Optimization in Transport and Logistics – RaceTrack Construction Heuristic

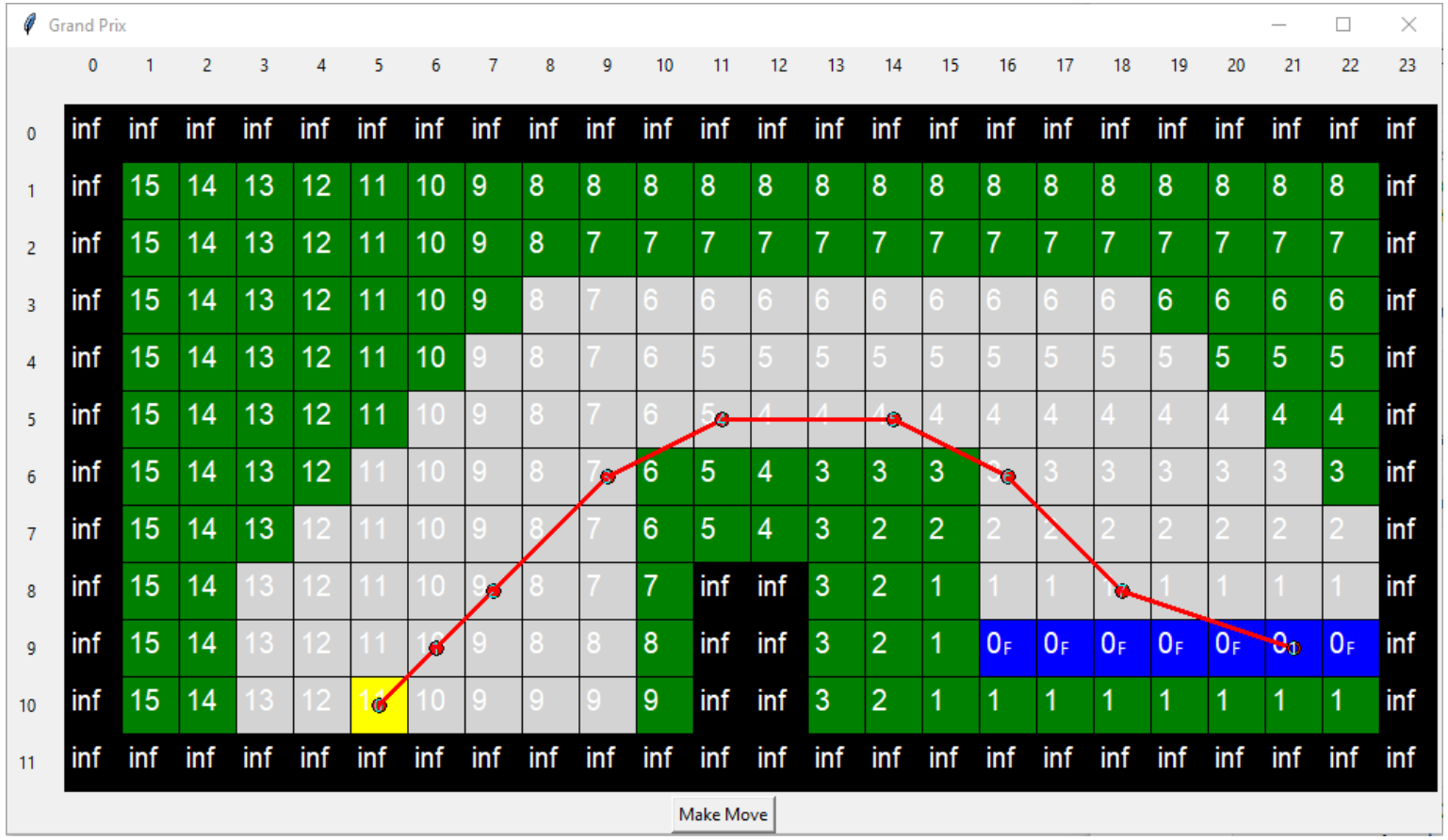
Janick Böhm

General Approach

- Link to code – Python 3.12.0
- Cost heuristic (position, inertia)
  - Distance to finish ($O(rows \cdot cols)$)
  - Distance from nearest object ($O(rows \cdot cols)$)
  - Current inertia
- Selection of next step based on best path ‘max_depth’ steps ahead
  - $O(9^k)$
  - Pruning of branches which do not improve current position → $\sim O(3^k)$
  - For branches with same minimum cost, additional criteria are considered:
    - Branch with lower depth selected
    - Branch with greater inertia selected
- Adjustment of ‘max_depth’ based on inertia
  - For $\text{inertia} > \frac{3}{4}\ \max \text{inertia}$ → current max depth = max depth + 1

