Tetris-AI - Triste

MOLLY-MITTON Clément
College of Engineering
dept. Computer Science
Paris, France
clement.mollymitton@gmail.com

BENDAVID OUYOUSSEF Sarah
College of Engineering
dept. Computer Science
Paris, France
sarahbendavid@dartybox.com

HUBNER James
College of Arts and Sciences
dept. Criminal Justice
Alabama, United States
jhub@uab.edu

ARROUET LE BRIGNONEN Aubin
College of Engineering
dept. Computer Science
Paris, France
aubin.arrouet_le_brignonen@edu.devinci.fr

MOBLEY Erin
College of Engineering
dept. Computer Science
Florida, United States
erin.mobley@email.saintleo.edu

***Abstract*—This project proposal aims to delve deeply into the game Tetris, focusing on two essential aspects. Firstly, it involves faithfully recreating the game Tetris and conducting an in-depth analysis of its functioning, design, and history. Secondly, this project seeks to develop an Artificial Intelligence (AI) capable of mastering Tetris to the extent of never losing.**

The first part of the project entails a meticulous analysis of Tetris, breaking down its mechanics, design principles, and historical evolution. This analysis will shed light on the elements that have contributed to making Tetris an enduring cultural phenomenon and explain its consistent appeal.

The second component of the project is equally fascinating, involving the development of an Artificial Intelligence (AI) system that can achieve exceptional mastery of Tetris, avoiding losses altogether. By creating an AI capable of surpassing human-like strategic thinking and adaptability within the Tetris environment, we aim to push the boundaries of AI capabilities and explore potential applications of this technology in the realms of gaming, problem-solving, and AI research.

ROLE ASSIGNEMENTS

Roles	Name	Task description
User	Erin	- Proofreading documents: Erin, as a native English speaker, will review and edit project documents for language accuracy and clarity. She will ensure that the content is linguistically flawless. - User Interface Development: Erin will be responsible for designing and developing the user interface, ensuring a user-friendly and visually appealing experience.
Customer	Aubin	- Resource Management: Aubin will oversee the allocation and utilization of project resources, ensuring efficient utilization of time and budget. - Support: Aubin will also have the mission of helping the various members in their tasks if necessary.
Software Developer	Clément / James	- Writing Code: James and Clément will be responsible for writing the code for the project, and implementing the game mechanics, AI algorithms, and system functionality. - AI Development: They will also focus on creating and fine-tuning the artificial intelligence (AI) component of the project, allowing it to excel at playing Tetris. - Documentation Assistance: In addition to coding, James and Clément will provide support in writing technical documentation, ensuring that the development process is well-documented.
Development Manager	Sarah	- Organizing the Project: Sarah will take charge of project organization, setting deadlines, scheduling meetings, and ensuring that project milestones are met. - Coordination: She will facilitate communication and coordination among team members, ensuring that everyone is aligned and working effectively towards project goals. - Technical Documentation: Sarah will be responsible for crafting technical documents that detail the project's architecture, design, and development processes.

I. Introduction

Motivation

Video games have always been a captivating realm of technological exploration and creativity, and among the most iconic titles in this industry, Tetris shines brightly. Created by Russian developer Alexey Pajitnov in 1984, Tetris has evolved into a global phenomenon, thanks to its addictively simple yet strategically rich gameplay. This puzzle game has enthralled entire generations of gamers, delivering a timeless gaming experience.

This is precisely why our interest was piqued by Tetris. While Tetris's rule set is very simple the multitude of strategies one can employ to strive for the highest score is particularly fascinating. Naturally, some strategies prove more effective than others. Given Tetris' universal recognition, we found it compelling to create an AI capable of playing the game endlessly while employing the most advanced strategies. To realize this project, we must rebuild Tetris from the ground up and design a sophisticated AI that excels in the game.

Developing an AI with the ability to indefinitely succeed at playing Tetris presents a complex problem-solving challenge. It necessitates critical thinking, the creation of innovative strategies, and the optimization of algorithms for maximum efficiency. Through this endeavor, we will not only employ and apply theoretical AI concepts but also enhance our software engineering skills.

Problem statement

The objective of Tetris is simple, Match the tiles to form a line so that it clears space in the board. Tetris was designed to be endless, However, there comes a point in the game where the player cannot continue. This happens when enough uncleared rows build up and there is no room left on the board to add the next generated game piece (tetrominoes).

