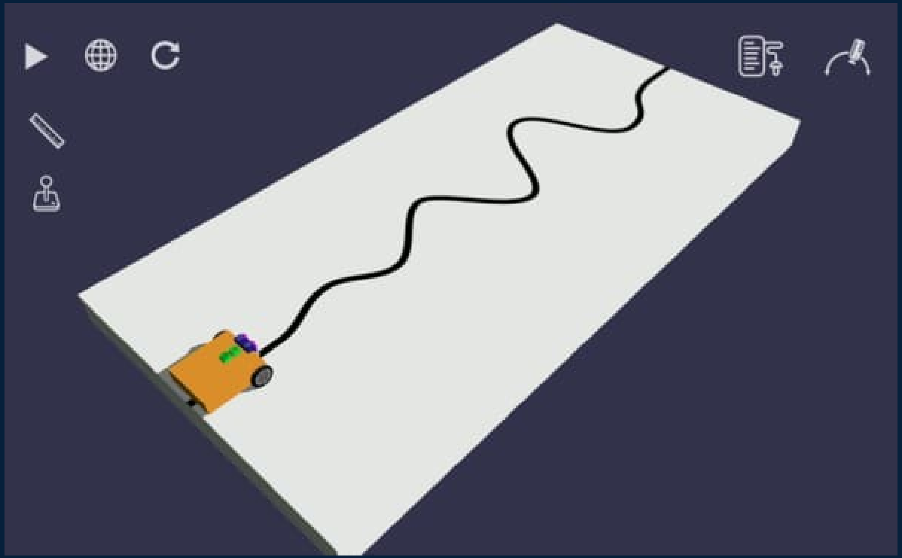
Single Sensor Line Follower

A POSTERIORI

Play · Experience · Learn



PLAY · EXPERIENCE · LEARN

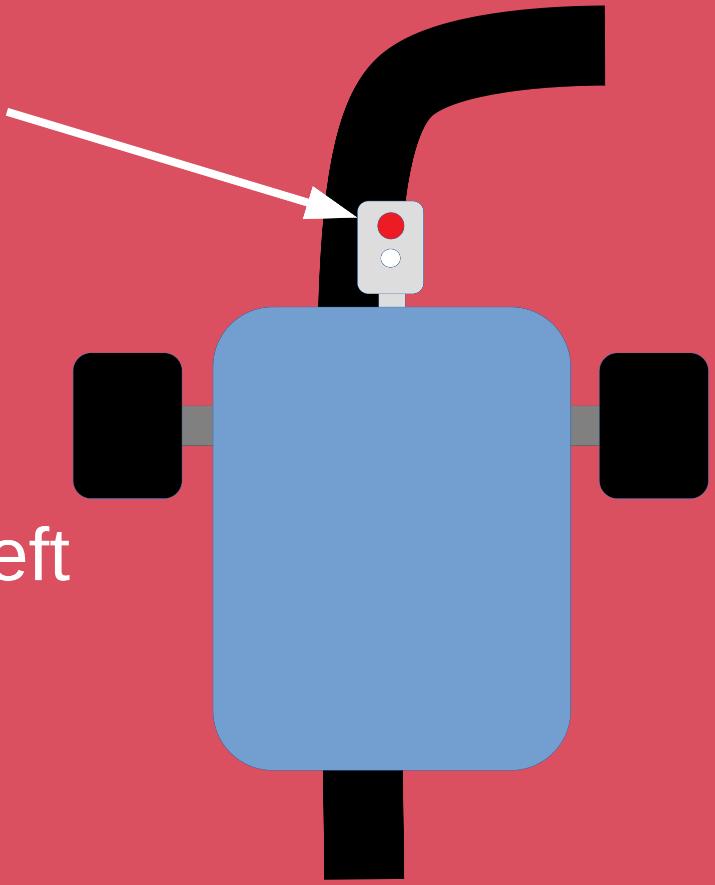
A POSTERIORI

Before We Start

- Robot
  - Use the “Single Sensor Line Follower” robot
- World
  - Use “Line Following Challenges” world
  - Select “Simple Curves” challenge