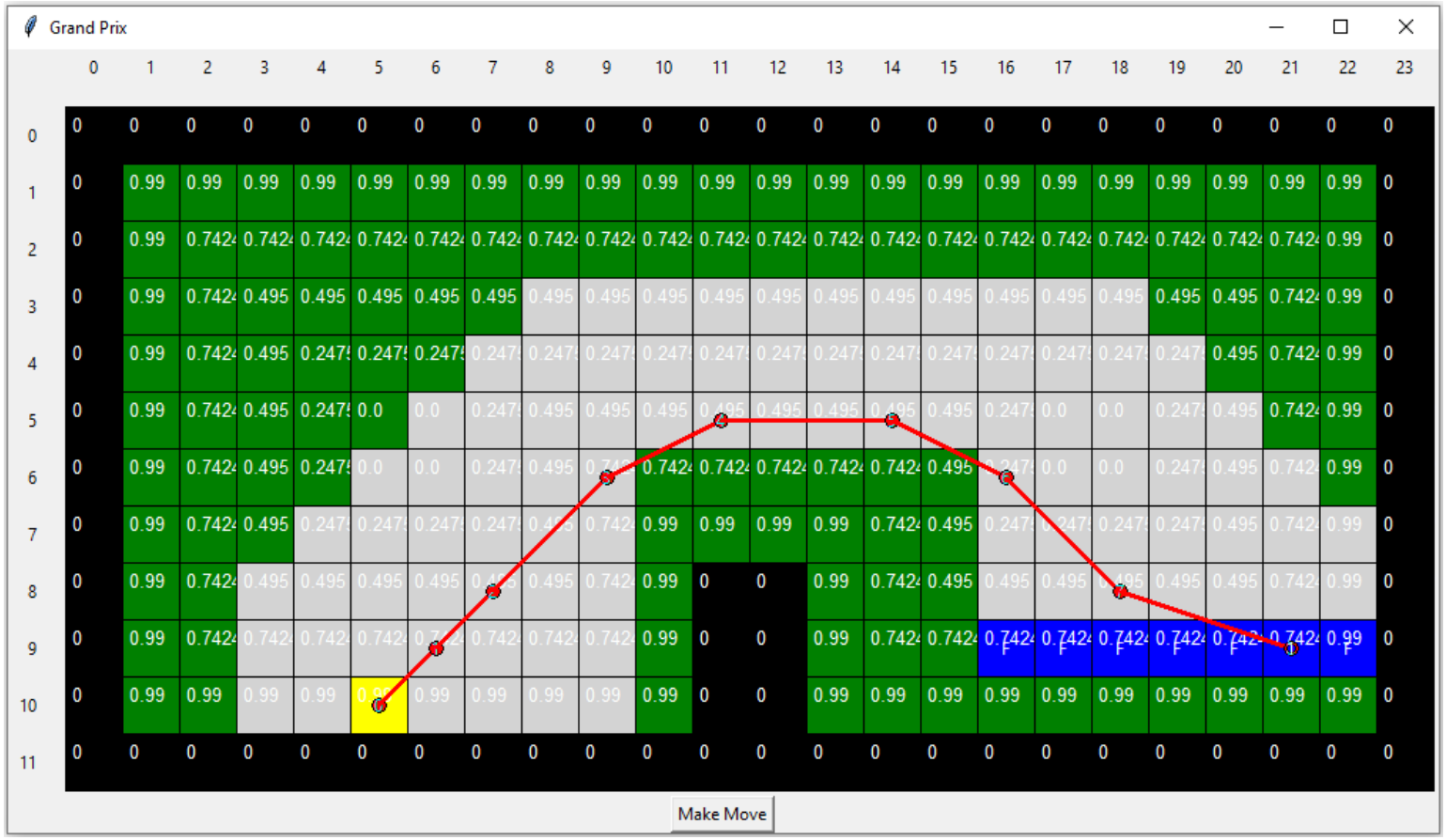
Cost heuristic – Distance to finish

- $O(rows \cdot cols)$
- Encourages movement towards finish
- Admissible, simple to calculate, good baseline
- Special case to handle, diagonal cells besides objects



Cost heuristic – Distance from walls

- $O(rows \cdot cols)$
- Encourages car to remain in the “middle”
  - More maneuverability
- Linearly distributed between 0 to 0.99



Cost heuristic – Inertia

$$f(x) = \frac{1}{1 + e^{-k(x - x_0)}}$$

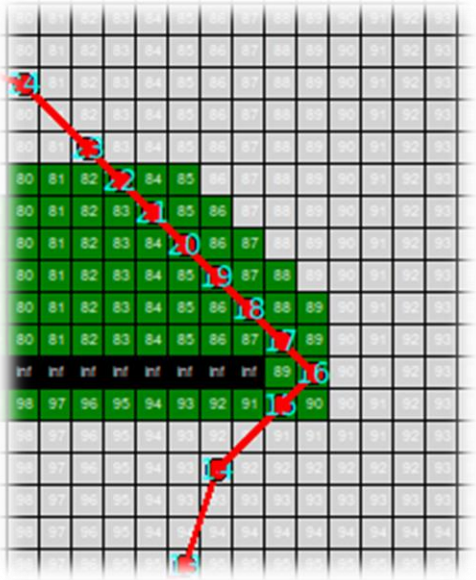
- Encourages car to pursue states with worse position cost, but better inertia
- Logistic function

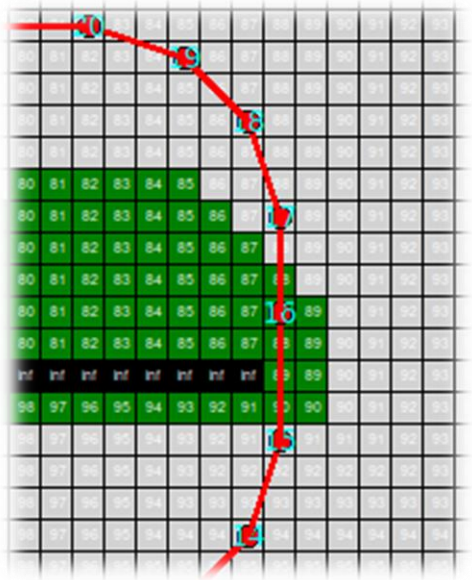
$$x = \max(abs(inertia))$$

$$k = speed\ importance\ (\text{param})$$

$$x_0 = \frac{\max speed\ (param)}{2}$$

Example of path with low speed importance (large k)





Example of path with high speed importance (small k)

Complete heuristic function

$$h(pos.inertia) = distanceToFinish(pos) + distanceToObject(pos) - f(\max(abs(inertia)))$$

All parameters

- max_depth (Maximum search depth when evaluating future positions)
- strategy (Heuristic approach)
- max_speed (Maximum allowed component of inertia)
- speed_importance (Steepness of logistic function)
- always_moving (Whether to allow car to stop (to reset inertia vector))

