

# **IxD Theory 2: Telecomunicazioni**

*IUAV University of Venice*

*Visual and Multimedia Design graduate programme*

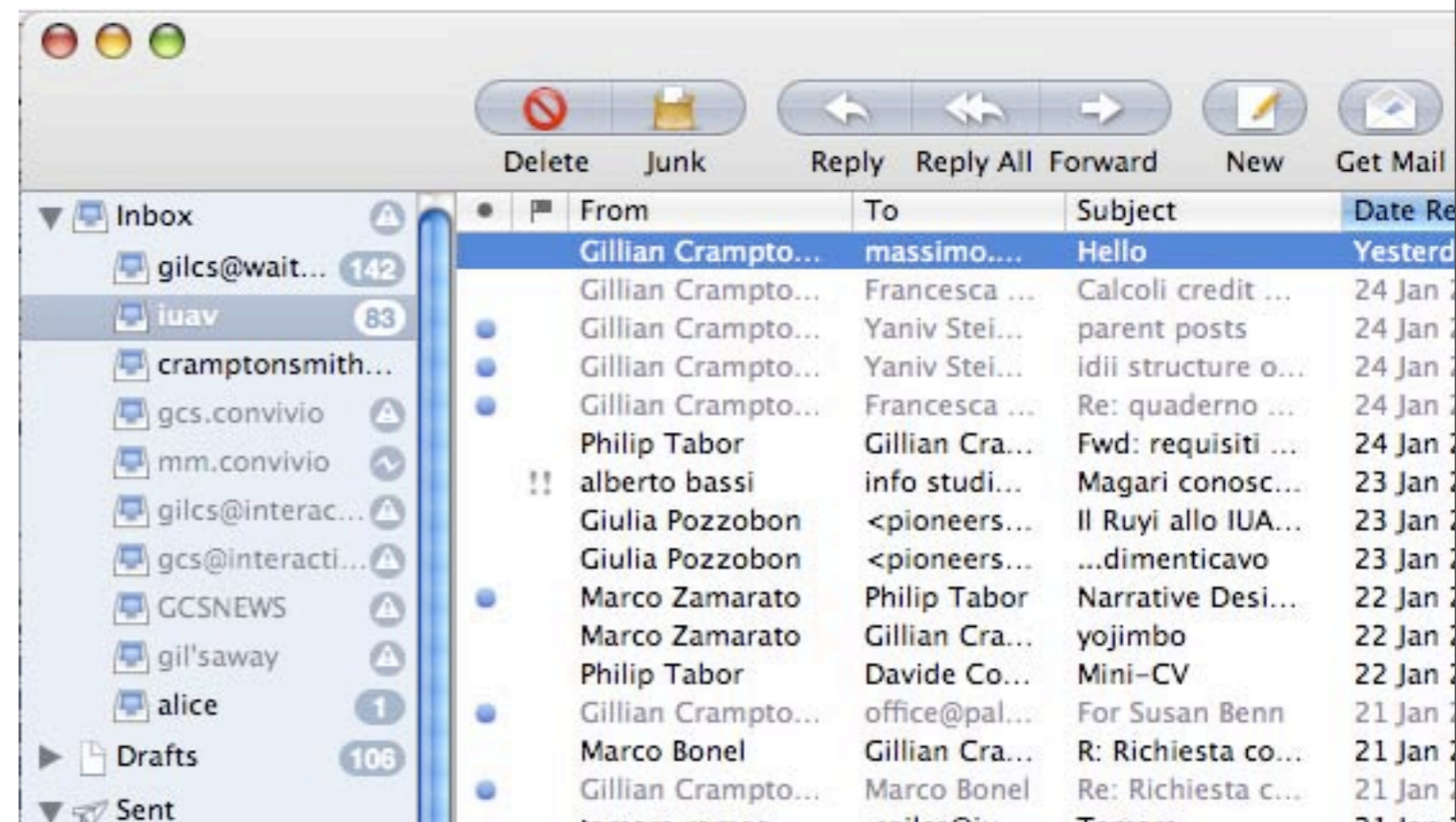
***Programming the computer***

© *Gillian Crampton Smith + Philip Tabor 2009*

# Programming

How do we get from manipulating 0s and 1s in the computer chip to an email program?

