Shover-World

Milestone 1 Specification & API Documentation

A.I 2025

November 8, 2025

A grid-based reinforcement learning environment for experimenting with discrete control, reward shaping, and environment design.

---

1 Learning Objectives

By completing Milestone 1, you will:

- Design and implement a Gym-compatible environment with a discrete action space.
- Encode states, actions, rewards, and termination conditions clearly and reproducibly.
- Implement stamina-based action costs, one-step default costs, and directional push dynamics.
- Detect and operate on perfect squares of boxes with special actions.
- Build a minimal GUI using Pygame and load maps from text files.
- Provide unit tests and documentation suitable for RL agents (e.g., random, Q-learning, DQN).

2 High-Level Overview

Shover-World is a grid world inspired by Sokoban. Shover was once a great worker in the dock, and his task was to make perfect squares so a robotic lifter could lift these boxes. All of a sudden, he became bored with his task and, due to some difficulties he had in university (he failed his A.I. course!), he became insane and his only goal is to clear the dock no matter the cost! He is also so athletic and can move in no time. A shover navigates a 2D grid and pushes boxes to interact with the world while avoiding barriers and lava. In this milestone you will implement the environment, stamina mechanics, perfect-square detection with two special actions, a Pygame GUI, and file-based map loading. The code base should expose a `ShoverWorldEnv` class and an executable `test.py` that demonstrates random play.

3 Environment Specifications

3.1 Initialization Parameters (`__init__`)

`render_mode`
: `'human'` to enable Pygame rendering; `None` for headless.

`n_rows`, `n_cols`
: Grid size; environment must support any $a \times b$ map.

`max_timestep`
: Maximum steps per episode (default: 400).

`number_of_boxes`
: Number of boxes when generating random maps.

`number_of_barriers`
: Number of barriers when generating random maps.

`number_of_lavas`
: Number of lava cells inside the map when generating random maps.

`initial_stamina`
: Starting stamina (default: 1000).

`initial_force`
: Positive scalar used in stamina cost (see 4).

`unit_force`
: Positive scalar used in stamina cost (see 4).

`perf_sq_initial_age`
: Initial age of perfect squares for automatic dissolution.

`map_path`
: Optional path to a map file to load (see 7).

`seed`
: RNG seed for reproducibility.

3.2 Spaces

Action Space The action space is a tuple `(x, y), z` where `(x, y)` is the position (row, column) of the targeted box on the grid that the shover pushes on, and `z` is the action type from the following discrete options:

z	Meaning
1	Move Up
2	Move Right
3	Move Down
4	Move Left
5	**Barrier Maker** (valid only if a perfect square of size $n \geq 2$ exists everywhere on map regardless of the target position)
6	**Hellify** (valid only if a perfect square of size $n > 2$ exists everywhere on map regardless of the target position)

If the chosen action is not currently available at the specified position, treat as an invalid action (see 3.3 and rewards).

State / Observation Space The observation is the map plus stamina, shover's current position, and shover's current direction. Representations allowed:

- **Dict** space: recommended.
  `{ "grid": Box(int), "agent": Box(int), "stamina": Box(float), "previous_selected_position": Box(int), "previous_action": Box(int) }`

- **Grid Encoding (int map):**

Value	Cell Type
−100	Lava
0	Empty
1–10	Box (strength levels)
100	Barrier

3.3 Episode Dynamics

Each call to `step(action)`:

1. Increments timestep.
2. **Movement actions (1–4):**
   - Compute the intended next agent cell.
   - If the next cell is barrier or out of bounds: invalid move → agent does not move.
   - If the next cell is box: attempt to push a chain of adjacent boxes in the same direction. Let k be the number of boxes in the chain (see 4.2).
   - If the cell beyond the chain is empty or lava, the chain shifts by one cell in the action's direction (boxes may fall into lava; see 4.4); otherwise the push is invalid and nothing moves.
   - Apply stamina costs per 4. Update "non-stationary" flags per-direction (see 4.3).
3. **Special actions:**
   - **Hellify (6):** If there exists at least one perfect square of boxes of size $n > 2$ erase the boundary boxes of the oldest perfect square, convert the interior boxes into lava. All n^2 erased boxes are counted as "destroyed". No movement occurs.
   - **Barrier Maker (5):** If there exists at least one perfect square of size $n \geq 2$, convert all the boxes of the oldest perfect square into barriers as a whole (all n^2 cells become barriers). Award stamina $+n^2$ (see 6); no movement occurs.
4. Compute rewards and termination:
   - Episode terminates if max timesteps reached, stamina ≤ 0, or (if used) all targets achieved.