With the inclusion of Artificial Intelligence, could the games be pushed to their limits, and a truly endless game of Tetris be achieved? How can we design a Tetris AI that not only places the tetrominoes efficiently but also can adjust to the gameplay mechanics of ever-increasing gravity with the randomness of block combinations, while still taking the player's inputs into account? The AI should still mimic the decision-making done by humans once it is aware of the upcoming tetrominoes, checking for the best placement, and executing its decision to allow the game to continue for as long as possible.

Research on any related software

Several AIs have been implemented in Tetris to cover all the different ways of playing the game. Indeed, Tetris has been implemented with different AIs over the years for several purposes: creating its computer opponents or creating its own recommendation system. Several videos have been posted on Youtube presenting AIs that could play Tetris, some indefinitely:

A. Code Bullet

The AI coded by Code Bullet [1] was reported as being the first AI to beat the human world record. It managed to get a score of around 15,000 points, he also said if he continued training and improving it, the AI could possibly become invincible.

B. StackRabbit

The AI coded by Greg Cannon [2] managed to break Tetris NES. It broke the game because when the game is played on the console, scores that are too high can cause the game to crash because of the way the score is calculated. After reaching a score of 102 million, the game crashed and so the AI broke the game and thus could be considered invincible.

C. Github

Making a Tetris-solving AI is a typical project for students, people who want to learn using AIs, and simply people who enjoy the game. A lot of people then posted their projects on Git Hub [3], consequently, there is a huge amount of GitHub projects for a Tetris AI.

II. Requirements

Creation of TETRIS

A. Game Grid

Create a graphical representation of the game grid using appropriate rendering techniques, clearly displaying the current state of tetrominoes and placed blocks.

B. Tetromino Generation

The tetromino generation will use a random generation algorithm. We would like, if possible, to use the exact random generation algorithm that the original Tetris uses to keep the maximum amount of fidelity possible.

C. Movement and Rotation

Development functions to handle the movement (left, right, down) and rotation of tetrominoes based on player input. Tetris also has more advanced mechanics in which when the player rotates the tetromino, this one stays in place and doesn't descend, this function should also be developed.

D. Controls

The control system in a Tetris game is a critical component of ensuring an enjoyable and engaging player experience. It should offer responsiveness and intuitiveness, allowing players to manipulate Tetris pieces with precision.

E. Collision Detection

Implement collision detection algorithms to determine when a tetromino reaches the limit of the grid or collides with another tetromino, triggering actions like line clearing.

F. Line Clearing

Implement the logic to detect and clear complete lines when a tetromino fills a row, adjusting the game grid and influencing the score.

G. Scoring System

Design and implement a scoring system that rewards points based on the number of lines cleared simultaneously (single, double, triple, or Tetris) and the level of the game.

H. Level and Game speed

The game speed of Tetris increases when the score increases. The game speed of Tetris increases by steps rather than linearly and is capped after reaching a certain score, the precise speeds and the scores at which they are used in the game should be the same as in the original Tetris.

I. Game Over

Define conditions for ending the game, typically when the game grid becomes too congested to accommodate a new piece. Display a game over screen that shows the player's final score and provides an option to restart the game.

J. HUD Elements

Design and integrate HUD elements, including score display, level indicator, and upcoming tetromino preview, providing essential information to the player during gameplay. For the AI, these functions should also be inputs that are directly provided to the AI so it can play without having to use the graphical interface. This interface is simply used for the human users of the game.

Creation of the AI

K. Data Collection

The AI system needs access to a comprehensive dataset of Tetri's gameplays from which it can learn. This dataset should encompass a variety of game scenarios, including different starting positions, levels, and player actions.

L. Input Processing

The AI should possess the ability to process the current game state, which includes the current Tetrimino, the matrix of filled blocks, and the current score.

M. Strategy Development

The AI system must develop a strategic approach to decision-making regarding the placement of Tetriminos. This strategy should take into account various factors such as maximizing score, clearing lines, avoiding stacking blocks, and ensuring long-term success.

N. Adaptability

The AI system must be designed to adapt to different levels of difficulty and game speeds. This adaptability ensures that the AI can effectively handle various game scenarios and continue to perform well as the game progresses.

By fulfilling these general requirements, it is possible to design and develop an AI system that can efficiently learn and play Tetris. However, it is important to note that the AI will require continuous refinement, debugging, and iteration to improve its performance and ensure its success in various game scenarios.

III. Development environment

A. Choice of software development platform

Software platform

Our project is being carried out on the Windows software platform. Windows is compatible with a wide range of Python development tools, including VisualStudio Code and IDLE, which make it easier to debug code. Windows also supports the Python libraries and frameworks commonly used for AI development, including Pygame [4] and PyTorch [5]. Our project is a game, which needs to be manipulated by as many people as possible. Our aim is to be able to share it with users. Windows seemed the best choice, because it supports a wide variety of hardware configurations. This means that if we develop our game, it will be compatible on a wide range of computers. As we are all familiar with the Windows environment, it was an obvious choice for us.