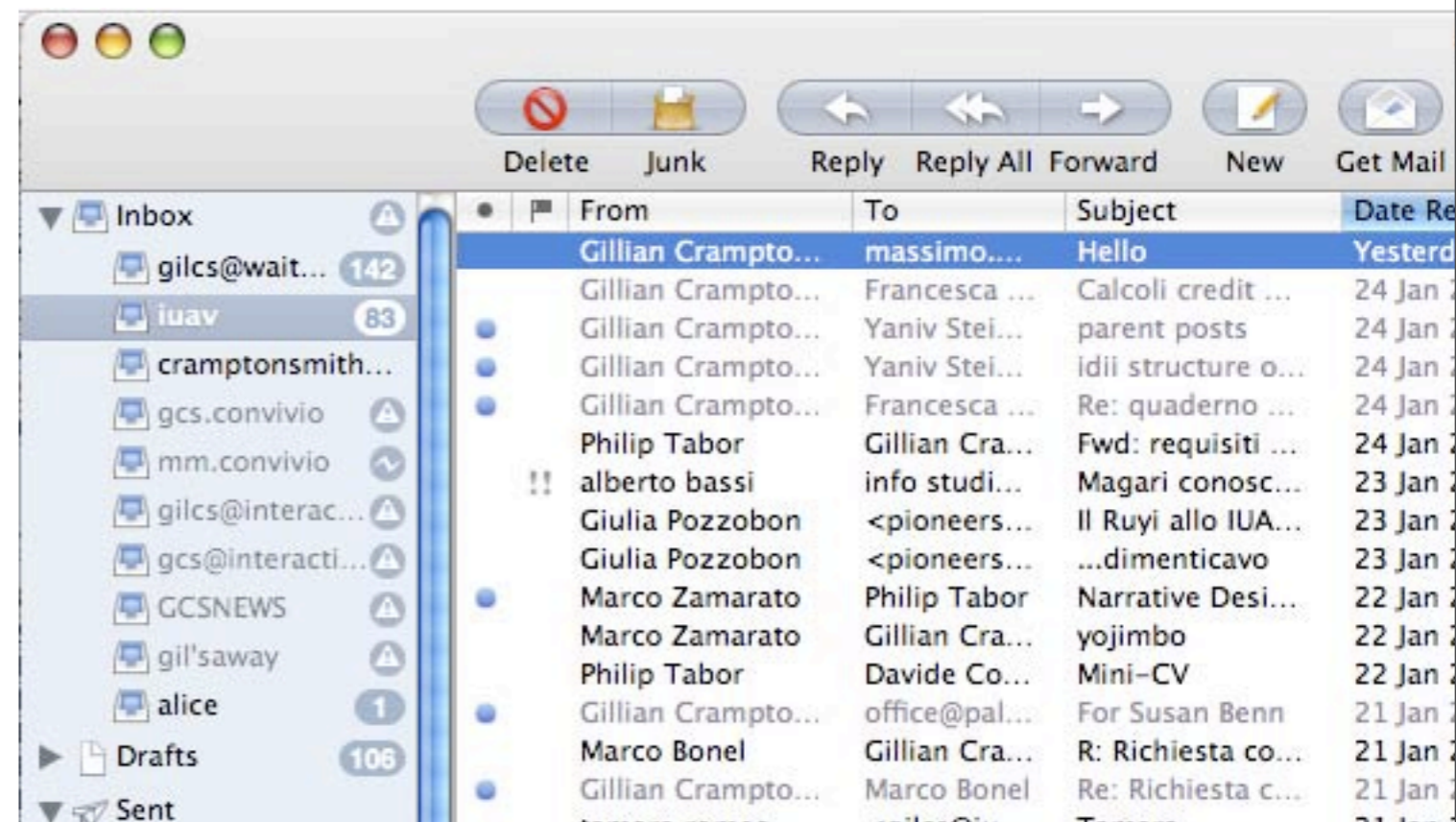
***PROGRAMMING CODE!***



# Programming

We need to program the computer—tell it what to do

This program has to be turned into the 0s and 1s which a computer understands



# Levels of programming languages

**Visual programming languages** – like Visual Basic or MaxSP

**Meta-languages** – that write code for you, like Dreamweaver, the web design program

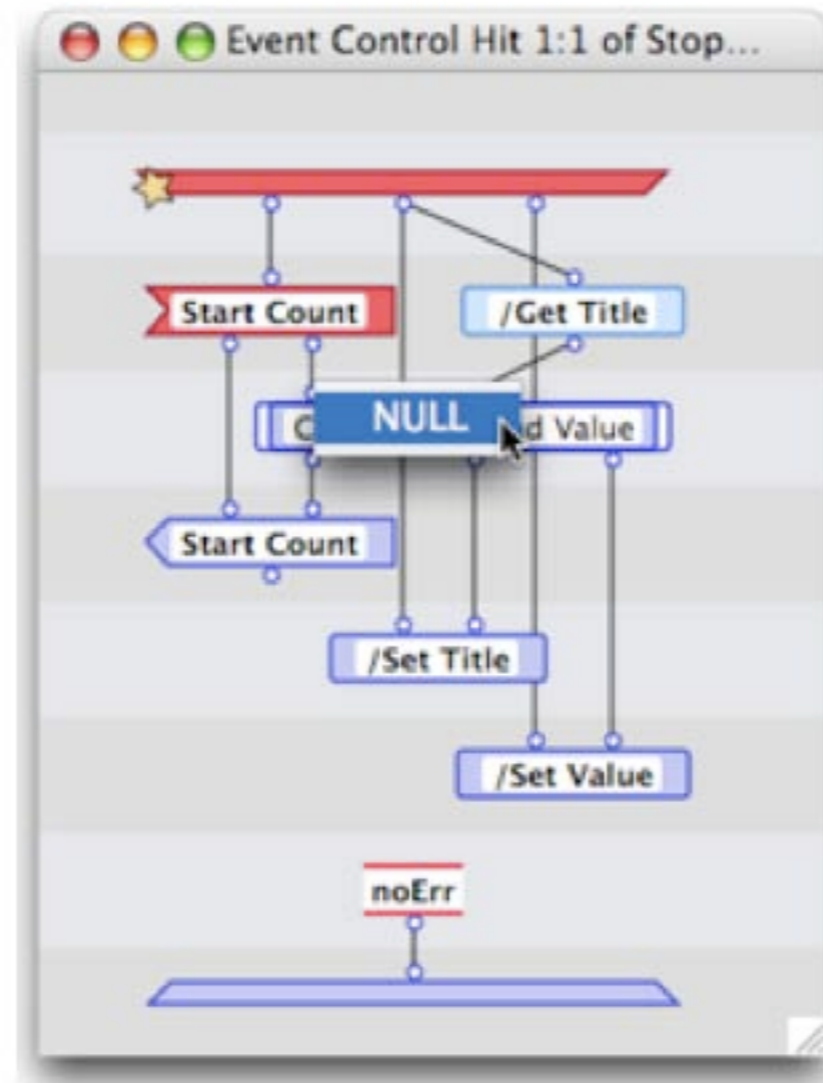
**High level languages** – like C or Java or Processing