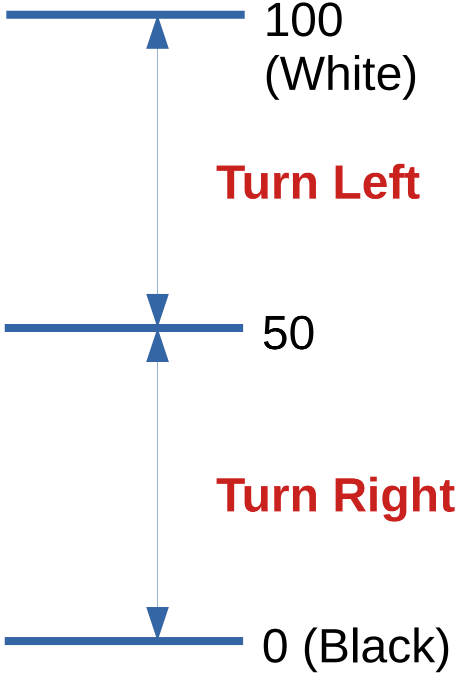
One Sensor Line Following

- Sensor on **edge** of line
- If sensor is reading…
  - White: Robot is too far right and needs to turn left
  - Black: Robot is too far left and needs to turn right



2 States Algorithm

Light Sensor Value

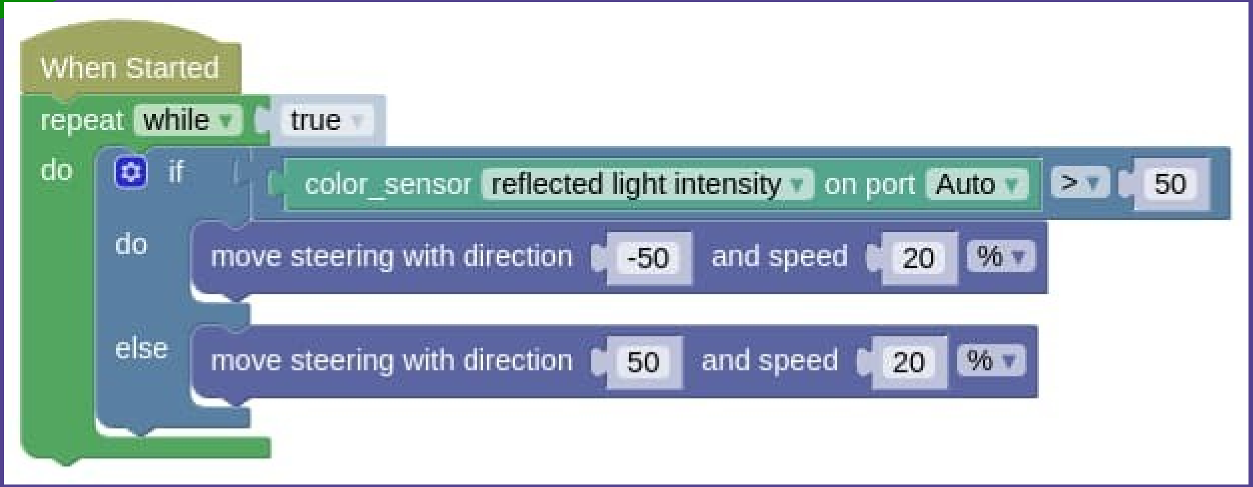


Pseudo Code

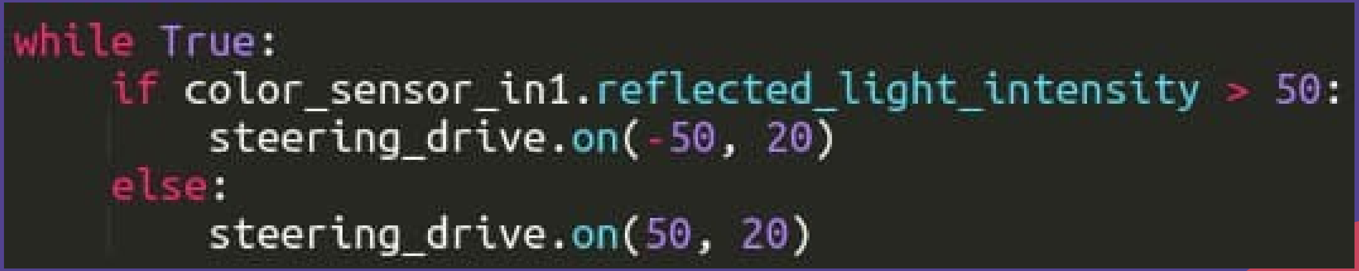
```
IF Value > 50:
  Turn Left
ELSE:
  Turn Right
```

- Loops forever
- Checks reflected light
  - White (>50): Turn Left
  - Black (<50): Turn Right
- Robot “wiggles” left and right

2 States Algorithm



```
When Started
repeat while ▾ true ▾
do
    if  color_sensor reflected light intensity ▾ on port Auto ▾  > ▾  50
    do   move steering with direction -50 and speed 20 % ▾
    else move steering with direction 50 and speed 20 % ▾
```



```
while True:
    if color_sensor_in1.reflected_light_intensity > 50:
        steering_drive.on(-50, 20)
    else:
        steering_drive.on(50, 20)
```

Initialization of motors and sensor not shown here.

