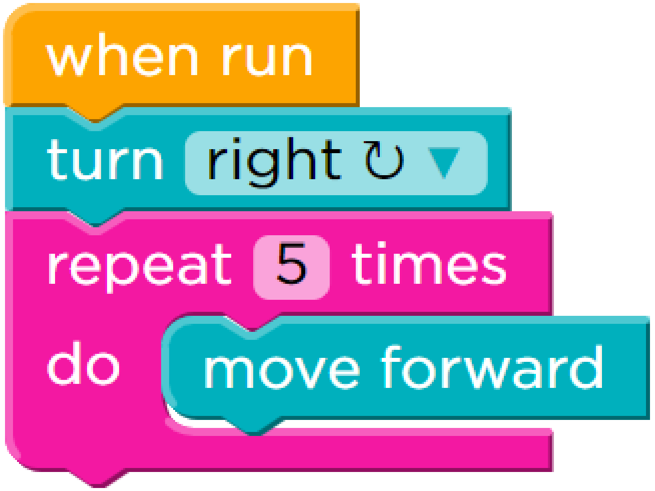
S

TurnRight

1

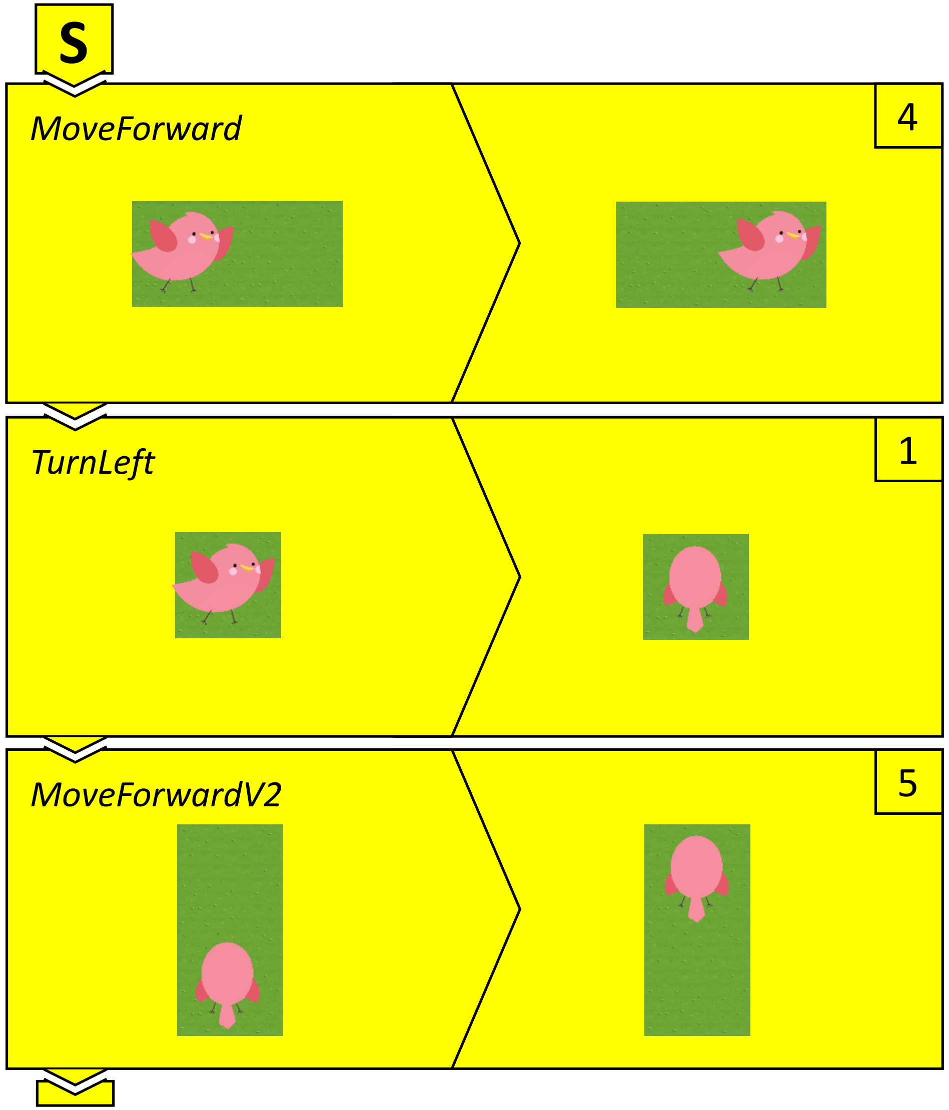
MoveForward