**Assembly language** – using mnemonics which are then translated into machine language

**Machine language** (using bit patterns) 0s and 1s

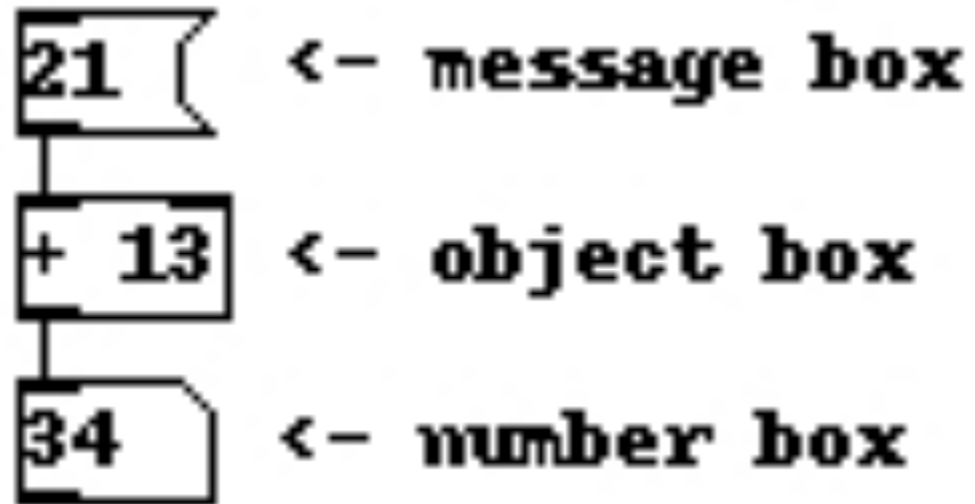
# Visual programming languages

Prograph (now Marten) for the Mac



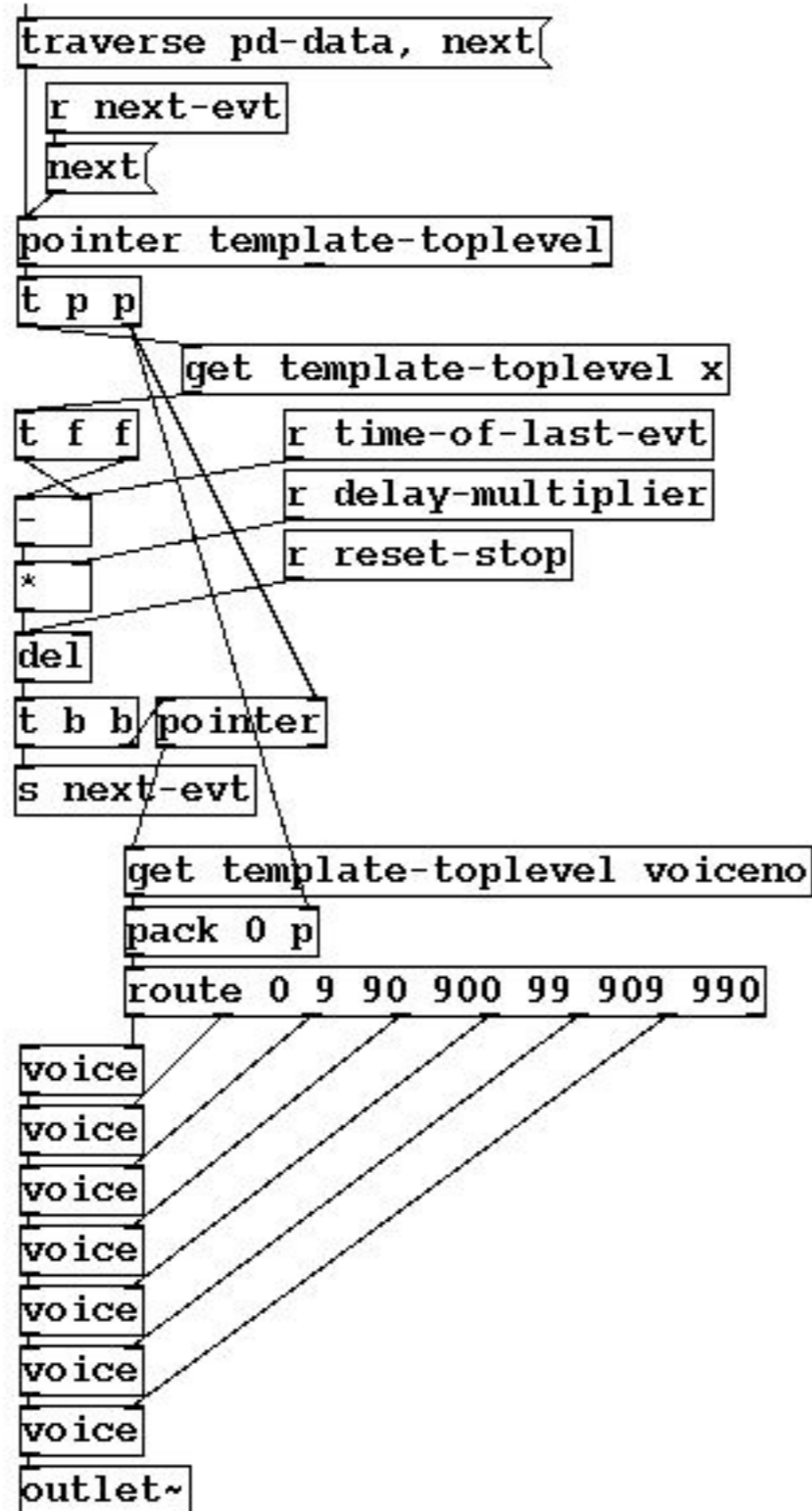
# Visual programming languages

Pure Data music programming



# Visual programming languages

Pure Data music programming



# Visual programming languages

Microsoft VPL

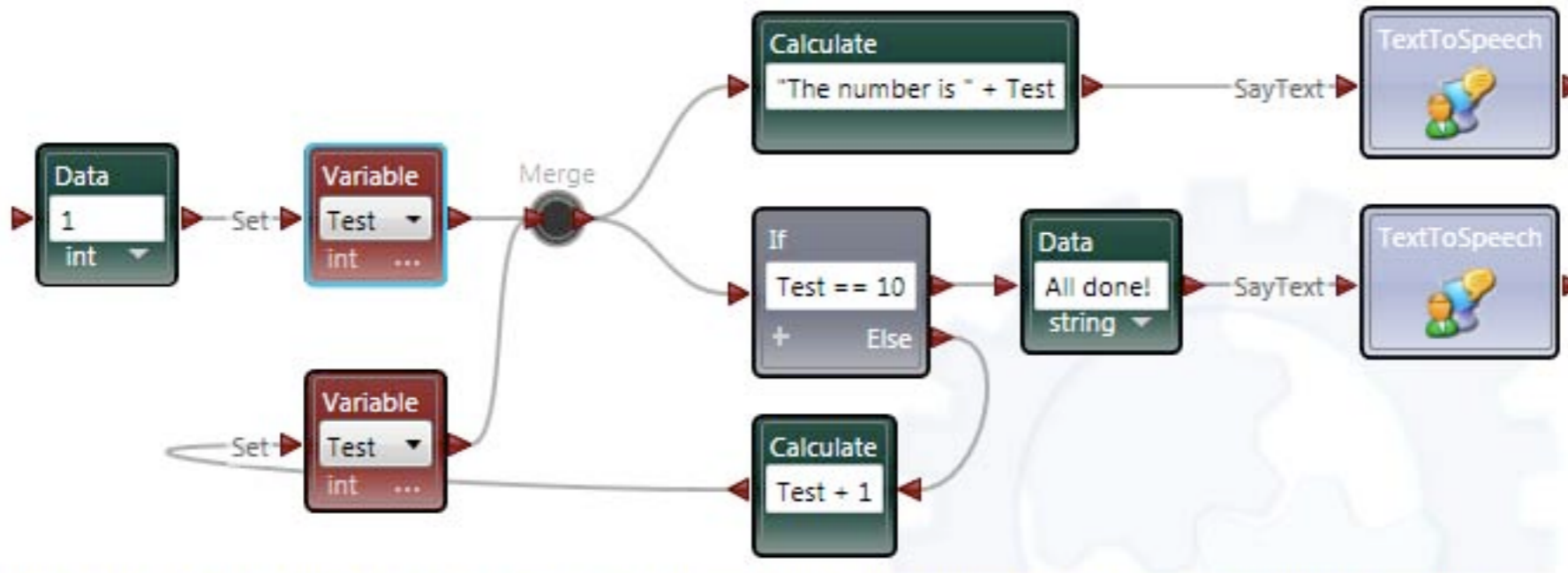
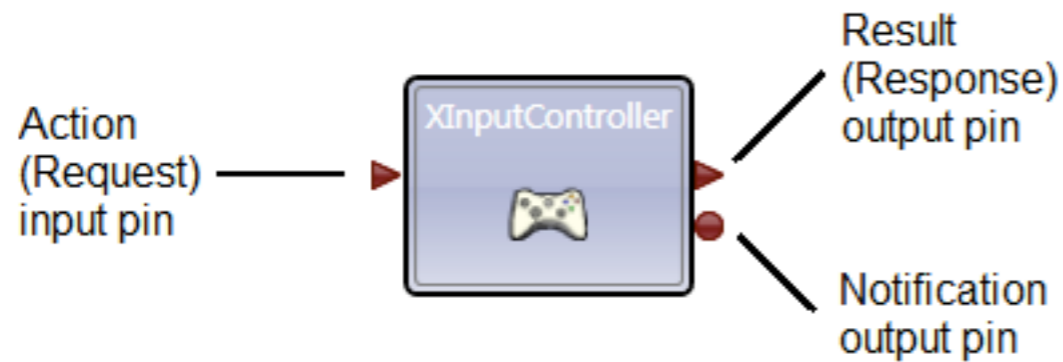