A POSTERIORI — PLAY · EXPERIENCE · LEARN

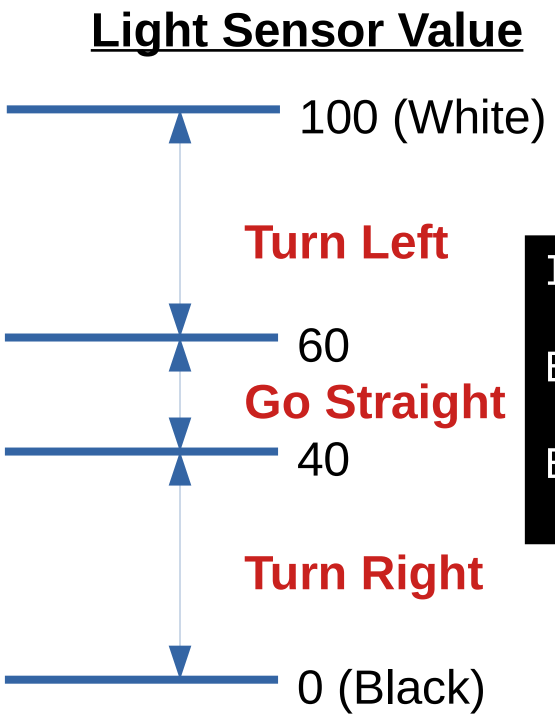
Common Problems

- Problem:
  - Movement is slow and jerky
- Why?:
  - Robot ONLY move left and right. It never goes straight.

- Problem:
  - Robot can’t handle very sharp turns
- Why?:
  - Robot only turns at “50” and “-50” (...max is 100)
  - What happens if we increase the turn to 100?

3 States Algorithm

- Check for Black, White, and Grey
  - White (>60): Turn Left
  - Grey (Between 40 to 60): Go Straight
  - Black (<40): Turn Right
- Robot runs smoother

