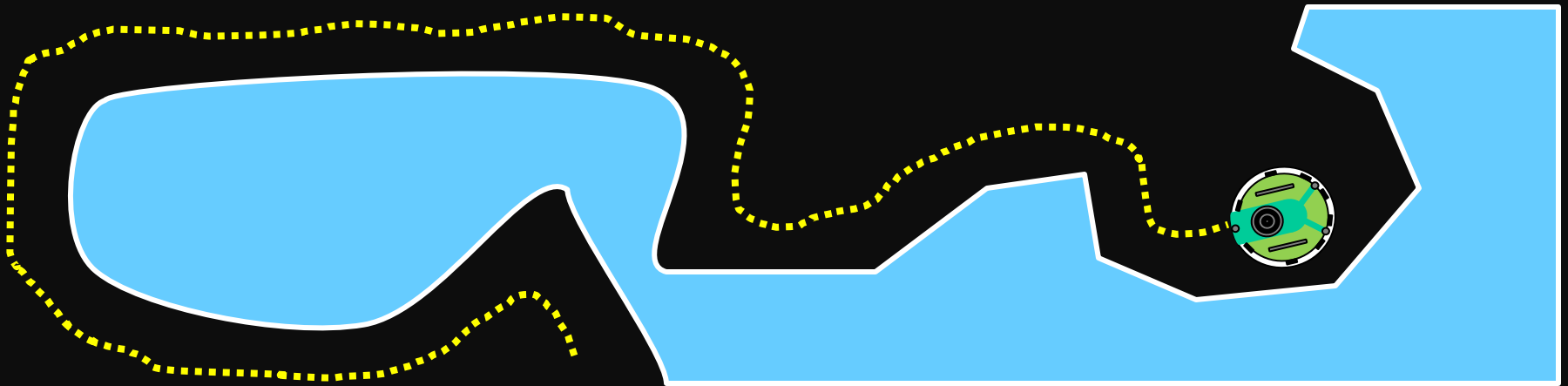


Wall-Following

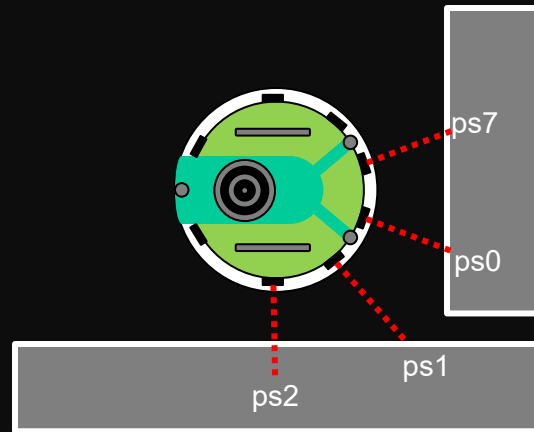
Wall Following

- Wall following behavior is useful for mapping, navigation, seeking wall outlets, performing cleaning tasks etc...
- Strategy varies depending on types of sensors.
- Robot usually follows wall by keeping itself aligned to the wall on its left or right side