Figure 2 - Activity blocks have connections that represent messages sent from one activity to another





# Visual programming languages

## Microsoft VPL

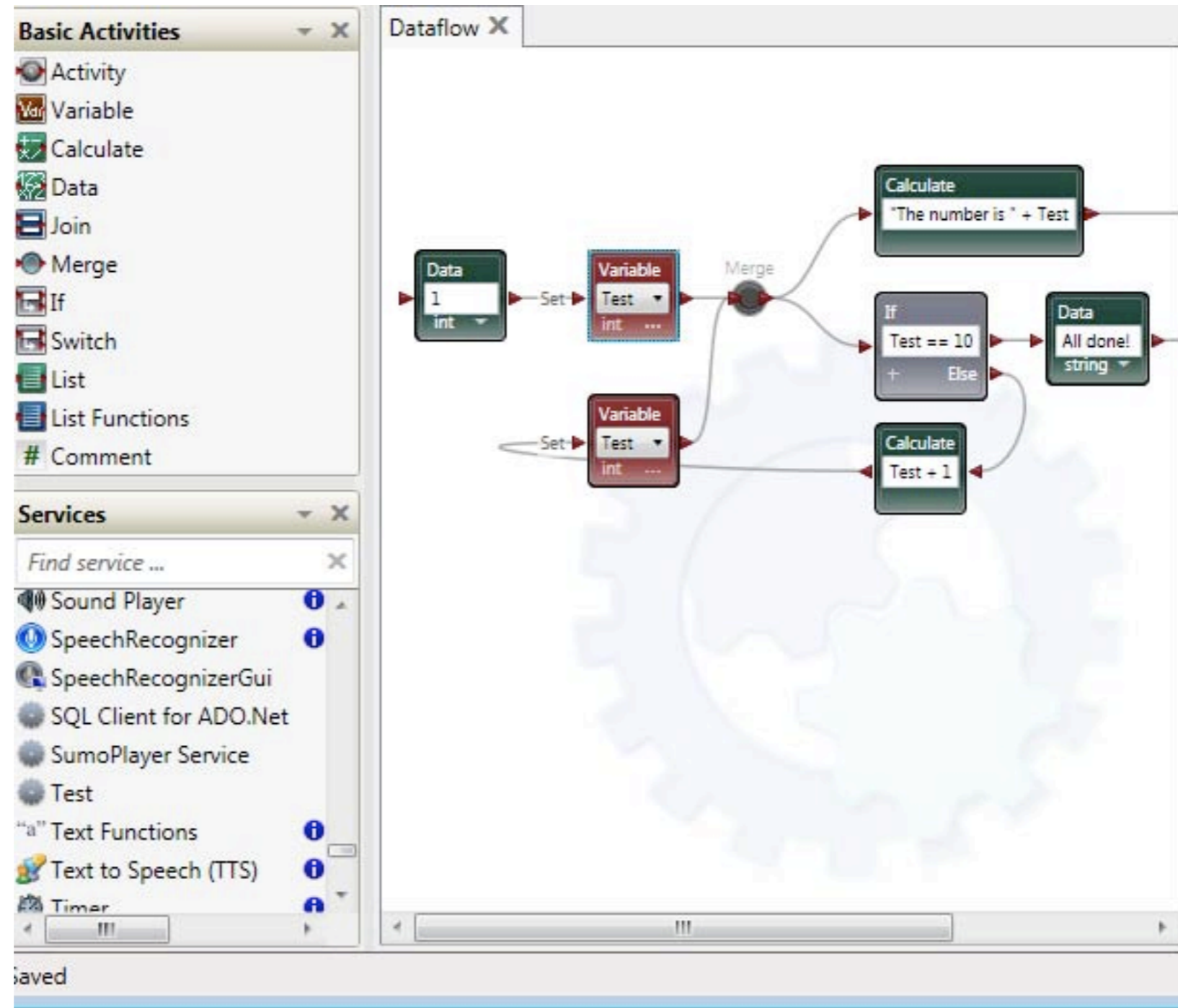


Figure 1 - Sample Microsoft VPL diagram

# Visual programming languages

For example: RobotProg



<http://www.physicsbox.com/supportrobotprogen.html>

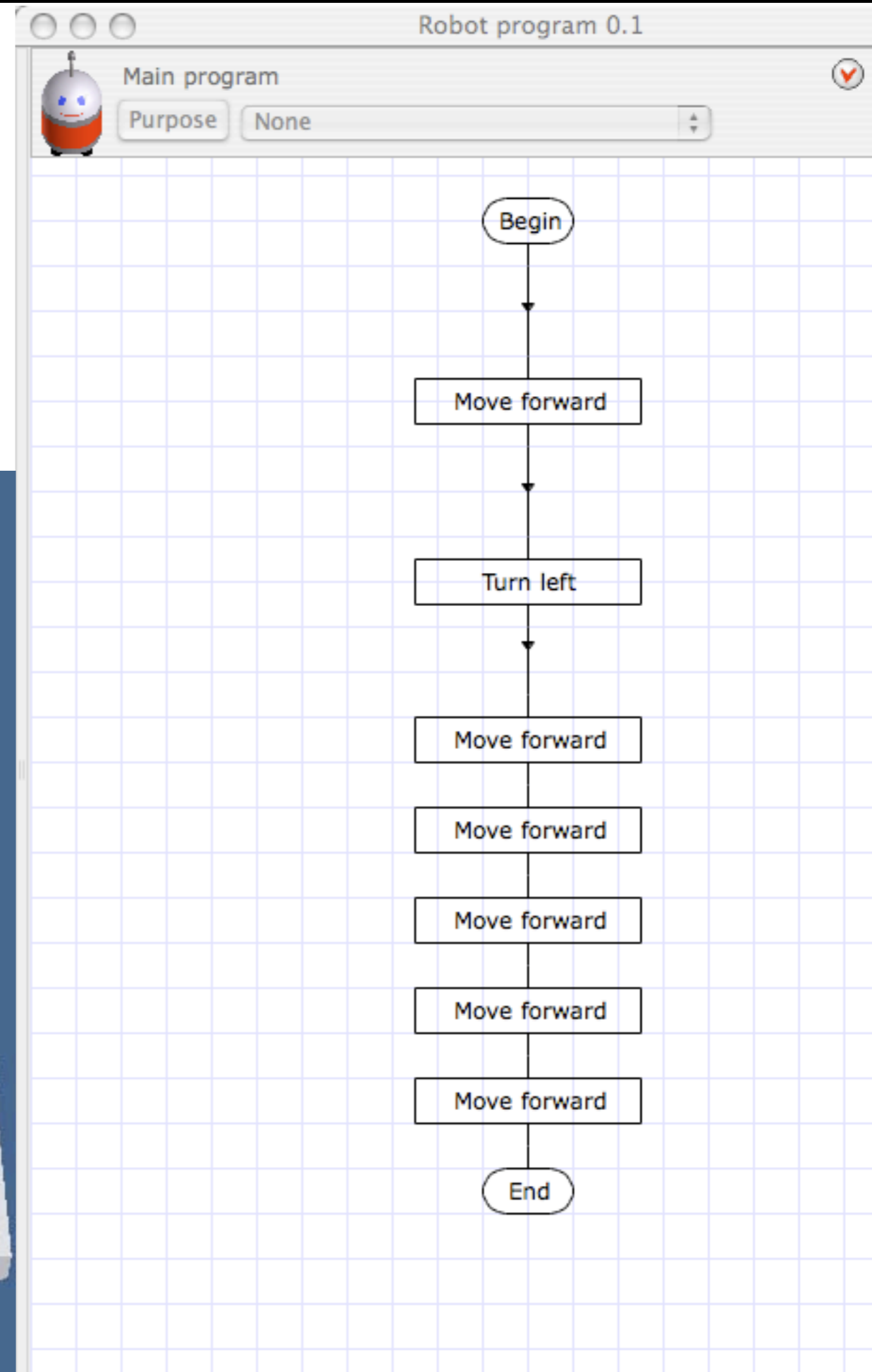
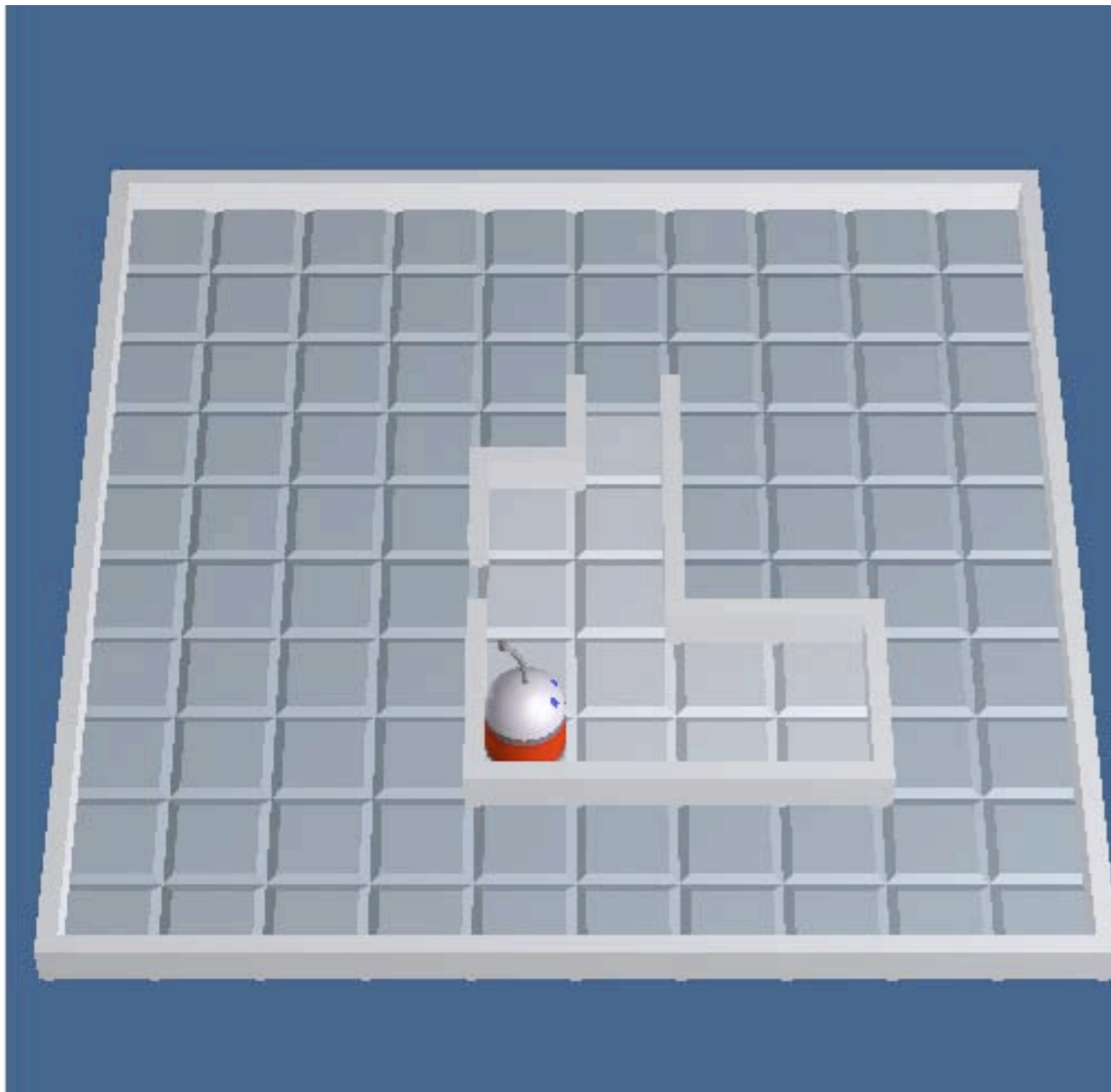
# Visual programming languages

The image displays a visual programming environment for a robot, divided into three main sections:

- Tools (Level 1):** A sidebar on the left containing various programming blocks: a mouse cursor, a red/blue pill, a downward arrow, a comment symbol (//c), a block labeled 'B', a block labeled 'E', a block with a question mark, and three blocks labeled 'TL', 'M', and 'TR'.
- Robot program 1:** The central workspace showing a flowchart for a 'Main program'. The purpose is set to 'None'. The flowchart starts with a 'Begin' terminal, followed by a 'Move forward' block. It then enters a loop: a 'WallAhead' decision block. If 'Y' (Yes), it goes to a 'Turn left' block, then another 'WallAhead' decision. If 'Y' here, it loops back to the first 'Turn left'. If 'N' (No), it goes to an 'Out' decision block. If 'Y' here, it goes to a 'Turn left' block, then a 'Move forward' block, then a 'WallAhead' decision. If 'Y' here, it loops back to the 'Out' block. If 'N' here, it loops back to the first 'WallAhead' block. The flowchart ends with an 'End' terminal.
- Execution:** A window on the right showing a 3D simulation of a robot on a grid. The robot is a small red and white figure. Above the grid are control buttons: 'INIT', a robot icon, a red 'STOP' sign, a blue 'Next' arrow, a blue 'Pause' icon, a blue 'Play' arrow, a help icon, and a 'Robot speed' slider.

