RBE 550 — MOTION PLANNING

Assignment 5: Wildfire

Liam Jennings

May 2026

Contents

0.1 Introduction

This project implements a competitive simulation between two autonomous agents in a wildfire scenario. The 'Wumpus', a grid-based arsonist, navigates the field igniting obstacles, while a Firetruck pursues and extinguishes them. The two agents use fundamentally different planning approaches: the Wumpus uses a discrete A* planner on a grid, while the firetruck uses a Probabilistic RoadMap (PRM) [1] with Reeds-Shepp [2] local connections to navigate under Ackermann kinematic constraints.

The simulation runs for 3600 simulated seconds (or until no further fire activity is possible) over five randomised obstacle fields per scenario, and a winner is declared by cumulative score.

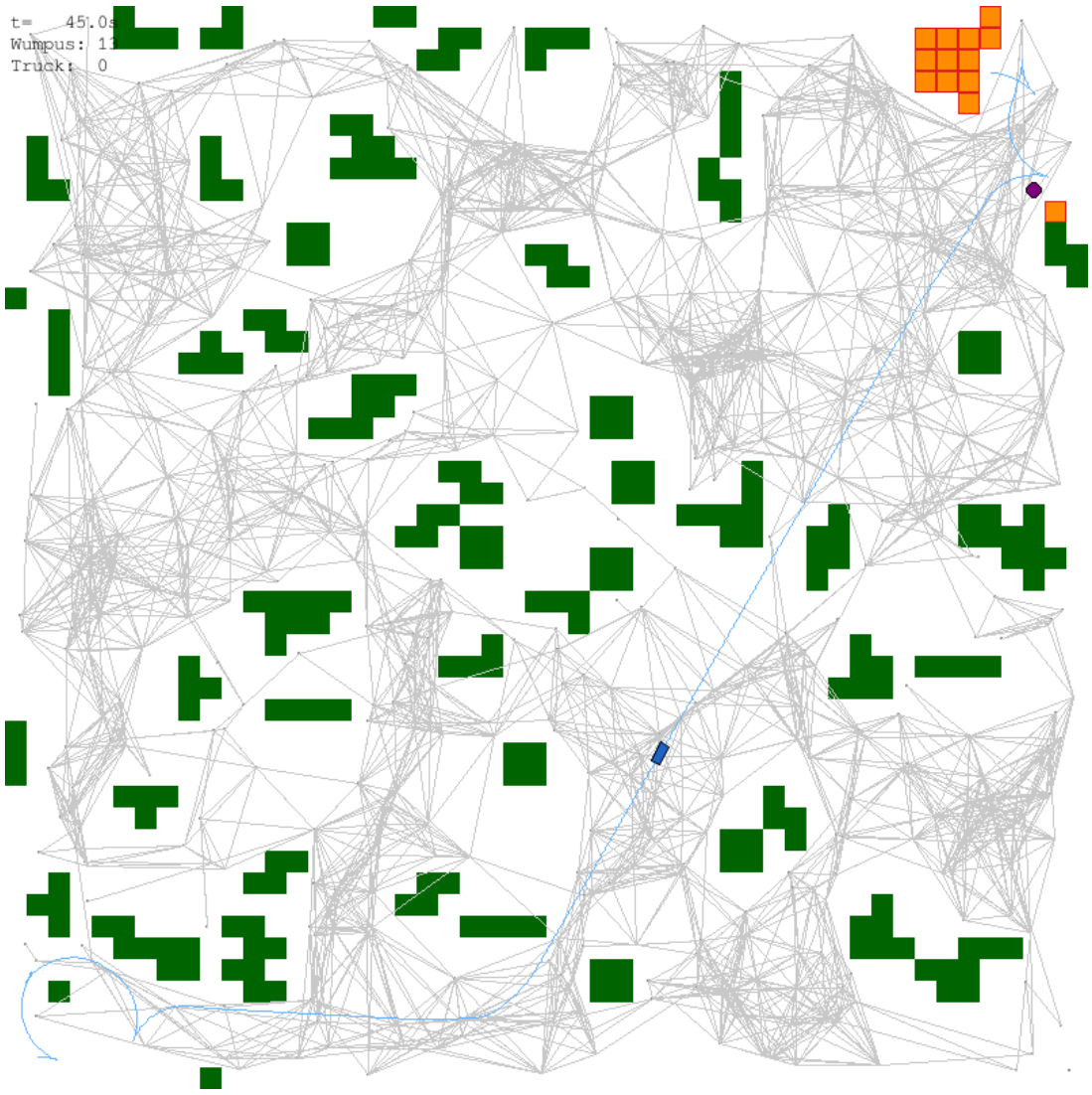

Figure 1: Screenshot of the simulation. Orange/red cells are burning obstacles; the firetruck (blue) navigates toward the nearest fire while the Wumpus (red) traverses the grid igniting new ones. The PRM graph is displayed in the background.

See Appendix A for installation and usage instructions. Appendix B lists all notable external dependencies. Appendix C contains assumptions made based on ambiguities in the assignment description.

0.2 Development Approach

Several components carried over directly from the valet assignment: the Ackermann kinematic model, Reeds-Shepp trajectory generation, and the two-phase collision checker. This left the PRM planner, fire mechanics, and both agents as the primary new work. Neither agent uses a sophisticated high-level strategy; the focus is on the planners rather than game-theoretic play. The PRM required the most iteration: directed-edge connectivity issues and goal-pose escapability bugs were the main obstacles.