Results

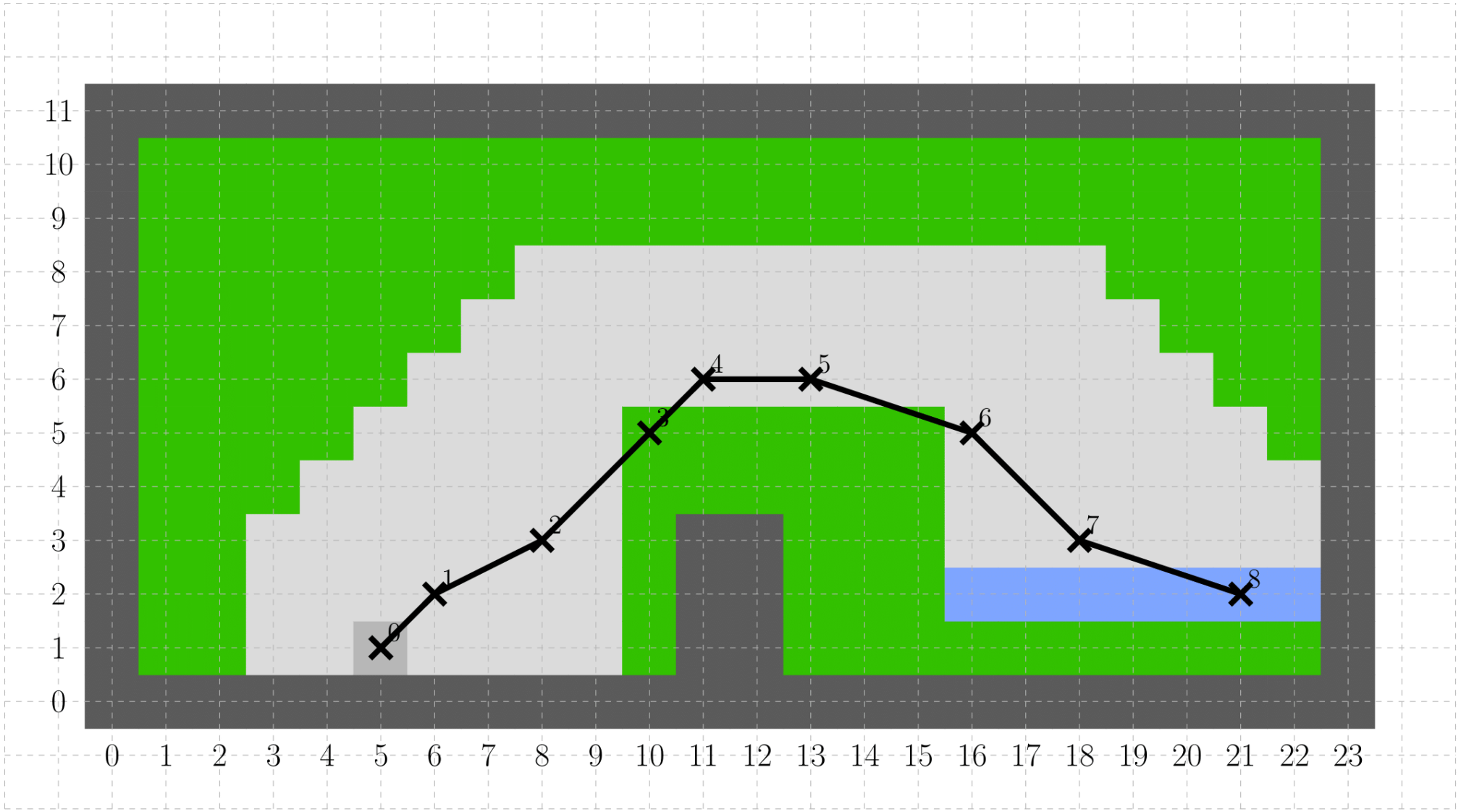
Track no.	Max_depth	Strategy	Max_speed	Speed_importance	Steps	Runtime [s] (average, 5 runs)
1	3	F	5	1	8	0.06
2	4	FO	5	1	14	0.30
3	4	FOS	8	100	42	8.89
4	5	FOS	8	1	41	31.50
5	5	FOS	7	1	91	25.13
6	5	F	7	1	91	24.92
7	4	FOS	8	1	35	5.44
8	5	F	8	1	36	10.52
9	4	FOS	8	1	35	5.02
10	3	FOS	8	1	26	2.73

Specs:
Intel Core i7-1065G7 @ 1.30GHz [8 336 cpubenchmark.net]
16 GB RAM

Strategy Legend:
F – distance to finish
O – distance from object
S – inertia (speed) importance