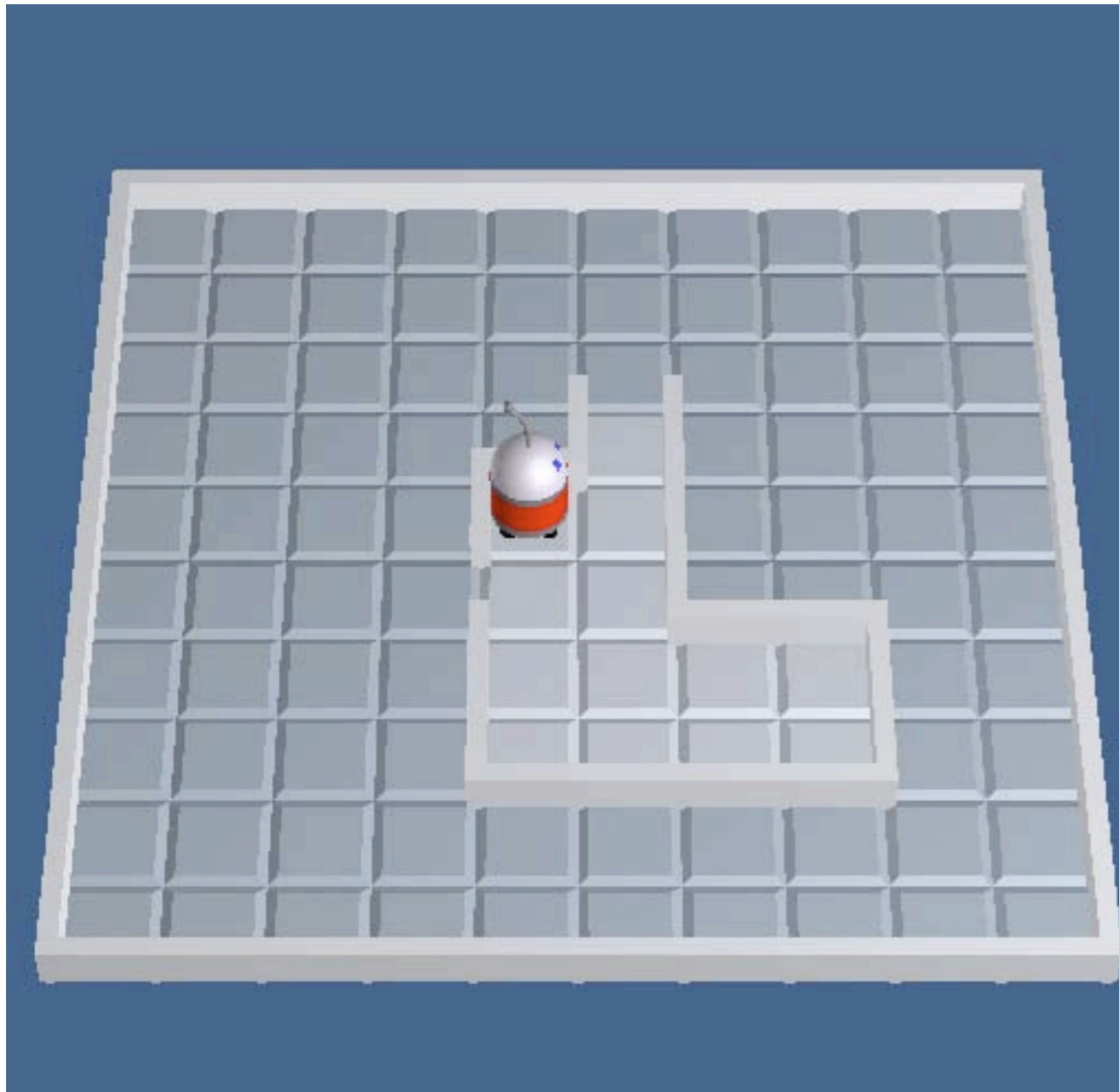
# RobotProg

RobotProg: a simple program for one particular starting point and situation



# RobotProg

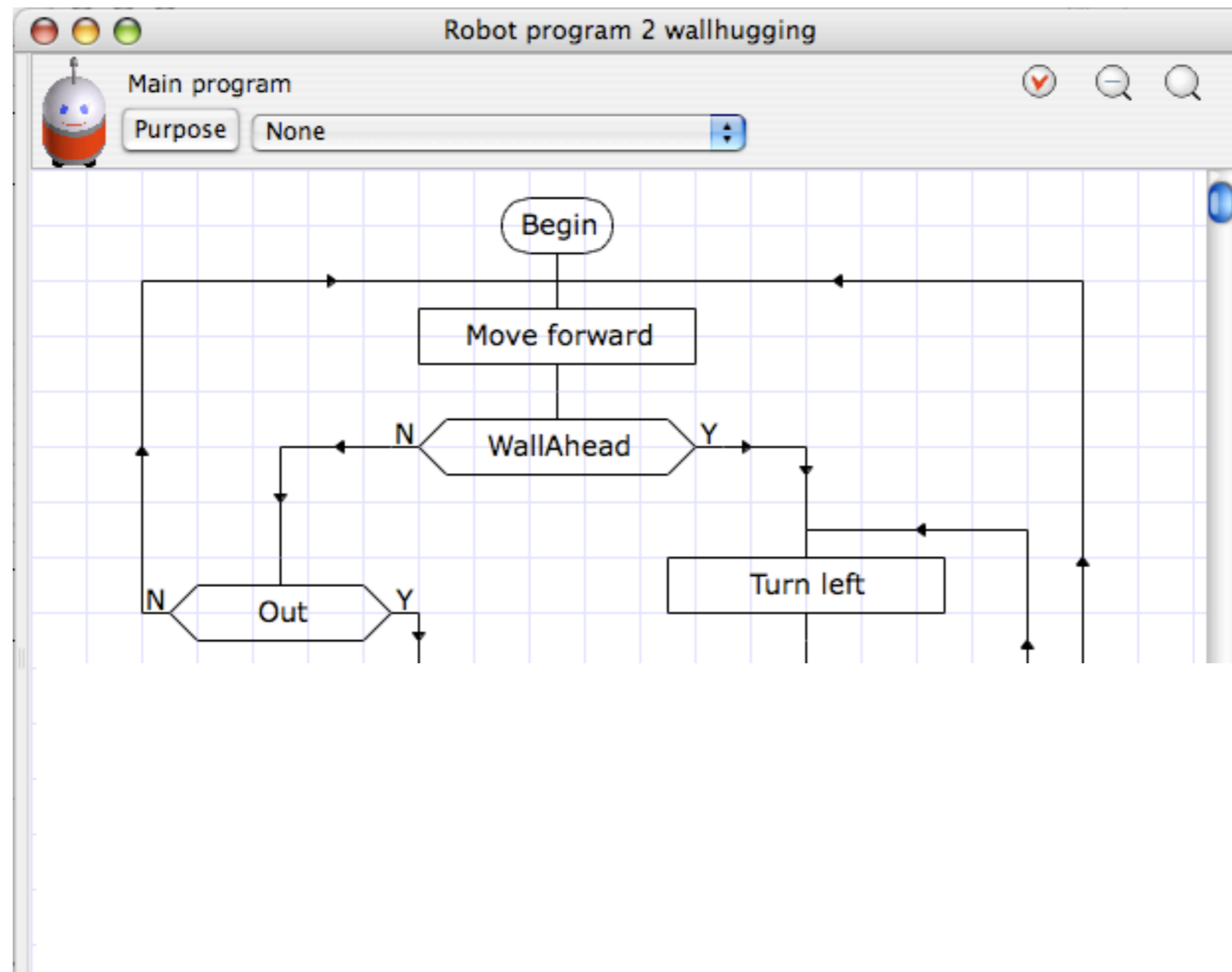
But what if the robot starts from another position? Can we make a program that works for every situation?



