UNIVERSITY · PRESENTATION 4 · LEARNING WITH C

Oil or Rocks

Two black letters, one colour: the camera tells an O from an R by its size and by the hole in the middle.

TheBotCup 2026 · AiBotCup Simulator · University



RECAP

Last time: colour tests

Camera

Camera[row][col]

24 × 18 pixels, [R, G, B] each; row 17 is just in front.

Colour test

Safe ranges

Gold: R > 195, G 135–200, B < 80.

Today

Shape test

Same colour, different shape: count, measure, look inside.

PART 1

The mission: Session 15

Take the Oil, leave the Rocks

Session 15 · Oil or Rocks

Collect 2 Oil letters (O). Leave the Rocks (R): trying to collect them costs −10.

Each Oil: gold + 10 = 30. 2 Oil: +150.

Oil 0/2

Camera tip: O has an empty centre; R has ink in its centre. Look at the middle of the black shape before you collect.

Sessions | Restart session

SESSION 15 · OIL OR ROCKS

2 × O, and never an R

Collect **2 Oil** letters (O): **30** each, then **+150**. Total **210**.

Trying to collect an **R** costs **−10**.

Time limit **5:00**.

Medals · Silver: under 3:00 · Gold: under 1:45 and perfect.



THE FIELD

Three O, three R, and metals

3 O and **3 R** letters, all black.

Also iron (black too), silver and gold.

Black is not enough: O, R and iron are **the same colour**.

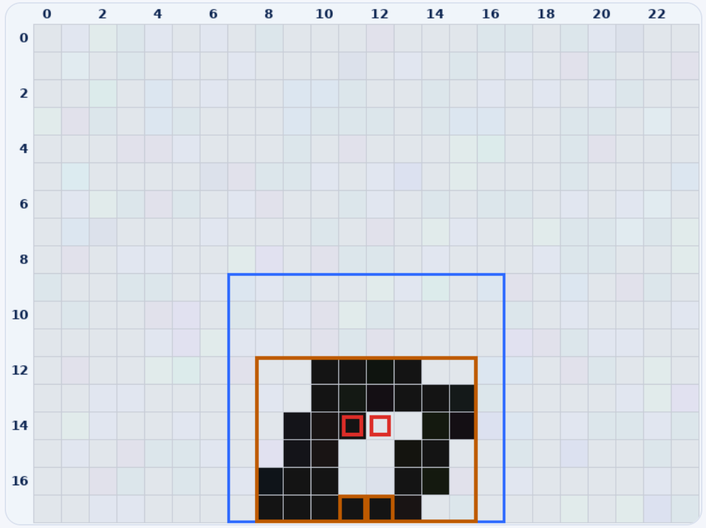
PART 2

Shape, not colour

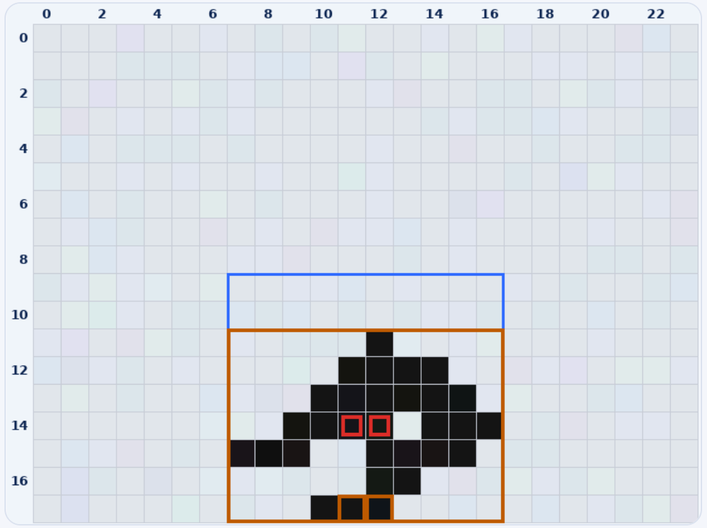
Count the ink · find the box · look in the middle

