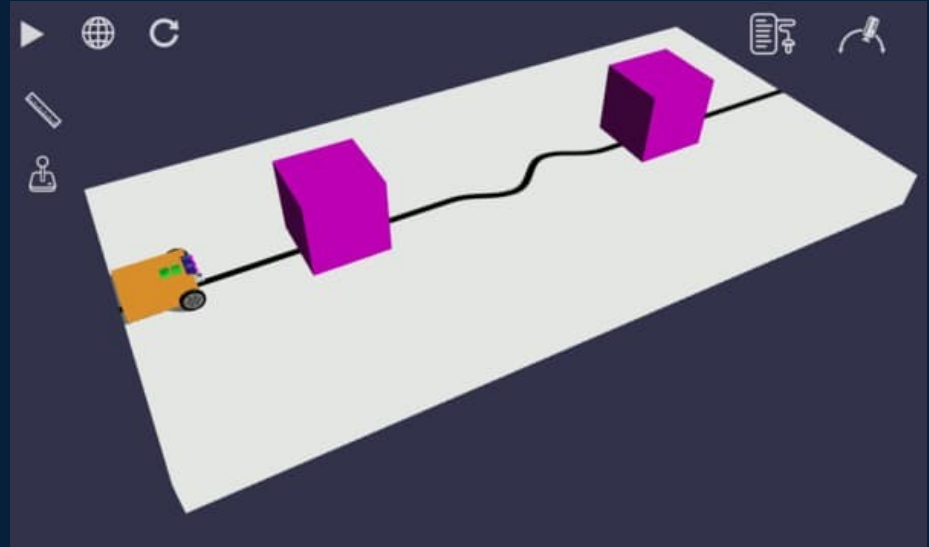


# Obstacle Avoidance

A POSTERIORI

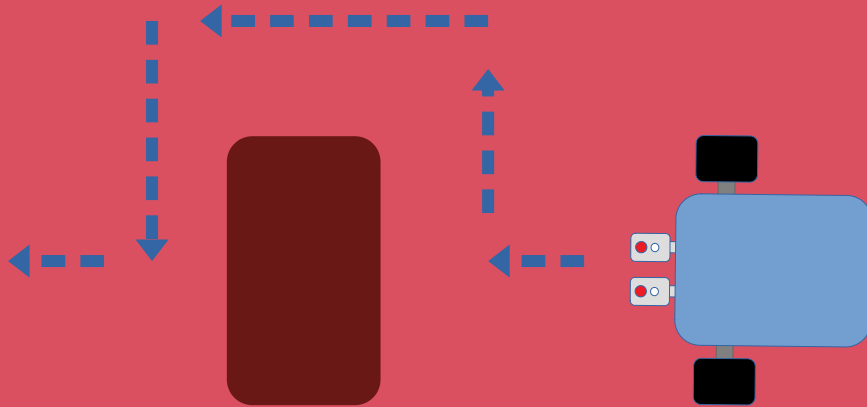
Play · Experience · Learn



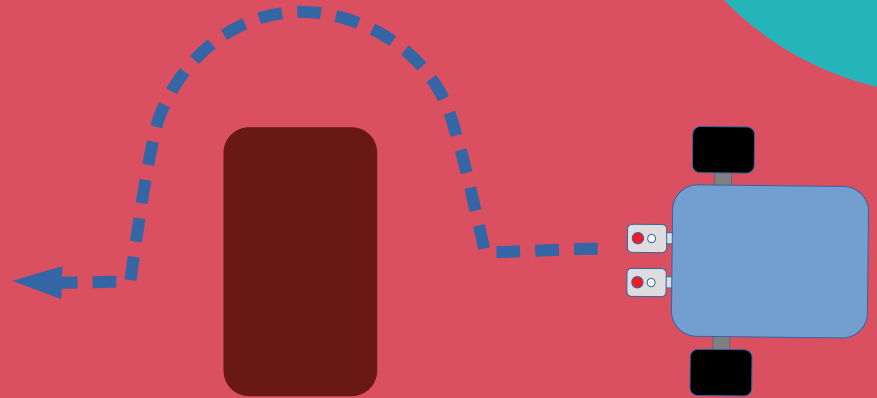
# Before We Start

- Robot
  - Use either the Single or Double Sensor Line Follower robot
- World
  - Use “Line Following Challenges” world
  - Start with the “Obstacles 1” challenge