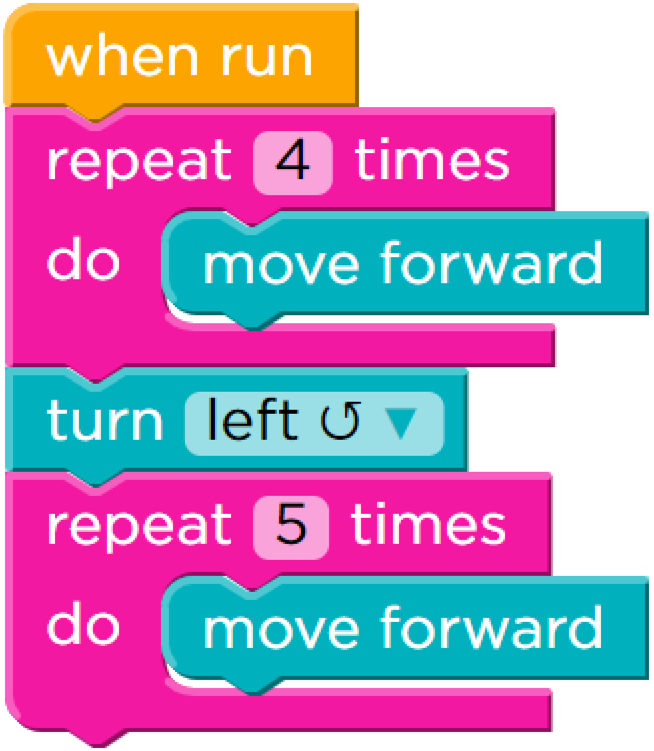
5



```
when run
turn right ↻ ▾
repeat 5 times
do  move forward
```