0.3 AI/LLM Usage Disclosure

This project was developed with assistance from Claude. Usage included pair programming, debugging, refactoring, and research. The overall structure, implementation decisions, and algorithm correctness are the author's own.

1 Approach

1.1 Environment

1.1.1 Field and Obstacles

The environment is a flat 250 m × 250 m square field discretised into 5 m grid cells (50 × 50 cells). Obstacles are generated by randomly placing and rotating tetrominoes until 10% of cells are occupied. The starting corners are cleared: the firetruck begins in the bottom-left, the Wumpus in the top-right.

Each obstacle cell progresses through a state machine:

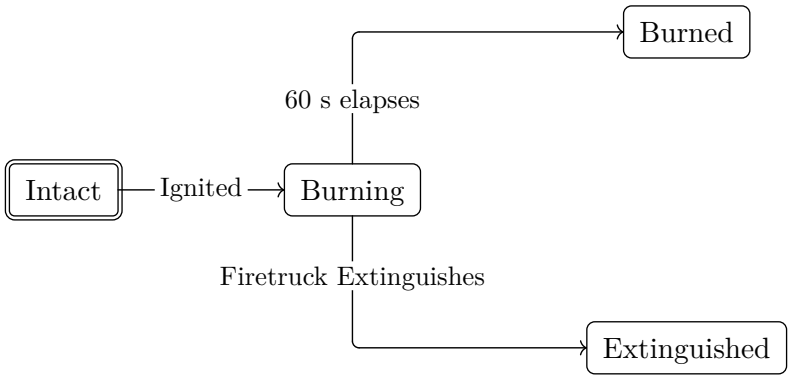

Figure 2: Obstacle state machine. Neither terminal state can be reignited.

Burned and extinguished obstacles cannot be reignited. After 10 seconds of burning, a Wumpus-ignited obstacle spreads fire to all intact obstacles within a 30 m radius; obstacles ignited by spread creep to their 8-connected grid neighbors after the same delay. Each burning obstacle burns out after 60 seconds if not extinguished.

1.1.2 Collision Detection

Collision checking for the firetruck footprint uses a two-phase approach: a broad-phase AABB filter against the occupancy grid, followed by an exact narrow-phase intersection test via a Shapely `STRtree`. A heading cache of pre-rotated footprints (one per 5°) eliminates per-query rotation cost.

```
function IS_VALID_STATE(footprint):
    compute translated AABB
    if AABB lies outside field boundary: return False
    if no obstacle cell under AABB:      return True   // fast path
    translate footprint geometry
    if footprint intersects any obstacle: return False
    return True
```

Listing 1: Firetruck collision check pseudocode.

