Design and implementation of a Tetris game and search algorithms in Java

BCS Project 1.1 Phase 2

Abstract

The growing popularity of video games has led to the development of intelligent agents capable of achieving remarkable gameplay performance in various games. This paper explores the design and implementation of a Graphical User Interface (GUI) for the classic game of Tetris. The goal in this case was to create an intuitive and user-friendly Tetris experience using the Java Swing library. This paper also presents an in-depth exploration of the design, development and implementation of a Tetris-playing bot by analyzing and evaluating the performance of different search algorithms.

Introduction

The game of Tetris has become an icon in the history of video gaming since its conception in 1994, entertaining generations of players with its simple but captivating gameplay. Despite the essence of the gameplay remaining mostly unaltered since its creation, the gaming landscape has instead experienced profound transformations, especially with regards to the introduction of Machine Learning models (AI Agents, or bots) in modern video games (Whitten, 2023).

Within the field of Artificial Intelligence, the development of a Machine Learning (ML) algorithm capable of beating humans at classic games, such as Chess or Go (Metz, 2016) has been a fundamental research area for its progress. In this context, challenging ML models by having them 'play' the world favorite game of Tetris has proven to be a viable field of study.

This paper will first explore the implementation of a user-friendly and modern Graphical Interface for the game of Tetris with the use of the object-oriented programming language Java and the Java Swing library as the sole UI library. Our aim is not only that of the mere recreation of the game's visual interface, but to delve into the intricacies of designing a dynamic and intuitive GUI that introduces modern aesthetics and usability while honoring the classic elements of Tetris.

Differently from the original Tetris game, this paper will explore an implementation that makes use of Pentominoes instead of the classic Tetris pieces. Such pentominoes have been implemented in a 2 dimensional matrix during the Phase 1 of the project, and all of their possible rotations have been produced algorithmically [REFERENCE?]. Furthermore, the UI taken from the simple Pentomino game of Phase 1 will be a starting step for the implementation of the Tetris game, as it already provides a possible solution to the required grid interface. [REFERENCE?]

Secondly, the paper will focus on the development of an algorithm capable of constantly achieving a high score at the implemented Tetris game. As mentioned previously, this challenge has been part of an extensive area of research in regards to the progress of Artificial Intelligence, and numerous models have been published to as a solution to this problem: in 2009 researches used NEAT, an algorithm for evolving artificial neural networks, to develop a neural network capable to perform well in the game s(Risto Miikkulainen & Stanley, 2009), and in 2017 Deep Q-Learning, a type of reinforcement learning algorithm, has been applied to Tetris in alignment with the broader trend of applying deep reinforcement learning to various games (Mnih et al., 2015)

This paper will not attempt to describe a design for a machine learning algorithm for Tetris, but will instead explore and compare various search algorithms that can be employed to explore and evaluate potential moves, such as Depth-First Search, Breadth-First Search, Minimax Algorithm and Alpha-Beta Pruning.

Depth-First Search (DFS), an algorithm that explores as far as possible along each branch before backtracking. In Tetris, such an algorithm could involve exploring possible moves or sequences of moves to find an optimal solution. (Cormen et al., 2009)

Breadth-First Search (BFS) explores all the vertices at the same level before moving onto the next one. In the context of Tetris, this could be used to systematically explore different sequences of moves to evaluate their possible outcomes. (Cormen et al., 2009)

Minimax Algorithm is a decision rule for minimizing the possible loss for a worst-case scenario. (John Von Neumann & Morgenstern, 1944/2007)

Lastly, Alpha-Beta Pruning is an optimization technique for the latter algorithm that reduces the number of nodes evaluated in the search tree by eliminating suboptimal branches. (Knuth & Moore, 1975)

The following sections of the paper will explore the design principles underlying the development of the Tetris GUI, as well as the analysis and comparison of different possible implementations of the Tetris search algorithm.

Methods

This section will discuss the main elements and methods employed to accomplish the project's goals, distinguishing between the design and coding of the game's logic, the implementation of a graphical user interface, and the development of four different search algorithms based on the literature proposed previously. The implementation will be heavily focused on the OOP principles to better compartmentalize UI, Game Logic and Search Algorithms. This would also facilitate and streamline testing for the experiments.

Game logic

The game itself will be implemented according to the classical Tetris rules, with the mere exception that the classic Tetris pieces will be replaced by a set of Pentominoes. The game involves falling pentominoes that the player can move around and rotate during the fall to create complete lines. When a line is created, it is eliminated from the board and the player receives points. The game also presents the possibility for the player to preview the next three pentominoes that will fall.

Graphical User Interface (GUI)

The Tetris GUI will be implemented using the Java object-oriented programming language, making use of classes imported from the Swing library such as JPane, JFrame and JButton. It will be designed to provide an intuitive and user-friendly gaming experience, as well as to be visually appealing. Key components included a game board display, a score indicator, and a preview of three incoming pentominoes.

Search Algorithms

This paper will focus on the implementation of four different search algorithms (Depth-First, Breadth-First Search, Minimax and Alpha-Pruning Search). These algorithms will be implemented in different files and as separate components entirely, but they will present the same class attributes to facilitate testing.

Implementation

The UI is being implemented with Java Swing, but is not complete as of yet.

Experiments

The Experiments we expect to work on are:

- Testing the Game Logic Functions, to make sure they run as expected and as efficiently as possible
- Testing and comparing the four search algorithms with a specific set of parameters.

Results

Results have not been produced as of yet.

Discussion

Discussion, evaluation and comparison of the implications of the Tetris algorithms will be produced following the results of the experiments.

Conclusion

As no results have been produced yet, a conclusion cannot be established.

References

Phon-Amnuaisuk, S. (2015). Evolving and Discovering Tetris Gameplay Strategies. Procedia Computer Science 60(1):458-467. https://doi.org/10.1016/j.procs.2015.08.167.

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). *Introduction to Algorithms* (Third). Mit Press.

John Von Neumann, & Morgenstern, O. (2007). *Theory of games and economic behavior*. Princeton University Press. (Original work published 1944)

Knuth, D. E., & Moore, R. W. (1975). An analysis of alpha-beta pruning. *Artificial Intelligence*, *6*(4), 293–326. https://doi.org/10.1016/0004-3702(75)90019-3

Metz, C. (2016, January 27). *In Major AI Breakthrough, Google System Secretly Beats Top Player at the Ancient Game of Go*. Wired. https://www.wired.com/2016/01/in-a-huge-breakthrough-googles-ai-beats-a-top-player-at-the-game-of-go/

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Risto Miikkulainen, & Stanley, K. O. (2009). *Evolving neural networks*. https://doi.org/10.1145/1570256.1570410

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence : a modern approach* (3rd ed.). Pearson.

Whitten, M. (2023, May 17). *Why we're excited about AI at Unity*. Unity Blog. https://blog.unity.com/news/why-we-are-excited-about-ai