Programming language

We decided to create our game in Python, mainly for its simplicity and readability, which makes it easier to create Tetris and implement the AI. Python is a rich computer language, full of libraries that make game creation easier. It has a vast collection of frameworks that are useful for game development (notably Pygame [4]). Python has also become a programming language that is frequently used to create AI (thanks in particular to the Sickit-learn and PyTorch libraries [5]). Because Python is an interpretive language, we can develop, test and iterate quickly. The debugging tools are effective and, being familiar with the language, we can move forward more quickly and correct our mistakes. That's why Python seemed to be the best choice.

Development Environment

The project will be developped using Spyder, which is a free and open source development environment for python coding. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package. In terms of computer ressources, we have access to five laptops, including two gaming laptops with advanced consumer available CPUs which could be helpful to the free training of the AI.

Cost estimation

By creating our code from scratch in Python, the project will cost nearly nothing. All we need to do is download Python (a programming language that can be downloaded free of charge). By downloading Python, various libraries can be installed, also free of charge. These are the ones we will be using to create our AI. Furthermore, to train our AI, we will likely use AWS, which costs almost nothing when we select certain specific configurations.

B. Software in use

Teams

Teams is an important tool for organizing our project. It allows us to assign tasks to each team member on a weekly basis and track the progress of these tasks. It also serves as the platform for our video conferences. Additionally, Teams facilitates file sharing, real-time document collaboration, and link sharing within our project channel.

KakaoTalk

KakaoTalk will be our primary tool for quick and spontaneous communication. Through our dedicated channel, team members can keep each other informed about tasks, their status, and deadlines. It provides an ideal platform for asking questions and staying updated.

LaTex

LaTeX serves as our primary document creation tool, enhancing the efficiency and quality of our project documentation. It excels in formatting, versatility, and facilitating collaboration. It ensures clear and standardized formatting, allows easy inclusion of complex elements, and simplifies version management and collaboration in our project.

Overleaf

To meet our project requirements, we have adopted Overleaf as a central platform for creating well-organized, scientific reports detailing the progress and completion of our project. Overleaf empowers us to work collaboratively on LaTeX documents, facilitating seamless and efficient document creation, editing, and revision.

GitHub

GitHub is a key tool in our project's software toolkit, streamlining version control, collaborative development, and project management. It serves as a web-based platform for hosting project source code, ensuring accessibility to all team members. Integration with Git simplifies remote storage management, while enabling seamless collaboration in our project. It facilitates concurrent work on different project aspects and eases code review and integration to maintain software quality.

PyGame

Pygame [4] is a library ideally suited for 2D games, making it a perfect choice for implementing Tetris. It offers a set of tools for managing 2D graphics, such as creating a game window, rendering Tetris pieces, and handling player input. Due to its versatility and simplicity, Pygame is a wise choice for developing Tetris.

Pytorch

Our Tetris game will also be played by an AI, developed using Pytorch [5]. Pytorch is particularly well-suited for reinforcement learning algorithms, which are essential for training the AI to play Tetris. Pytorch can also simulate the game environment and allow the AI to interact autonomously while learning from its experiences. It enables the AI to generate actions, such as moving and rotating pieces, using a neural network. Importantly, the AI can continue learning in real time without interrupting the game, making Pytorch a complementary tool to Pygame, which provides the game interface and piece manipulation capabilities.

C. Task distribution

IV. Specifications

Creation of TETRIS

A. Game Grid

The game grid is represented as a two-dimensional data structure, specifically, a list of lists of integers. In this grid, the value 0 denotes an empty cell, while the value 1 signifies a cell that is occupied by a block. Additionally, a secondary matrix is employed to store the requisite color information for rendering the game grid.

B. Tetromino Generation

Each game piece consists of three key attributes: a shape, a color, and its position coordinates within the grid. To ensure diversity in the generation of these pieces, we employ Python's "random" [6] library, which allows us to select randomly the colors and shapes.

C. Movement and Rotation

Facilitating player control over the game pieces involves verifying the validity of the requested movements and executing them accordingly. For horizontal (right and left) and vertical (down) movements, the piece's coordinates are simply updated in the respective direction. When it comes to rotation, we perform a 90-degree rotation of the matrix representing the piece's shape.

D. Controls

Players can interact with the game pieces using the arrow keys: left, right, and down for lateral movement, and the up key for a 90-degree rotation.