# Navigating Around Obstacles



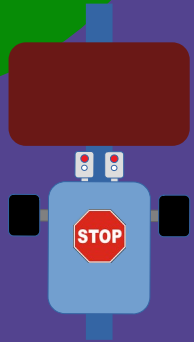
Straight Path



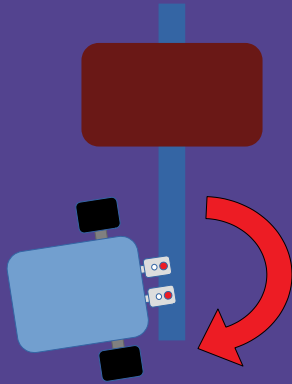
Arc Path

Both methods are workable.  
Both methods can be improved with the use of side facing sensors.

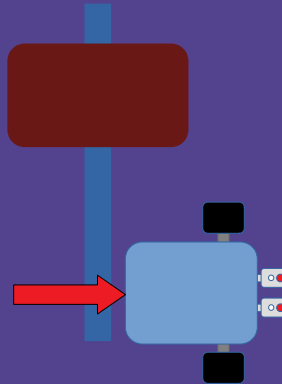
# Naive Approach



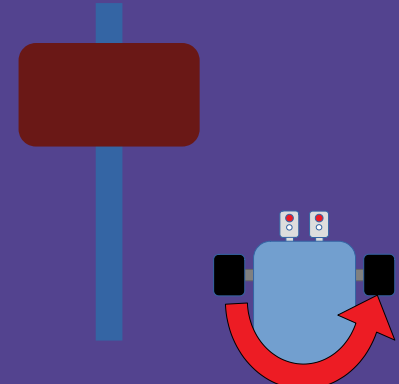
1) Stop



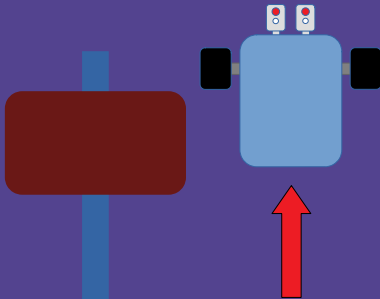
2) Turn



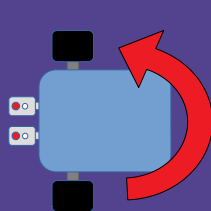
3) Straight



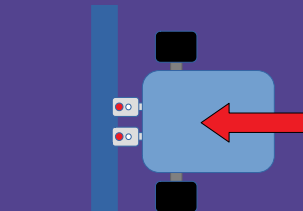
4) Turn



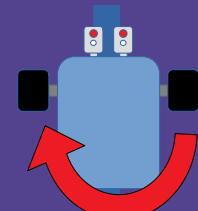
5) Straight



6) Turn



7) Straight



8) Turn

# Naive Approach

Repeat until less than 3cm from obstacle

Proportional line follower

The diagram shows a Scratch-style block structure. It starts with a 'When Started' block, followed by a 'repeat until' loop. The loop condition is 'ultrasonic distance on port Auto in cm' with a value of 3. Inside the loop is a 'do' block containing 'move steering with direction' (set to 1) and 'color\_sensor reflected light intensity on port 1' (set to 50), followed by 'and speed' (set to 20%). The loop ends with a 'stop moving and hold' block.

```
while not ultrasonic_sensor_in2.distance_centimeters < 3:  
    steering_drive.on(1 * (color_sensor_in1.reflected_light_intensity - 50), 20)  
    tank_drive.off(brake=True)
```

Need to tell the robot to stop, or it'll continue moving even after the program ends

1) Our first step is to line follow until we are close to the obstacle, then stop

# Naive Approach

The image shows a Scratch script for a robot's movement. It starts with a 'When Started' block, followed by a 'repeat until' loop. The loop condition is 'ultrasonic distance on port Auto in cm' less than 3. Inside the loop, there is a 'do' block containing a 'move steering with direction' block with a direction of 1 and a speed of 20%. This is followed by a 'color\_sensor reflected light intensity on port 1' block with a value of 50. Below the loop, there are seven 'move steering with direction' blocks, each with a different direction and speed, and a 'for' block with a specific number of rotations. A red bracket groups these seven blocks, and a black arrow points from the text below to the bracket.

```
When Started
repeat until [ultrasonic distance on port Auto in cm < 3]
do
  move steering with direction 1 and speed 20%
  color_sensor reflected light intensity on port 1 50
  move steering with direction -50 and speed -20% for 1.4 rotations
  move steering with direction 0 and speed 20% for 1.2 rotations
  move steering with direction -50 and speed 20% for 1.4 rotations
  move steering with direction 0 and speed 20% for 1.9 rotations
  move steering with direction -50 and speed 20% for 1.4 rotations
  move steering with direction 0 and speed 20% for 0.35 rotations
  move steering with direction 50 and speed 20% for 1.4 rotations
```

2) Navigate around the obstacle.  
These are basically step 2 to 8 in the naive approach.

