**Tutorial**

*General idea:*

The main idea of the project “Last Hope” is to create entertaining teaching materials for those making first steps in programming. Our game is structured in a way that it asks user to control character (mars explorer) by selecting and ordering certain programming commands.

No matter how distinct programming languages are, they all are based on the same concepts:

1. Functions to perform a certain action;
2. Conditional statements to perform an action only if something is true;
3. Loops, which help to set up iterative processes;

User will be able to use each of these elements to help our mars explorer, named “Last Hope” (as it is the last hope for humanity to explore the red planet, in our game scenario) to reach his target – a place, from which it will be able to move on to more adventures.

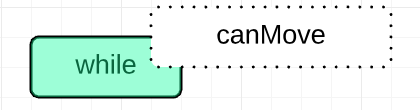
We use graphical interface to visualise programming concepts, that might sometimes be hard to understand for beginners. Our game directly shows which processes occur when a program is executed. Normally, these are hidden, and a user can only deal with input and output (I/O) of an application. We believe, that such background activities are especially interesting and have a large teaching potential for the novice programmers. Additionally, we try to illustrate the importance of these concepts, why programmers use them and the way efficiency of code can be improved.

As the user develops programming skills, we introduce new computing statements that will make every next level more complicated, interesting and enjoyable. We also make every level harder to develop a set of skills from making correct assumption to predicting character’s behaviour.

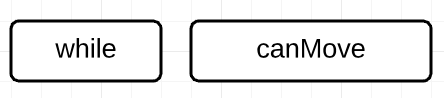
*GamePlay:*

Press “Play” button and proceed to the first level of the game. You will see the actual game on the right and “Drag&Drop” feature at the left. The “Drag&Drop” is composed of two rectangular-form areas: one is for dragging functions from, the other is to drop them into. In order to move the character, drag the necessary programming statement and drop on to “Start\_program” block. Every next block must be dropped on to either its predecessor, or before it (to change the order of the blocks).

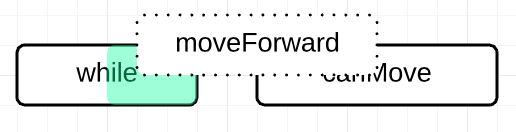
Certain blocks (which are “if”, "if else" statements and “while” loop) will require conditions, and steps that will be followed if a specified condition has been met. To add condition, move the corresponding block on to the loop or statement – the condition will be on the same line with it:



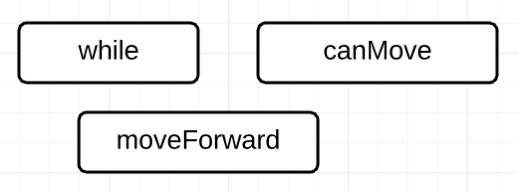
This is the way it should look:



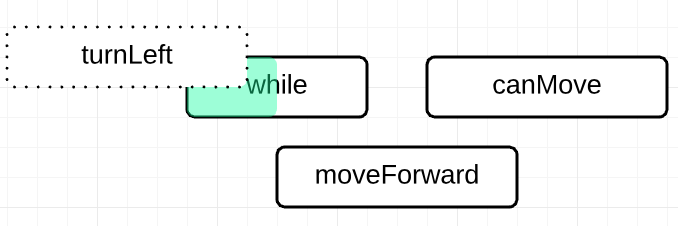
To add blocks within the loop, divide, for instance "while" loop, in two parts: left and right – the proportions will be 50% for each. If you place the next block to closer to the right part of the loop, it will be added inside:



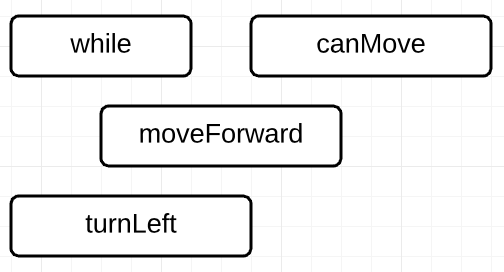
Inside the loop, which is shown by slightly moved to the right block:



Otherwise, the new block will be separate from the loop:



This has function “moveForward” inside the loop and “turnLeft” outside:



You can insert loops and statements inside other loops and statement following exactly the same procedure.

Functions:

* moveForward – a function that moves mars explorer forward by one cell in the direction it is rotated to
* turnRight or turnLeft – turns mars explorer right/left by 90 degrees relatively to the currently faced position

Loops:

* while – a loop that will repeat until the condition is not true; requires statement condition to be specified

Statements:

* if – a statement that when true executes the code inside, otherwise has no impact; runs once only
* else – additional part to construct “if else” statement; specifies default action when condition is not met

Conditions:

* canMove – checks if the character can move forward (true) or not (false)
* not\_canMove – a condition opposite to canMove; true for when cannot move forward, false for when can
* notReached\_target – will run “while” loop until the character finishes the level; if placed inside “if” statement, will be always true
* noObstacles\_around – checks if there are no obstacles around, within 1 cell; no obstacles (true), there is at least one (false)

*More Information:*

* check “Level Specification” word document for detailed description of each level in the game

This project has been created for the purposes of Coding Curriculum course at University College London, Engineering Department.