Maze 8

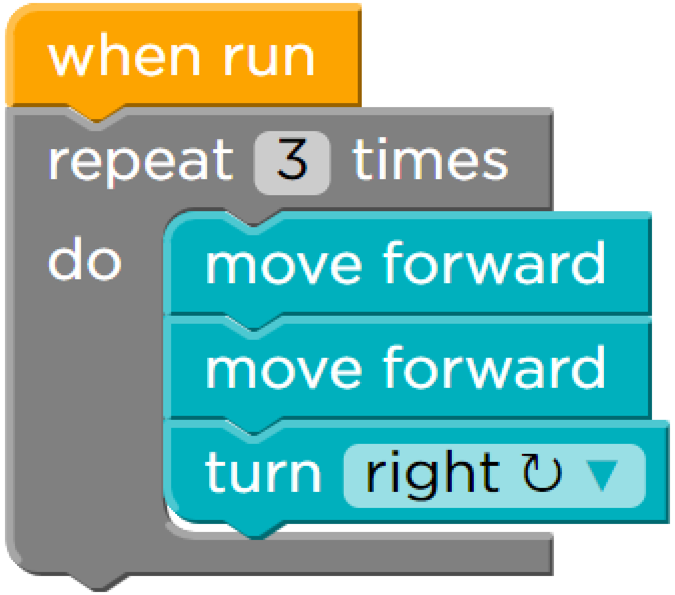




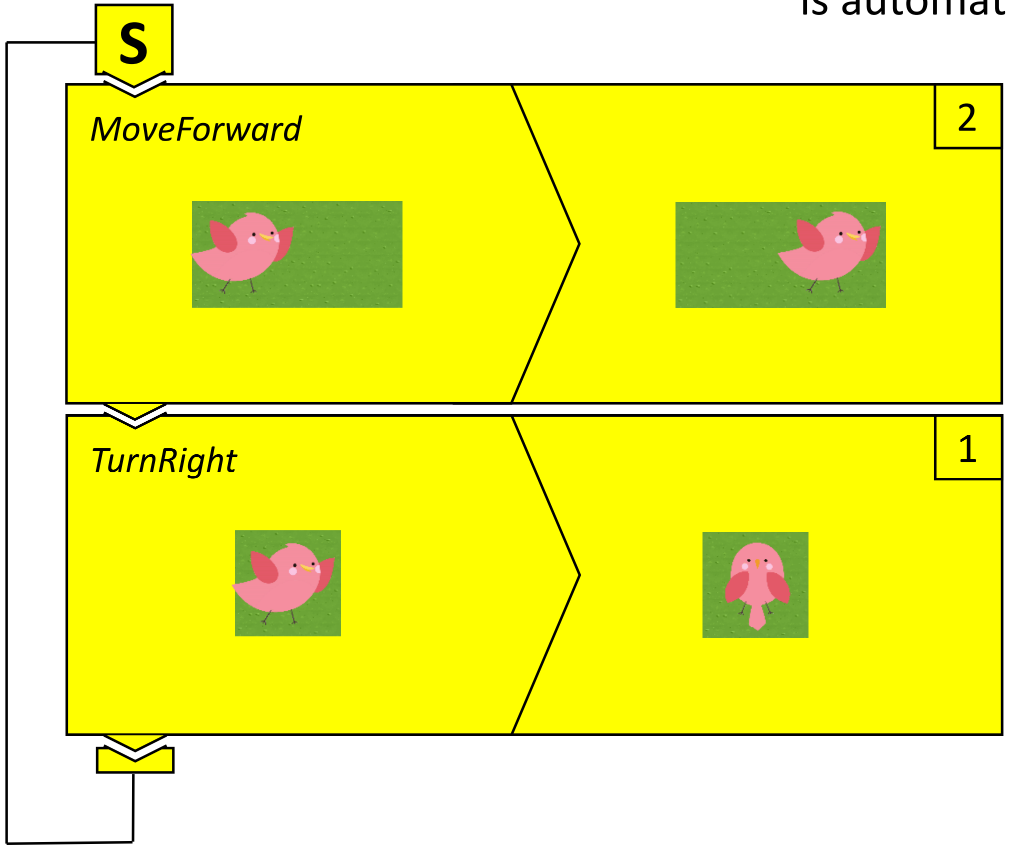
```
when run
repeat 4 times
do  move forward
turn left ↺ ▼
repeat 5 times
do  move forward
```

Maze 9

Grey blocks mean, they can’t be deleted. In the model-driven solution, we might use query or not (again if we assume, catch rule is automatically provided).

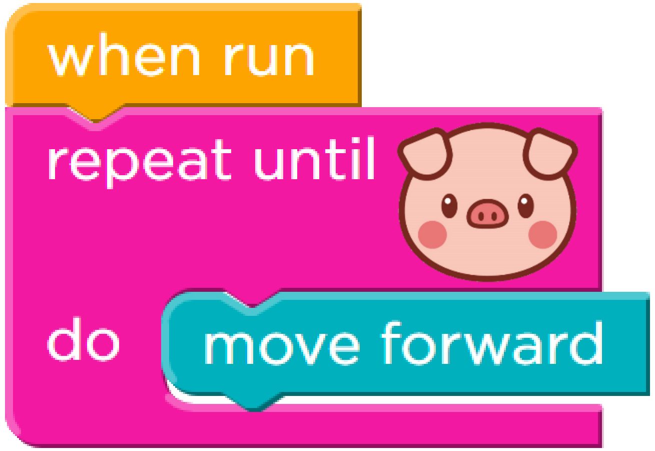


```
when run
repeat 3 times
do  move forward
    move forward
    turn right ↻ ▾
```



