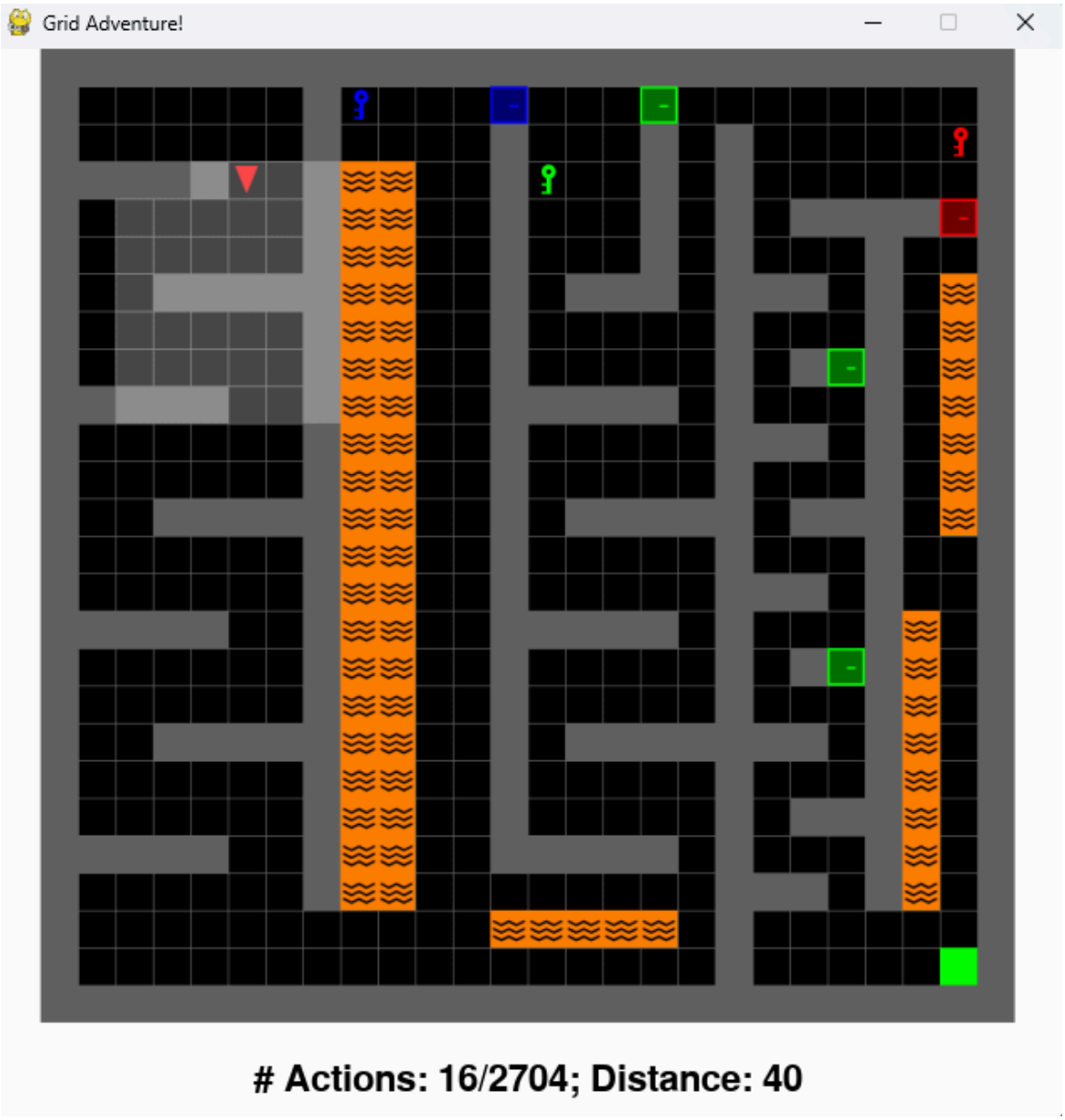
2024-02 KNU AI Team Competition Round 1

Grid Adventure!



