UNIVERSITY · PRESENTATION 2 · LEARNING WITH C

Two targets by position

Two pads in a fixed order: a program that changes its own target while it runs.

TheBotCup 2026 · AiBotCup Simulator · University



RECAP

Last time: compute, then steer

dx, dy

TX − PositionX, TY − PositionY

Distance

sqrt(dx² + dy²)

Heading

atan2d(dy, dx) − 90

Error

angleDiff(heading, Compass)

Then: error over 8° → turn on the spot; otherwise drive and steer by 0.2 × error.

PART 1

The mission: Session 13

Target 1, then target 2



SESSION 13 · TWO TARGETS BY POSITION

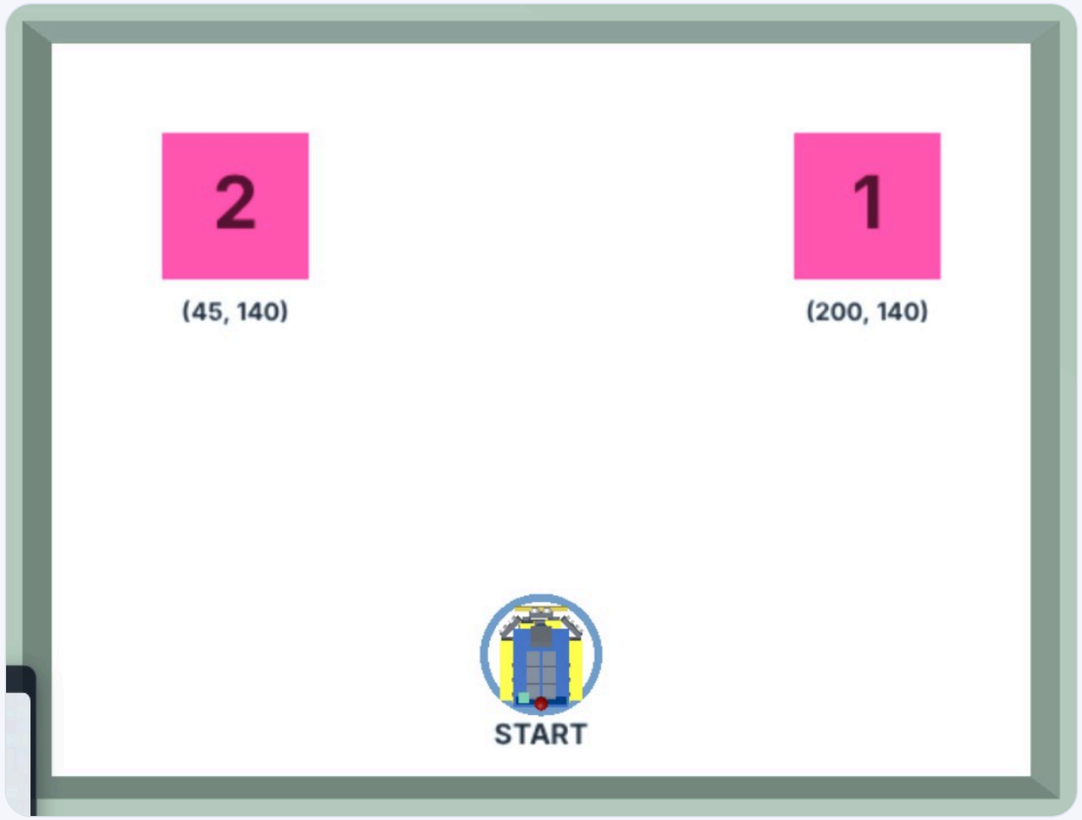
Only the next target counts

Start at **(120, 30)**, facing north.

Target 1 **(200, 140)**: +100 · Target 2 **(45, 140)**: +150.

Only the **next** target in order counts. Time limit **4:00**.

Medals · Silver: under 2:24 · Gold: under 1:24 and perfect.



THE FIELD

Two pads, one route

240 × 180 cm, empty.