1.2 Wumpus (Discrete Planner)

The Wumpus moves one grid cell per move interval and can ignite any intact obstacle adjacent to its current cell. It perceives the firetruck's position but does not interact with it physically.

1.2.1 Target Selection

At each replanning step the Wumpus selects the intact obstacle that maximises a score trading off two objectives: obstacles far from the firetruck are preferred (to give fire time to spread before the truck arrives), and obstacles close to the Wumpus incur a smaller travel penalty:

$$\text{score}(o) = d(p_{\text{truck}}, o) - 0.5 \cdot d\big(p_{\text{wumpus}}, o\big) \tag{1}$$

where d is Euclidean world-space distance. When the firetruck position is unavailable, the nearest intact obstacle is chosen instead. Once a target is selected, the closest free grid cell adjacent to it is chosen as the navigation goal.

1.2.2 A* Path Planning

The Wumpus's navigation uses grid A* [3, Chapter 2] on the 50×50 occupancy grid with 8-connected movement. The heuristic is octile distance:

$$h(a, b) = \max(\Delta r, \Delta c) + \left(\sqrt{2} - 1\right) \cdot \min(\Delta r, \Delta c) \tag{2}$$

which is well suited for 8-connected grids. Move costs follow the same heuristic. The path is replanned if the target has already been lit by fire spread, or whenever the current path is exhausted.

```
function UPDATE(env, dt, current_time, truck_pos):
    if move timer has not elapsed: return
    reset move timer

    if needs new target:
        replan(env, truck_pos); return

    if adjacent to target:
        ignite target
        replan(env, truck_pos); return

    advance one cell along path, or replan if path exhausted

function REPLAN(env, truck_pos):
    target ← SELECT_TARGET(env, truck_pos)
    if no target: clear state; return
    goal ← closest free neighbor of target
    if no free neighbor: clear state; return
    current_path ← GRID_ASTAR(current_cell, goal)
```

Listing 2: Wumpus update and replanning pseudocode.

1.3 Firetruck (Sampling-Based Planner: PRM)

The firetruck is a Mercedes Unimog modelled as a car-like robot with Ackermann steering: width 2.2 m, length 4.9 m, wheelbase 3.0 m, minimum turning radius 13.0 m, maximum velocity 10 m/s. It has omniscient perception of the environment and extinguishes a burning obstacle by stopping within 10 m of it for 5 seconds.

1.3.1 Roadmap Construction

A Probabilistic RoadMap [1], [3, Chapter 5] is built once at simulation start. 500 collision-free poses (x, y, θ) are sampled by rejection: candidates are drawn uniformly and checked for collision with the environment using the firetruck footprint. Each node is connected to its 15

nearest spatial neighbors (by Euclidean position) with directed edges. A KDTree (B.3) is used to spatially index every sampled pose via their (x, y) position, greatly speeding up these queries.

An edge is accepted when:

1. The Reeds-Shepp path length between the two poses does not exceed 50 m.
2. An obstacle-ignorant Reeds-Shepp trajectory at 1 m resolution can be generated.
3. Every sampled state along that trajectory is collision-free.

```
function BUILD_ROADMAP(env, truck):
    nodes ← REJECTION_SAMPLE(env, truck, N=1500)
    tree  ← KDTree(node positions)
    seen  ← empty set
    for each node_i in nodes:
        for node_j in K_NEAREST(node_i, k=15):
            if (node_i, node_j) already seen: continue
            mark (node_i, node_j) seen
            if RS_LENGTH(node_i, node_j) > MAX_EDGE: continue
            path ← RS_TRAJECTORY(node_i, node_j, resolution=1 m)
            if every point of path is collision-free:
                add undirected edge (node_i ↔ node_j)
```

Listing 3: PRM construction pseudocode.

