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. Movement Decision Strategy

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. 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. Finnaly 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.

L. Training

The primary goal of training the AI is to refine its decision-making capabilities in Tetris. By exposing the AI to various game scenarios and optimizing parameters, we aim to enhance its ability to make strategic and effective moves.The training process seeks to identify and refine the optimal configuration of parameters governing the AI's decision-making. This involves adjusting parameters.
The ultimate objective is to improve the AI's overall performance in playing Tetris. This includes achieving higher scores, clearing lines more efficiently, and demonstrating a strategic understanding of the game dynamics.

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]. 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). 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.

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.

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" [5] 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.

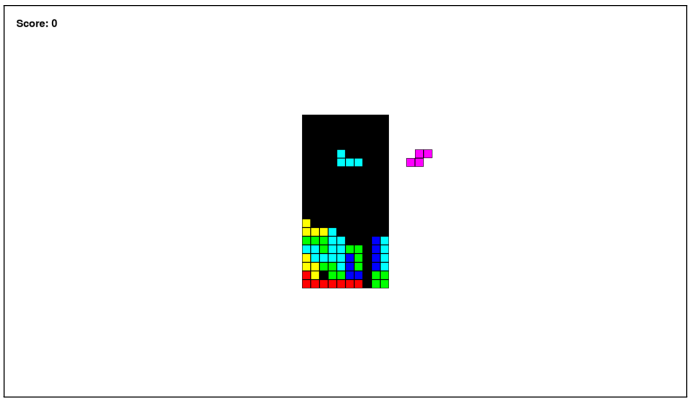

Fig. 1. Our first display of Tetris

CREATION OF THE AI

K. Movement Decision Strategy

Developing an adept Tetris-playing AI requires a meticulous approach to evaluate the desirability of each potential move. Our strategy involves identifying pivotal parameters crucial for the AI's decision-making process. These parameters include metrics such as hole count, bumpiness, lines cleared, blocks in the rightmost column, average peak, open holes, and maximum line height. The incorporation of these parameters empowers us to calculate the cost associated with every conceivable move, considering both the current active piece and the upcoming piece.

L. Training

1) Genetic Algorithm: To pinpoint the optimal parameters, we employ a genetic algorithm, allowing our AI to undergo iterative improvements across generations. This entails creating a diverse population of AIs, implementing a mutation method to introduce genuine genetic diversity, and subsequently identifying and reproducing the most successful AIs from each generation.

2) AI Population Formation: Initiating the training process necessitates the establishment of a robust AI population. This involves creating a substantial number of AIs to comprehensively explore the parameter space and identify the most effective configurations. Additionally, the development of essential functions is imperative for the seamless functioning of the training process. These functions collectively contribute to the AI's evolution and proficiency in playing Tetris.