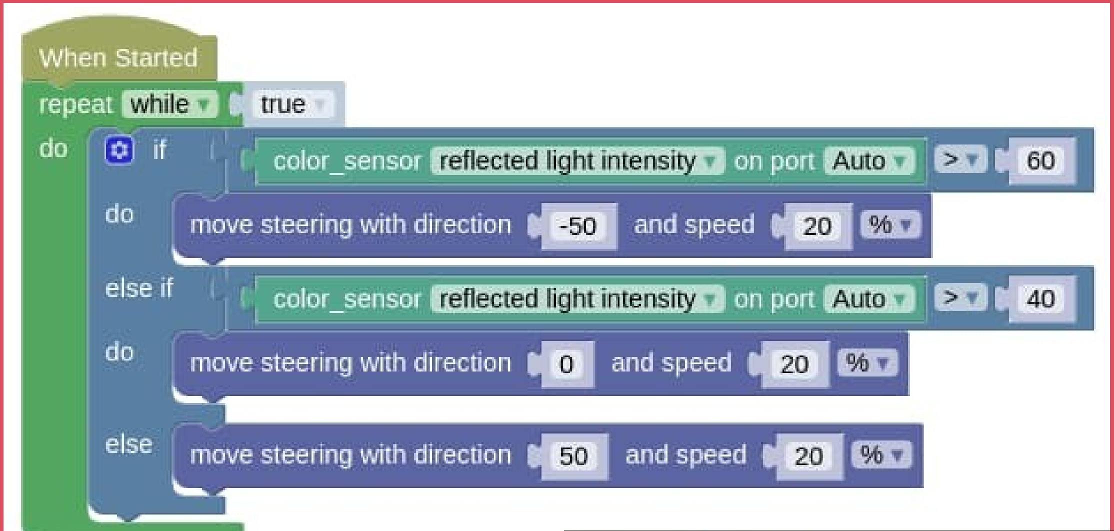


Light Sensor Value

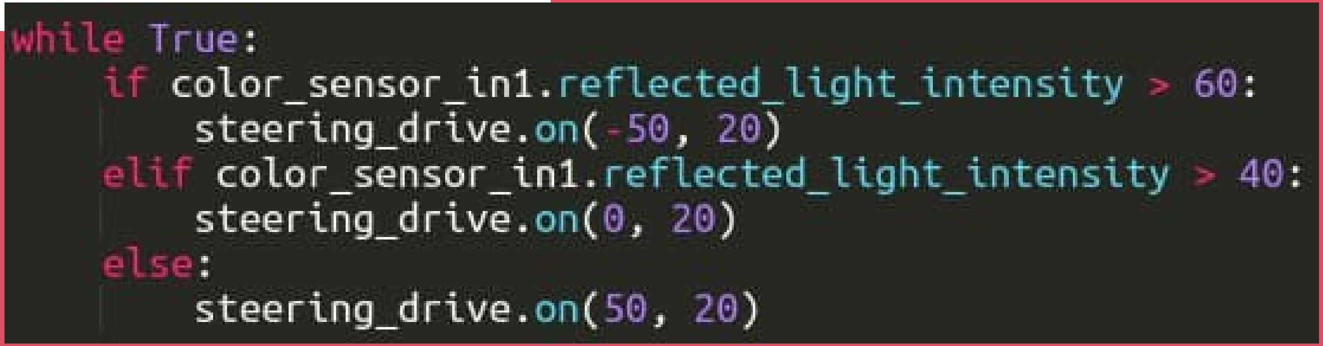
Pseudo Code

```
IF Value > 60:
  Turn Left
ELSE IF Value > 40:
  Go Straight
ELSE
  Turn Right
```

3 States Algorithm



```
When Started
repeat while ▾ true ▾
do
    if color_sensor reflected light intensity ▾ on port Auto ▾ > ▾ 60
    do move steering with direction -50 and speed 20 % ▾
    else if color_sensor reflected light intensity ▾ on port Auto ▾ > ▾ 40
    do move steering with direction 0 and speed 20 % ▾
    else move steering with direction 50 and speed 20 % ▾
```

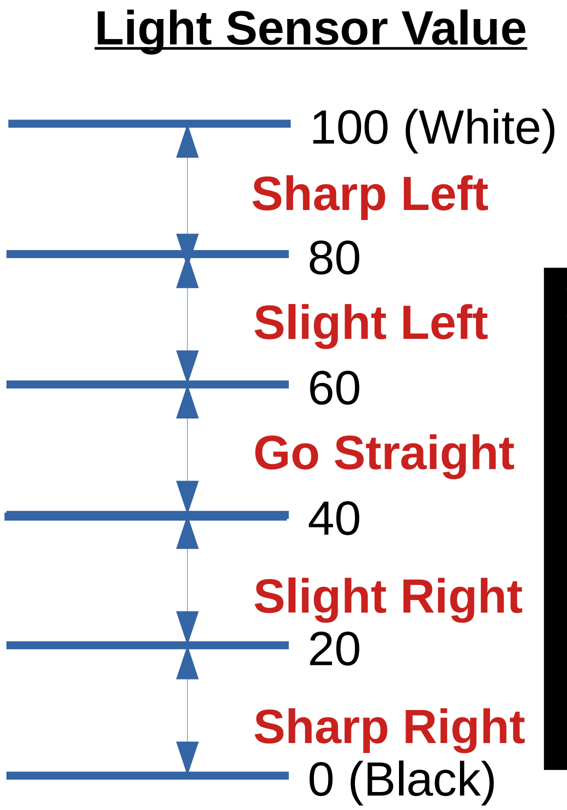


```
while True:
    if color_sensor_in1.reflected_light_intensity > 60:
        steering_drive.on(-50, 20)
    elif color_sensor_in1.reflected_light_intensity > 40:
        steering_drive.on(0, 20)
    else:
        steering_drive.on(50, 20)
```

Common Problems

- Problem:
  - Better than 2 states, but still a little jerky
  - May be good enough
- Can we do better?