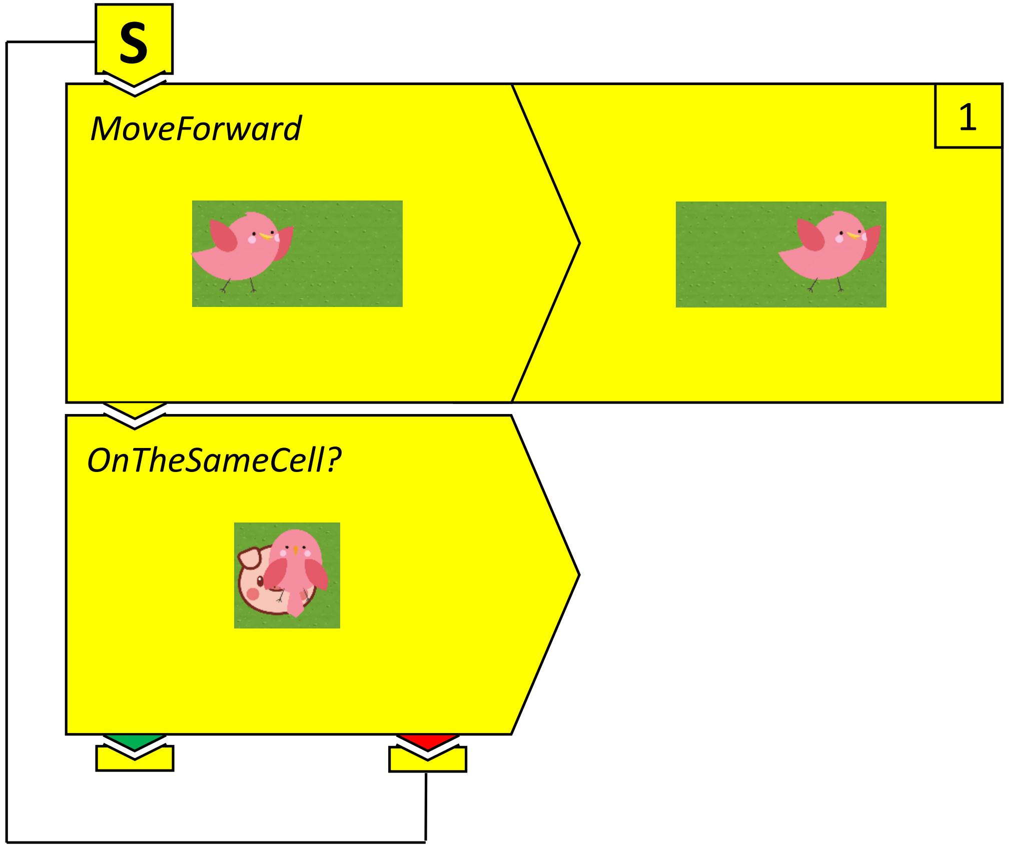
We already discuss how we will structure the puzzles in research directions. Even from start, some of our rules will not be deleted to force the programmer to model the inside of the rules. Then, slowly we will give the users freedom.

Maze 10

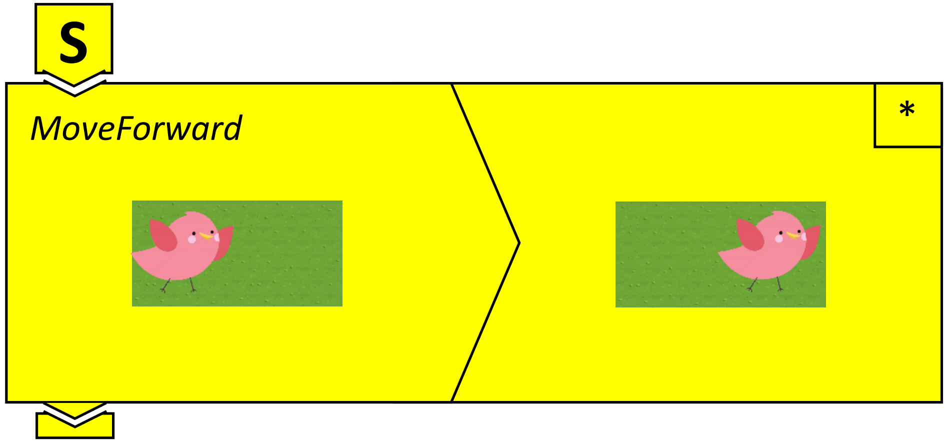


```
when run
repeat until
do  move forward
```