SHAPE · THE PROBLEM

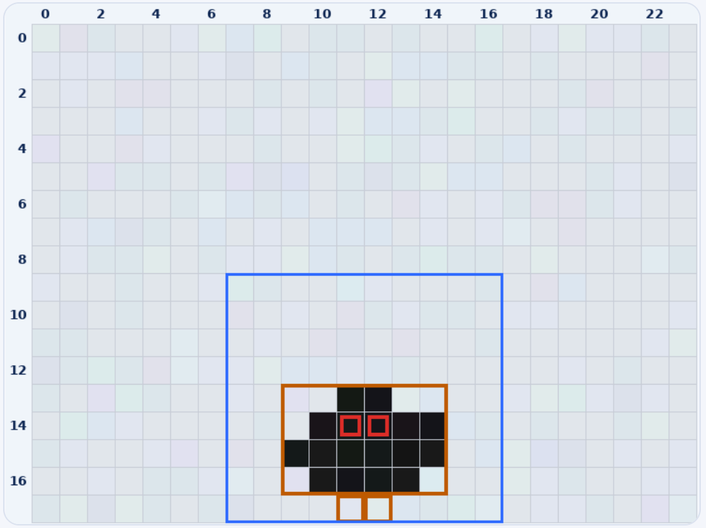
Three black things, one colour test



O (Oil): a ring, **30** black pixels.



R (Rocks): **30** black pixels, solid middle.



Iron: a small blob, **17** black pixels.

All three pass the same test: R, G and B all below 60 (measured 21–29). Real camera images from the simulator.

SHAPE · STEP 1

Count the ink: big or small?

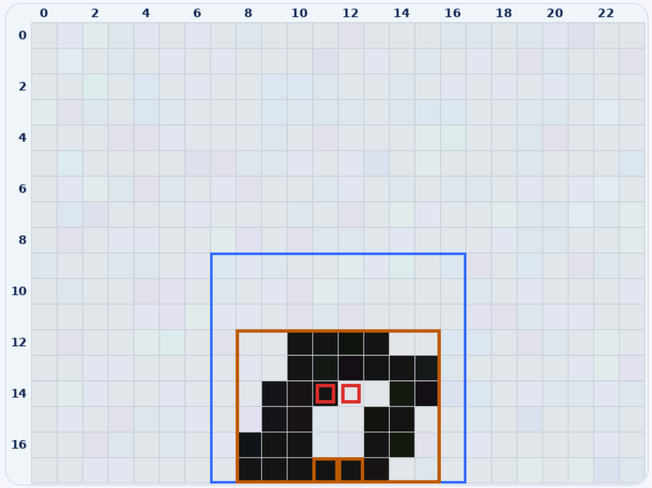
```
int n = 0; int r0 = CAM_H; int r1 = -1; int c0 = CAM_W; int c1 = -1;
for (int y = CAM_H - 9; y < CAM_H; y++)     // the 9 rows nearest the robot
    for (int x = mid - 5; x < mid + 5; x++)  // the 10 middle columns
        if (black(y, x)) {
            n++;                              // how much ink?
            r0 = min(r0, y); r1 = max(r1, y); // top and bottom row
            c0 = min(c0, x); c1 = max(c1, x); // left and right column
        }
```

Search only a **window**: the 9 rows nearest the robot, the 10 middle columns.

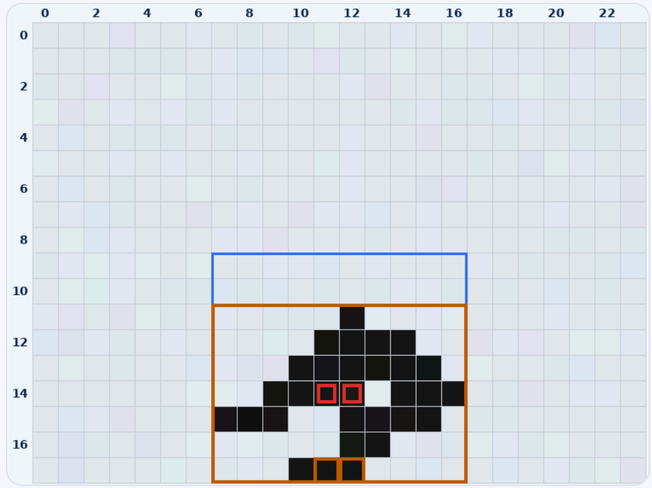
Our samples: iron **17–20** black pixels, letters **27–31**.

Also remember the **box**: top, bottom, left and right of the ink.

n ≤ 23: an iron object. More: a letter.