5 States Algorithm



Light Sensor Value

100 (White)

Sharp Left

80

Slight Left

60

Go Straight

40

Slight Right

20

Sharp Right

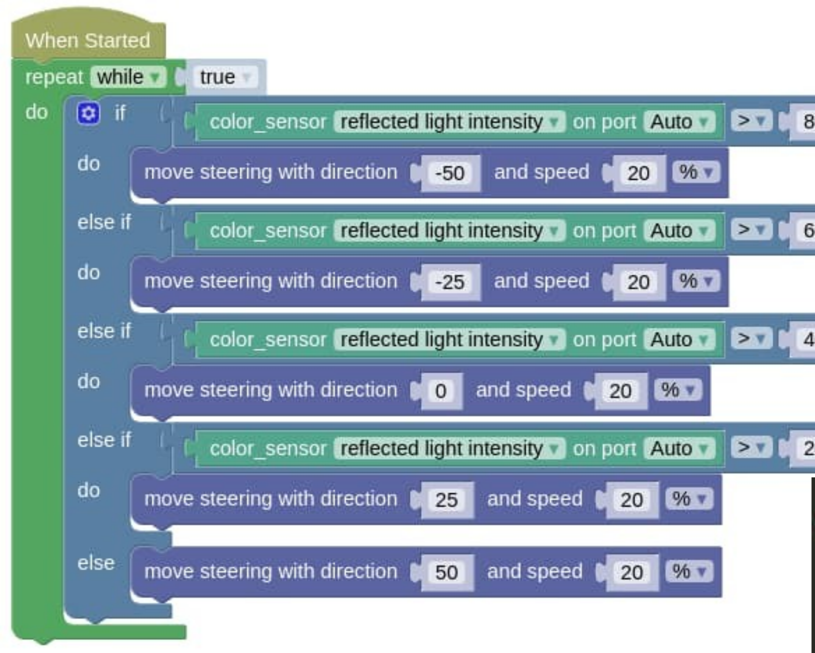
0 (Black)

Pseudo Code

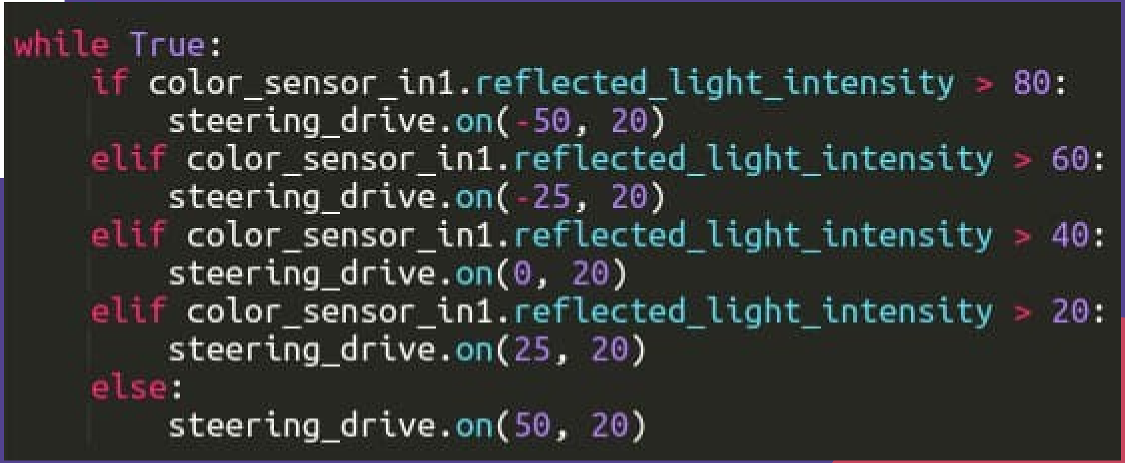
```
IF Value > 80:
  Turn Sharp Left
ELSE IF Value > 60:
  Turn Slight Left
ELSE IF Value > 40:
  Go Straight
ELSE IF Value > 20:
  Turn Slight Right
ELSE
  Turn Right
```

- Checks for 5 levels of light sensor value
- Robot runs even smoother than 3 states

5 States Algorithm



```
When Started
repeat while ▾ true ▾
do  if  color_sensor reflected light intensity ▾ on port Auto ▾ > ▾ 80
    do  move steering with direction -50 and speed 20 % ▾
    else if  color_sensor reflected light intensity ▾ on port Auto ▾ > ▾ 60
    do  move steering with direction -25 and speed 20 % ▾
    else if  color_sensor reflected light intensity ▾ on port Auto ▾ > ▾ 40
    do  move steering with direction 0 and speed 20 % ▾
    else if  color_sensor reflected light intensity ▾ on port Auto ▾ > ▾ 20
    do  move steering with direction 25 and speed 20 % ▾
    else  move steering with direction 50 and speed 20 % ▾
```