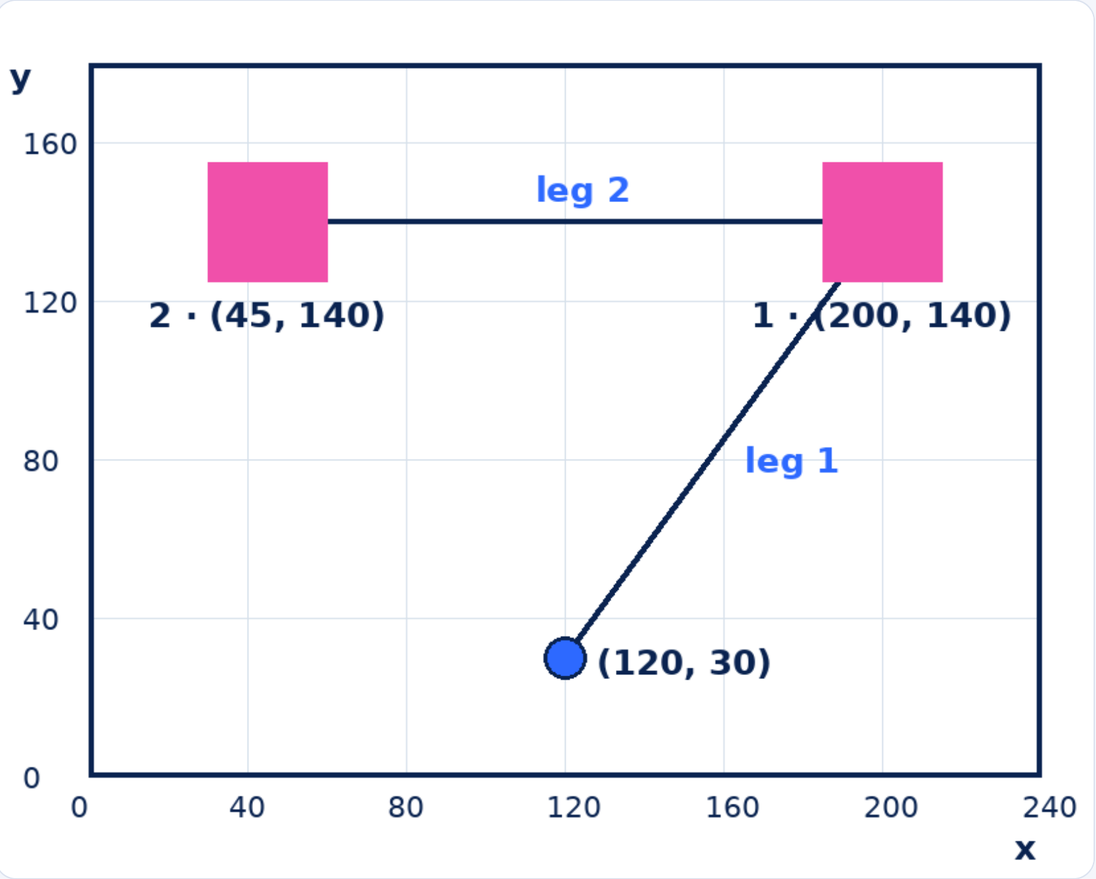
Pad 1 top right, pad 2 top left, **155 cm** apart.

Same geometry as last time, just **twice**, in order.

PART 2

Two goals in order

A program that changes its own target



THE ROUTE · TWO LEGS

Compute each leg the same way

Leg 1 · dx 80, dy 110 → distance **136.0 cm**, heading **−36.0° = 324°**

Leg 2 · dx −155, dy 0 → distance **155 cm**, heading **180 − 90 = 90°** (west)

Tested: the robot drove leg 1 at Compass **323–324** and leg 2 at **89**.