V. Architecture Design Implementation

A. Overall Architecture

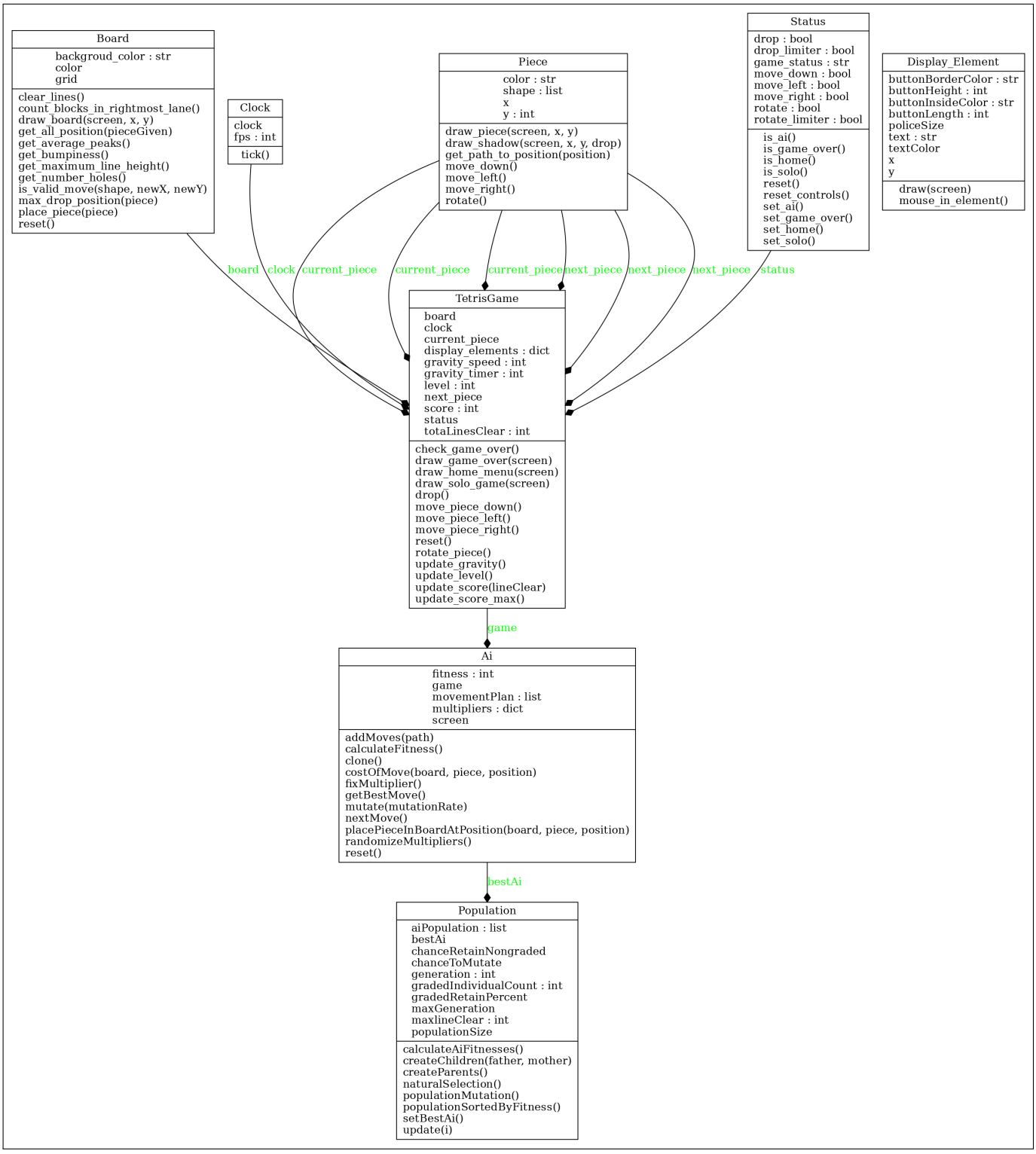

Fig. 2. UML

B. Directory organization

Directory	Files name	Description
Tetris-AI	tetrisgame.py	- The purpose of this code is to create our Tetris game base. It uses nearly all the class in the other py files to create a Tetris game. The current piece is managed using methods such as move_piece_left, move_piece_right, move_piece_down rotate_piece and drop methods - The class updates the score with update_score and the level with update_level according to the lines cleared. The graphical display, including the home and end of game screens, is handled by the draw_home_menu, draw_solo_game and draw_game_over methods using the Pygame [4] library.
Tetris-AI	piece.py	- The Piece code defines a Piece class to represent the pieces in the Tetris game. Each instance of the class is a piece with a random shape, a corresponding colour, and x and y coordinates for its position. This code can be used to direct the part to the left, right or bottom, to rotate it, and to draw the part and its 'shadow' on the screen. Moreover The get_pathto_position method returns the list of movements necessary to move the piece to a specific position.
Tetris-AI	population.py	- The population code implements a genetic algorithm to evolve an AI population in Tetris. It promotes the reproduction of the best adapted individuals. The Population class manages the initial creation of the population, natural selection, mutation and the creation of new generations of AIs.
Tetris-AI	population.py	The update function is used to progress each individual through the game simulation, calculateAiFitnesses to evaluate the performance of each AI, naturalSelection to carry out natural selection.
Tetris-AI	training.py	- The aim of the training is to train an AI capable of maximising the number of lines deleted in the game Tetris. It uses the Population class by instantiating it, then iterating over the generations, updating the individuals and performing natural selection.
Tetris-AI	status.py	- The Status class in Python is a status handler for the Tetris game. It includes methods such as set_game_over, set_home, set_solo, set_ai to change the status of the game. The is_home, is_game_over, is_solo, and is_ai functions are used to check the current state of the game. The class also provides reset methods, including reset_controls to reset controls, and reset to reset the entire state of the game.
Tetris-AI	main.py	- This code uses the Pygame library to implement the Tetris game, integrating essential Pygame functions such as keyboard/mouse event management with pygame.event.get() and display updating with pygame.display.update(). Tetris features are managed by the class instance of tetrisgame.
Tetris-AI	display.py	- This code implements a class called Display_Element, which creates graphical elements, such as buttons, for the Tetris user interface. Moreover this class contain methods that are really useful for the user action detection like the mouse_in_element(), which checks whether the mouse is on the element.