4 Stamina Mechanics

Stamina begins at $S_0 = 1000$ (configurable). Additional costs or refunds apply based on actions:

4.1 Push Cost Formula

When a push succeeds (moving a chain of boxes), the stamina cost for that action is:

$$\Delta S_{\text{push}} = \underbrace{\text{initial\_force} \cdot \mathbf{1}\{\text{head box is stationary in this direction}\}}_{\text{paid once per push action}} + \underbrace{\text{unit\_force} \cdot k}_{\text{proportional to number of boxes}}$$

where k is the number of boxes moved together in the chain.

4.2 "Boxes Together" (the Chain)

Boxes are together when they are axis-adjacent in a straight line in the agent's push direction, and the agent attempts to move the last box toward the others. The chain length k counts all contiguous boxes starting from the first box in front of the agent through the last box before the first non-box cell.

4.3 Stationary vs. Non-Stationary (Per Direction)

A box is **stationary in direction** d if it did not move in direction d on the immediately preceding timestep for that same direction chain. Rules:

- When a box is pushed in direction d this timestep, it becomes **non-stationary in** d for the next timestep.
- If that box does not move on the following timestep, it reverts to **stationary** in d thereafter.
- Non-stationary status is direction-specific: moving a box up does not affect its stationary status in left/right.
- For the push cost, the indicator $\mathbf{1}\{\text{head box is stationary in this direction}\}$ is 1 iff the heading box in the chain is stationary in direction d at the moment of pushing; otherwise 0. (A valid, simpler alternative for Milestone 1 is: pay the initial force if and only if the front-most box of the chain is stationary in d. Document which choice you implement.)

4.4 Lava Refund

If a box is pushed into a lava cell (-100), it is destroyed and removed from the map, and the agent **gains** stamina equal to `initial_force` once per box that enters lava during that action.

5 Perfect Squares & Special Actions

5.1 Definition of a Perfect Square

A perfect square is an $n \times n$ ($n > 1$) axis-aligned block of **only boxes** such that:

1. Every cell in the $n \times n$ region is a box.
2. No other boxes are adjacent to the perimeter of that region (i.e., the 8-neighborhood just outside the square contains no boxes). In other words, an $(n-1) \times (n-1)$ squared region would be identified where all $n \times n$ interior cells are boxes but none of boundary is a box.

Multiple disjoint squares may exist simultaneously.

5.2 Automatic Dissolution of Perfect Squares

After a perfect square is formed, if none of the boxes within it change (i.e., no boxes are moved, added, or removed from the square) and no Hellify or Barrier Maker action is performed on it by perf_sq_initial_age steps, the perfect square will automatically dissolve: all boxes in the square become empty spaces (`0`). No rewards or stamina changes occur during dissolution. This mechanic encourages timely action on perfect squares, shortening the agent's path to the terminal state with reduced stamina expenditure.

5.3 Action Availability

- **Hellify (ID 6)** is available iff there exists a perfect square with $n > 2$ on map.
- **Barrier Maker (ID 5)** is available iff there exists a perfect square with $n \geq 2$ on map.

If an action is chosen while unavailable, treat as invalid (no change in state apart from the stamina decreased and any invalid-action penalty as you would define).

5.4 Hellify

Erase exactly one perfect square with $n > 2$: the outer border boxes are removed (becoming `0`/empty), and the interior $(n-2) \times (n-2)$ square becomes lava (`-100`). All affected boxes are counted as "destroyed" for logging. No additional stamina cost beyond the baseline step cost.

5.5 Barrier Maker

Convert exactly one perfect square (with $n > 2$) into barriers: all n^2 cells become `100`. Award stamina $+n^2$ immediately.
If more than one perfect square exist on the map, then only the oldest one will convert to barriers per this action. According to this definition, the player should repeat this action as many times as any perfect square exists.

6 Rewards, Termination, and Info

While the exact reward shaping is flexible, you must implement the stamina mechanics above and at least the following default signals:

- Pushing boxes: no inherent reward, but stamina costs apply (discouraging unnecessary pushes).
- Box into lava: optionally, positive reward as equal to `+initial_force` per destroyed box to mirror the stamina refund; required stamina refund per 4.4.

Termination: Episode ends when any occur:

1. No boxes remain.
2. Stamina ≤ 0.
3. Timestep $\geq$ `max_timestep`.

Info dict: Include at least: timestep, stamina, number of boxes, number destroyed, last action valid?, chain length k, initial_force_charged? (bool), lava_destroyed_this_step, perfect_squares_available (list of $(n, \text{top-left})$).