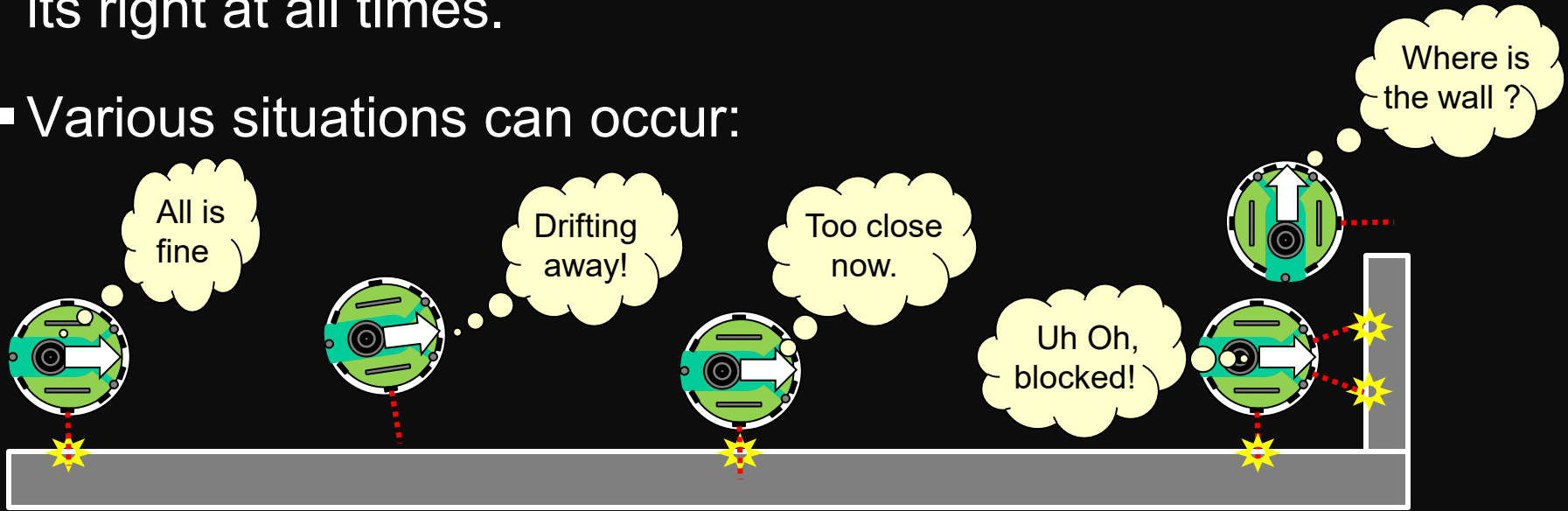
Wall Following

- At any time, robot simply moves forward or turns right or left depending on the shape of the contour that it is following.
- We will consider right-handed wall following only
- Consider the e-puck robot using 4 sensors as follows:



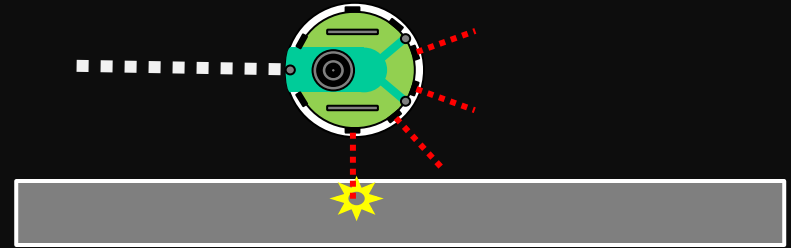
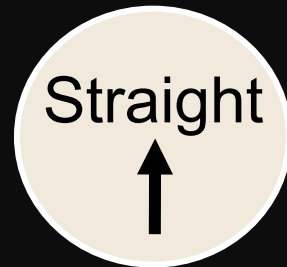
Wall Following

- Robot will try to maintain the same distance from the wall on its right at all times.
- Various situations can occur:

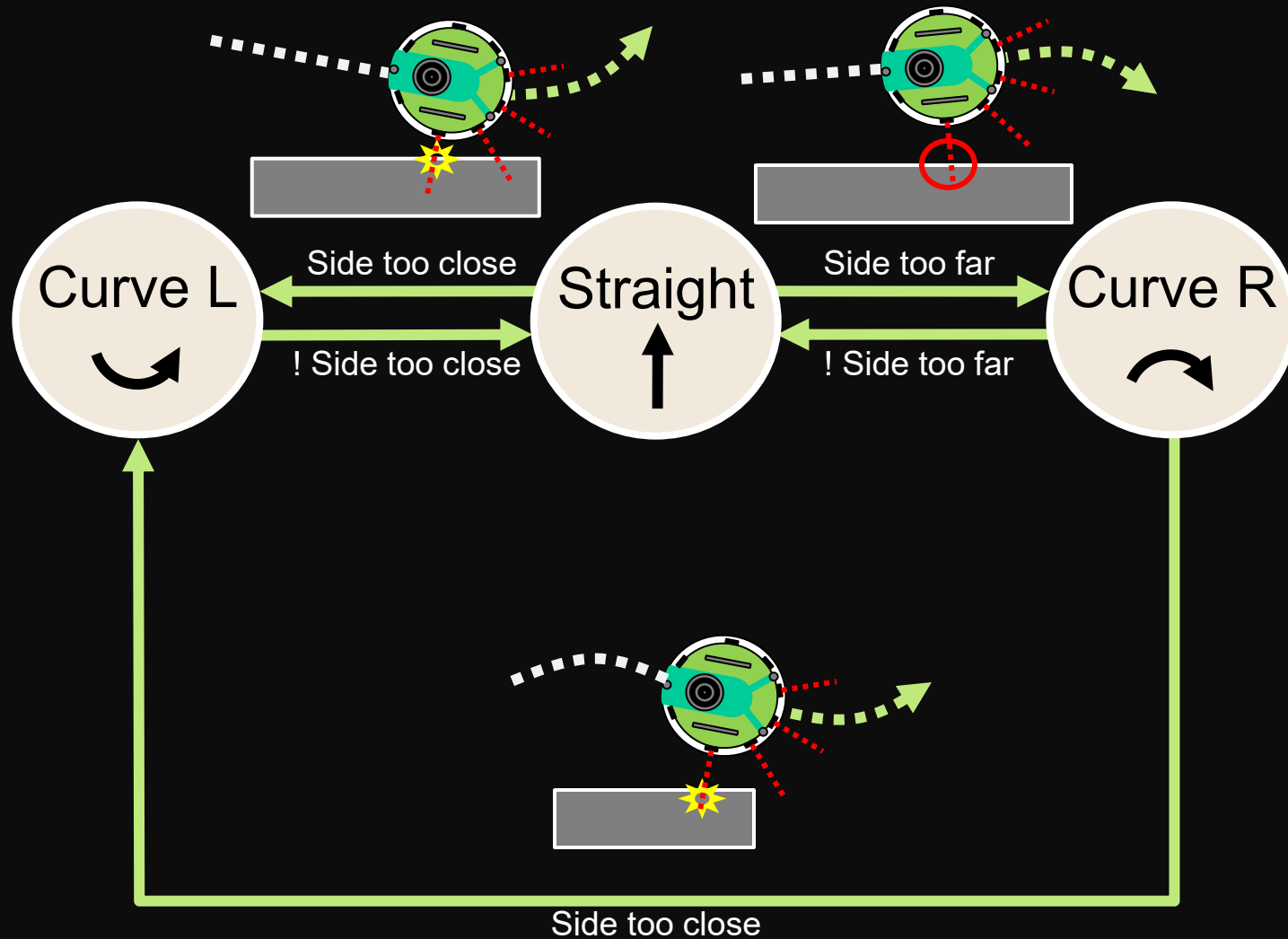


- Robot will thus be in certain *modes* to cause it to travel ahead or make appropriate turns to re-align with the wall or orient itself to a new edge.
- Can use a state machine to do this...