1.3.2 Query

When the truck needs to reach a target position, the planner:

1. **Connects the start pose**: finds the nearest roadmap node reachable via a collision-free RS trajectory.
2. **Finds an escapable goal pose**: tries eight candidate headings at the goal (x, y) and commits to the first from which a connection back to the roadmap is possible, ensuring the truck will not get stuck on arrival.
3. **Runs A* on the roadmap graph**: finds a path from the start node to the goal node, using Euclidean distance as the heuristic.
4. **Stitches the full path**: concatenates start $\rightarrow$ roadmap waypoints $\rightarrow$ goal, generating a dense RS trajectory for each segment.
5. **Smooths the path** via probabilistic shortcutting [4]: two random waypoints are selected and, if a direct RS trajectory between them is collision-free, it replaces the segment. After 200 iterations, unnecessary detours are removed.

```
function QUERY(start, goal_xy):
    s_node ← CONNECT_TO_ROADMAP(start_pose)
    if s_node is None: return None

    goal_pose, g_node ← ESCAPABLE_GOAL_POSE(goal_xy)
    if g_node is None: return None

    node_path ← ASTAR(s_node, g_node)
    if node_path is None: return None

    full_path ← STITCH_TRAJECTORIES(start_pose, node_path, goal_pose)
    return SMOOTH(full_path)

function ESCAPABLE_GOAL_POSE(goal_xy):
    for heading in {0, π/4, π/2, ..., 7π/4}:
        node ← CONNECT_TO_ROADMAP((goal_xy, heading))
        if node is not None: return ((goal_xy, heading), node)
    return None
```

Listing 4: PRM query pseudocode.

A known failure mode is the start pose being unreachable from the roadmap: this happens when the starting region is geometrically isolated from the rest of the sampled nodes, so no collision-free RS trajectory can connect the start to any roadmap node. When this occurs the truck cannot plan at all and sits idle for the entire run (Figure 3).

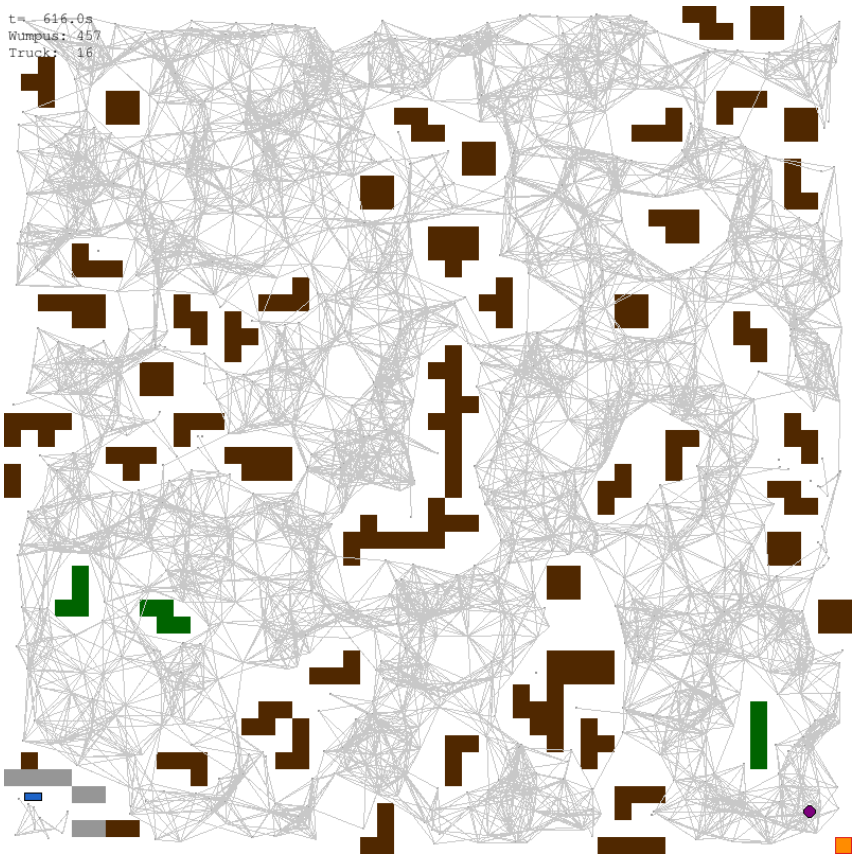

Figure 3: A seed where the firetruck's starting corner is disconnected from the roadmap. The truck has no valid connection to any roadmap node and remains stationary.