O: centre row 14 → col 11 ink, col 12 **floor**



R: centre row 14 → cols 11 and 12 both **ink**

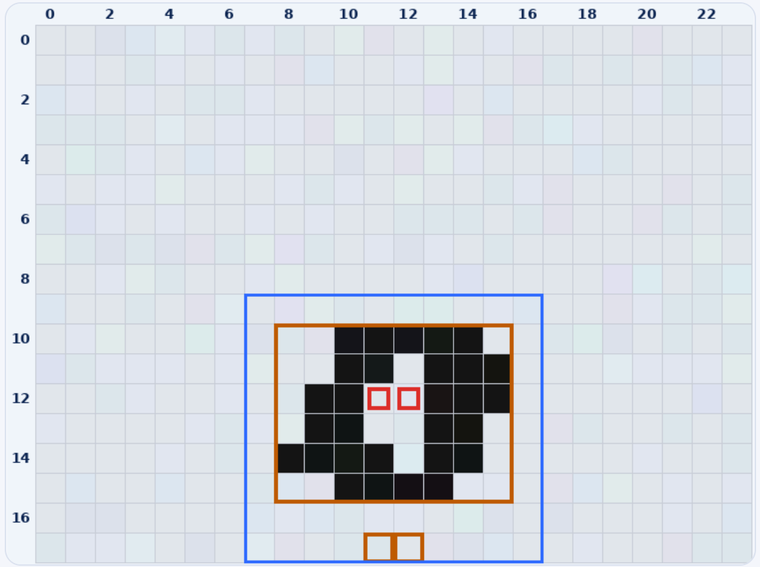
SHAPE · STEP 2

The box, and its centre

Box: rows r0–r1, columns c0–c1. Centre row = (r0 + r1) ÷ 2.

If the width is even, the centre lies **between two columns**: check both.

Both ink → **R** · any floor → a hole → **O**.



Another O, further away: a clear hole.

SHAPE · HOW GOOD IS IT?

Tested on real camera images

We placed the robot in front of all **6 letters** and **1 iron** object at different distances: **23** camera images.

The rule was right in **21**. Both mistakes were letters **not yet at the pickup point**.

So the program decides **only at the pickup point**, where it was always right.

24 × 18 pixels is coarse: **test your rules on real images**.

PART 3

Step by step in C

Find black, check the shape, collect or leave

STEP 1 · THE PLAN

Five rules, most important first

1. Black at the pickup point and the shape is **O** → **Collect**.
2. Black at the pickup point, not an O → **leave it**: turn away, ignore black for 2.4 s.
3. Wall or block close → **turn**.
4. Black in view (and not ignoring) → **steer** to it.
5. Nothing seen → **search** in a wide curve.

STEP 2 · LINES 6–8 AND 10–15

Black pixels, and the nearest one

```
6  int black(int y, int x) {                    // iron, O or R ink?
7      return Camera[y][x][0] < 60 && Camera[y][x][1] < 60 && Camera[y][x][2] < 60;
8  }
```

```
10 int nearestBlack() {                         // column of the nearest black, or -1
11     for (int y = CAM_H - 1; y >= 0; y--)
12         for (int x = 0; x < CAM_W; x++)
13             if (black(y, x)) { gy = y; return x; }
14     return -1;
15 }
```

STEP 3 · LINES 17–30

The shape test

```
17  int shape() {                               // 0 iron, 1 Oil (O), 2 Rocks (R)
18      int mid = CAM_W / 2;
19      int n = 0; int r0 = CAM_H; int r1 = -1; int c0 = CAM_W; int c1 = -1;
20      for (int y = CAM_H - 9; y < CAM_H; y++)
21          for (int x = mid - 5; x < mid + 5; x++)
22              if (black(y, x)) {
23                  n++;
24                  r0 = min(r0, y); r1 = max(r1, y); c0 = min(c0, x); c1 = max(c1, x);
25              }
26      if (n <= 23) return 0;                  // small: an iron object
27      int cy = (r0 + r1) / 2;                 // centre row of the box
28      if (black(cy, (c0 + c1) / 2) && black(cy, (c0 + c1 + 1) / 2)) return 2;  // solid: R
29      return 1;                               // a hole in the centre: O
30  }
```

Count ink and find the box in the window.

n ≤ 23 → 0, iron.

Centre row, both centre columns ink → 2, R; else 1, O.