Wall Following – State Machine

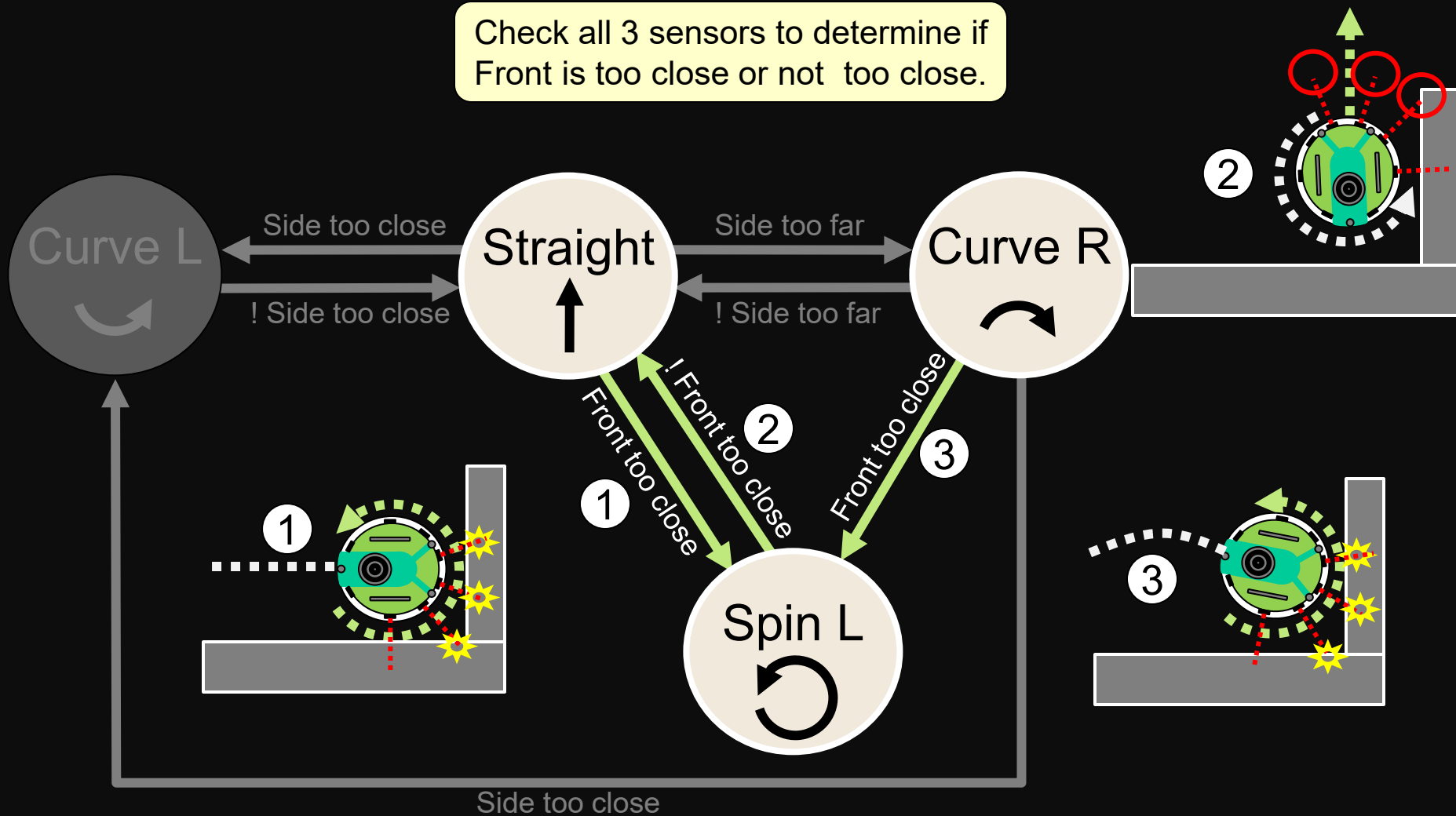


Wall Following – State Machine

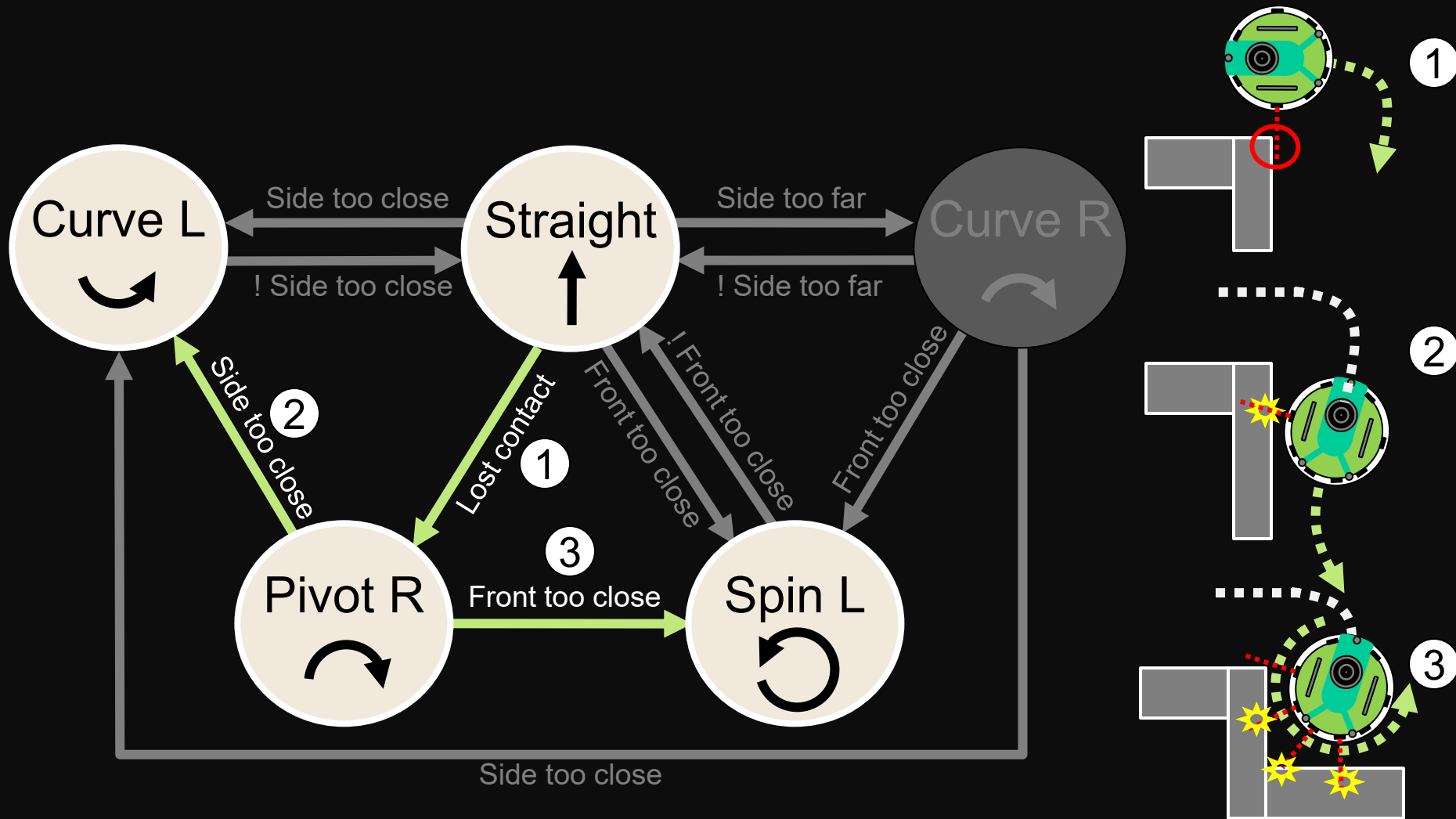


Wall Following – State Machine

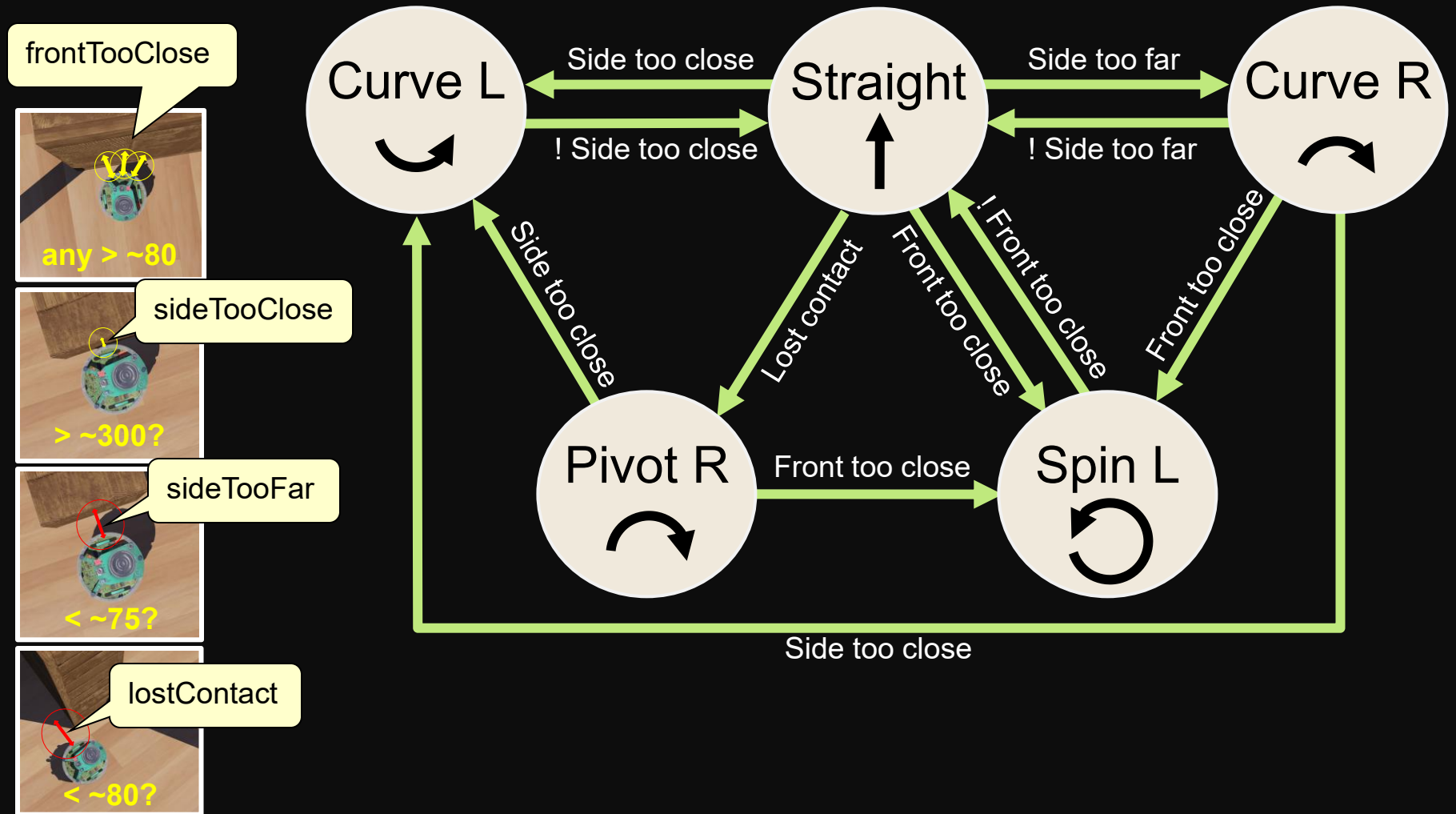
Check all 3 sensors to determine if Front is too close or not too close.



Wall Following – State Machine



Wall Following – Completed



Wall Following – Code Structure

```
// States represented as unique #'s
static final byte STRAIGHT = 0;
static final byte SPIN_LEFT = 1;
static final byte PIVOT_RIGHT = 2;
static final byte CURVE_LEFT = 3;
static final byte CURVE_RIGHT = 4;

byte state = STRAIGHT;
while(robot.step(timeStep) != -1) {