# Naive Approach

The code is written in Scratch and starts with a 'When Started' event. It contains two identical 'repeat until' loops, each with a 'do' block. The first 'do' block is a 'move steering with direction' block. The 'direction' field is set to '1' with a multiplier 'x'. The 'color sensor' block is set to 'reflected light intensity' on 'port 1', with a value of '50'. The 'speed' field is set to '20 %'. The 'repeat until' block is set to 'ultrasonic distance on port Auto' in 'cm', with a value of '3'. Following the first 'do' block are seven 'move steering with direction' blocks with the following parameters: direction: -50, speed: -20 %, for: 1.4 rotations; direction: 0, speed: 20 %, for: 1.2 rotations; direction: -50, speed: 20 %, for: 1.4 rotations; direction: 0, speed: 20 %, for: 1.9 rotations; direction: -50, speed: 20 %, for: 1.4 rotations; direction: 0, speed: 20 %, for: 0.35 rotations; direction: 50, speed: 20 %, for: 1.4 rotations. The second 'do' block is identical to the first one. An arrow points to the '50' value in the 'color sensor' block of the second 'do' block.

3) Resume line following

# Naive Approach

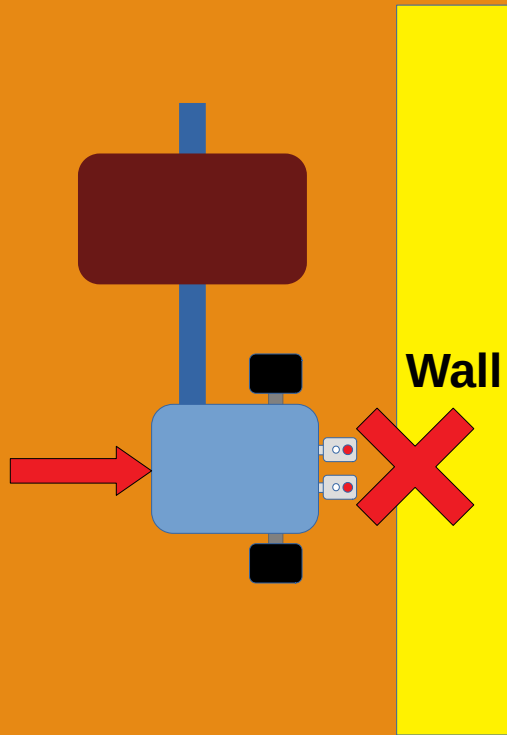
- It's “naive”, because it assumes...
  - Robot can turn and move accurately
  - Path on the right is unobstructed
  - No gaps in the floor
  - Obstacle is a fixed size
  - The line continues on the opposite side of the obstacle



# Inaccurate Turns and Moves

- Robots won't turn and move accurately, due to...
  - Uneven ground
  - Tires slipping
  - Motors inaccuracies
- Solve this by...
  - Using the gyro to control your turns
  - Using a gyro follower code to control your moves
  - Turning slower
  - Reducing acceleration (ie. Change speed gradually)