STEP 4 · LINES 32–43

Collect the O, leave the rest

```
32  void Game1() {
33      LED_1 = 0;
34      int mid = CAM_W / 2;
35      ignore = max(0, ignore - 1);
36      int here = black(CAM_H - 1, mid - 1) && black(CAM_H - 1, mid) && ignore == 0;
37      int col = nearestBlack();
38      if (here && shape() == 1) {                          // Oil: collect
39          WheelLeft = 0; WheelRight = 0;
40          LED_1 = 2; LED_R = 255; LED_G = 0; LED_B = 0;
41          Duration = 3200;
42      } else if (here) {                                   // Rocks or iron: leave it
43          WheelLeft = 3; WheelRight = -3; Duration = 600; ignore = 40;
```

ignore counts down by 1 every loop, never below 0.

here: both pickup pixels black, and not ignoring.

Not an O: turn 0.6 s, **ignore = 40**.

STEP 5 · LINES 44–54

Walls, steering and search

```
44      } else if (US_Front < 18) {                                  // Wall
45          WheelLeft = 3; WheelRight = -3; Duration = 400;
46      } else if (col >= 0 && ignore == 0) {                        // SteerToBlack
47          float err = col - (CAM_W - 1) / 2.0;
48          float speed = 3;
49          if (gy > CAM_H - 5) speed = 1.5;
50          WheelLeft = speed + err * 0.25; WheelRight = speed - err * 0.25;
51      } else {                                                     // Search
52          WheelLeft = 3.5; WheelRight = 2.5;
53      }
54  }
```

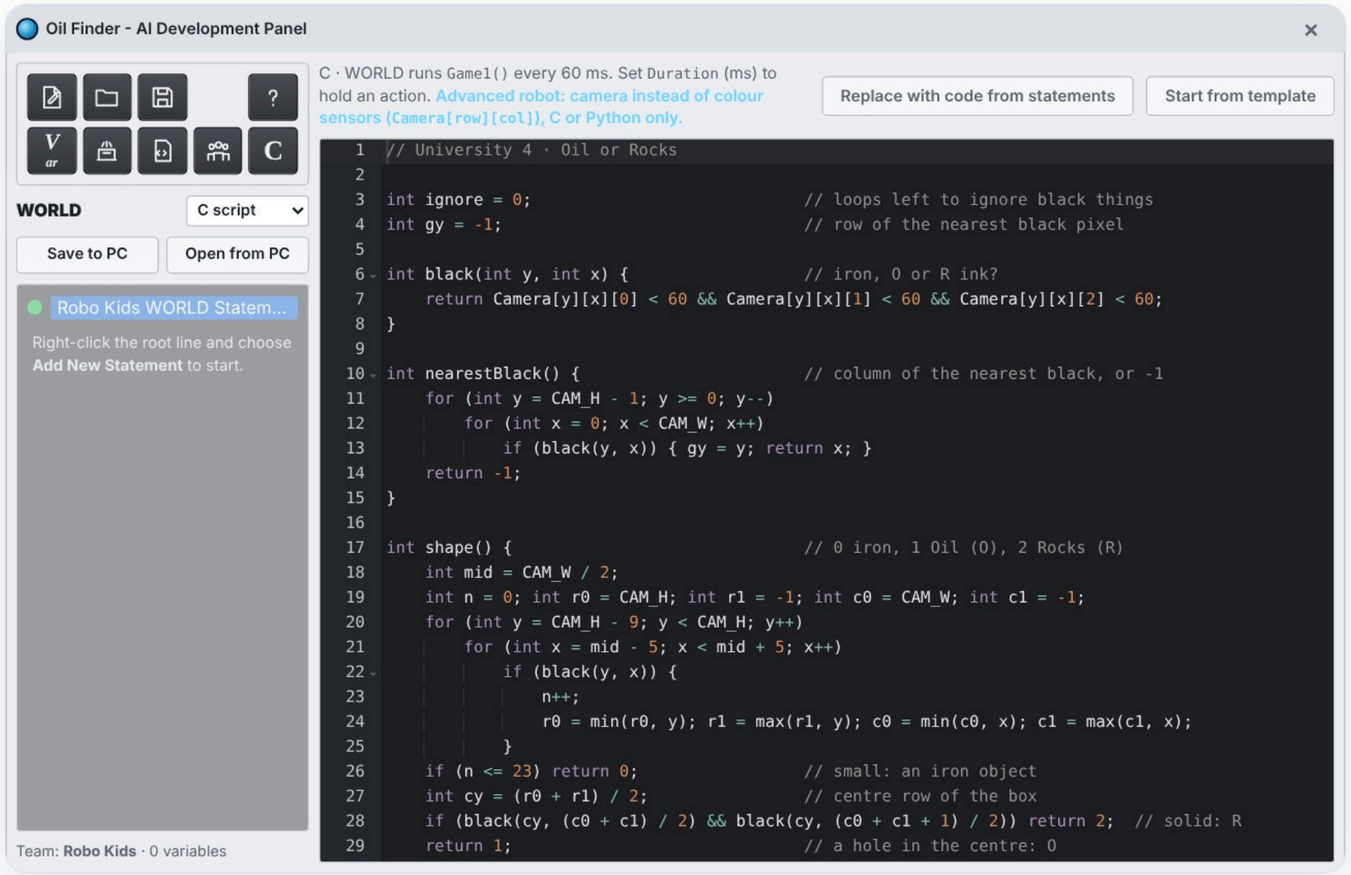
Steering exactly as in Gold hunt: 0.25 × err, slower in the last 4 rows.

