PiMaze: Teaching Programming through Tangible Interfaces

Danny Barthaud, Amel Bennaceur, and Vikram Mehta

SEAD - http://sead.open.ac.uk
Motivation

- OpenSTEM Challenge 2018

- Improve programming and problem solving skills using a fun exercise that includes a tangible interface.

Objectives

- Basic programming constructs, especially conditions and loops
- Backtracking and related data structures
- Incrementally build a program by considering different scenarios
Main Steps of the experiments

This experiment can be associated with Block 1 – Part 2 of TM112 entitled “Problem solving with Python”.

- 5 minutes – familiarise yourself with the interface and read instructions
- 5 minutes – use the forward action and solve the first maze
- 5 minutes – use the turn action and solve the second maze
- 10 minutes – learn about backtracking and solve the third maze
- 5 minutes – reflect about a generic algorithm and test with all mazes
Programming a Maze - Step 1

Using the forward function

While (current_position != EXIT):
    current_position = move_forward()
Programming a Maze - Step 2

*Using the turn function*

```python
While (current_position != EXIT):
    current_position = move_forward()
    If (current_position == old_position):
        # there must be a wall in front
        current_position = turn_right()
```
Programming a Maze - Step 3

Backtracking

```python
markedChoice[] = false

While (current_position != EXIT):
    current_position = move_forward()

    If (current_position == old_position):
        # there must be a wall in front
        if (!markedChoice[current_position]):
            current_position = turn_right()
            markedChoice[current_position] = true

        elif:
            current_position = turn_left()
```
Comments

""
Single or multiple line comments
can be between quotation marks
""
# Single line comments can be written after a hash (#)
Variable assignment

# assign the value 5 to the variable x
x = 5
Print
Invoking a Function
ABC - Python
Let’s Code
Possible Extensions

- Optimisation and complexity
  - Including keys and finding the shortest path → M269
  - Moving walls to convey notions of software adaptation and resilience
- Different programming languages
  - Java Programming in M250
- Robot programming
  - Using iRobot Create (Programmable Vacuum Cleaner)
  - Highlights the connection between computing and engineering modules such as T212 (Electronics: sensing, logic and actuation)
Thank you

SEAD - http://sead1.open.ac.uk