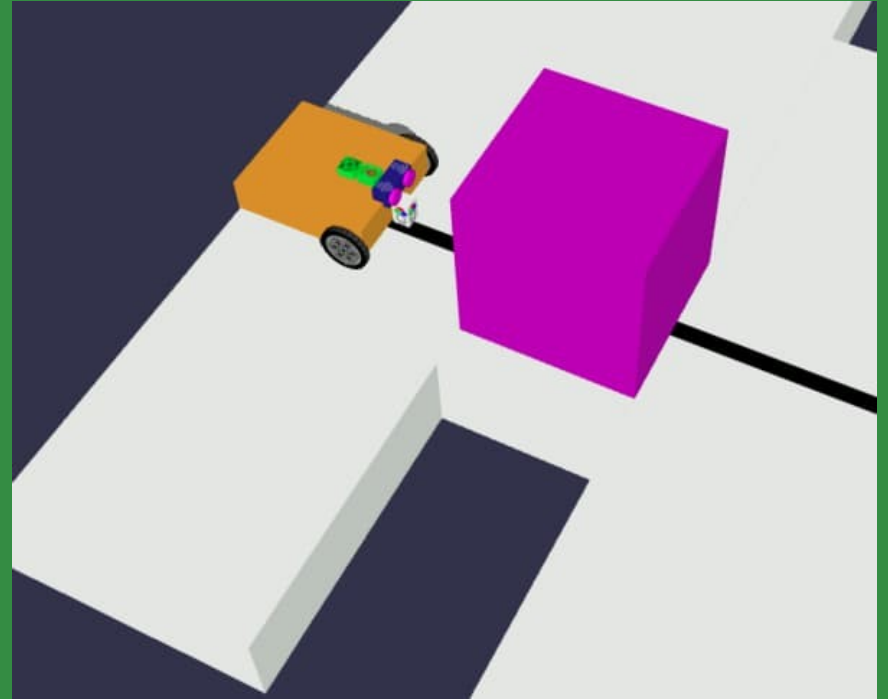
# Obstructions on Path



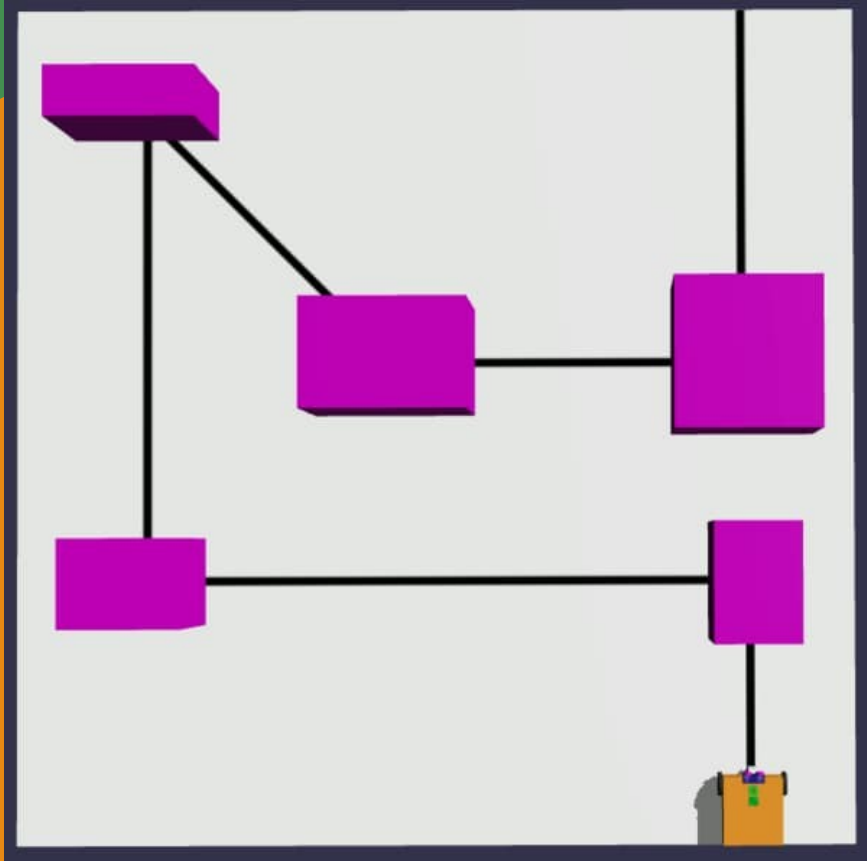
- One side of the obstacle may be blocked
- Solve by...
  - Detect obstructions with ultrasonic
  - Backtrack to starting position if detected
  - Navigate around using the other path (ie. Going left instead of right)

# Gaps in Floor

- There may be a gap in the floor on one side
- Solve by...
  - Detect floor gaps using color sensor or downward facing ultrasonic
  - Backtrack to starting position if detected
  - Navigate around using the other path (ie. Going left instead of right)



# Random Size Obstacles



- Obstacle may not be a fixed size
- Solve by...
  - Use a sideways facing ultrasonic to detect obstacle; this will let you know when it's safe to turn
  - You can also try using a wall following algorithm to navigate around the obstacle



# Challenges

- Program your robot to complete these challenges
  - Obstacles 1
  - Obstacles 2
  - Obstacles 3
  - Obstacles 4



# A POSTERIORI

Play · Experience · Learn

- Created by A Posteriori LLP
- Visit <http://aposteriori.com.sg/> for more tips and tutorials
- This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.