첫 번째 Team Competition은 뜨거운 용암을 피하고 잠겨진 문을 열어서 미로를 탈출하는 ***Grid Adventure*** 입니다!

Description

플레이어는 좌표 [0, 0]에서 출발해서 좌표 [25, 25]에 있는 도착 지점으로 가야 합니다. 하지만, 그 앞을 가로막는 건 벽과, 용암과, 잠겨진 문이죠. 여러분은 벽과 용암을 피하고, 잠겨진 문을 열어서 미로를 탈출하는 에이전트를 훈련해야 합니다!

Installation

*Grid Adventure*를 사용하기 위해서는 `Python 3.8` 이상 `3.11` 이하를 추천하며, 그 외의 환경에서는 정확한 동작을 보장할 수 없습니다.

*Grid Adventure*의 설치는 다음과 같이 할 수 있습니다:

비공개

그 외 실행에 필요한 `gymnasium`, `pygame` 등의 라이브러리는 함께 설치되므로, 별도 설치할 필요는 없습니다.

Class and Methods

*Grid Adventure*는 다음과 같이 4개의 클래스와 함수로 구성됩니다:

knu_rl_env.grid_adventure.GridAdventureAgent

*Grid Adventure*를 플레이하는 에이전트의 추상 클래스입니다. 아래처럼 이 추상 클래스를 상속해서 여러분만의 에이전트를 정의해보세요. 여기서 여러분이 필수로 구현해야 할 추상 함수는 상태를 받아 행동을 출력하는 **act(state)**입니다.

```
from knu_rl_env.grid_adventure import GridAdventureAgent


class GridAdventureRLAgent(GridAdventureAgent):
    def act(self, state):
        '''
        Return value is one of actions following:
        - GridAdventureAgent.ACTION_LEFT
        - GridAdventureAgent.ACTION_RIGHT
        - GridAdventureAgent.ACTION_FORWARD
        - GridAdventureAgent.ACTION_PICKUP
        - GridAdventureAgent.ACTION_DROP
        - GridAdventureAgent.ACTION_UNLOCK
        '''
        pass
```

knu_rl_env.grid_adventure.make_grid_adventure(show_screen)

Grid Adventure 환경을 만들어주는 함수입니다. **show_screen = True**일 경우 실제 게임 화면을 보여줍니다.

이 함수는 **gymnasium.Env**의 인스턴스를 반환하므로,

gymnasium.Env.step(action), **gymnasium.Env.reset()**,

gymnasium.Env.render(), **gymnasium.Env.close()** 의 핵심 함수 또한 활용 가능합니다.

knu_rl_env.grid_adventure.evaluate(agent)

여러분들이 훈련시킨 에이전트를 실제 시연 평가에서 보여주는 함수입니다. **agent** 인자에 여러분이 직접 구현한 에이전트를 넣으세요.

knu_rl_env.grid_adventure.run_manual()

에이전트 구현 없이 직접 플레이도 가능합니다.

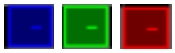
- 방향키 ←: 플레이어가 좌측으로 90도 회전합니다.
- 방향키 →: 플레이어가 우측으로 90도 회전합니다.
- 방향키 ↑: 플레이어가 보는 방향으로 앞으로 1칸 이동합니다.
- 탭 키: 플레이어 앞에 존재하는 열쇠를 획득합니다. 단, 이미 획득한 열쇠가 있다면 획득할 수 없으며, 아래 놓기 행동을 먼저 해야 합니다.
- 좌측 쉬프트 키: 플레이어 앞에 이미 획득한 열쇠를 버립니다.
- 스페이스 바: 잠긴 문을 엽니다. 단, 이전에 문의 색상과 일치하는 열쇠를 획득해야 합니다.