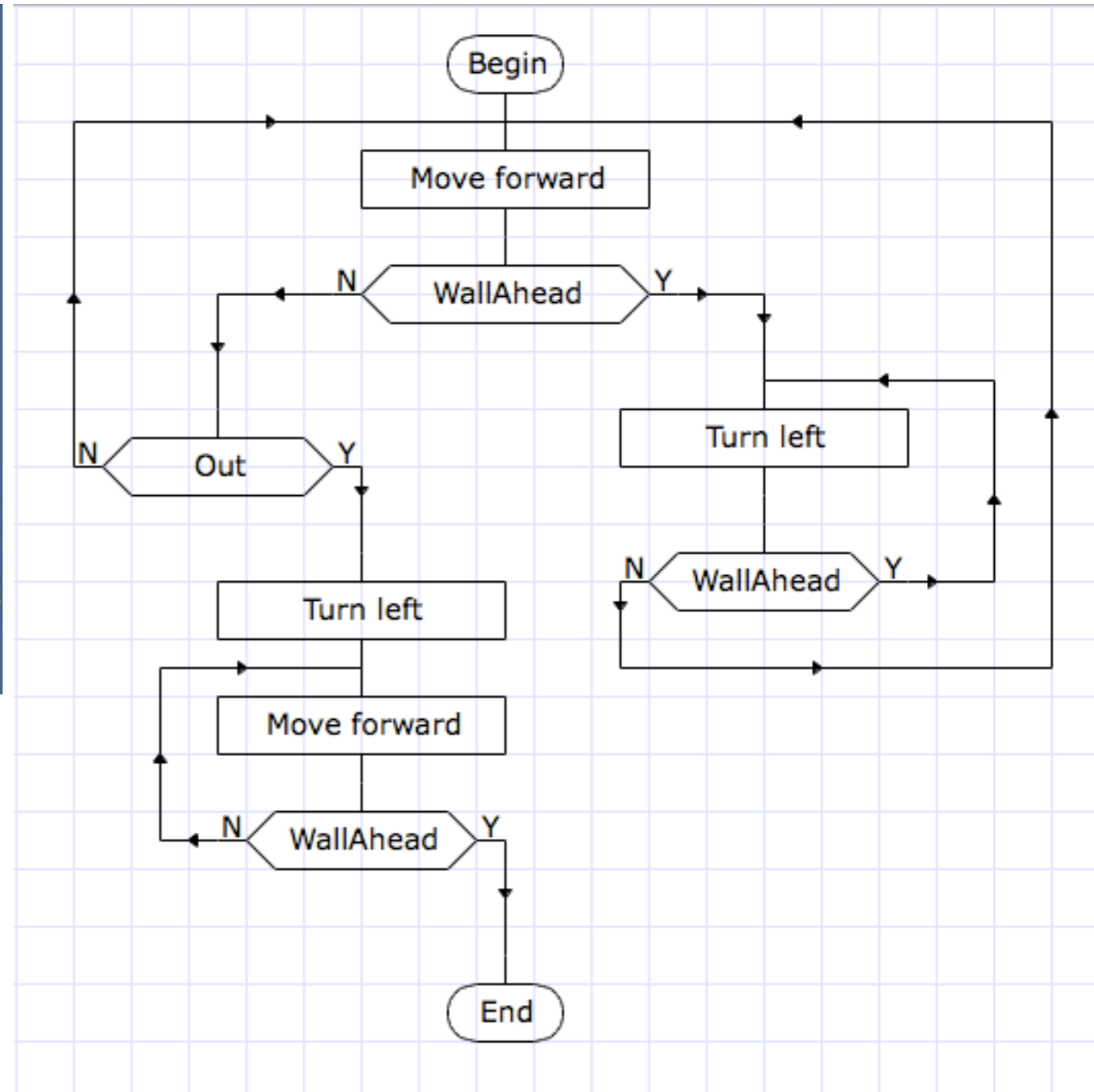
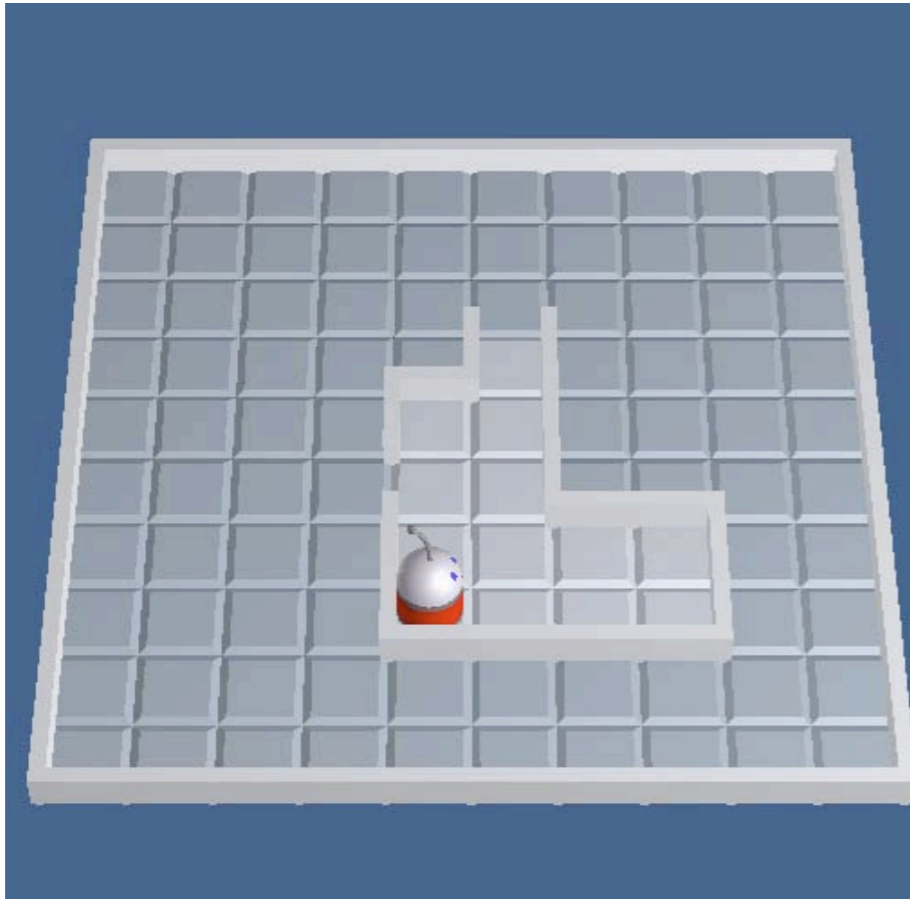
# RobotProg

We need some conditional statements: is there a wall ahead? yes/no.