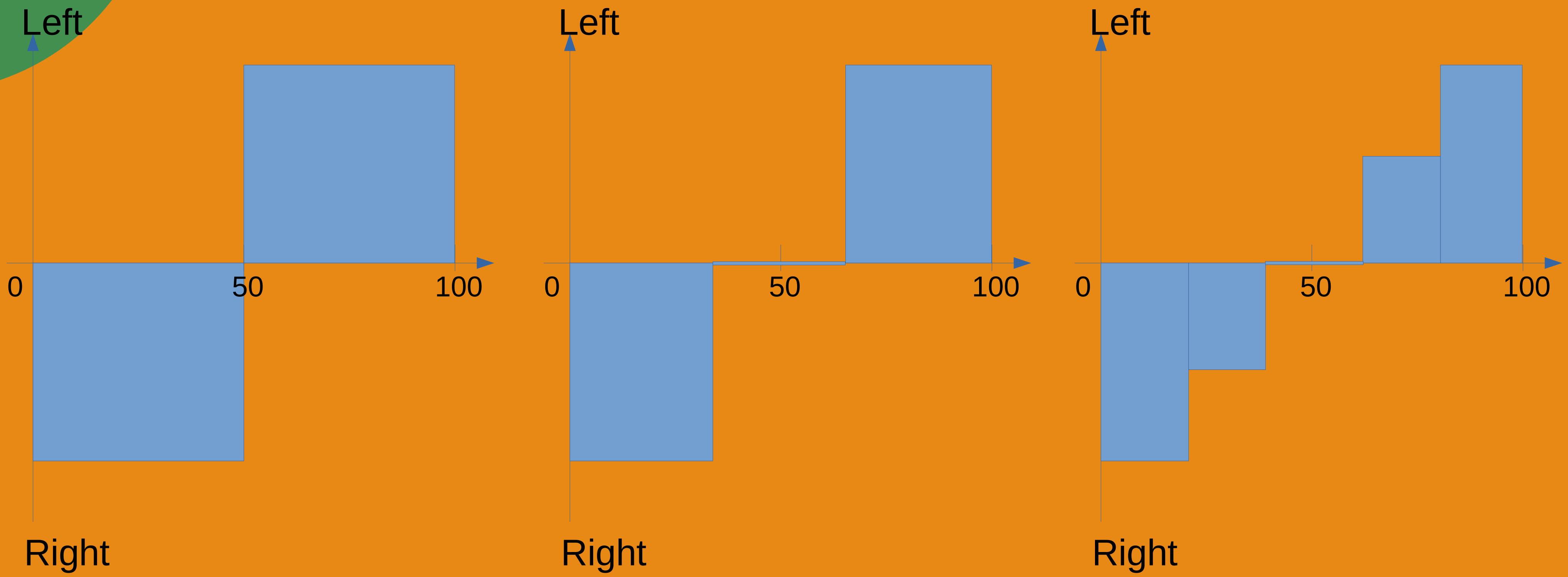


```python
while True:
    if color_sensor_in1.reflected_light_intensity > 80:
        steering_drive.on(-50, 20)
    elif color_sensor_in1.reflected_light_intensity > 60:
        steering_drive.on(-25, 20)
    elif color_sensor_in1.reflected_light_intensity > 40:
        steering_drive.on(0, 20)
    elif color_sensor_in1.reflected_light_intensity > 20:
        steering_drive.on(25, 20)
    else:
        steering_drive.on(50, 20)
```

