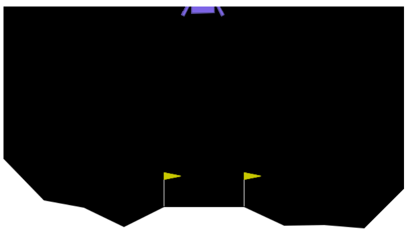
Lunar Lander



This environment is part of the Box2D environments which contains general information about the environment.

Action Space	`Discrete(4)`
Observation Space	`Box([ -2.5 -2.5 -10. -10. -6.2831855 -10. -0. -0. ], [ 2.5 2.5 10. 10. 6.2831855 10. 1. 1. ], (8,), float32)`
import	`gymnasium.make("LunarLander-v3")`

Description

This environment is a classic rocket trajectory optimization problem. According to Pontryagin's maximum principle, it is optimal to fire the engine at full throttle or turn it off. This is the reason why this environment has discrete actions: engine on or off.

There are two environment versions: discrete or continuous. The landing pad is always at coordinates (0,0). The coordinates are the first two numbers in the state vector. Landing outside of the landing pad is possible. Fuel is infinite, so an agent can learn to fly and then land on its first attempt.

To see a heuristic landing, run:

```
python gymnasium/envs/box2d/lunar_lander.py
```

v1.1.1 (latest)

There are four discrete actions available:

- 0: do nothing
- 1: fire left orientation engine
- 2: fire main engine
- 3: fire right orientation engine

Observation Space

The state is an 8-dimensional vector: the coordinates of the lander in `x` & `y`, its linear velocities in `x` & `y`, its angle, its angular velocity, and two booleans that represent whether each leg is in contact with the ground or not.

Rewards

After every step a reward is granted. The total reward of an episode is the sum of the rewards for all the steps within that episode.

For each step, the reward:

- is increased/decreased the closer/further the lander is to the landing pad.
- is increased/decreased the slower/faster the lander is moving.
- is decreased the more the lander is tilted (angle not horizontal).
- is increased by 10 points for each leg that is in contact with the ground.
- is decreased by 0.03 points each frame a side engine is firing.
- is decreased by 0.3 points each frame the main engine is firing.

The episode receive an additional reward of -100 or +100 points for crashing or landing safely respectively.

An episode is considered a solution if it scores at least 200 points.

Starting State

The lander starts at the top center of the viewport with a random initial force applied to its center of mass.

v1.1.1 (latest)

The episode finishes if:

1. the lander crashes (the lander body gets in contact with the moon);
2. the lander gets outside of the viewport (`x` coordinate is greater than 1);
3. the lander is not awake. From the Box2D docs, a body which is not awake is a body which doesn't move and doesn't collide with any other body:

> When Box2D determines that a body (or group of bodies) has come to rest, the body enters a sleep state which has very little CPU overhead. If a body is awake and collides with a sleeping body, then the sleeping body wakes up. Bodies will also wake up if a joint or contact attached to them is destroyed.

Arguments

Lunar Lander has a large number of arguments

```
...                enable_wind=False, wind_power=15.0, turbulence_power=1.5)
>>> env
<TimeLimit<OrderEnforcing<PassiveEnvChecker<LunarLander<LunarLander-v3>>>>>
```

- `continuous` determines if discrete or continuous actions (corresponding to the throttle of the engines) will be used with the action space being `Discrete(4)` or `Box(-1, +1, (2,), dtype=np.float32)` respectively. For continuous actions, the first coordinate of an action determines the throttle of the main engine, while the second coordinate specifies the throttle of the lateral boosters. Given an action `np.array([main, lateral])`, the main engine will be turned off completely if `main < 0` and the throttle scales affinely from 50% to 100% for `0 <= main <= 1` (in particular, the main engine doesn't work with less than 50% power). Similarly, if `-0.5 < lateral < 0.5`, the lateral boosters will not fire at all. If `lateral < -0.5`, the left booster will fire, and if `lateral > 0.5`, the right booster will fire. Again, the throttle scales affinely from 50% to 100% between -1 and -0.5 (and 0.5 and 1, respectively).
- `gravity` dictates the gravitational constant, this is bounded to be within 0 and -12. Default is -10.0
- `enable_wind` determines if there will be wind effects applied to the lander. The wind is generated using the function `tanh(sin(2 k (t+C)) + sin(pi k (t+C)))` where `k` is set to 0.01 and `C` is sampled randomly between -9999 and 9999.
- `wind_power` dictates the maximum magnitude of linear wind applied to the craft. The recommended value for `wind_power` is between 0.0 and 20.0.
- `turbulence_power` dictates the maximum magnitude of rotational wind applied to the craft. The recommended value for `turbulence_power` is between 0.0 and 2.0.

Version History

- v3:
  - Reset wind and turbulence offset (`C`) whenever the environment is reset to ensure statistical independence between consecutive episodes (related GitHub issue).
  - Fix non-deterministic behaviour due to not fully destroying the world (related GitHub issue).
  - Changed observation space for `x`, `y` coordinates from ± 1.5 to ± 2.5, velocities from ± 5 to ± 10 and angles from $\pm \pi$ to $\pm 2\pi$ (related GitHub issue).
- v2: Count energy spent and in v0.24, added turbulence with wind power and turbulence_power parameters
- v1: Legs contact with ground added in state vector; contact with ground give +10 reward points, and -10 if then lose contact; reward renormalized to 200; harder initial random push.
- v0: Initial version

v1.1.1 (latest)

There are several unexpected bugs with the implementation of the environment.

1. The position of the side thrusters on the body of the lander changes, depending on the orientation of the lander. This in turn results in an orientation dependent torque being applied to the lander.
2. The units of the state are not consistent. I.e.

- The angular velocity is in units of 0.4 radians per second. In order to convert to radians per second, the value needs to be multiplied by a factor of 2.5.

For the default values of VIEWPORT_W, VIEWPORT_H, SCALE, and FPS, the scale factors equal: ‘x’: 10, ‘y’: 6.666, ‘vx’: 5, ‘vy’: 7.5, ‘angle’: 1, ‘angular velocity’: 2.5

After the correction has been made, the units of the state are as follows: ‘x’: (units), ‘y’: (units), ‘vx’: (units/second), ‘vy’: (units/second), ‘angle’: (radians), ‘angular velocity’: (radians/second)

Credits

Created by Oleg Klimov

Copyright © 2025 Farama Foundation

v1.1.1 (latest)