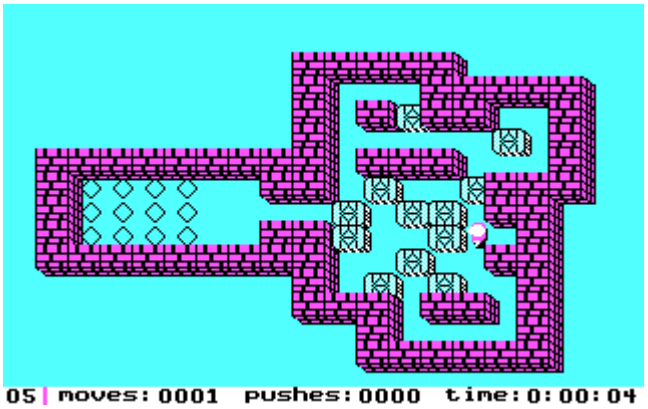
PROPOSED ENVIRONMENT FOR SEMINAR WORK

Environment properties inferred from:
https://www.doc.ic.ac.uk/project/examples/2005/163/g0516302/environments/environments.html

Proposed environment for seminar work: **SOKOBAN** (http://en.wikipedia.org/wiki/Sokoban):



a. Description:
   - Is a type of transport puzzle, in which the player pushes boxes or crates around in a warehouse, trying to get them to storage locations.
   - The game is played on a board of squares, where each square is a floor or a wall. Some floor squares are marked as storage locations, and some of them have boxes.
   - The player is confined to the board, and may move horizontally or vertically onto empty squares (never through walls or boxes). The player can also move into a box, which pushes it into the square beyond. Boxes may not be pushed into other boxes or walls, and they cannot be pulled. The puzzle is solved when all boxes are at storage locations.

b. Scientific research:
   - The problem of solving Sokoban puzzles was proven to be NP-hard.
   - This is also interesting for artificial intelligence researchers**, because solving Sokoban can be compared to designing a robot which moves boxes in a warehouse.**

c. Variations:
   - In CyberBox, each level has a designated exit square, and the goal is to reach that exit.
   - Sokonex, Xsok, Cyberbox and Block-o-Mania all add new elements to the basic puzzle. Examples include holes, teleports, moving blocks and one-way passages.

d. Placement within template:
   - Class Environment:
     1. __init__: We have normal 2D grid, with starting state and end state. We would also need to define box locations and locations of where certain box should be.
     2. getStartingState: Each tile will have unique ID (I would use hash of indexes used to address array).

3. do: We need to define possible actions!
    a. Actions:
        i. Move (up, down, left, right)
        ii. Push (up, down, left, right)
        iii. Pull (up, down, left, right)
    b. Reward:
        i. Setting boxes to proper location.
        ii. Exit the maze (reaching terminal state).
4. getActions: Possible actions which depend whether agent is located near box or wall. If is located near box, then actions PUSH and PULL are possible.

e. Literature:

- Reinforcement Learning Agents with Primary Knowledge
- Learning state features from policies to bias exploration in reinforcement learning
- Reinforcement learning based on local state feature learning and policy adjustment