THE IDEA · CHANGE THE TARGET

When target 1 is reached, aim at target 2

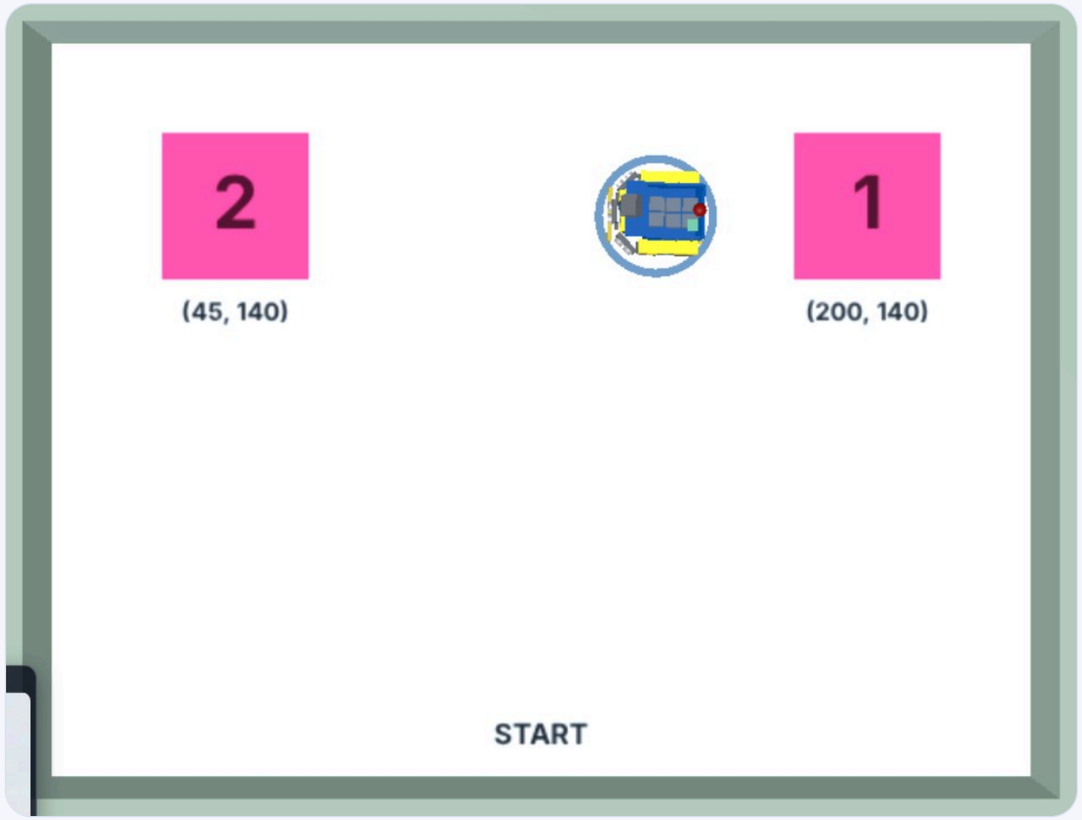
```
12      if (dist < 8 && TX == 200) {                    // NextTarget
13          TX = 45; TY = 140;
14      } else if (dist < 8) {                          // Arrived
15          WheelLeft = 0; WheelRight = 0;
```

TX == 200: we are still on leg 1.

TX and TY live **outside** Game1(): the change **stays**, and the next loop aims at target 2.

After the switch TX is 45, so this branch never runs again.

The variables are the robot's **memory**: which leg it is on.



THE IDEA · WHEN TO SWITCH

Switch only once the pad has counted

A pad is **30 × 30 cm**: it counts as soon as the robot drives onto it.

We switch at **dist < 8**: by then the robot is surely on pad 1.

Switch too early (say dist < 25) and the robot could turn away before the pad counts.

At 9.0 s: score **100**, Compass **89**: on its way west.

PART 3

Step by step in C

The same maths, plus one switch

STEP 1 · THE PLAN

Five rules, most important first

1. Compute dx, dy, distance, heading and error (as in University 1).
2. Within 8 cm and still on leg 1 → **switch** the target to (45, 140).
3. Within 8 cm of target 2 → **stop**.
4. Error more than 8° → **turn** on the spot.
5. Otherwise → **drive** and steer by 0.2 × error.