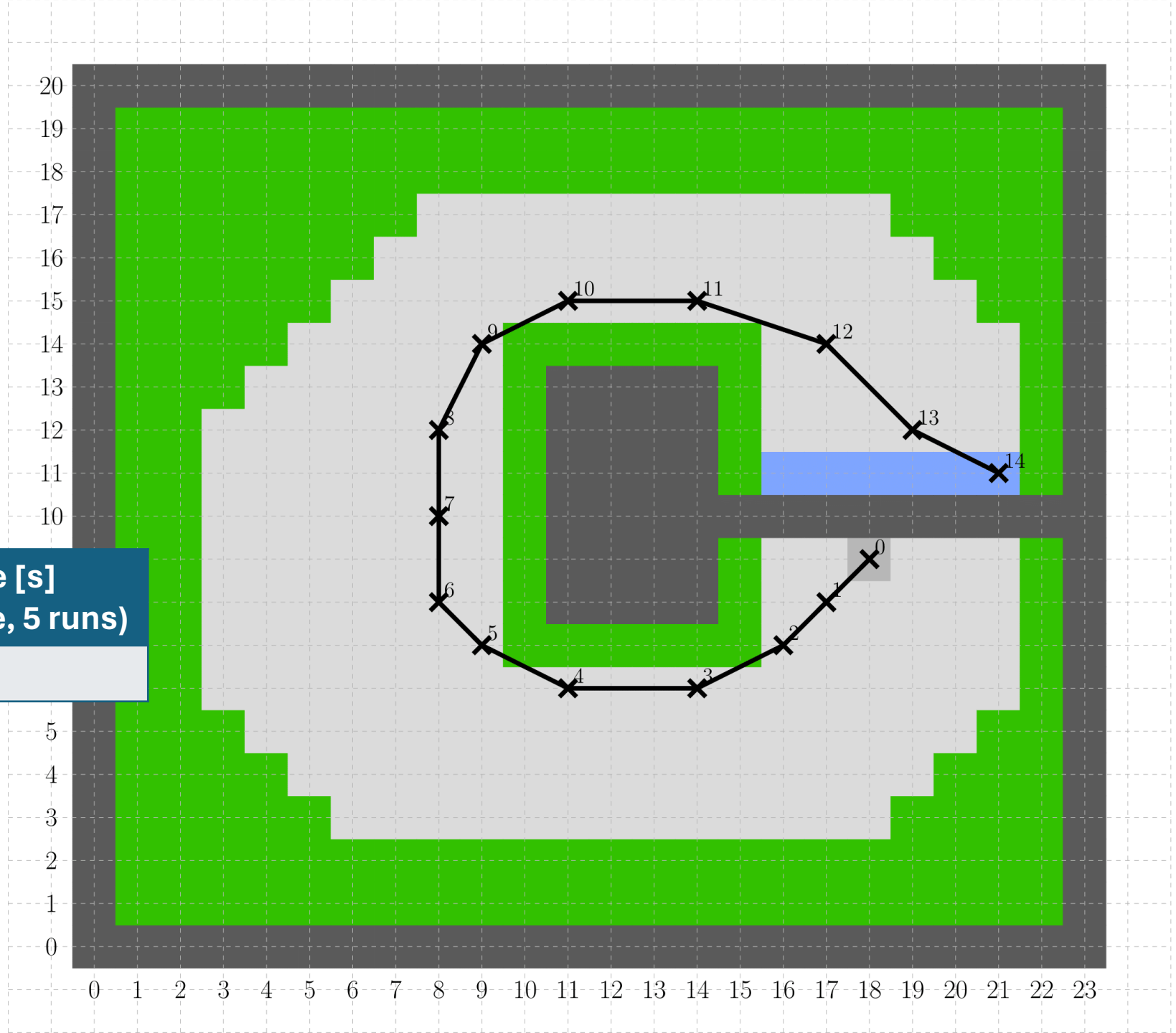
Track 1

Steps	Runtime [s] (average, 5 runs)
8	0.06



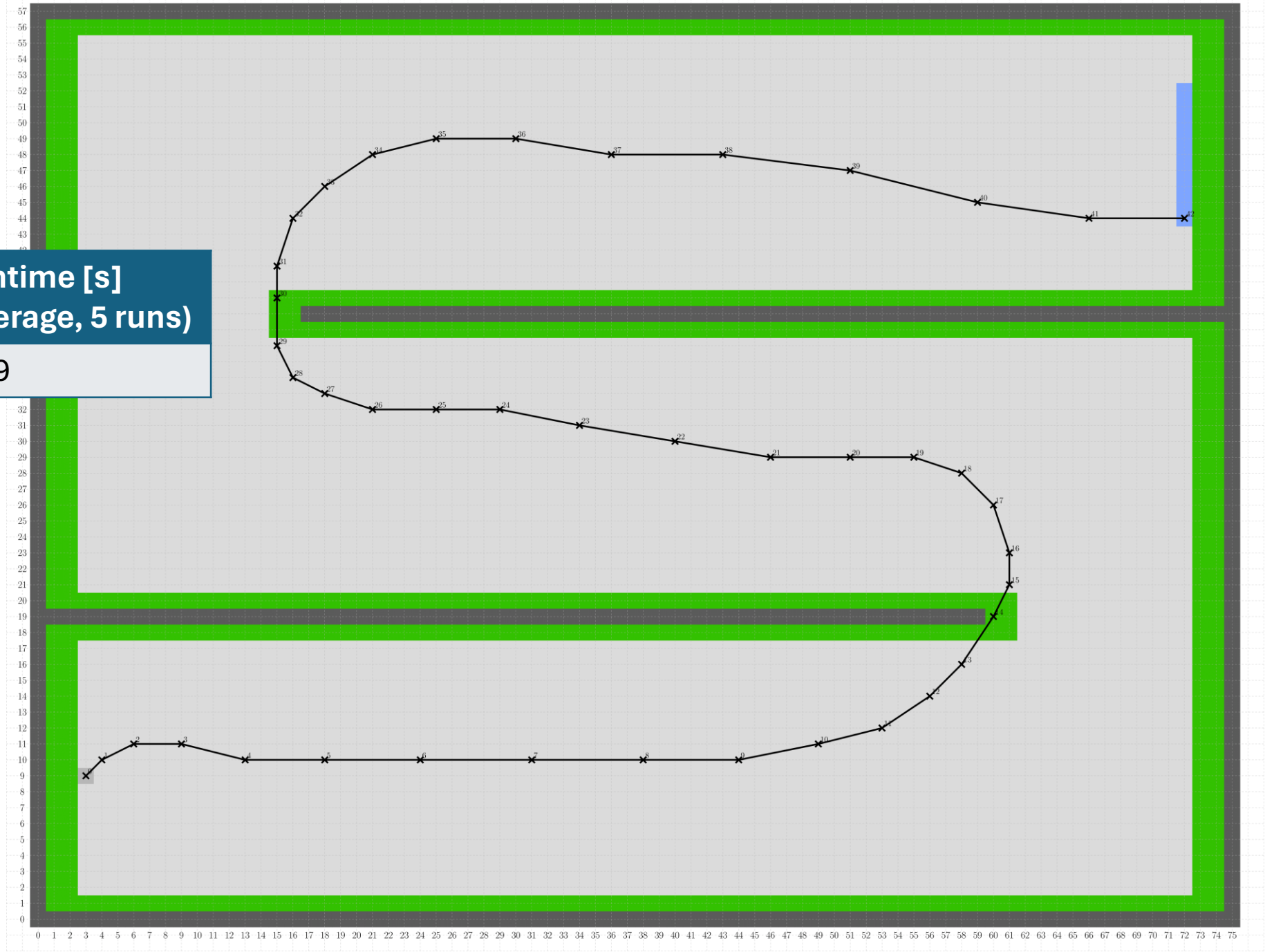
