Development of an autonomous agent for the game **Tetris**

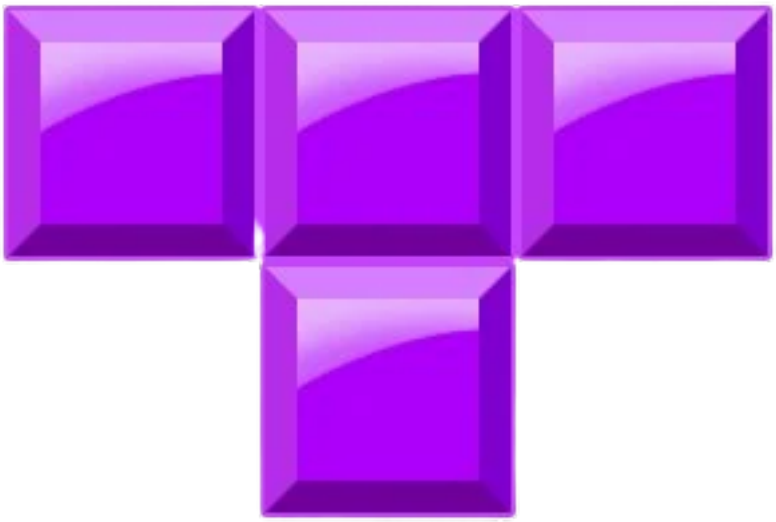
Group Assignment

Inteligência Artificial

Ano Lectivo de 2021/2022

Diogo Gomes
Luís Seabra Lopes

October 9, 2021



I Important notes

1. This work must be carried out in groups of 2 to 3 students. In each Python module submitted, you must include a comment with the authors' name and mechanographic number.

2. A first version of the program must be submitted by 26th of November 2021. The final version must be submitted by 10th of December 2021. In both submissions, the work can be submitted beyond the deadline, but will be penalized in 5% for each additional day.

3. Each group must submit its code through the *GitHub Classroom* platform. In the final submission, include a presentation (Powerpoint type) in `.pdf` format and named `presentation.pdf`, with a maximum of five pages, where you should summarize the architecture of the developed agent.

4. The code should be developed in Python 3.7. The main module should be named `student.py`.

5. If you discuss this work with colleagues from other groups, include a comment with the name and mechanographic number of these colleagues. If you use other sources, cite them as well.

6. All code submitted must be original; though trusting that most groups will comply with this requirement, tools will be used to detect plagiarism. Students involved in plagiarism cases will have the assignment canceled.

7. The project will be evaluated taking into account: performance; quality of architecture and implementation; and originality.

II Overview

This work involves the application of concepts and techniques from three main chapters of the IA/AI course, namely: Python programming; agent architectures; and search techniques for automated problem solving.

Within the scope of this work, you should develop an agent capable of playing intelligently the game Tetris, a game developed by Alexei Pajitnov in the former USSR and popularized in the early 80s of last century by Nintendo.

In *Tetris*, the player packs up pieces/tiles (tetrominoes) that fall vertically to fill lines. Whenever a line is completely filled it disappears and the player receives a score. The simultaneous filling of multiple lines gives the player bonus scores. The pieces that fall have a random order, but there is information about the future pieces, allowing the planning of the game in order to optimize the score. As the game evolves, the pieces fall at a higher speed, increasing the game's difficulty. If a new piece does not fit on the game screen, it ends. The objective of the game is therefore to pack all the pieces maximizing the score.

The game's score takes into account the number of packed pieces by the agent and the efficiency of the solutions found (privileging multiple lines in the shortest possible time).

1 Objectives

- To obtain a positive mark, the agent must be able to score a minimum of 100 points.
- Score as much as possible. (see below details on marking)

III Game Rules

- *Tetris* starts with an empty corridor and a falling piece.
- The *Tetris* player (student agent) has access at all times to the entire corridor, with the location of all occupied positions and to a list of the next N pieces (default 3) . [1]
- *Tetris* allows you to move the pieces using the commands **"w"** (*rotate piece*), **"s"** (*down*), **"a"** (*left*) and **"d"** (*right*).
- The score obtained mirrors the number of lines removed and the fact of removing more than one line at the same time. 1 line is 1 point, 2 lines is 4 points, 3 lines is 9 points, 4 lines is 16 points.
- As the game progresses, the speed of the game increases, therefore increasing the difficulty of the game.

IV Code and Development Support

A *Tetris* game engine written in Python is available at `https://github.com/dgomes/ia-tetris`.

All game entities are represented by classes.

Each group develops an agent creating a client that implements the exemplified protocol in the `client.py` file. No modifications to other files are necessary (you can't change `game.py`), but you can create new files, folders, etc.

If you implement a new feature or implement any improvement to the game engine and/or viewer, you can create a "Pull Request"(PR) on the GitHub platform. If your change is accepted, you will be credited with a bonus on the final evaluation up to a maximum of 1 value.

The developed agent must be delivered in a module named `student.py` and the agent should connect to the local game server, using as *username* the mechanographic number of one of the elements of the group (any).

There is a support channel in `https://detiuaveiro.slack.com/messages/ai/` where students can ask questions and receive notifications of changes.

Given the novelty of this work/code, it is expected that there are some bugs and tweaks during the course of work. Be on the watch out for server modifications (`git pull`) and notifications in *e-mail*, *Slack* and *e-learning*.

To start the work, you must form a group with colleagues and access the link `https://classroom.github.com/a/GUehLVOB` which will *fork* the code for the group. Only one *fork* should be done per group. One of the elements of the group creates the group associating the other elements, after this step the *fork* will be created automatically (do not create a new *fork* without all elements being registered).

V Recommendations

1. Start by studying `client.py`. The code is very basic and simple, so start by *refactoring* the client to something more oriented to a AI Agent.
2. Periodically `git fetch` from the original repository to update your code.
3. Run `git log` to keep track of small changes that have been be made.

[1] The exact value of N will be established on November 10, and may change after the first delivery.

4. Follow the channel #ai on Slack

VI Clarification of doubts

Clarifications on the main doubts that may arise during the performance of the work will be placed here.

1. **Question**: How will the performance of agents be evaluated?

   Answer: Agents will be evaluated on their performance in a set of games based on the sum of the final scores in the various games. The final score in a game is the one the agent has at the time of *gameover* (when the next piece doesn't fit on the screen).

2. **Question**: How will the practical work be evaluated?

   Answer: Total game scores for each submitted agent are mapped to marks taking into acount the distribution of total scores. A total score of P points [2] is mapped to 10/20. The average total score is mapped to 16/20. The maximum total score is mapped to 20/20. Other total scores are mapped linearly.

   1st delivery evaluation

   - In this delivery, only the agent code must be submitted.
   - Each agent will play 10 games and the average of these 10 games will be obtained.

   2nd delivery evaluation

   - Each agent will play 10 games.
   - In this delivery it is necessary to submit a presentation (see point I.3 above).
   - The evaluation of the second delivery reflects the agent's performance (90%), the design quality (according to the delivered presentation, 7.5%) and the quality of the implementation (2.5%).

[2] P will be announced by November 10; as a reference, consider P=100 points.