Environment

Observation Space

미로 내의 객체들은 다음과 같습니다:

-  : 일반적인 바닥입니다. 플레이어는 이곳은 마음대로 움직일 수 있어요.
-  : 벽입니다. 이 곳으로는 이동할 수 없고, 플레이를 이리저리 헤매게 만듭니다.

-  : 용암입니다. 이 곳으로 도착하는 순간 게임 오버입니다.
-  : 목표 지점입니다. 이곳에 도착하면 미로를 탈출하게 됩니다.
-  : 플레이어입니다. 화살표의 방향이 현재 향하고 있는 방향입니다.
-  : 잠겨진 문입니다. 문 색상과 일치하는 색의 열쇠로만 열립니다.
-  : 열쇠입니다. 당연히, 문을 여는데 사용됩니다.

에이전트는 이러한 미로의 구성 요소들을 [26, 26] 행렬로 알 수 있으며, 각 행렬의 원소는 객체 및 객체의 상태를 나타내는 알파벳 문자로 구성됩니다:

- **`E`**: 일반적인 바닥
- **`W`**: 벽
- **`L`**: 용암
- **`G`**: 목표 지점
- **`AL`**: 좌측 방향을 보고 있는 플레이어
- **`AR`**: 우측 방향을 보고 있는 플레이어
- **`AU`**: 윗 방향을 보고 있는 플레이어
- **`AD`**: 아랫 방향을 보고 있는 플레이어
- **`DBL`**: 잠겨진 청색 문
- **`DBC`**: 닫힌 청색 문
- **`DBO`**: 열린 청색 문
- **`DGL`**: 잠겨진 녹색 문
- **`DGC`**: 닫힌 녹색 문
- **`DGO`**: 열린 녹색 문

- **DRL**: 잠겨진 적색 문
- **DRC**: 닫힌 적색 문
- **DRO**: 열린 적색 문
- **KB**: 청색 열쇠
- **KG**: 녹색 열쇠
- **KR**: 적색 열쇠

Action Space

플레이어는 다음과 같은 6개의 행동을 할 수 있으며, 에이전트의 구현체**knu_rl_env.GridAdventureAgent**에 각 행동을 상수로 지정해두었습니다:

- **knu_rl_env.GridAdventureAgent.ACTION_LEFT**: 플레이어가 좌측으로 90도 회전합니다.
- **knu_rl_env.GridAdventureAgent.ACTION_RIGHT**: 플레이어가 우측으로 90도 회전합니다.
- **knu_rl_env.GridAdventureAgent.ACTION_FORWARD**: 플레이어가 보는 방향으로 앞으로 1칸 이동합니다.
- **knu_rl_env.GridAdventureAgent.ACTION_PICKUP**: 플레이어 앞에 존재하는 열쇠를 획득합니다. 단, 이미 획득한 열쇠가 있다면 획득할 수 없으며, 아래 놓기 행동을 먼저 해야 합니다.
- **knu_rl_env.GridAdventureAgent.ACTION_DROP**: 플레이어 앞에 이미 획득한 열쇠를 버립니다.

- **knu_rl_env.GridAdventureAgent.ACTION_UNLOCK**: 잠긴 문을 엽니다. 단, 이전에 문의 색상과 일치하는 열쇠를 획득해야 합니다.

Reward Space

어떤 행동이든 보상의 양을 0으로 설정했습니다. 여러분들이 자유롭게 보상을 설계해보세요!

Episode End

목표 지점에 도달하거나, 용암에 빠지거나, 2,704회 이상 행동을 하게 되면 에피소드가 종료됩니다.

Submission

이루리에 제출해야 하는 파일은 두 개입니다.