PLAY · EXPERIENCE · LEARN

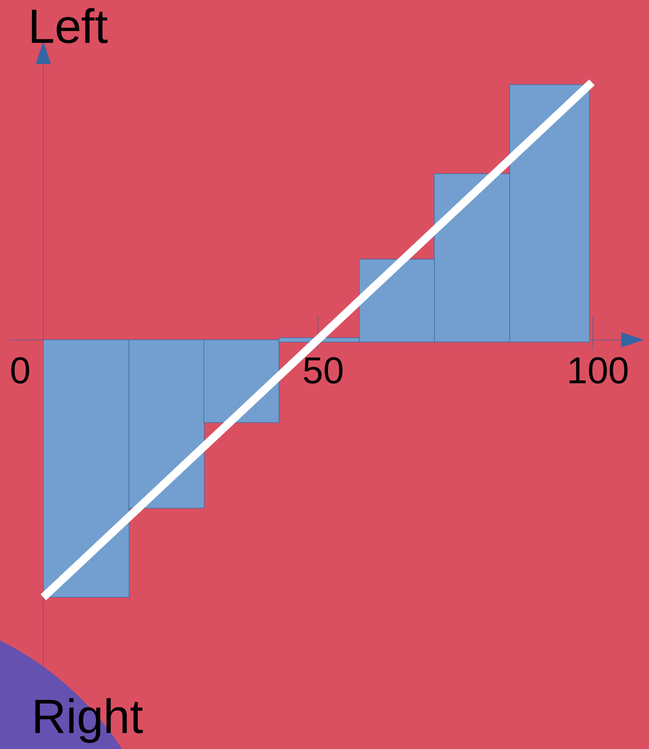
A POSTERIORI

Comparison of 2, 3, 5 States

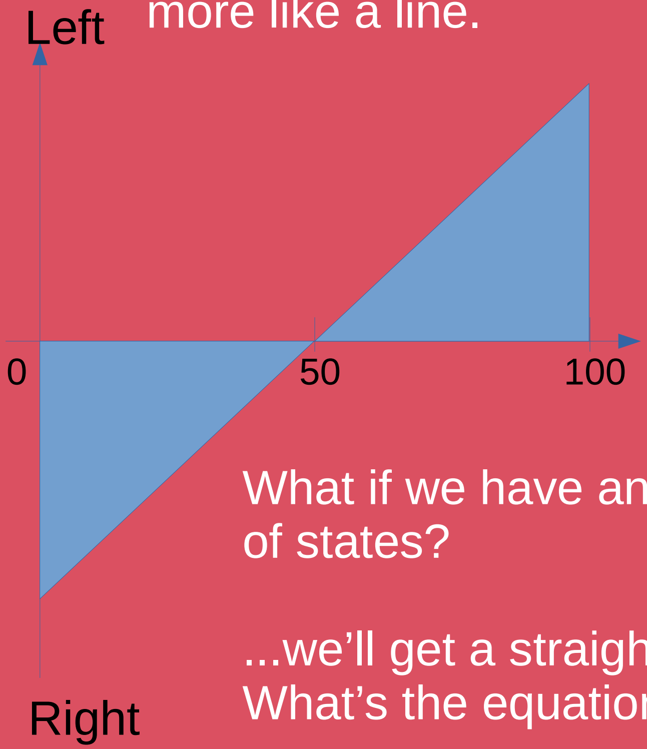


What happens if I increase the number of states?
(eg. 7 states, 9 states, 11 states)

Increasing number of states



As we increase the number of states, the diagram starts to look more like a line.



What if we have an infinite number of states?

...we’ll get a straight line!
What’s the equation of the line?