Tetris-AI	constants.py	- This code defines useful constants for the game, including the dimensions of the grid (gridWidth and gridHeight), the size of each cell (cellSize), the frame rate (fps), and the maximum possible score (maxscore). It has also a lists the different shapes of Tetris pieces with their representation in matrices of 1 and 0, as well as the colours associated with each shape (color_per_shape). Moreover, include colour dictionary (colors) to represent the different colours used in the game. -Finally it as a maxScore line for the save of the best score. All these element necessary to ensure that the game is consistent and easy to maintain and having them in one file is pratical.
Tetris-AI	clock.py	- This code defines a Clock class in the Pygame module. The class is used to manage time and updates at a specific rate of frames per second (fps). In the constructor (__init__), a pygame.time.Clock object is initialised, and the constant fps from the constant module is assigned to the fps attribute of the class. The tick method is defined to obtain the time elapsed since the last call to the tick function and maintain the defined frame rate. This class is useful for controlling the speed of the game, and ensuring that it runs at a constant rate.
Tetris-AI	board.py	- The Board code uses the Pygame module. It tests the validity of piece movements with is_valid_move, place_piece to place a piece on the board, clear_lines to clear complete lines. - Also max_drop_position to determine the maximum drop position of a piece, and methods evaluating various aspects of the grid such as the height of peaks (get_average_peaks), irregularity (get_bumpiness), number of holes (get_number_holes), and so on.
Tetris-AI	ai.py	- AI functions include calculateFitness to evaluate the AI's performance, getBestMove to determine the best possible move, mutate to introduce mutations into the evaluation multipliers, and clone to create a copy of the AI. The AI uses multipliers to evaluate moves according to various criteria, such as the number of holes, the irregularity of the board, the number of lines erased, and so on.

C. Module 1 : Main

The purpose of main is to provide the user with the ability to choose between a solo game, the AI game and exiting the program. Moreover, when Solo game is selected, main will check the key inputs of the player and assign the correct actions to the Tetris game. Finally, when the player loses, it shows the game over screen. If AI game is selected, main will create a tetris game and then use the AI class to play it. For each move, main calls getBestMove function from the AI class to determine the best move. After the best move is determined, main will go to the next move. This is repeated untill game over. Main's components are: display (the display of the game), game (which is an instance of the Tetris game class), ai (which is an instance of the Ai class) and running, which is used to check if the user wants to exit the program, it is initialized as true at the start)

D. Module 2 : Constants

As it's name suggests, constants is a file in which all of the global constants necessary to the well functioning of the program are stored. Those include constants for the board such as the grid width and height and cell size. It also stores the frames per seconds, the best score ever on the game, the different shapes of the tetrominos and the different colors of the shapes and all the color useful for the display of the game.

E. Module 3 : Clock

The clock class is simply used to act as the clock in the game, adapting it's ticks to the fps variable stored in constants

F. Module 4 : Board

Board has two main variables, the grid which stores the different cells that are filled by placed tetrominos, and the colour matrix which stores the colours of each cell for the display of it to the player/ observer. Board has several functions that are used both by solo players and the AI, those functions are used in piece movements and placement, such as verifying if a movement is valid (not out of bounds or clipping with already placed tetrominos), placing a piece on the board, clearing completed lines. Also, the function draw_board manages the display of the board, and reset_board resets it Moreover, Board also has specific functions for the AI. In those, some functions help to extract some data from the board for the AI to use, such

as the height of the highest column, the average height of the columns, the bumpiness, the number of holes in the structure of placed tetrominos and finally the number of filled cells in the right column (this helps determine the chances of making a Tetris). Also, the function get_all_positions will return all of the possible positions in which a piece can be placed on the board in the current configuration, this function will allow the AI to determine the best move according to it's model. All of the functions of board are written here because of their strong reliance on using data from the grid of the game to display the game, check the validity of moves, and extract useful data for the AI

G. Module 5 : Piece

Piece is initialized by creating a random shape from the shapes array in constants. This piece can then be used in the Tetris game. The piece class has functions dedicated to movement (right, left, down, rotate), display of the piece and it's shadow on the drop position right under it's current position, and finally it has the get_path_to_positon which is useful for the AI to write the necessary movement the piece need to do to attain a specific position on the board.

H. Module 6 : Display

Display is a class used to make the tools that are used to display information to the rest of the program or button. This class was principally created to facilitate the creation of display element and also the detection of the user behavior for each element created.

I. Module 7 : Status

Status stores different data on the status of the game. The variable game_status can be "Home" for the home screen, "solo" for the solo game, "Ai" for the AI game and finally "game over. Status also stores data relative to the movement of pieces such as if it's moving right, left, down, rotating and dropping. Finally it has a function to reset the controls and to resets itself entirely.

