README.md

Sokoban Generator - NYU CS-GY6613 Extra Credit

Prerequisites

-- See Sokoban Environment README section *prerequisites*

Install libraries

```
$ pip install –r requirements.txt
```

Run the Generator

Create the levels

`$ python level_generator.py` -- **Note**: Internal variables including number of levels, solution length, agent evaluator, and iterations, are modified within the code itself.

Test the levels

-- See Sokoban Environement README section *Run the Game* -- This environment has been modified to run the generated levels instead of the original framework levels. To change the directory for the levels played, change this line `LEVEL_DIR = "gen_levels"` in the `fast_game.py` and `game.py` codes.

Agent Types

- **RandomAgent()** - agent that tries to find a randomly generated solution of length 20 within a set number of iterations -- For all other agents, see Sokoban Environment README section *Agent Types*

Code Functions

Note: You are allowed to change ONLY the functions in the `level_generator.py` file (and `agent.py` file if you need to modify your agents). Do NOT change any of the functions in the Sokoban framework.

- level_generator.py
  - **lev2Str(level2D)** - converts a 2d char array into a string with newline characters (return type: str) Use with exporting the level to a file.
  - **makeEmptyLevel(width, height)** - creates an empty Sokoban level of set `width` and `height` with wall characters bordering the level and floor tiles filling it (return type: 2d char array). Use to create a blank level.
  - **buildALevel()** - creates a new Sokoban level using procedural content generation (return type: str).
  - **solveLevel(level,bot)** - uses an agent from the `agent.py` file and specified by the `EVAL_AGENT` variable to attempt to solve the level within #(`SOLVE_ITERATIONS`) iterations and returns whether it was successful and a solution (return type: bool, dict array).
  - **generateLevels()** - iteratively generates #(`NUM_LEVELS`) solvable levels using the `buildALevel()` and `solveLevel()` functions. Exports these levels to the `LEVEL_DIR` directory to be played later.

FAQ

- *What are you looking for in terms of a "good" generator?*
    - We will run your code individually, so the levels generated on our end should be different than the levels generated on your end. The minimum requirements for the generator are as follows:
        a. Generates X number of *SOLVABLE* levels and exports them to an external folder in the correct file format
        b. No hard-coded levels. Each level is generated procedurally using some form of randomness
        c. Creates different levels every time that have variability in terms of content (i.e different level dimensions, different number of goals and crates, internal wall placements, varying solution length, varying levels of difficulty across multiple agents, etc)
    - This assignment is graded for completion of the 3 criteria, however the more creative you are with your PCG code, the better.
    - The `DEMO_level_generator.py` is included as a sample file to help you get started with the minimum criteria. Your own generator should be much more complex than this file's in terms of procedural content generation methods.
- *Can I create more functions in level_generator.py?*
    - Yes! It's strongly encouraged you do so.
- *I don't think my ___ agent is correctly implemented, what do I do?*
    - Use the feedback from the assignments to fix your code as much as possible. Unfortunately, we cannot release the master code for these agents to prevent plagarism in next year's class. However, you are allowed to choose any of the agents in the `agent.py` file as your evaluator.
- *Can I import more libaries?*
    - Other than numpy or any other built-in python libraries, no. Please do not include any external libraries (i.e. libraries that require a pip installation or that require code to be downloaded from another site.) We want to keep the assignment simple, but still allow you the freedom to be creative with the tools already available.