E. Collision Detection

Collision detection plays a critical role in gameplay by ensuring the legality of movements. We incorporate conditions within the verification method of the grid to ensure that the coordinates stay within the grid boundaries and do not conflict with cells already occupied by other pieces.

F. Line Clearing

To clear complete lines, we will simply check if an entire row of the board is composed entirely of 1's. In that case, we will remove that row from the list of lists and add a new empty row to the beginning of the grid. To detect and remove completed lines, we simply examine each row of the grid to ascertain if it consists entirely of 1's. In such cases, we remove that row from the list of lists and append an empty row at the top of the grid.

G. Scoring System

Scoring in the game is incrementally updated each time the board lines are cleared. Whenever a line is successfully cleared, the player's score is increased accordingly.

H. Level and Game speed

Controlling the pace of the game is achieved by adjusting the frequency at which tetrominos descend. By manipulating the clock, we can reduce the time interval between piece drops, effectively increasing the game's speed as the player progresses to higher levels.

I. Game Over

The game over condition is triggered when it becomes impossible to place a newly generated piece due to the presence of obstructing pieces. A dedicated function will be implemented to detect and respond to this scenario.

J. HUD Elements

For rendering various game elements, the "pygame" [4] library is employed. This library enables the creation of a customized game window that can be tailored to present the game in a visually appealing and informative manner.

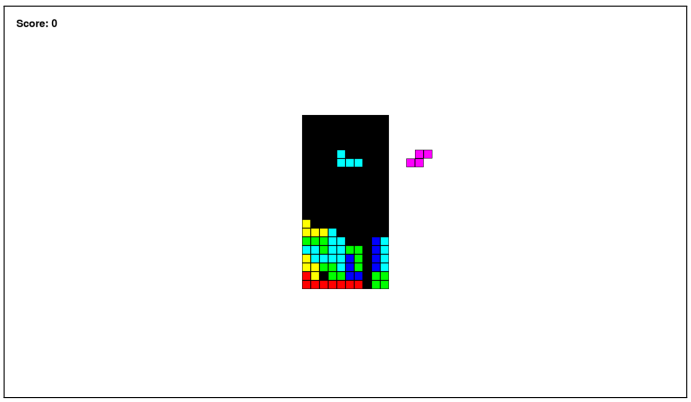

Fig. 1. Our first display of Tetris

CREATION OF THE AI

K. Data Collection

In order to train the AI for playing Tetris, we need to collect a dataset of game states and corresponding actions. This involves recording the current state of the game grid, including the positions and shapes of the tetrominoes, as well as the potential actions that can be taken. The dataset should also include the corresponding action that the AI should have taken in that state. This data can be collected by simulating gameplay with human players or through self-play.

L. Input Processing

Once the dataset is collected, we need to process the input data to prepare it for training the AI model. This involves converting the game state information, such as the positions of the tetrominoes and the game grid, into a suitable format that can be fed into the AI model. The input data may also need to be normalized or preprocessed to ensure consistent and meaningful representations.

M. Strategy Development

The AI needs to develop a strategy for playing Tetris effectively. This involves training a machine learning model, using the collected dataset. The model should be trained to predict the best action to take given a particular game state.

N. Adaptability

To make the AI adaptable, it should be able to learn and improve its strategies over time. This can be achieved by continuously updating and retraining the AI model with new data as more gameplay is recorded. The model should be able to generalize its knowledge and adjust its strategies based on the changing game dynamics and patterns observed in the data. This adaptability will allow the AI to become increasingly proficient at playing Tetris.

REFERENCES

[1] Code Bullet, "I Created An A.I. to DESTROY Tetris", YouTube, [Online]. Available : https://youtu.be/QOJfyp0KMmM?si=dKykHlZGzpcrUu4L [Accessed Oct. 7, 2023]

[2] Greg Cannon, "AI BREAKS NES TETRIS! - 102 MILLION and level 237", YouTube, [Online]. Available : https://youtu.be/l_KY_EwZEVA?si=QNukpvk903yk_8xA [Accessed Oct. 7, 2023]

[3] "Tetris ai", Github, [Online]. Available : https://github.com/search?q=tetrisai&type=repositories [Accessed Oct. 7, 2023]

[4] "Pygame Front Page", Pygame, [Online]. Available : https://www.pygame.org/docs/ [Accessed Oct. 10, 2023]

[5] "PyTorch Front Page", PyTorch, [Online]. Available : https://pytorch.org/ [Accessed Oct. 10, 2023]

[6] "random — Generate pseudo-random numbers", Python Docs, [Online]. Available : https://docs.python.org/3/library/random.html [Accessed Oct. 10, 2023]