STEP 2 · LINES 3–4 AND 6–11

Target 1 first, then the same maths

```
3  float TX = 200;                                  // target 1 first
4  float TY = 140;
```

```
6  void Game1() {
7      float dx = TX - PositionX;
8      float dy = TY - PositionY;
9      float dist = sqrt(dx * dx + dy * dy);
10     float heading = atan2d(dy, dx) - 90;
11     float err = angleDiff(heading, Compass);
```

STEP 3 · LINES 12–15

The switch, then the stop

```
12      if (dist < 8 && TX == 200) {                                    // NextTarget
13          TX = 45; TY = 140;
14      } else if (dist < 8) {                                          // Arrived
15          WheelLeft = 0; WheelRight = 0;
```

First branch: the robot’s **own decision** to move on.

Second branch: the same test, now on leg 2, means **arrived**: stop.

STEP 4 · LINES 16–23

Turn or drive, as before

```
16      } else if (err > 8) {                                   // TurnLeft
17          WheelLeft = -2; WheelRight = 2;
18      } else if (err < -8) {                                  // TurnRight
19          WheelLeft = 2; WheelRight = -2;
20      } else {                                                // Drive, steering gently
21          WheelLeft = 4.5 - err * 0.2; WheelRight = 4.5 + err * 0.2;
22      }
23  }
```

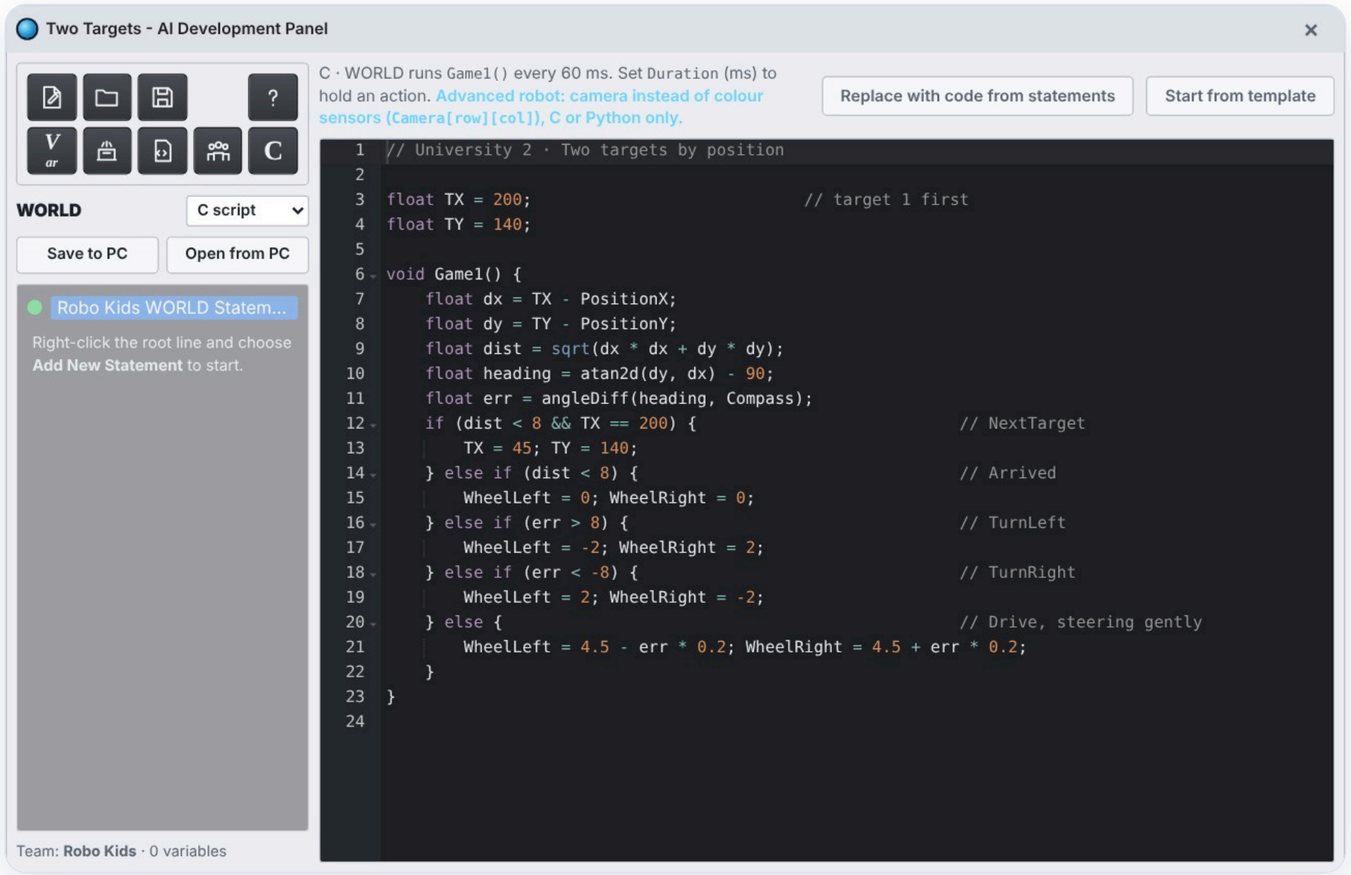
Same three zones: left, right, drive.