While ignoring, the robot searches instead of steering to black.

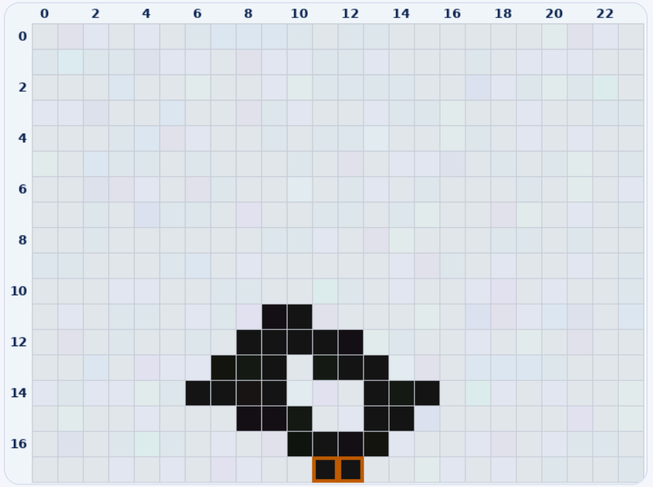
STEP 6 · BUILD AND LOAD

The whole program, built

1. Category **Advanced**, Learning, Session 15.
2. WORLD menu: **C script**.
3. Name it **Oil Finder**.
4. **Build**, then **Load** on BLUE.







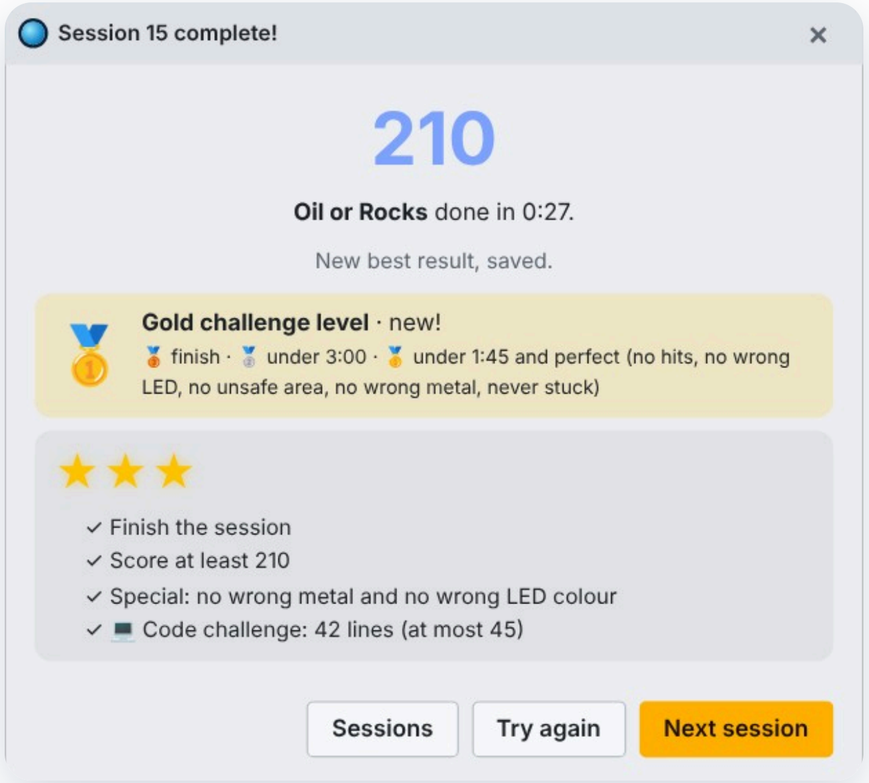
STEP 7 · RUN · 19.7 S

An O at the pickup point

At **19.7 s** the robot is on an O: the shape test finds the hole.

