Development of an autonomous agent for the game **Dig Dug**

Group Assignment

Inteligência Artificial

Ano Lectivo de 2023/2024

Diogo Gomes
Luís Seabra Lopes

October 5, 2023

I Important notes

1. This work must be carried out in groups of 2/3 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 17th of November 2023. The final version must be submitted by 15th of December 2023. 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 at least Python 3.9. 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 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 Dig Dug, an arcade game developed by Namco in Japan in 1982.

In *Dig Dug*, the player plays the role of a miner. It is a maze-style game that combines elements of action and strategy. The gameplay of Dig Dug revolves around the player controlling a character known as "Dig Dug"who must eliminate underground-dwelling monsters while digging through the game's underground levels.

The main objective of Dig Dug is to clear each level of all the underground-dwelling monsters while accumulating points. The player progresses through increasingly challenging levels as they advance in the game.

The primary antagonists in Dig Dug are the Pookas and Fygars, which are colorful, round creatures. To defeat them, Dig Dug uses its air pump. When a monster is adjacent to Dig Dug, the player can press the pump button (A), and Dig Dug will inflate the monster until it bursts (it does not die immediately). Points are earned for defeating monsters according to their type, position and kill weapon.

The game's strategy involves luring monsters into tunnels and then quickly inflating them to defeat them. Players must plan their movements to avoid getting cornered by monsters, as contact with them results in losing a life. Additionally, players must be cautious while digging through the ground, as falling rocks can also be lethal (to both Dig Dug and Monsters). Killing monsters with rocks grants extra points compared to the inflatable pump.

Occasionally, vegetables and fruits appear on the screen. These items can be collected for extra points when touched by Dig Dug.

1 Objectives

- To obtain a positive mark, the agent must be able to score more than 10000 points.
- Score as much as possible. (see below details on marking)

III Game Rules

- *Dig Dug* starts with a map divided into 4 layers and with tunnels populated with monsters, as well as randomly placed rocks.
- The *Dig Dug* player (student agent) has access at all times to the entire map, with the location of all occupied positions. The agent controls a cursor, through which he can move Dig Dug either vertically or horizontally.
- *Dig Dug* allows you to move the cars using the commands **"w"** (*move up*), **"s"** (*move down*), **"a"** (*move left*), **"d"** (*move right*) and **"A"** (*shoot*).
- The game is composed of several maps of incresed difficulty. For each map, the player must kill all monsters. Level ends as soon as no monster is left in the game. Monsters might exit the game on their own by reaching the surface.
- Each map in the game has a timeout value. If you don't find a solution before timeout you loose the game.

IV Code and Development Support

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

All game entities are represented by classes.

Each group develops an agent creating a client that implements the protocol exemplified 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 point (in 20).

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 the game engine, it is expected that there are some bugs and tweaks during the course of work. Be on the watch out for server modifications (configure the professor repository as an upstream repository and fetch/merge often) 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/yvOMUkkU` 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 and associates 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 an autonomous 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 made.
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 these games. The final score in a game is the one the agent has at the time of *gameover* (when the agent fails to exit the current map within the time limit) or when all maps have been completed.

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 10000 is mapped to 10/20. The median of the total scores 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%).