J. Module 8 : Tetris

Tetris game stores the main tools used to make a Tetris game work. It imports the constants grid width,grid height,cellSize and maxscore. It also imports an instance of Board, Clock, Piece, Status and Display_Element. With those, it also has specific variables such as the current and next pieces to be played, the score, the level, the gravity (speed at which the pieces fall), the game status and the display elements associated with that status. First Tetris is used to display all of the home screen (Home solo button, home ai button, home exit button, best score, etc), and does similarly with the game over screen and the in game screen. Then it has functions dedicated to game controls for moving left, right, and rotate while checking that the moves are valid. It also checks for game overs. Finally, it also has tools to update the game, such as updating the levels and gravity (speed at which the piece falls), updating the score and displaying those updates on the in game screen.

K. Module 9 : AI

The AI class is the main tool to use an AI according to a model to play Tetris. The AI class is not only used to play the game, but also provides useful data for the training using a genetics algorithm. The AI class imports TetrisGame. In terms of variables, AI has the current game being played for the instance of TetrisGame, multipliers for the different data collected, the fitness of the AI (used in training), the movement plan (the list of movements to do in order in the game) and a screen. In terms of functions, AI has two functions for the multipliers of data, one is fixed and predermined for the best multipliers currently found, and the other is to set the multipliers randomly. The calculatFitness function calculates how good the AI is depending on it's best score. The nextMove function applies the first move (moving right, left, rotating or dropping) and switches to the next one to do. The placeInBoardAtPosition function directly places a piece at a specific position. The costOfMove function calculates how good a specific move is for the AI depending on the multipliers that were previously defined. getBestMove chooses the best move amongst all of the possible moves in the current game configuration. Then AI has two functions for the genetic algorithm part: mutate and clone. Finally it has a reset function.

L. Module 10 : Population

The population class has for goal to manage a population of AIs through different generations, it creates generations and then has functions to make them evolve through time with a genetics algorithm. Population imports AI. It has several class variables, those are populationSize, maxGeneration (maximum number of generations), chanceToMutate, gradedRetainPercent (the percentage of the best AIs to retain), chanceRetainNongraded (chance to retain bad AIs), screen and then AI population which is a list of AIs. In terms of functions, population has one for updating the AI, calculating the overall fitness for all AIs, sorting the population based on fitness, and selecting the best AI in the generation. Moreover population has functions for keeping the AIs (the parents) and creating offspring to complete the population (createParents and createChildren). Additionally, Population has functions to potentially mutate the population and the function responsible for the entire natural selection process.

M. Module 11 : Training

The training.py file can be executed to train the AI. It uses population.py to manage the different generations and use the genetics algorithm. In training you can set the parameters you desire for the training in the following variables: populationSize, maxGeneration, chanceToMutate, gradedRetainPercent, chanceRetainNongraded While the number of generations or the chosen "win" criteria (for us it was clearing 1000 lines) isn't completed it repeats the following operation: executing a population generation through using multithreading (which was faster than multiprocessing for our case) and applying the genetics algorithm to create a new generation. It also prints the

max number of lines cleared by an AI to check the progress made during each generation. Finally, it prints the generation at which the ”win” criteria was completed and shows the time it took.

VI. Use Cases

1) Game Launch and Menu Screen:
When the user launches the game, they will be presented with the following Menu Screen. From this Screen, the user can select to run the Tetris game in Solo Mode, AI Mode or Exit the game.

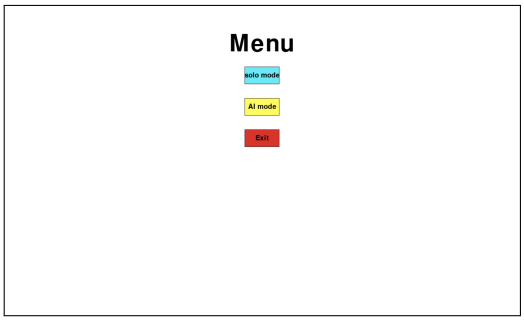

Fig. 3. Screenshot of Triste - Menu

2) Solo Mode:
When the user clicks the “Solo Mode” option, they will be brought to the following screen showing a basic game of Tetris. To control the movement of the Tetrominoes, the user can use the Left and Right arrow keys to move the game pieces, the Up arrow key to rotate the game pieces, the Down arrow key to drag the game pieces down slowly, and the Spacebar to drop the game pieces.