If yes, do this, if no, do something else

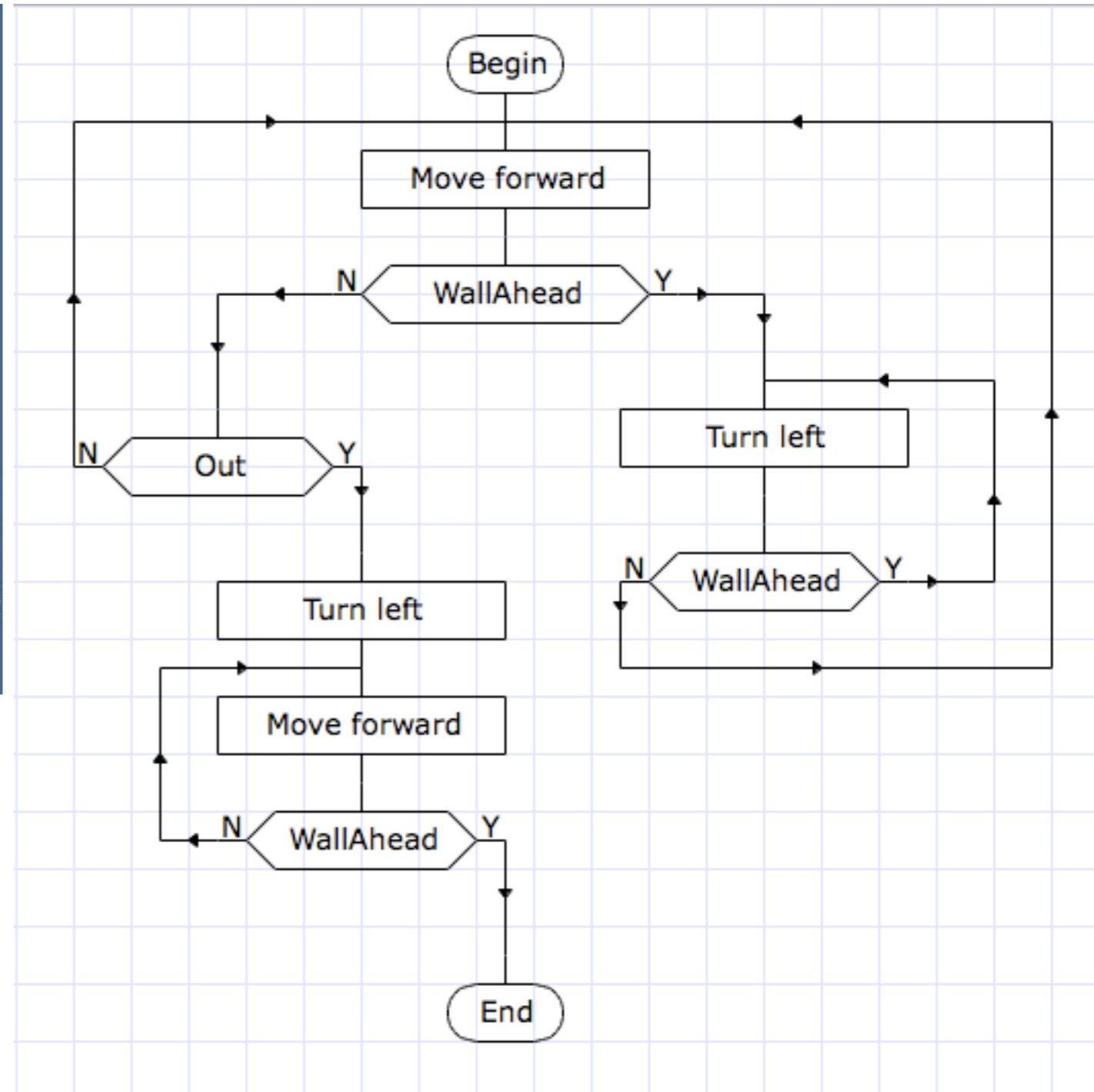
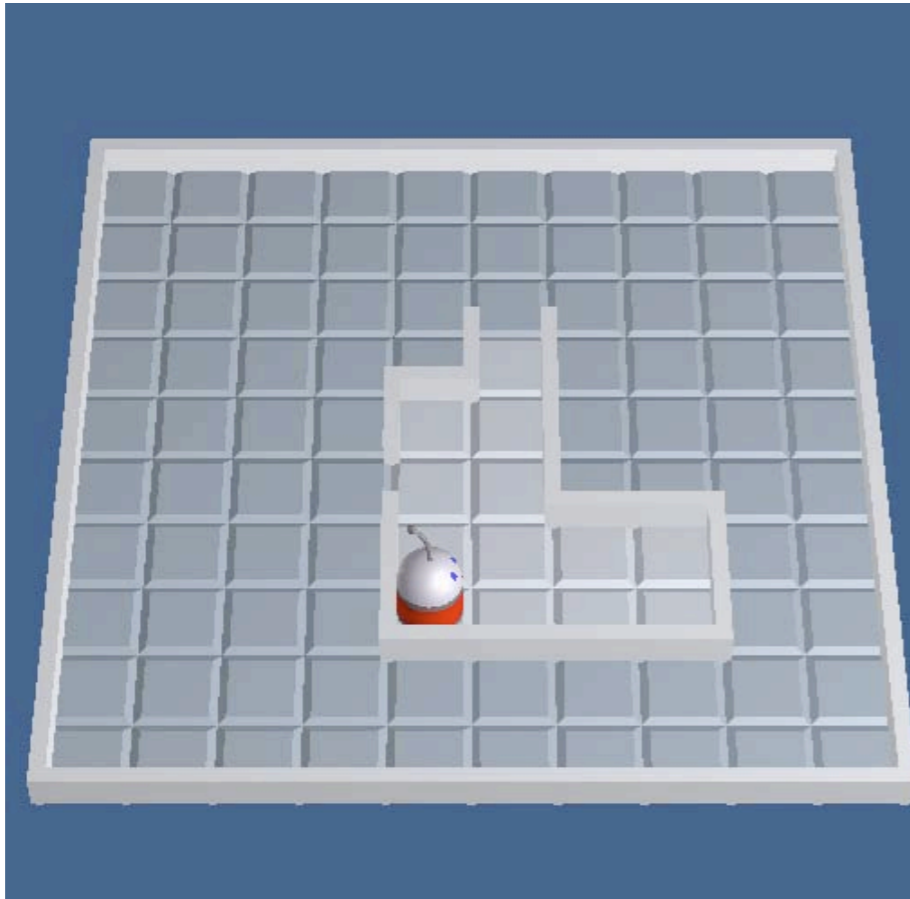


# Exercise: PROGRAMME A ROBOT'S ESCAPE



Does this program allow the robot to escape?

# Exercise: PROGRAM A ROBOT'S ESCAPE



<http://www.physicsbox.com/indexrobotprogen.html>



# ROBOTPROG