Track 2

Steps	Runtime [s] (average, 5 runs)
14	0.30



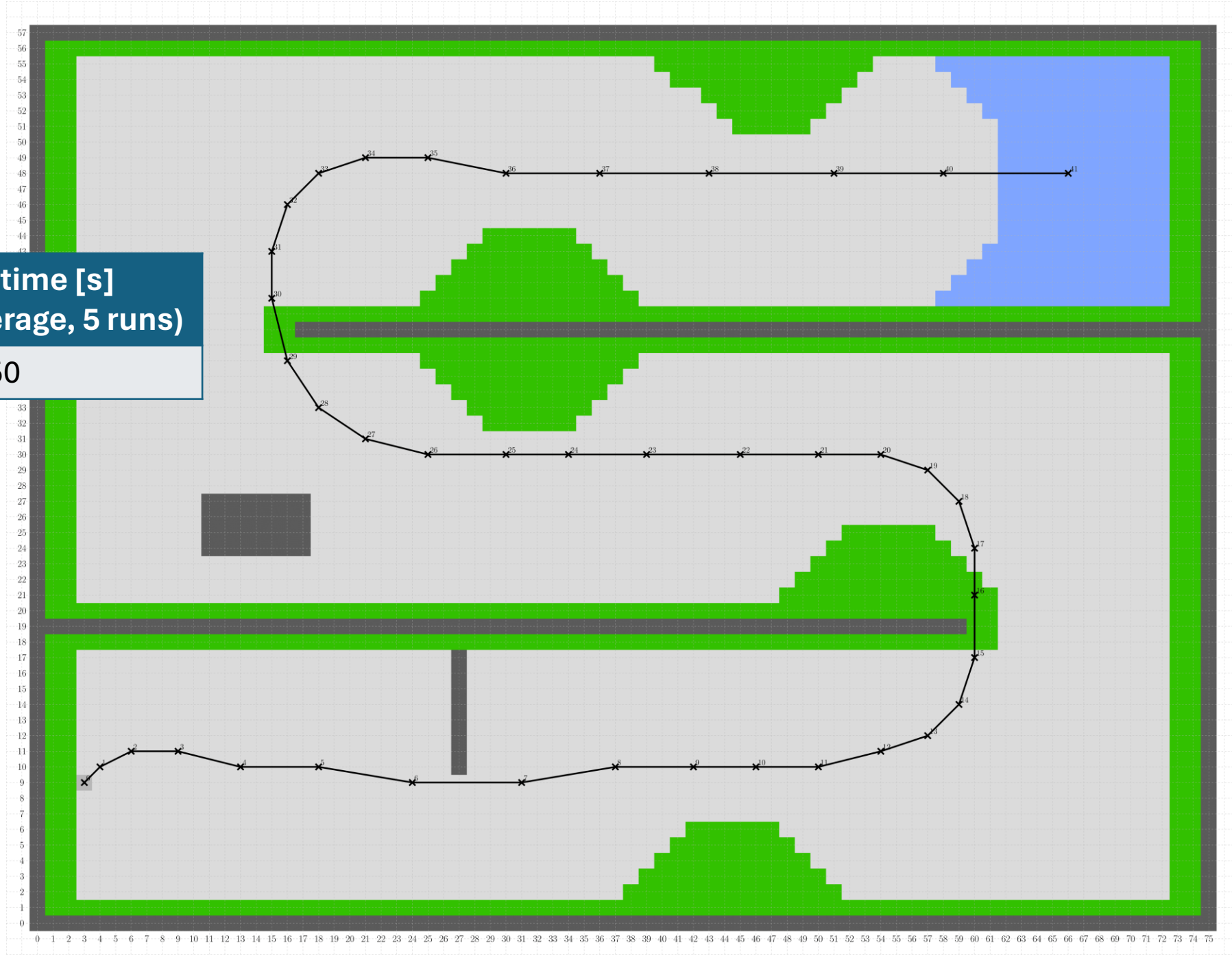
Steps	Runtime [s] (average, 5 runs)
42	8.89

