

# Blockr: Use Cases

Group 05

Jesse Geens, Oberon Swings,  
Aram Khachaturyan, Bert De Vleeschouwer

21 February 2020

## # Use Cases

### ## Use Case 1: Add Program Block

### Primary Actor  
User

### Stakeholders and interests  
The User wishes to expand his program by adding a new block to the Program Area

### Preconditions  
There are empty cells in the Program Area  
The block limit has not been reached

### Success guarantee  
No blocks are added if the block is not dragged in to the program area  
Otherwise the program has been expanded by one block and the block has re-appeared in the Palette

### Main Success Scenario  
1. The user moves the mouse cursor over a block in the Palette,  
then presses the left mouse key,  
then moves the mouse cursor to the Program Area,  
and then releases the left mouse key.  
2. The system adds a new block of the same type to the Program Area.

### Extensions  
1. The new program block connects with another program block, go to Use Case 2  
2. The user releases the mouse key when the mouse is not positioned in the Program Area  
2a. The system adds no new block to the Program Area  
2b. The dragged block returns to its original position in the Palette.  
3. The number of blocks in the Program Area has reached its maximum  
3a. All of the blocks in the Palette disappear  
3b. If the user drags a block back to the Palette, all blocks in the Palette reappear

## ## Use Case 2: Moving Program Blocks in the Program Area

### Primary actor  
User

### Stakeholders and interests  
User wants to play the game and doesn't want to run into issues.

### Preconditions  
At least one block is in the Program Area.  
There are positions in the Program Area which are empty (no wall no other blocks).

### Success guarantee  
The block doesn't break apart the program in the Program Area.  
The block is placed on a legal tile (correct connection, not on wall or other block).

### Main Success Scenario (connecting)

1. The user moves their cursor over a block.  
The user left clicks on the block,  
in the Program Area and holds the button down.  
The user moves the block into the Program Area.  
While moving the block one or more of the following things happen:
  - 1.1 The bottom socket of the block comes near a top plug of another block.
  - 1.2 The top plug of the block comes near a bottom socket of another block.
  - 1.3 The left plug of the block comes near a right socket of another block.
  - 1.4 The right socket of the block comes near a left plug socket of another block.
2. The user lets go of the left mouse button.  
When this happens the aforementioned plug(s) and socket(s) connect with each other  
The user loses "control" over the block it was moving  
(the mouse no longer holds down the block)  
This block gets placed in relation to the the other block  
so that the plugs are in the sockets  
The connected elements are now considered as a group of blocks

### Extensions

1. Just moving blocks inside of the Program Area
  - 1a. The user moves their cursor over a block
  - 1b. The user left clicks on the block in the Program Area and holds the button down
  - 1c. The user moves the block in the Program Area
  - 1d. The user lets go of the left mouse button
  - 1e. The block gets placed in the Program Area at the cursor's location
2. Moving blocks to an invalid location (outside of the Program Area, Palette Area or placed top of another block)
  - 2a. Follow 1a. to 1d.
  - 2b. The block will relocate to it's original place in the Program Area or Palette.
3. Disconnecting connected blocks
  - 3a. The user moves their cursor over a connected block
  - 3b. Follow 1b. to 1c.
  - 3c. The selected block will unconnect from the other block(s)
  - 3d. Follow 1e.
4. Deleting a block
  - 4a. Follow 1a. to 1b.
  - 4b. The user moves the block in the Palette Area
  - 4c. Follow 1d.
  - 4d. The block disappears into the Palette Area

## #Use Case 3: Running the program

### Primary actor  
User

### ### Stakeholders and interests

The User wishes to run the program successfully and lead the robot to the goal cell.

### ### Preconditions

There is only one group of blocks (can be a single block) in the Program Area,  
in this group only the top socket and bottom plug are not connected

### ### Success guarantee

The robot in the Game World has moved from its starting position  
to the location which results from following the instructions  
on the group of blocks in the Program Area

### ### Main Success Scenario

1. The user presses F5
2. It is not possible to add Program Blocks  
or move Program Blocks until the program execution has halted
  - a. If the robot in the Game World is at the Goal Cell  
the program execution halts successfully
  - b. Otherwise
    - \*. The next Program Block is selected (if it has not been selected yet before)
      - . If the program just started the next Block is the first block
      - . Else if the last block was the end of a set of blocks in a while block,  
the next block is this while block
      - . Otherwise the block that is connected below the last block gets selected
    - \*. This Program Block in the Program Area gets highlighted
    - \*. The robot in the Game World processes the current block
      - . If the current block is a while or if block, the conditions get evaluated
        - \_. If the condition is true, the next Program Block will be  
the first block inside of this if or while block
        - \_. If the condition is false, the next block will be  
the first block below the if or while block
      - . If the current block is a move block, the robot moves in the direction  
its arrow is pointing in the grid
      - . If the current block is a turn left or turn right block,  
the arrow showing the robot's current direction in the Game World turns 90  
left or right respectively.  
For future moves this is the direction in which the robot will move.
  - c. The program execution pauses until the user presses F5 again (go back to step 1.)

### ### Extensions

1. During executions, when the user presses Esc, the program execution halts.  
The robot in the Game World is reset to its original location  
and pressing F5 again starts the program again
2. If the robot in the game world is pointing towards a wall  
and the next block it needs to process is a move block, the program execution stops.  
The robot in the Game World is reset to its original location  
and pressing F5 again starts the program again
3. If the instruction blocks in the Program Area run out without the robot reaching the goal area  
the robot in the Game World is reset to its original location  
and pressing F5 again starts the program again