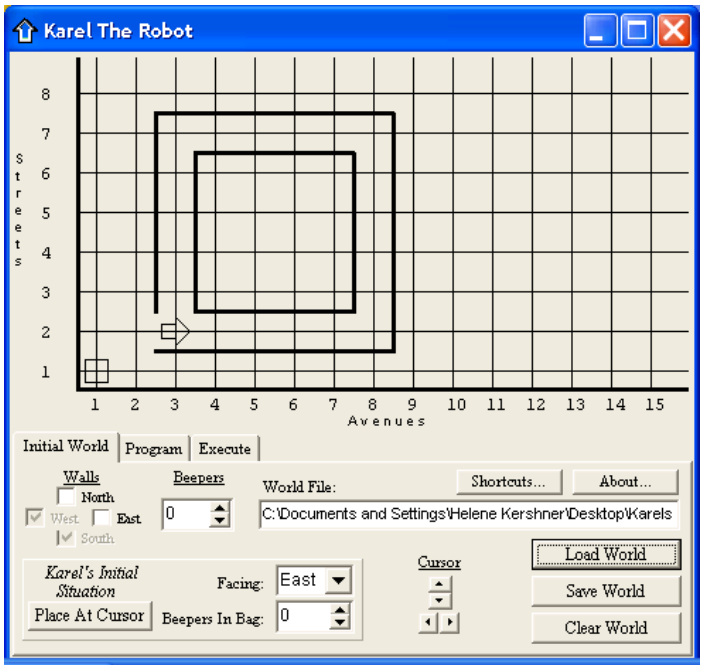
Nested If Statements
While Loop

Five Block Square Maze

Problem statement; Karel needs to move through the five block square maze and stop when he gets to the opening.



Define Output: Karel stops when he locates the opening in the maze.
Define Input. Karel is at the beginning of a 5-block square maze, facing east.

Version 1)
If we didn't know more than the primitive instructions we would do something like:

```
Move;
Move;
Move;
Move;
Move;
Turnleft;
Move;
Move;
Move;
Move;
Move;
Turnleft;
…
```

Nested If Statements
While Loop

But we do know more than the primitive instructions and we have tools to let Karel make some basic decision. Remember decisions are made using an IF statement.

Version 2)
How do we get Karel to move around the maze with less of the programmer's help? Look at the maze and look at the very straight forward instructions we have written.

What question would Karel need to ask to determine IF it were time to turn left?

```
If front-is-clear then move
     Else
          Turnleft;
```

What happens when we enter that code?
Karel moves exactly one block. Why?
Because, we only told Karel to move one time or turnleft one time.

Version 3)
How do we get Karel to move a bunch of times? Iterate

Would this work?

```
Iterate 30 times
Begin
If front-is-clear then move
     Else
          Turnleft;
End;
Turnoff;
```

Karel certainly moves around the maze when we combine the iterate command with the IF statement.

Version 4)
Are we solving the problem we were asked to solve? NO
Why?
Because, Karel stops when the iterate command finishes, wherever Karel happens to be. Karel does not stop when he reaches the opening as the problem statement required.

Nested If Statements
While Loop

How does Karel know when Karel is at the opening? How does Karel know when Karel has completed his walk through the maze?

We, as the programmer, could count the number of times, but that sort of misses the point. We need to create a decision (IF) that asks essentially “Am I done?”

How can Karel know when to stop? How does Karel know Karel is at the opening?

What does that mean?
Remember Karel can test any of the following conditions

front-is-clear	front-is-blocked
left-is-clear	left-is-blocked
right-is-clear	right-is-blocked
next-to-a-beeper	not-next-to-a-beeper
any-beepers-in-beeper-bag	no-beepers-in-beeper-bag

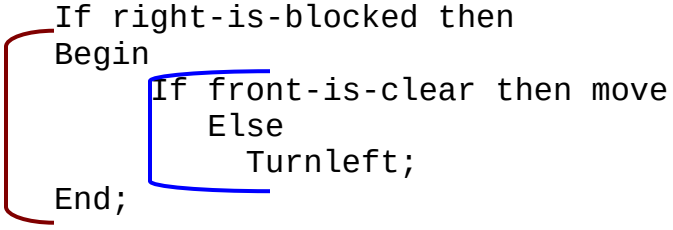
Look at the 5-block maze. Which of these tests can we use to determine if Karel is done?

When the right-is-clear Karel is done.

We could write the following:

```
If right-is-blocked then
Begin
     If front-is-clear then move
         Else
          Turnleft;
End;
```

This is called a **nested IF statement**.
A nested IF statement means there is one If statement wrapped inside another IF statement.



Nested If Statements
While Loop

Version 5)
Let's go back to our earlier program and add this critical test. We will use iterate 30 times again just to see what happens.

We know this is too much, but hopefully he will stop when he comes to the opening.

```
Iterate 30 times
begin
If right-is-blocked then
Begin
     If front-is-clear then move
       Else
          Turnleft;
End;
End;
```

This seems to work.

But watch what happens at the end.

Karel finds the opening, knows not to turn. But the iterate statement means Karel has to really stand in place and do nothing for a bit until the iterate statement has run its course.

So, we have solved our problem but not in a very efficient way.
One more tool the **While instruction.**

The While Instruction or While Loop

Karel uses the While instruction whenever a task is to be repeated an undetermined number of times. Like the Iterate instruction it is used to have Karel repeat a task. But with Iterate, Karel must know IN ADVANCE how many times the programmer wants Karel to repeat a task.

The While instruction essentially combines an IF with an Iterate allowing Karel to determine when Karel has completed a given task based on one of the tests Karel is capable of performing.

Nested If Statements
While Loop

A while statement looks like this:

WHILE < test > DO
    < instruction >

For example:

WHILE front-is-blocked DO turnleft;

Or

```
WHILE front-is-clear DO
    Begin
        Putbeeper;
        Move;
    End;
```

Going back to our program, moving around our maze seems a perfect opportunity to use the While statement. We already have Karel testing for the ending condition (right-is-blocked) only now we have a statement Karel can use that will end Karel's movement when he meets that ending condition.

Let us transform this program into one using a While statement.

```
Iterate 30 times
begin
If right-is-blocked then
Begin
      If front-is-clear then move
        Else
            Turnleft;
End;
End;
```

Using the While loop we replace the IF right-is-blocked test with a While right-is-blocked, and we replace the Iterate 30 times statement. Now we are letting Karel do more of the work.

```
While right-is-blocked DO
Begin
     If front-is-clear then move
       Else
          Turnleft;
End;
```

Notice, now when Karel reaches the end of the Maze Karel stops without sort of "marching" in place as Karel was forced to do using the Iterate statement.