A POSTERIORI — PLAY · EXPERIENCE · LEARN

Equation of the Line

- Standard form

  $y = mx + c$

- Crosses x axis at x = 50, y = 0

  $0 = m(50) + c$

  $m = -c / 50$

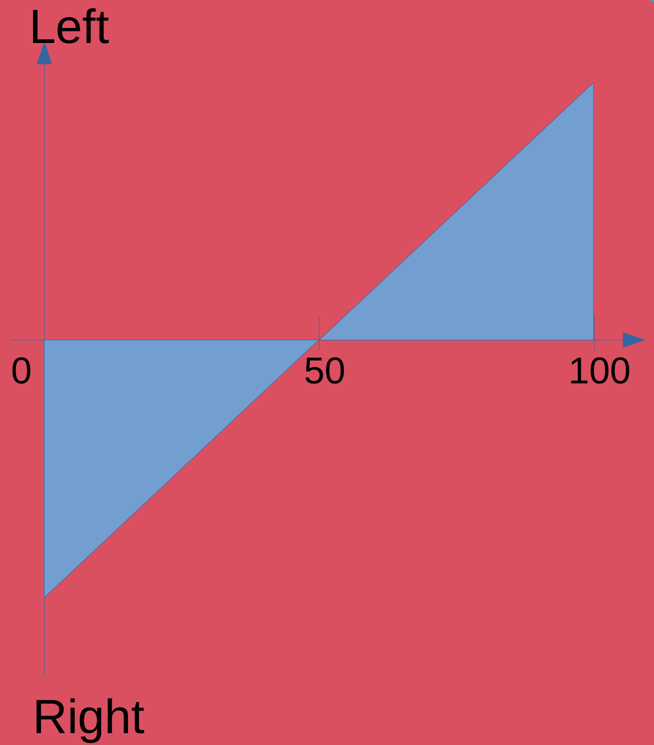
- Substitute and rearrange

  $y = (-c / 50)x + c$

  $y = -c\ (x / 50 - 1)$

  $y = -c / 50\ (x - 50)$

  $y = k\ (x - 50) \quad , \quad \text{where } k = -c / 50$

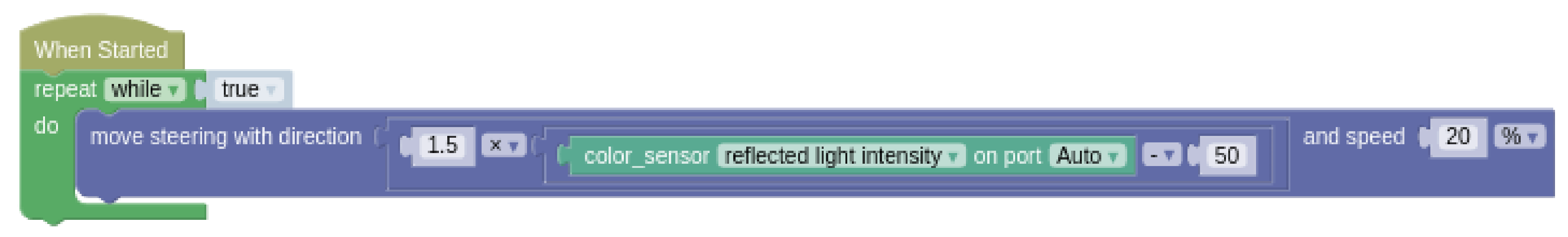


Equation of Line (Engineering Style)

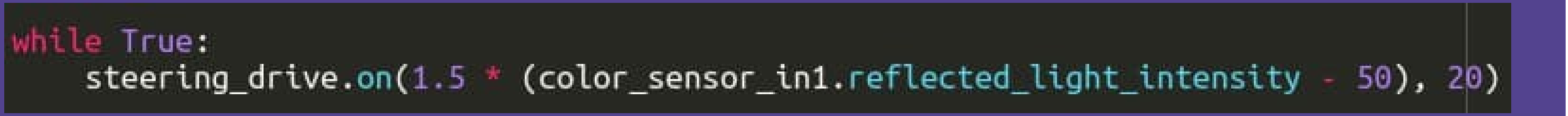
$$\underbrace{\overset{\text{Gain}}{\mathrm{Kp}} \times \underbrace{(\overset{\text{Sensor value}}{S} - \overset{\text{Midpoint}}{50})}_{\text{Error}}}_{\text{Correction}}$$

* The “p” in “Kp” stands for proportional. In a full PID (Proportional, Integral, Derivative) control, you will also have an “Ki” and “Kd”.

Proportional Control



```
When Started
repeat while ▾ true ▾
do
    move steering with direction 1.5 × ▾ color_sensor reflected light intensity ▾ on port Auto ▾ - ▾ 50 and speed 20 % ▾
```



```
while True:
    steering_drive.on(1.5 * (color_sensor_in1.reflected_light_intensity - 50), 20)
```

I’m using a gain of “1.5”. Experiment with other gain value (eg. 0.2, 1.0, 2.0) and see how that affects your robot.

Changing Gain

- Increase Gain
  - Turns more sharply, may wobble
- Decrease Gain
  - Tuns more smoothly, may fail at sharp turns
- Negative Gain?
  - Try it out

Setting gain higher than “2” may cause an error. Try it out, read the error message, and see if you can figure out why.

Is Proportional Control the Best Solution?

- Depends. Proportional controls have a straight line response, and you can only tune the Gain (gradient of the line)

- High gain may wobble too much, low gain may fail at sharp turns. Depending on the map and robot, there may not exist a Gain value that is both smooth and can handle sharp turns.

Areas to Explore

- Non-proportional controls (ie. not a straight line eqn).
  - Will a quadratic eqn work? (spoiler: No it won’t, but why not?)
  - What about a cubic eqn?
- Add in Integral and Derivative terms to make it a PID controller

Challenges

- Try to complete the “Sharp Turns” challenge
- Create a modified version that follows the right edge of the line
- Use the same concept to create a wall / gyro follower

A POSTERIORI

PLAY . EXPERIENCE . LEARN

A POSTERIORI

Play · Experience · Learn

- Created by A Posteriori LLP
- Visit http://aposteriori.com.sg/ for more tips and tutorials
-