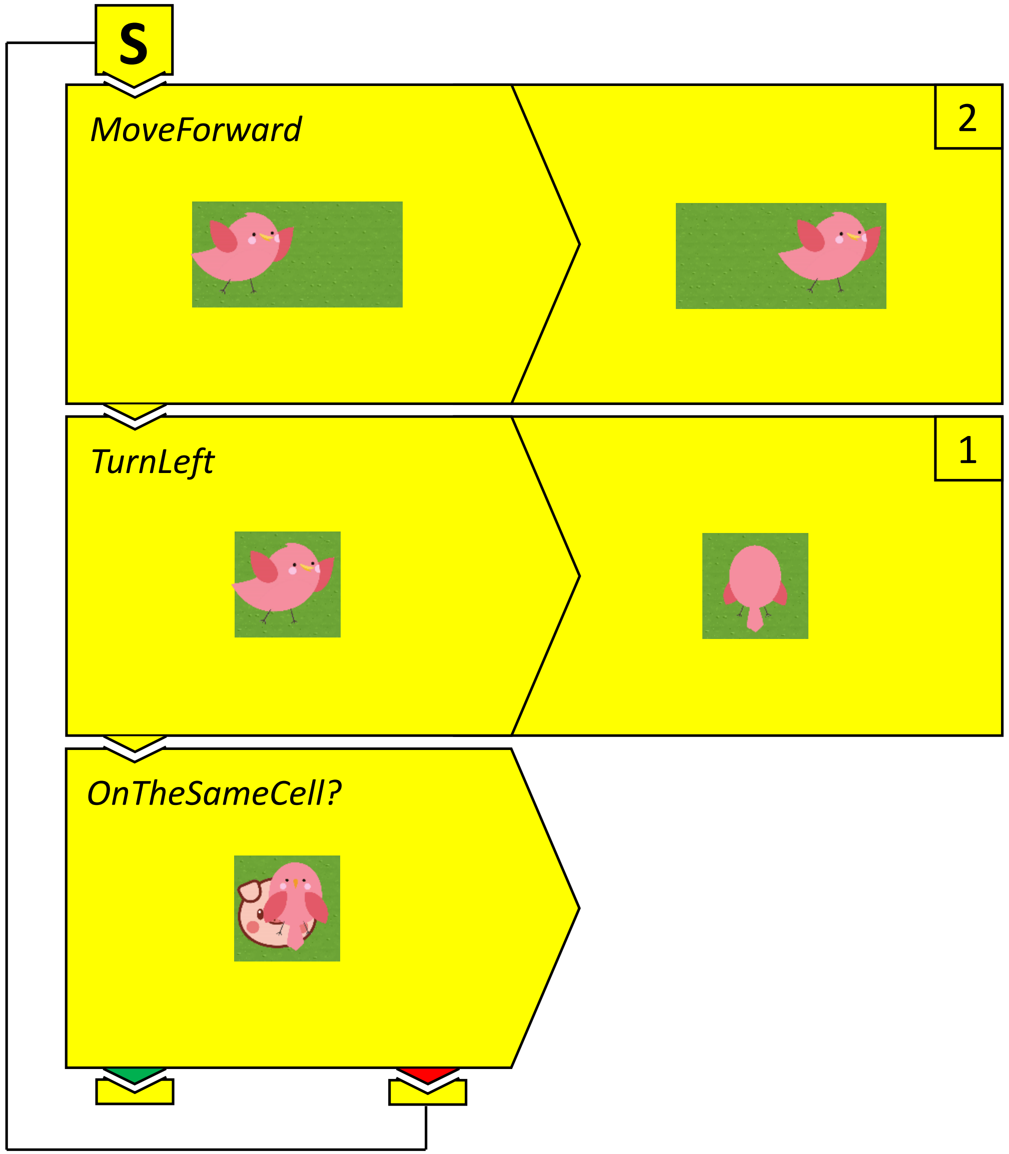
Repeat until is with queries (again assuming CatchPig rule is automatic.)



Another way to implement this is with an asterisk, which means ‘execute a rule as long as possible’.



Maze 11



S

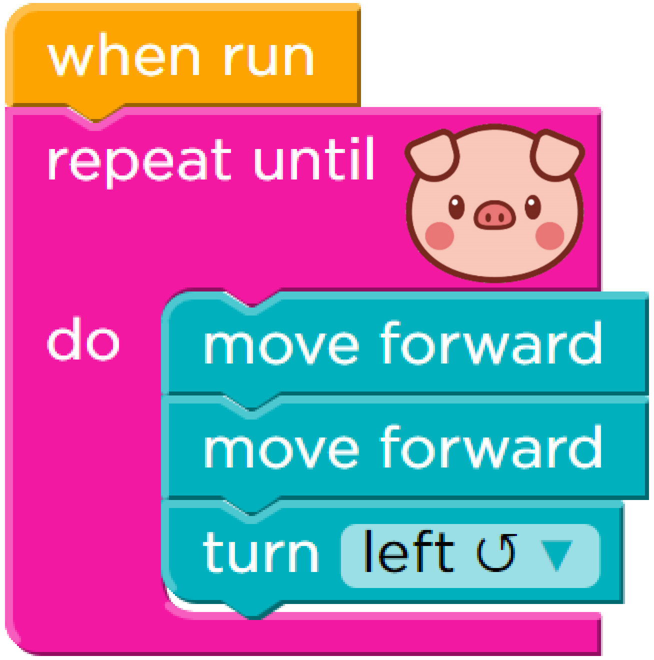
MoveForward

2

TurnLeft

1

OnTheSameCell?



```
when run
repeat until
do  move forward
    move forward
    turn left ↺ ▾
```

We can change the exact numbers to asterisk and it will still do the task.