1. version: draw a black road that is twice broad as the robot, by using most of the white play ground; let the robot follow the road

2. version: place 2 rectangular obstacles at random positions on the road; after touching/avoiding an obstacle, let the robot turn 90° randomly left/right and move straight, until a road/line found again; avoid touching an obstacle a 2. time; due Feb 28

obstacle: size is 30 pixels x 30 pixels; avoid collisions by calculating a collision area relative to the co-ordinates of the first collision (x,y) with x±30 and y±30

collision area: sample calculation

collision area: ((x-30,y-30),(x+30,y+30))

additional rules: the robot should never stop; if domain boundary is hit, turn 180°

knowledge representation: represent your solution for the whole problem above in form of a set of primitive rules, like IF light=black AND touch=no THEN gearForward=1second

documentation: use LDD for mechatronics; use UML2 for software
3. version: add a black line in shape of an 8 inside the circle; let the robot follow the line; test the robot starting i) initially on the road and ii) initially on the line

4. version: place 2 rectangular obstacles at random positions on the line; after touching/avoiding an obstacle, let the robot turn 90° randomly left/right and move straight, until a road/line found again; avoid touching an obstacle a 2. time; due Mar 01

obstacle: size is 30 pixels x 30 pixels; avoid collisions by calculating the full shape of the obstacle relative to the co-ordinates of the first collision (x,y) with x±30 and y±30

additional rules: the robot should never stop; if domain boundary is hit, turn 180°

knowledge representation: represent your solution for the whole problem above in form of a set of primitive rules, like IF light=black AND touch=no THEN gearForward=1second

documentation: use LDD for mechatronics; use UML2 for software