Implementation

구현 템플릿에 에이전트의 추상 클래스**knu_rl_env.GridAdventureAgent**를 상속하여 자신만의 에이전트를 정의하고, 이 에이전트를 훈련하는 코드를 구현하세요.

```
from knu_rl_env.grid_adventure import GridAdventureAgent, make_grid_adventure, evaluate


'''
knu_rl_env.grid_adventure.GridAdventureAgent을 상속해서
자신만의 에이전트를 구현하세요.
```

```
'''
class GridAdventureRLAgent(GridAdventureAgent):
    def act(self, state):
        '''
        다음 중 하나를 반환해야 합니다:
        - GridAdventureAgent.ACTION_LEFT
        - GridAdventureAgent.ACTION_RIGHT
        - GridAdventureAgent.ACTION_FORWARD
        - GridAdventureAgent.ACTION_PICKUP
        - GridAdventureAgent.ACTION_DROP
        - GridAdventureAgent.ACTION_UNLOCK
        '''
        pass

'''
에이전트를 훈련하는 코드를 구현하세요.
'''
def train():
    '''
    Grid Adventure 환경은 다음과 같이 생성할 수 있습니다.
    '''
    env = make_grid_adventure(
        show_screen=True # or, False
    )
    '''
    여기서부터는 이 환경에 대해서 에이전트를 훈련시키는 코드가 필요합니다.
    '''
```

Dependent Libraries

에이전트 구현 시에 사용한 의존성 라이브러리의 목록을 다음 명령어로 생성된 **requirements.txt** 파일로 제출해야 합니다.

```
pip list -format=freeze > requirements.txt
```

Evaluation

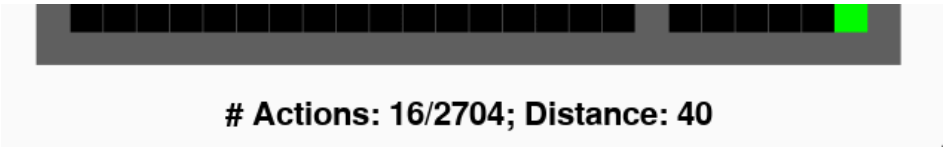
Team Competition은 다음의 코드를 실행하여 플레이한 결과로 결정됩니다

(별도의 실행 코드는 인정하지 않습니다):

```
from knu_rl_env.grid_adventure import evaluate

if __name__ == '__main__':
    agent = '''여러분이 정의하고 학습시킨 에이전트를 불러오는 코드를 넣으세요.'''
    evaluate(agent)
```

Grading Criteria



 # Actions: 16/2704; Distance: 40

최소한의 행동을 수행하면서 목표 지점에 도착하는 것이 기본적인 목표이며, 세부 평가 기준은 다음과 같습니다:

- 목표 지점에 도착하는 데 **성공**했을 시, **행동 수행 횟수(# Actions)가 작을수록 좋음**
- 목표 지점에 도착하는 데 **실패**했을 시**, 플레이어와 목표 지점의 거리(Distance)가 짧을수록 좋음**

위 기준에 따라 순위를 매기며, 순위에 따른 점수는 다음과 같습니다:

- 1위: 25%
- 2위: 22%

- 3위: 19%
- 4위: 16%
- 5위: 13%
- 6위: 10%
- 7위: 7%
- 8위: 4%

0 Points Policy

단, 다음 중 하나에 해당하는 경우에는 무조건 0% 처리가 됩니다.

- 제출 마감 내에 제출물 (에이전트 구현체 및 **`requirements.txt`**)을 이루리에 제출하지 않은 경우 (지연 제출 또한 허용 안됨)
- **`requirements.txt`**에서 정의된 모든 의존성 라이브러리를 설치했음에도 불구하고 제출한 코드의 실행이 되지 않는 경우
- 제출한 코드의 실행 결과로 학습된 에이전트가 실제 시연 결과와 크게 다른 경우
- 시연을 하지 않거나, 시연이 불가능한 경우
    - 자신의 팀이 시연을 하는 일자에 출석하지 않는 경우, 해당 학생만 0% 처리
- 정책 또는 가치 함수를 학습하지 않고, 직접 손으로 지정한 정책 또는 가치 함수를 사용하는 경우
- 환경을 바꾸는 경우

- 자신이 구현하지 않은 별도의 강화 학습 구현체 또는 라이브러리를 활용하는 경우