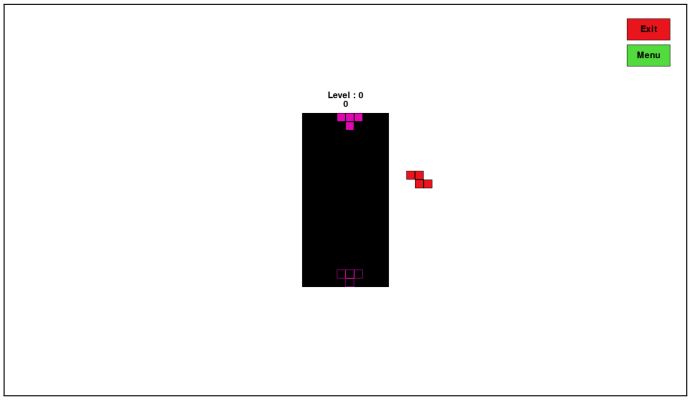

Fig. 4. Screenshot of Triste - Solo Mode

3) AI Mode:
When the user clicks the “AI Mode” option, the user will be brought to a screen exactly like the screen displayed in Solo Mode, but the AI will do all of the work to play the game until the limit is reached and the Game Over screen is displayed.

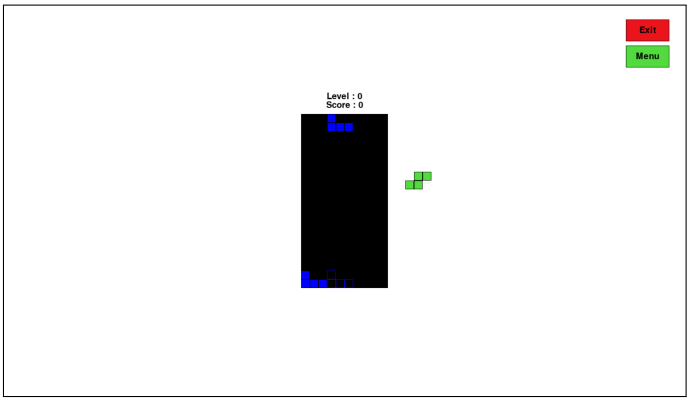

Fig. 5. Screenshot of Triste - AI Mode

4) Game Over:
Regardless of whichever mode the user selects, when the game has reached its limit and no more Tetrominoes can be placed, the Game Over screen will be displayed. From this screen, the user will see their score for the current game, and the best score achieved from the game. Then, the user can chose to either restart in the mode that was previously selected, return to the main menu and select a different mode, or exit the game. When the User exits the game, the window is closed.

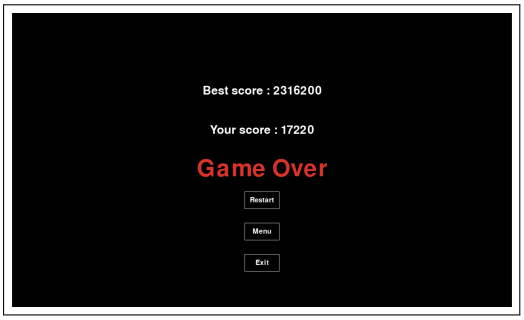

Fig. 6. Screenshot of Triste - Game Over

VII. Software Installation Guide

To install our game, you must first install a Python distribution (https://www.python.org/). You must then install the various libraries necessary to run it, using the command ’pip install -r requirements.txt’ in the command prompt. Finally, just run the main.py file and the game starts.

VIII. Discussion

The realization of our project has been marked by various difficulties. Firstly, the lack of prior experience in the field of artificial intelligence posed a major challenge. None of our team members had a deep understanding of the methods to

adopt for developing an AI, leading to uncertainties about the appropriate approach and slowing down our initial progress.

Additionally, communication within the team proved to be a challenge, resulting in significant disparities in task distribution. Difficulties in clearly assigning responsibilities had a negative impact on team cohesion and effectiveness. This lack of coordination led to delays and additional complications in the project's execution.

Concerning the training of the AI, we encountered significant issues. Despite our efforts, the application of various genetic algorithm methods did not yield satisfactory results. Moreover, these approaches consumed a considerable amount of time, adding time pressure to our existing challenges. Attempts to reduce the training duration faced obstacles, with a generation of 100 AIs playing 100 Tetris games taking over an hour.

Lastly, the investment and interest in the project were not uniform across the team, creating a significant imbalance in individual motivation. This disparity affected the quality of work and overall team cohesion. To overcome these obstacles, it is imperative to strengthen communication, explore new training approaches, and rebalance the commitment of each member to ensure the future success of the project.

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