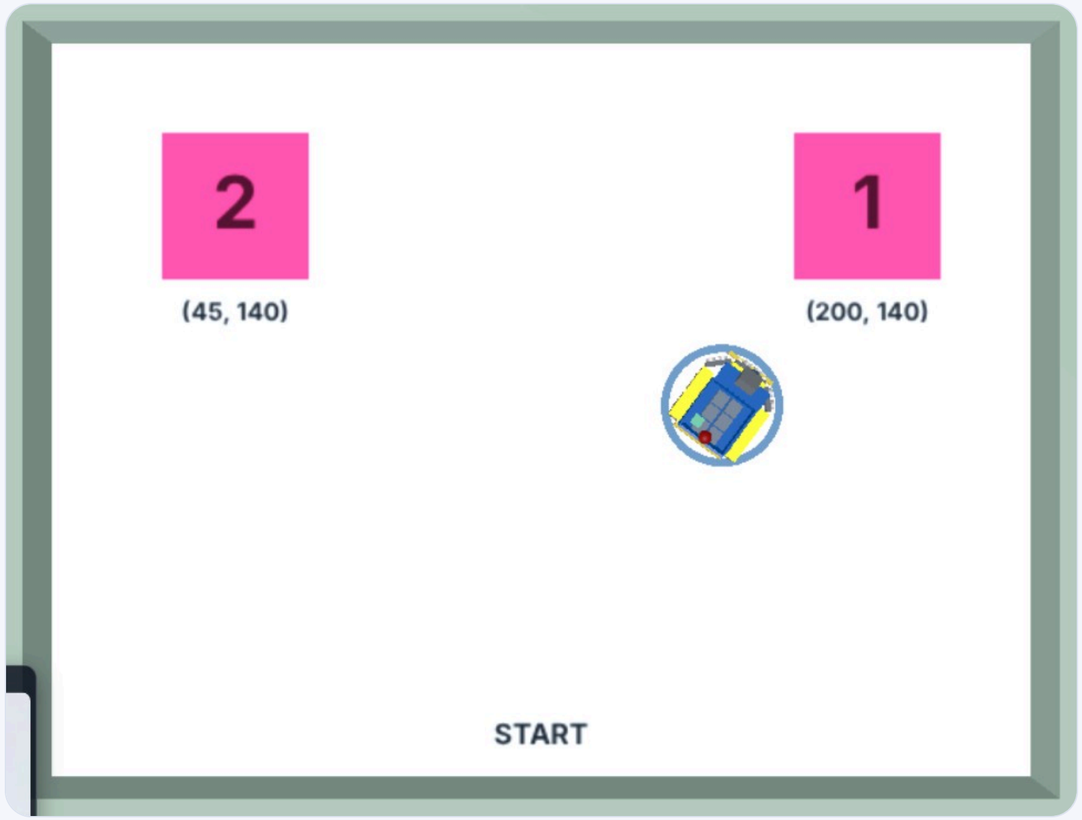
Same gains: **2** to turn, **4.5 ∓ 0.2 × err** to drive.

