Report on project at Artificial Intelligence Seminar
- team Marek & Alexandra -

1. Initial task

Initial vision consisted of:

- Shapes detection using simple geometry
- Shape counting
- Image/Object recognition (by certain characteristics)

And it was packed in the task of "reading" the Winnie the Pooh comic.

2. Progress development

A. Learning how to work with the ozobot and write in ozoblockly.

B. Testing what we would be able to achieve using the given functionalities of the ozobot.

C. Developing different playgrounds(maps) for the ozobot.
   - Black & white
   - Colorful
   - Only rectangles & squares
   - With other known geometrical shapes
   - Random shapes

D. Trying different implementations of functions which would be helpful in fulfilling the final task:

   - Determine all the encountered colors on the playground
     - Line by line(on length), ozobot scans the color playground by small steps (chosen step length: 5 mm x 50 times – in order to fit the dimensions of a A4 sheet)
     - At the end of a line, it rotates and goes further (on width) and then continues with the following line (chosen widthlength between lines: 40 mm)
     - if color is different from the last scanned color, it notices the event of changed color

   - Counting the number of lines
     - constraint: objects have black borders
     - each time a black color is found, the counter is increased

- Parsing a line
  - implementing our own "parseALine" function would give more precision in knowing the position of the ozobot and in knowing the dimensions of the objects
  - built-in function was not following the line precisely
  - we didn't manage to finish the function

- Parsing different geometrical shapes
  - a rectangle has 4 edges & 90 degrees between them -> ozobot has to rotate 4 times with 90 degrees and then it can stop and go to another object
  - an equilateral triangle -> the ozobot has to rotate 3 times with 60 degrees and follow the line
  - a circle -> the ozobot parses the interior of the circle as an rectangle -> it doesn't follow the line, but counts how many time it "hits" the line (4)
  - we didn't manage to finish the function

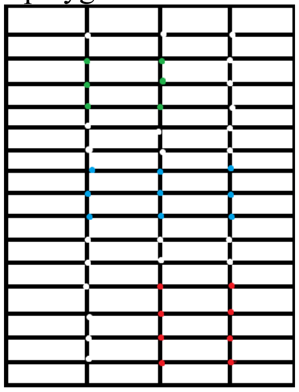
E. Using the functions to complete the task (description in "3.Final results")

3. <u>Final results</u>

Determining different ways of computing the number of objects on the playground:

A. using the information about the number of lines found on the playground:
   - the ozobots intersects an object twice when parsing the playground
   - in order for this method to work, exists the following constraint: the relation between the width between the lines (LW) and the width of an object(OW) has to be: LW < OW < 2*LW

B. using the information about the colors found on the playground:
   - first, there was a constraint that each object has a unique color
   - i.e. the number of different colors determines the number of the objects found
   - then, I partitioned the playground in the following way:



- on width, with 40 mm distance between the lines
- on length, with 25 mm
- the dimension can be changed in order to be more precise (depending on the size of the objects)
- idea: at each intersection, the encountered color is saved in an array
- initial step: the arrays are initialized with the white color
- parse line by line and saving the encountered colors
- after parsing each 2 lines, it compares the neighbor colors:
    i. if both are white or of the same color, nothing changes
    ii. if first line has a color != white and different from the corresponding color from the next line, we increase the counter
    iii. at the end of each comparison, for the first line we compute the number of neighbors with the same color and we decrease it form the counter

- when at the last line, it just adds to the counter the number of different colors found on the line

4. <u>Distribution of work between the team members</u>

Marek Černý: - presentation about the vision
- presentation about the working progress

Alexandra Maior: - implementation of different ways to compute the number of objects on a „playground” (including functions used to solve this task)
- final results presentation
- final report