1.3.3 Simulation Loop

The truck operates as a rough state machine on each simulation tick:

```
function UPDATE_TRUCK(dt):
    if following a path:
        advance along path by velocity × dt;

    else if within extinguish range of a burning obstacle:
        accumulate extinguish timer
        if timer ≥ 5 s: extinguish obstacle


    else if target-check interval elapsed:
        targets ← nearest burning obstacles (up to 5)
        for each target (up to 3 attempts):
            path ← QUERY(current_state, target)
            if path found: set path;
```

Listing 5: Firetruck simulation loop pseudocode.

The truck only attempts to extinguish when it has no active path — i.e., it has arrived at the planned approach point. This prevents the truck from stopping mid-path to extinguish an obstacle it happens to pass near, which would leave it at an unvalidated pose. Unvalidated poses have no guarantee of being able to reach back onto the PRM graph, so they are rendered motionless.

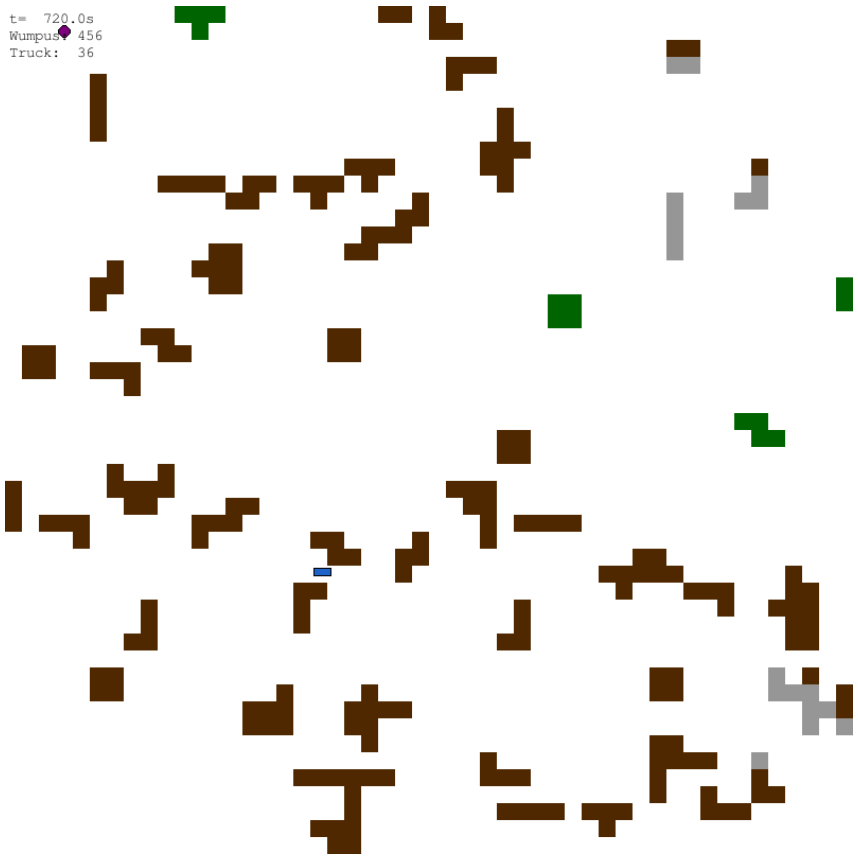

Figure 4: The firetruck stopped at an unvalidated pose, unable to connect back to the roadmap. The truck will remain motionless for the rest of the simulation.

2 Results

Run	Seed	Wumpus	Truck	Winner	W-plan (s)	T-plan (s)
1	3235666703	264	214	Wumpus	0.039	2.809
2	3235666704	246	254	Truck	0.039	3.581
3	3235666705	236	260	Truck	0.036	2.790
4	3235666706	222	260	Truck	0.036	3.356
5	3235666707	276	230	Wumpus	0.034	2.831
Total		**1244**	**1218**	**Truck 3–2**	0.184	15.367

Table 1: Simulation results across five iterations. Planning time is summed over the full run. Champion: Truck (3 wins to 2).

