User Guide for the `StaticGridEnv` Environment

Version 1.0

October 18, 2024

Contents

1 Introduction

Welcome to the `StaticGridEnv` user guide. This environment is a grid-based simulation, designed for testing reinforcement learning algorithms like Q-learning or SARSA. It provides a straightforward and effective platform to develop and assess agents in a controlled setting.

2 Prerequisites

Before using the environment, ensure you have the following software installed:

- **Python 3.x**
- **NumPy**: Required for numerical computations.
- **Pygame**: Required for rendering the environment visually.

You can install these packages using `pip`:

```
pip install numpy pygame
```

3 Installation

Follow these steps to set up the environment:

1. **Download the Environment Code**: Save the `StaticGridEnv` class code into a file named `static_grid_env.py`.
2. **Create an Images Directory**: In the same directory as your script, create an `images` folder that contains:
   - `agent.png`: An image representing the agent.
   - `goal.png`: An image representing the goal.
   - `obstacle.png`: An image representing obstacles.

4 Environment Overview

Key characteristics of the environment include:

- **Grid Size**: The grid is composed of 10×10 cells.
- **Cell Size**: Each cell measures 64×64 pixels when rendered.
- **Obstacles**: These are located at the following coordinates: $(1,1)$, $(2,2)$, $(3,3)$, $(4,4)$, and $(5,5)$.
- **Goal Position**: The goal is placed in the bottom-right corner of the grid, at $(9,9)$.
- **Agent's Starting Position**: This is randomly chosen, ensuring that it is not on an obstacle or the goal.

- **Action Space**: The agent can perform one of four possible actions:
  - `0`: Move up.
  - `1`: Move down.
  - `2`: Move left.
  - `3`: Move right.
- **Rewards System**:
  - Penalty of -1 for each normal step.
  - Penalty of -5 for hitting an obstacle (the agent stays in its current position).
  - Reward of 20 for reaching the goal (the episode ends).
- **Movement Constraints:**
  - **Boundary Constraints**: The agent cannot move beyond the grid's boundaries. If it attempts to do so, it remains in its current position but no additional penalty is applied (however, the regular reward remains).
  - **Obstacle Constraints**: The agent cannot move into a cell occupied by an obstacle. If it tries to, it will stay in place and receive a penalty.

5 Using the Environment

5.1 Initialising the Environment

To begin using the environment, import the module and create an instance:

```
from env import StaticGridEnv

# Optional: Set a seed for reproducibility
env = StaticGridEnv(seed=42)
```

Note: Setting the `seed` parameter ensures consistent random number generation across runs, aiding reproducibility.

5.2 Resetting the Environment

To start a new episode, reset the environment:

```
state = env.reset()
```

- `state`: A NumPy array indicating the agent's initial position.

5.3 Taking Actions

Move the agent by taking an action:

```
action = 0  # Replace with the desired action (0: up, 1: down, 2:
    left, 3: right)
next_state, reward, done, info = env.step(action)
```

- `next_state`: The agent's position after the action.
- `reward`: The reward earned from the action.
- `done`: `True` if the goal is reached, otherwise `False`.
- `info`: Additional information (unused in this environment).

5.4 Rendering the Environment

To visualise the environment, use the following command:

```
env.render(
    delay=0.1,
    episode=1,
    learning_type="Q-learning",
    availability=0.8,  # Optional: Teacher availability (80%)
    accuracy=0.9       # Optional: Teacher accuracy (90%)
)
```

Parameters:

- `delay`: Time delay between frames (in seconds).
- `episode`: Current episode number.
- `learning_type`: Type of learning algorithm used.
- `availability`: (Optional) Teacher availability percentage.
- `accuracy`: (Optional) Teacher accuracy percentage.

5.5 Closing the Environment

After finishing with the environment, close it properly:

```
env.close()
```

6 Example Usage

Here's an example of how to utilise the `StaticGridEnv` within a basic agent loop:

```
import random
from env import StaticGridEnv

# Initialise the environment
env = StaticGridEnv()
```

```
# Start a new episode
state = env.reset()
done = False

while not done:
    # Take a random action for this demonstration
    action = random.choice([0, 1, 2, 3])

    # Execute the action
    next_state, reward, done, _ = env.step(action)

    # Render the environment
    env.render(episode=0, learning_type="Random Policy")

    # Update the state
    state = next_state

# Close the environment
env.close()
```

7 Troubleshooting

- **Pygame Window Not Displaying**: Ensure your system supports graphical user interfaces (GUI) and that Pygame is installed correctly.
- **Image Files Not Found**: Check that the `images` folder is in the correct location and contains `agent.png`, `goal.png`, and `obstacle.png`.
- **Performance Issues**: Reduce the `delay` parameter in the `render` method to speed up the simulation.