7 Map Loading & Encoding

Your environment must load maps from a text file with UTF-8 encoding. Two supported formats:

Format A: Integer Grid (required)

Each line is a row of whitespace-separated integers in $\{-100, 0, 10, 100\}$.

```
// map.txt
0 0 0 0 0 0
0 10 10 0 100 0
0 0 -100 0 0 0
0 0 0 0 0 0
0 100 0 10 0 0
0 0 0 0 0 0
```

Format B: Symbolic (recommended)

Provide a simple legend and parse to the integer grid:

Symbol	Meaning
`.`	empty (0)
`B`	box (10)
`#`	barrier (100)
`L`	lava (-100)
`A`	agent start (separately recorded as position)

8 Rendering (Pygame GUI)

Implement a minimal GUI:

- Draw each cell type with distinct colors; draw agent as an overlay glyph.
- Display current timestep, stamina and other info on screen (e.g., top bar).
- Keyboard bindings: arrows/WASD for moves; `B` for Barrier Maker, `H` for Hellify; `R` to reset; `Q` to quit.
- Mouse click: change the agent's current position.
- Cap frame rate (e.g., 30 FPS).
- Optional: smooth animations for box pushes.(extra point)

9 API Surface & Expected Behavior

9.1 `reset()`

Resets the world, returns initial observation (see 3.2). Reinitialize:

- Agent position, map/grid, box set, barrier set, lava set.
- Stamina = S_0; timestep = 0.
- Clear all non-stationary flags.

10 Testing Requirements

Unit Tests (required)

Provide `pytest` tests for:

1. Push chain formation (k) and blocking behavior.
2. Stamina updates: baseline -1, push cost formula, lava refunds, barrier-maker gains, hellify effects.
3. Stationary/non-stationary transitions across timesteps and directions.
4. Perfect-square detection for $n = 2$ and $n = 3$, including adjacency exclusion.
5. Invalid actions (unavailable specials; pushing into barrier; moving into lava).
6. Map loading: malformed rows, invalid tokens, box placed on edges.

Smoke Test Script

Include `test.py` demonstrating:

```
import numpy as np
from environment import ShoverWorldEnv

env = ShoverWorldEnv(render_mode=None, n_rows=6, n_cols=9,
                     initial_force=4.0, unit_force=1.0, seed=0)
obs = env.reset()
done = False
total_r = 0.0
while not done:
    a = env.action_space.sample()
    obs, r, done, info = env.step(a)
    total_r += r
print("Episode return:", total_r)
env.close()
```

11 Deliverables (Milestone 1)

1. **Code**: `environment.py`, `test.py`, `gui.py` (if separate), `maps/` with at least two sample maps, `tests/` with unit tests.
2. **README**: build/run instructions; parameter table; brief design notes on stationary logic choice.
3. **Documentation**: This PDF with any deviations clearly marked.
4. **Short demo video (optional)**: ≤ 2 minutes showing GUI and special actions.

12 Grading Rubric (100 pts)

Criteria	Points
Gym-compatible API (`reset`, `step`, spaces, info)	10
Core dynamics (movement, pushing, blocking, boundaries)	15
Stamina mechanics (baseline, push formula, lava refund)	20
Stationary/non-stationary logic (per-direction)	10
Perfect-square detection & validation	10
Special actions (availability, Nullify & Barrier Maker effects)	10
Map loading (robust validation)	5
Pygame GUI (rendering, HUD, controls)	10
Tests (coverage of edge cases)	8
Documentation quality (clarity, completeness, reproducibility)	2
Total	**100**

13 Design Notes & Constraints

- **Solvability:** A solver is not required for Milestone 1. You may optionally reject clearly deadlocked random maps.
- **Rewards vs. Stamina:** Rewards are your shaping signal; stamina is the episodic resource. Keep both consistent and documented.

14 Parameter Reference

Name	Type	Meaning
`initial_force`	float	One-time cost (per push) if the pushed chain is stationary in that direction (default: 40)
`unit_force`	float	Linear cost per box in the pushed chain (default: 10)
`initial_stamina`	float	Starting stamina (default: 1000)
`r_lava`	float	Optional reward per box destroyed in lava (default: `initial_force`)
`r_barrier_maker(n)`	func	Reward when creating an $n \times n$ barrier (default: $+10 \times n^2$)

15 Example: Push Cost Walkthrough

Suppose $k = 3$ boxes form a chain to the right. Parameters: `initial_force`$= 5$, `unit_force`$= 2$.

- If the chain is stationary in "right": push cost $5 + 2 \cdot 3 = 11 \Rightarrow$ stamina change -11.
- If non-stationary in "right": push cost $0 + 2 \cdot 3 = 6 \Rightarrow$ stamina change -6.
- If the front-most box falls into lava: add $+5$ refund; net -6 (stationary) or -1 (non-stationary).

16 Minimal File Structure (Suggested)

```
shover_world/
- environment.py
- gui.py
```

```
- test.py
- maps/
  - map_small.txt
  - agent_small.txt
- tests/
  - test_push.py
  - test_stamina.py
  - test_squares.py
  - test_maps.py
- README.md
```

17 Academic Integrity

You may consult public references on Gym, Pygame, and RL. Your implementation must be your own. Cite any external code snippets in `README.md`.

18 Project Example

In this link, you can see an example implementation of the Shover-World environment as an illustration of how to structure your code, including the Gym-compatible API, stamina mechanics, and special actions.