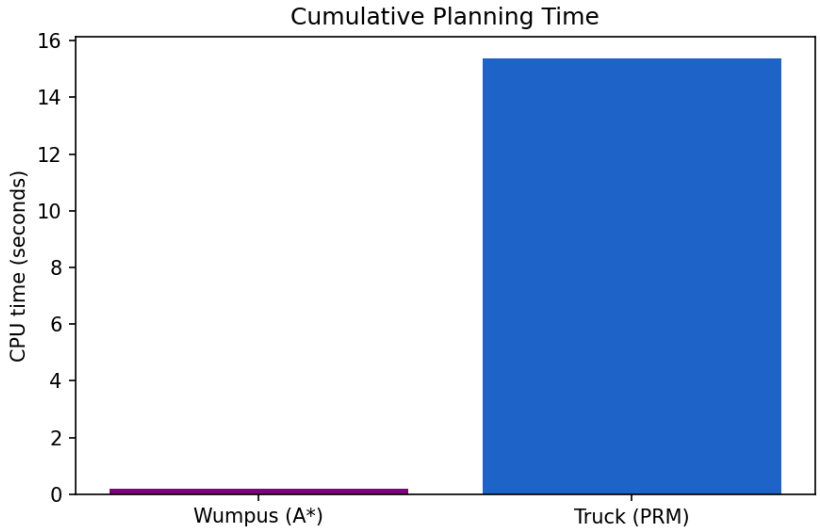

Figure 5: Cumulative planning CPU time per agent summed over all five iterations. Wumpus time covers A* calls only; truck time covers PRM construction plus all query calls.

2.1 Discussion

Neither agent's high-level strategy is particularly sophisticated. The Wumpus uses a greedy scoring function and the truck always targets the nearest fire. Results therefore reflect planner capability more than strategic depth, which is the intent.

Which method is better suited?

Both are well-matched to their constraints. The Wumpus's discrete grid naturally fits its cell-by-cell movement and cheap per-move replanning. The firetruck's continuous Ackermann kinematics and minimum turning radius make a sampling-based planner with kinematically-correct local connections the appropriate choice.

Does one plan more efficient paths?

The Wumpus plans grid-optimal paths by construction (A* with an admissible heuristic). The firetruck's paths are suboptimal at multiple levels: sparse random sampling misses shortcuts, k-nearest connection limits graph density, and probabilistic shortcutting only partially recovers quality. Arrival reliability matters more than path length for the truck.

Advantages of an a priori roadmap?

Building the roadmap once up front means most of the expensive path-checking work occurs before the simulation starts. During the simulation, each query simply connects the start and goal poses to the nearest nodes and runs A*.

Which requires more computational resources?

The truck requires significantly more computation; see Figure 5. Each run spends roughly 1 s building the roadmap and another 1.5-2.5 s on queries over the full run. The Wumpus's A* calls are individually very cheap, and even accumulated many replanning events the total stays under 0.04 s per run.

Best performance accounting for computational resources?

The truck scored 1218 points over 15.4 s of planning (79 pts/s); the Wumpus scored 1244 points over 0.18 s (6900 pts/s). By raw score-per-CPU-second the Wumpus is far more efficient — grid A* is orders of magnitude cheaper than PRM. Of the truck's per-run total (2.8-3.6 s), roughly 1 s is the one-time roadmap build; the remainder is query time accumulated across the full run.

3 Trophy

The firetruck wins the fancy trophy.

Bibliography

[1] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996, doi: 10.1109/70.508439.

[2] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: http://lavalle.pl/planning/

[4] R. Geraerts and M. H. Overmars, "Creating High-quality Paths for Motion Planning," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007, doi: 10.1177/0278364907079280.

[5] G. Liemann, "pyReedsShepp." Feb. 2020.

[6] G.-H. Liu, "pyReedsShepp." Aug. 2016.

Appendices

Appendix A: Setup

A.1 Installation

The project requires Python 3.12 and uses uv for dependency management. The `reeds_shepp` dependency is vendored under `deps/pyReedsShepp` and built automatically on install.

```
git clone <repo-url>
cd wildfire
uv sync
```

See `README.md` for platform-specific notes on building the vendored C extension.