STEP 5 · BUILD AND LOAD

The whole program, built

1. Category **Advanced**, Learning, Session 13.
2. WORLD menu: **C script**.
3. Name it **Two Targets**.
4. **Build**, then **Load** on BLUE.





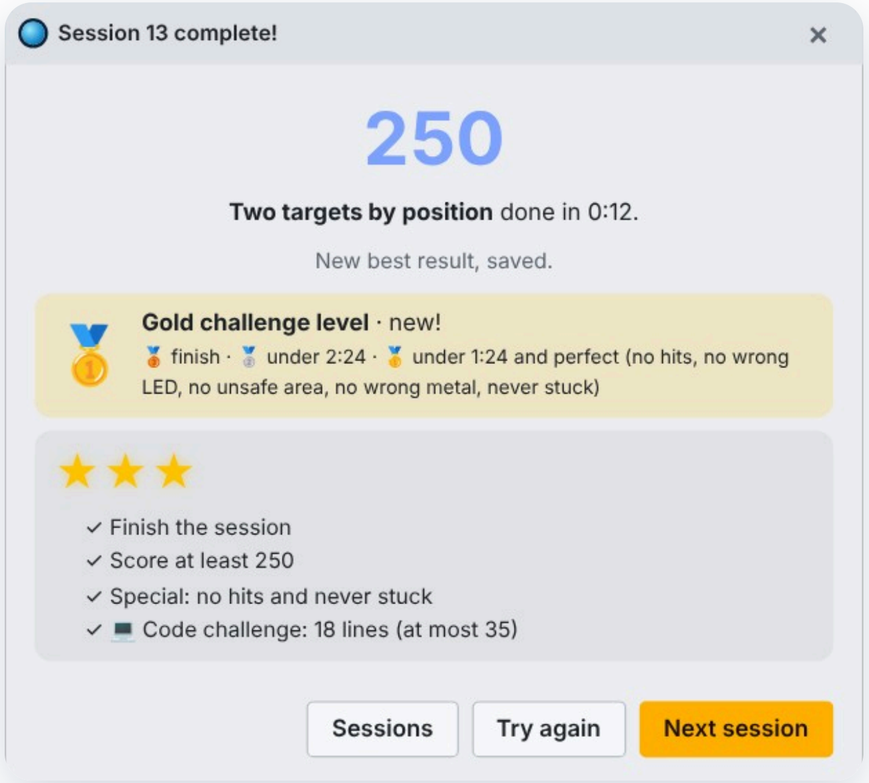
STEP 6 · RUN · 3.7 S

Leg 1: straight at pad 1

At **3.7 s**: (174, 105), Compass **324**, score 0.

A short right turn at START, then straight up the diagonal.

On the pad: **+100**, the switch runs, and the robot turns west.