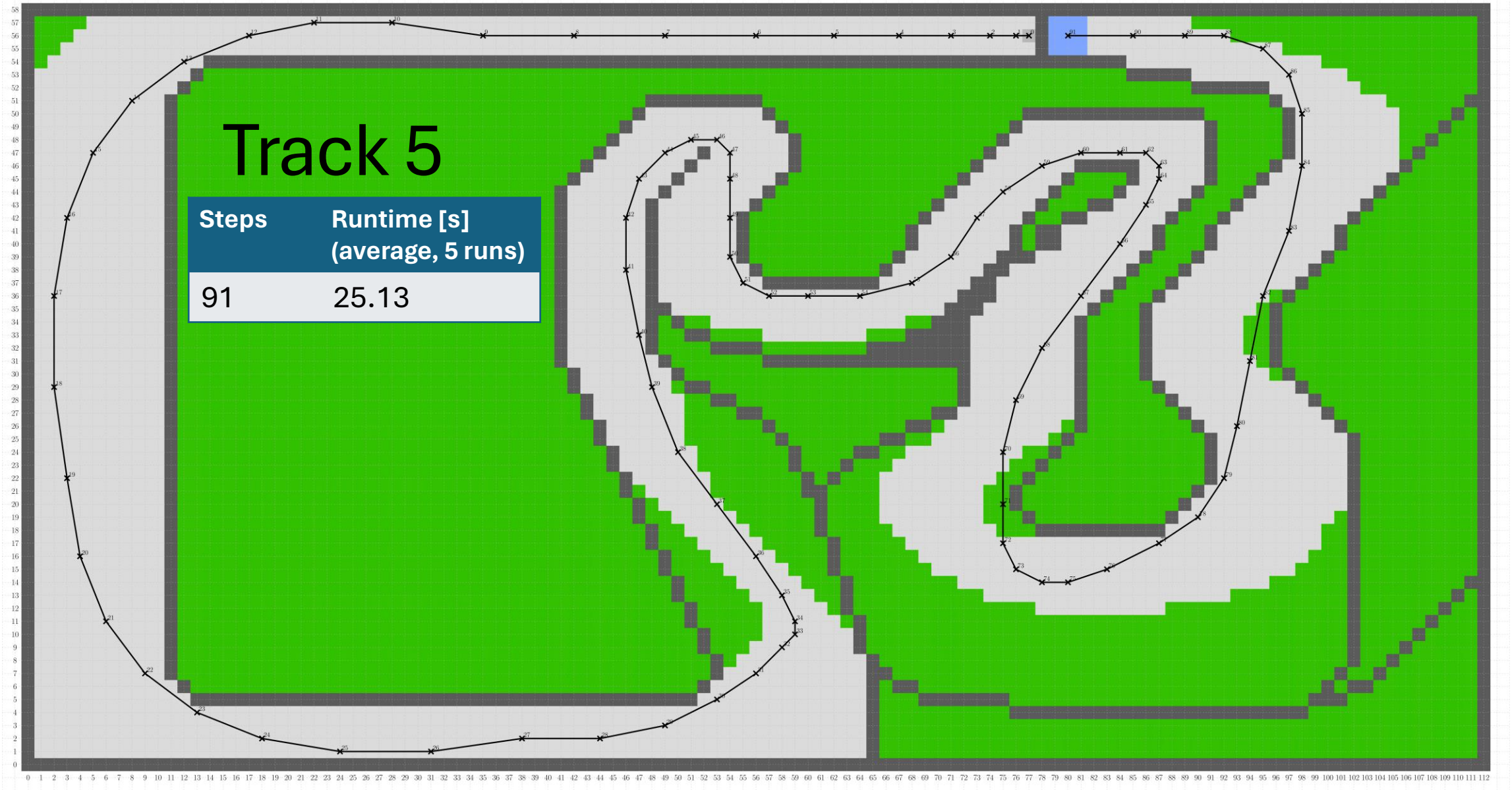
Track 3

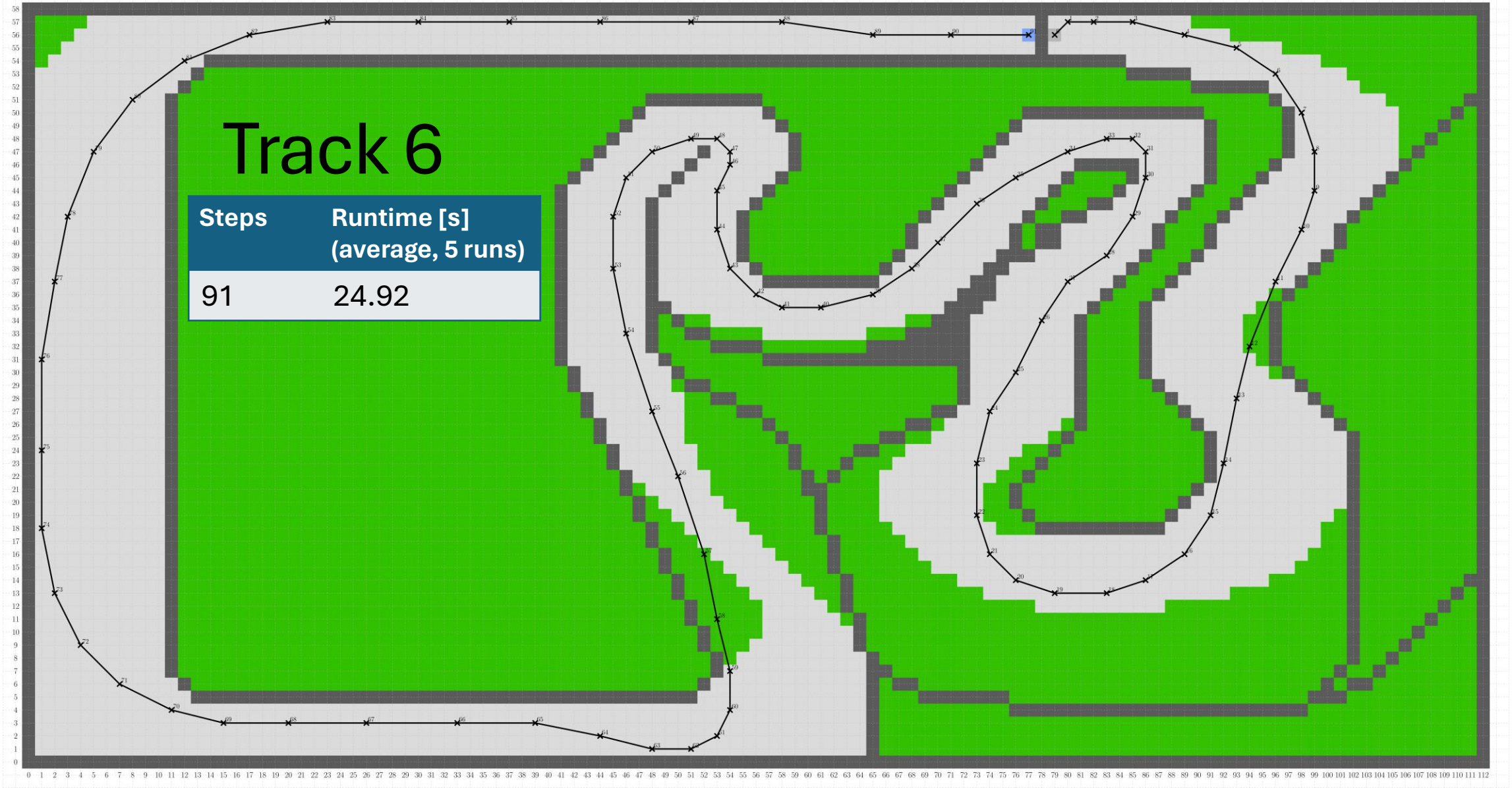


Steps	Runtime [s] (average, 5 runs)
41	31.50

Track 4

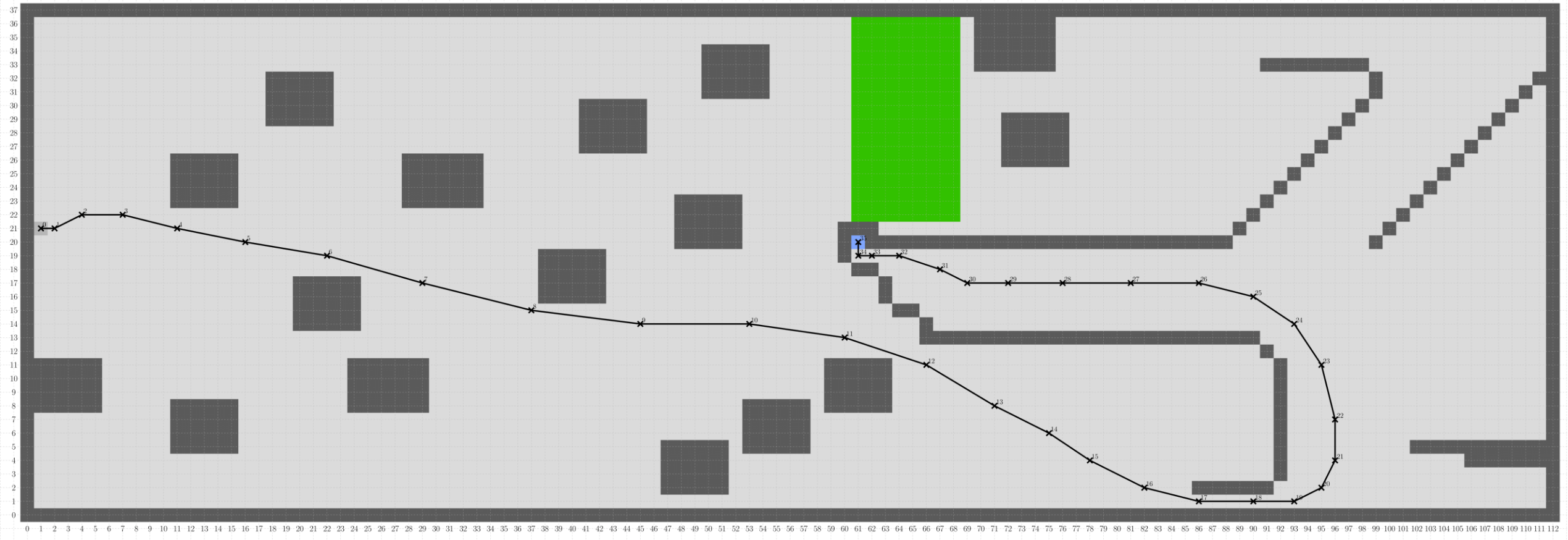



Track 5
Steps
Runtime [s] (average, 5 runs)
91
25.13


Track 6
Steps