Red LED, 3.2 s: **+30**.

Then on to the next black shape.



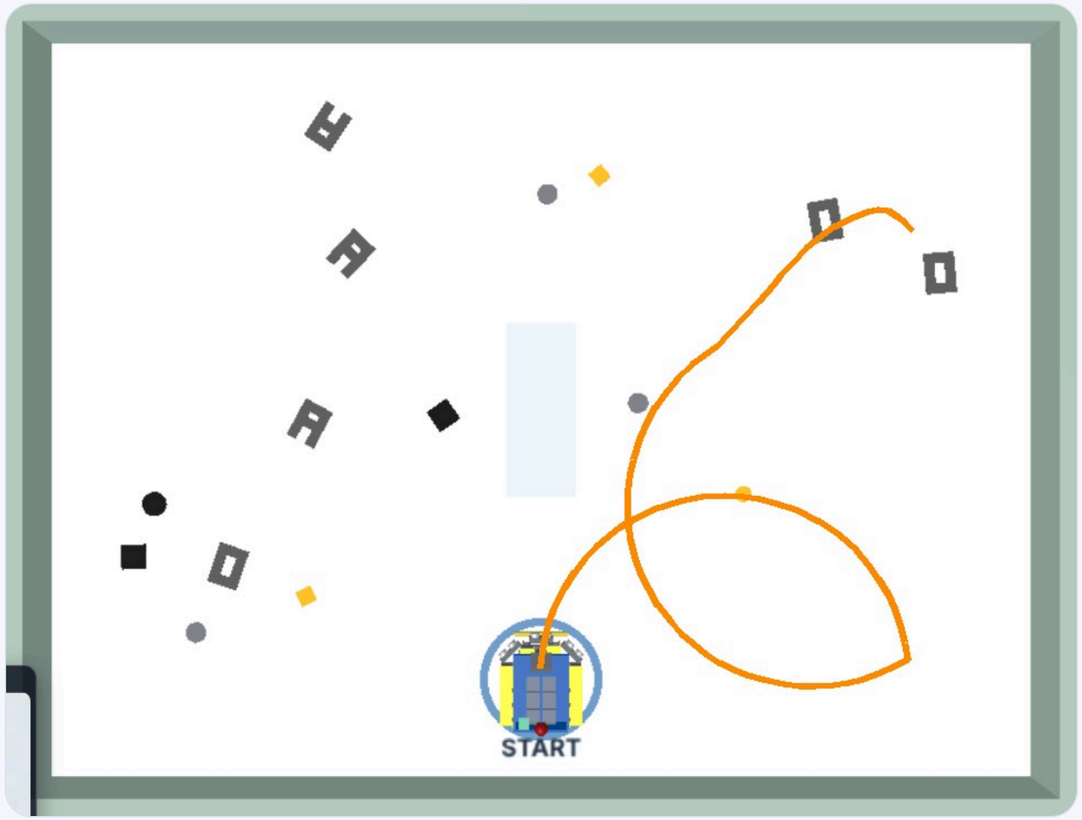
RESULT · TESTED

210 points in 0:27 · Gold

Medals · Silver: under 3:00 · Gold: under 1:45 and perfect.

Stars · finish · score 210 · no wrong metal and no wrong LED colour.

Code challenge · at most 45 lines. We used 42: a tight fit.



THE PATH · RECORDED

Curves, then straight to the letters

Wide curves while searching; straight runs once black is in view.

Colour finds **candidates**; shape decides **what they are**.

DID NOT WORK?

Check first, then tune carefully

What you see	What to do
Collects an R (−10)	Check both centre columns; print n and the box
Drives past an O without stopping	The O’s ink must cover both pickup pixels: slow down earlier
Keeps going back to the same R	ignore must be written outside Game1(), and set to 40
Never steers to anything	ignore never reaches 0: check the countdown line
Too many lines	Write short branches on one line; lines with only a bracket do not count

NEXT TIME

University 5 · Line up and collect

Turn until the object is dead ahead, **then** drive: line up first, and watch out for look-alike colours.