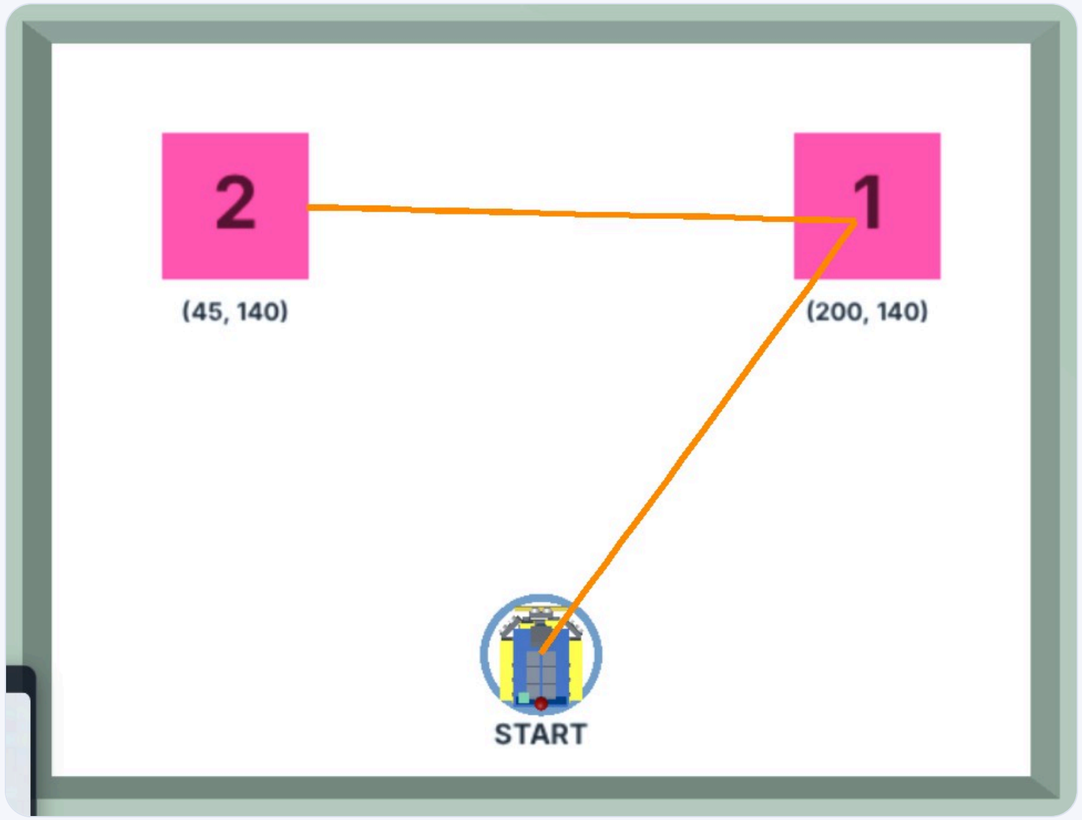
RESULT · TESTED

250 points in 0:12 · Gold

Medals · Silver: under 2:24 · Gold: under 1:24 and perfect.

Stars · finish · score 250 · no hits and never stuck.

Code challenge · at most 35 lines. We used 18.



THE PATH · RECORDED

Two straight legs

Diagonal to pad 1, a turn on the pad, straight west to pad 2.

Change the target, keep the maths: **any number of waypoints**.

YOUR TURN

Three or more targets with MyState

```
if (dist < 8 && MyState == 0) {     // target 1 reached
    TX = 45; TY = 140; MyState = 1;
} else if (dist < 8 && MyState == 1) { // target 2 reached
    TX = 120; TY = 90; MyState = 2; // a third target
}
```

Use **MyState** as a leg counter: 0, 1, 2 ...

Each switch sets the next TX, TY and moves MyState on.

Try a third point, for example (120, 90). It scores nothing here, but it is the pattern for every longer mission.

DID NOT WORK?

Check first, then tune carefully

What you see	What to do
Reaches pad 1, then stops	The switch must come **before** the stop in the if chain
Pad 2 gives no points	Pad 1 did not count: switch later (smaller distance)
Circles around pad 1	Stop or switch distance too small for your speed: try 10
Goes back to pad 1	TX and TY must be written **outside** Game1()
Build error with a line number	Check semicolons, { } and float declarations

NEXT TIME

University 3 · Gold hunt

The camera: what it is, how to read its pixels and colours, and a robot that **finds gold** by itself.