A.2 Usage

```
uv run runsim [options]
```

Flag	Effect
`--no-render`	Run headless (no pygame window). Faster for batch runs. (default: false)
`--iterations N`	Number of simulation runs (default: 5).
`--seed N`	Base RNG seed; run i uses seed $+i$ (default: random integer between 0 and 2^{32}).
`--record`	Save the first iteration as `recording.mp4`. (default: false)

Keyboard shortcuts during a rendered run:

Key	Effect
`+` / `=`	Double simulation speed (up to 200×).
`-`	Halve simulation speed.
`s`	Save a screenshot.

Appendix B: Dependencies

B.1 Pygame

Real-time 2D rendering, keyboard event handling, and the simulation clock.

B.2 Shapely

Polygon geometry library used to represent the firetruck footprint and query it against obstacle geometry via an `STRtree` spatial index.

B.3 SciPy (`KDTree`)

k-nearest-neighbour queries for PRM node lookup during roadmap construction and query-time roadmap connection.

B.4 NumPy

Occupancy grid representation, seeded random sampling via `np.random.Generator`, and array operations throughout.

B.5 reeds_shepp (pyReedsShepp)

Python bindings to a C++ implementation of Reeds-Shepp path length and trajectory sampling. Vendored at `deps/pyReedsShepp`; a fork of [5], which is itself a fork of the original library [6], updated for Python 3.12 compatibility.

B.6 imageio / imageio-ffmpeg

MP4 recording of simulation runs when `--record` is passed; imageio-ffmpeg supplies the FFmpeg backend.

Appendix C: Specification Assumptions

Several points in the assignment specification were ambiguous or silent. The decisions below were made to produce a fair and well-defined simulation.

Question	Decision
Fire spread: chain reaction or not? The spec says a burning obstacle spreads fire after 10 s, but does not say whether spread-to obstacles also spread.	Unlimited chain reaction was tested and gave the Wumpus an overwhelming advantage. Lighting one obstacle auto-won the simulation via cascade. The rule was restricted: Wumpus-ignited obstacles spread to all intact obstacles within 30 m (the "explosion"). Obstacles that catch fire via spread creep only to their immediate 8-connected grid neighbors, limiting but not eliminating chain behaviour.
Wumpus scoring for extinguished obstacles. The spec says "+1 per obstacle ignited" but does not say whether credit is granted if the truck later extinguishes the obstacle.	No credit awarded for extinguished obstacles. "+1 per obstacle ignited" is interpreted as obstacles still burning or fully burned at scoring time — awarding credit for extinguished obstacles would give the Wumpus too large an advantage, since the truck's effort would score points for both players.
Multiple obstacles within firetruck's extinguish range simultaneously. The spec says the truck extinguishes by stopping within 10 m for 5 s, but is ambiguous on what happens when several burning obstacles are in range at once.	The truck extinguishes one obstacle at a time, specifically the closest one in range. The timer resets if the nearest obstacle changes.
Extinguished obstacle state. The spec says "burned obstacles may not be relit" but says nothing about extinguished ones.	Extinguished obstacles cannot be reignited. The Wumpus's ignition and fire-spread code only targets intact obstacles.
Wumpus movement speed. The spec establishes grid-based A* movement but never specifies how many cells per second the Wumpus can move.	Move interval tuned to 2.6 simulated seconds per cell.
Wumpus grid shape and density. The spec allows "any graph shape or density."	The Wumpus grid matches the obstacle grid: 50 × 50 cells at 5 m spacing.
Starting positions. The spec says "opposite sides of the field, either random or fixed."	Fixed positions: the firetruck starts in the bottom-left corner, the Wumpus in the top-right. Both starting regions are cleared of obstacles.

Question	Decision
Steady-state termination. The spec mentions early termination at steady state but does not define it.	Steady state is defined as all obstacles being either burned or extinguished: no intact or burning obstacles remain.
Instantaneous acceleration. The spec notes that instantaneous acceleration is allowed but describes it as "destructive to the transmission."	No acceleration model is implemented. The truck moves at maximum velocity from the first step of each path segment, but its SM-465 transmission breaks down after the fires burn out, and needs to be repaired in the shop.