Runtime [s] (average, 5 runs)
91
24.92

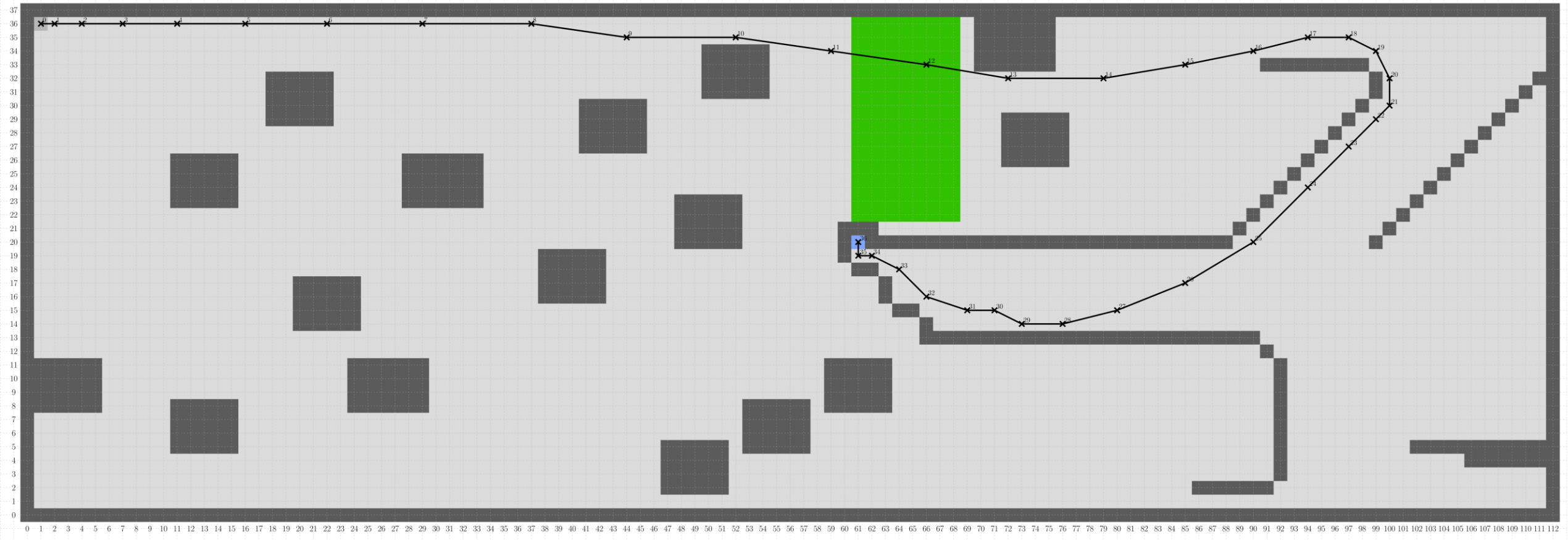
Track 7

Steps	Runtime [s] (average, 5 runs)
35	5.44



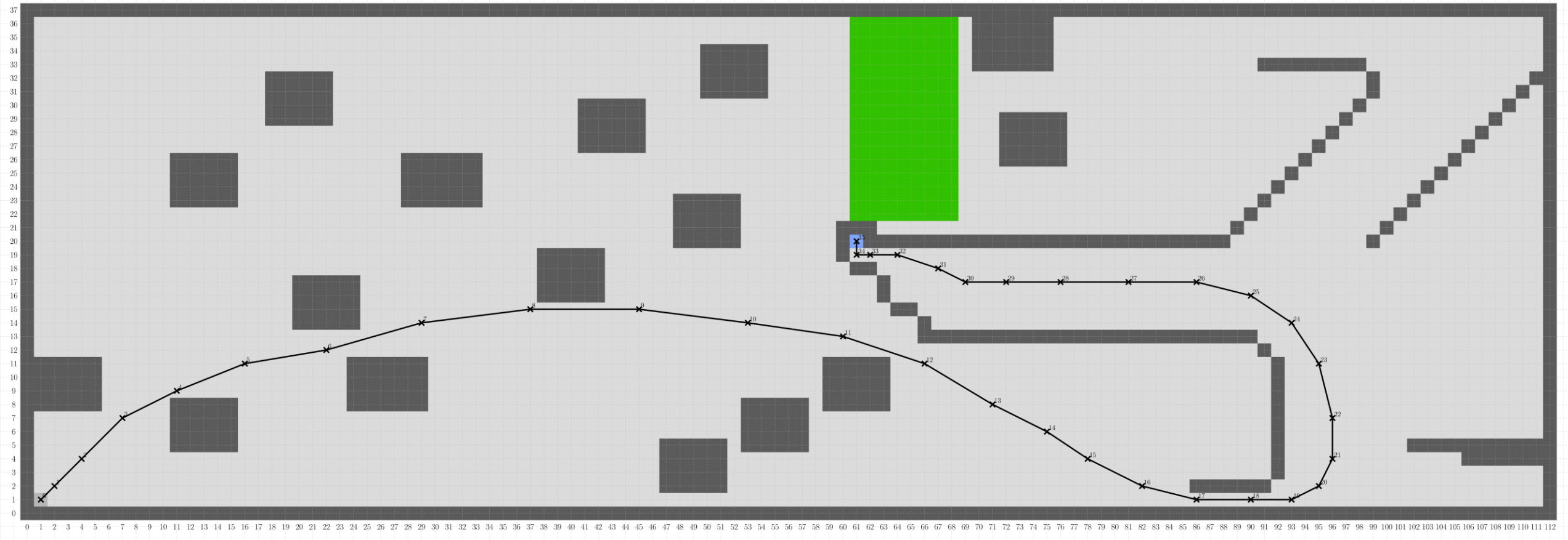
Track 8

Steps	Runtime [s] (average, 5 runs)
36	10.52



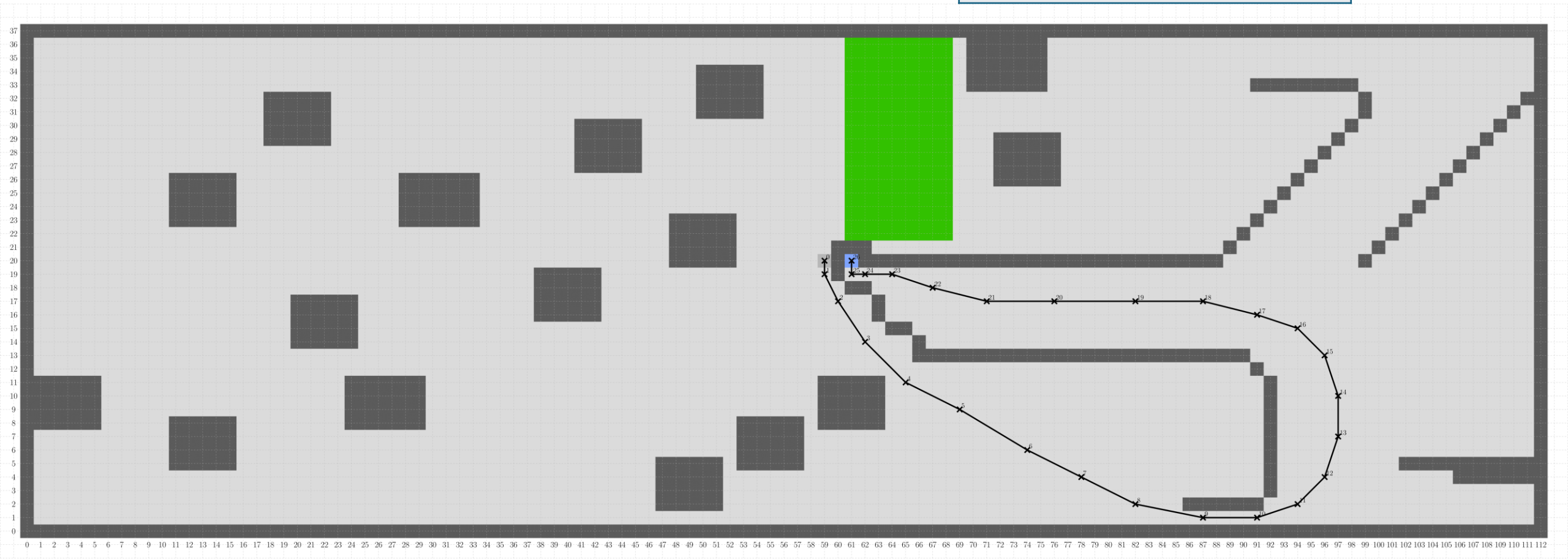
Track 9

Steps	Runtime [s] (average, 5 runs)
35	5.02



Track 10

Steps	Runtime [s] (average, 5 runs)
26	2.73



References

- https://playtechs.blogspot.com/2007/03/raytracing-on-grid.html (validation of path line overlapping with objects)