TAZI Usability Study with Implementation Modifications

Allison Reiss, Julia Jingxuan Wei, Erick Soto, Isaiah Aponte
Advisor: Steven Swanson

Problem
With its blockly interface, TAZI aims to make non-engineers easy to program robots, but it is still confusing for some users.

How to make TAZI more intuitive for both novice and experienced programmers?

Background
GADGETRON
The user drags and drops the desired parts onto the board to design their gadget. They can also drag corners of the board to adjust the board’s shape and size.

TAZI
TAZI is a block-based, visual programming environment that is tailored to each robot’s design.

- Users don’t need to have programming experience to understand how to drag and drop the desired blocks
- Users don’t need to worry about programming syntax errors such as type checking, loops and conditionals visual show what is within that block, and proper use of semicolons, brackets, etc.

User Study

Programming Tasks Given:

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>Draw a happy face on the LED display</td>
</tr>
<tr>
<td>Task 2</td>
<td>Draw neutral face, Drive forward at speed 100, Stop within 15 cm of wall, Draw sad face</td>
</tr>
<tr>
<td>Task 3</td>
<td>Move in square at speed 100, Draw a happy face when moving up or down, Draw a sad face when moving left or right, Move forward 500 times for each side of the square</td>
</tr>
</tbody>
</table>

Diagram of Task 3 → Robot after completing Task 1

Data Collection:
We collected data including:
- Time taken per task
- Number of trials,
- Where they look for specific blocks
- What confusions they had

We asked the participants to think aloud as they programmed each task.

Results
Some of the more common problems in TAZI encountered by our users were:

- When using the distance sensor users were confused about the units of measurement used. What units does distance sensor return?
- When using the motors and trying to turn, users could only specify how long to turn rather than the angle. How do you make a 90 degree turn?
- For some blocks, it looks like they should connect based on the matching notch shape, but when the user tried it wouldn’t allow it.
- Users were confused by what “change variable by” block means. Does “change variable by” block increment or decrement the variable?
- Where are the “else” and “else if” blocks to go with the “if” block? The cogwheel was not intuitive to use.
- Too small of a divider between “Setup” and “Loop forever and ever”, and sometimes users accidentally put blocks in the wrong section

Improvements

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>DistanceSensor #1 get distance</td>
<td>DistanceSensor #1 get distance (cm)</td>
</tr>
<tr>
<td>change x by 1</td>
<td>increment x by 1</td>
</tr>
</tbody>
</table>

To make it more visually clear which blocks can connect, we changed the shape of integer connectors to be long and round, while non-value is still represented by the original notch shape, as seen below.

Future Work
In addition to our type checking, we could implement Polymorphic blocks which would:

- Make connectors visually represent the type of data being passed in
- In comparative blocks, once you pass in the first argument the connector shape of the second argument will be updated to match

Acknowledgements
Many thanks to Christine Alvarado, Aditi Mavalankar and Michael Gonzalez for all of their guidance and support. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1339335.