    // SENSE: Read all the sensors.

    // To make your life easier,
    // create some booleans to match
    // the state transition labels
    // based on the sensor values.
    // Then use these booleans in
    // your switch statement if
    // conditions.
    boolean sideTooClose = ...;
    boolean sideTooFar = ...;
    boolean lostContact = ...;
    boolean frontTooClose = ...;

    // THINK
    // REACT
}
```

```
switch(state) {
    case STRAIGHT:
        if (sideTooClose) {}
        // etc..
        break;
    case SPIN_LEFT:
        if (...) {}
        // etc..
        break;
    case PIVOT_RIGHT:
        if (...) {}
        // etc..
        break;
    case CURVE_LEFT:
        if (...) {}
        // etc..
        break;
    case CURVE_RIGHT:
        if (...) {}
        // etc..
        break;
}
```

```
switch(state) {
    case STRAIGHT:
        // Set motor speeds ...
        break;
    case SPIN_LEFT:
        // Set motor speeds ...
        break;
    case PIVOT_RIGHT:
        // Set motor speeds ...
        break;
    case CURVE_LEFT:
        // Set motor speeds ...
        break;
    case CURVE_RIGHT:
        // Set motor speeds ...
        break;
}
```

... = any code that results in a true/false value
(e.g., contents of an IF statement condition).



Start the Lab ...