Table of Contents

About the Program

Structure of Project

Details about the structure of the project can be found in the README.pdf file.

src/ → the source files of the project
- src/main → the sources files for the main program
  - src/main/java → the .java files for the main program
    - src/main/java/ai → the .java files relating to the AI component
    - src/main/java/backend → the .java files relating to the backend processing
    - src/main/java/frontend → the .java files relating to the GUI components
      - **src/main/java/Game.java → entry point into the main program**
  - src/main/res → the resource files for the main program
- src/test → the source files for the tests for the program
  - src/test/java → the .java files for the tests
    - src/test/unittests → the source file for testing individual components
      - src/test/unittests/backend → the tests for the backend components
      - src/test/unittests/frontend → the tests for the frontend components
      - **src/test/unitests/TetrisGameAITests.java → entry point for unit tests**
    - src/test/TestDataGenerator → generates test data for the unit tests
  - src/test/res → the resources files for the tests

lib/ → the external libraries for the project

Main Menu Options

Instructions

This opens a window that shows the keys used to control the game.

Single Player

This mode is where a human player plays by themselves. The game area is displayed on the left side of the window and the next Tetromino with the stats of the game are displayed on the right side of the window.

Play Against AI

This mode is where a human player plays alongside an AI. See Saving and Loading AI Files - Loading Organism for more details on loading an AI Organism. The AI game is shown on the left, the stats for both games in the middle, and the player controlled game on the right. The next Tetromino for the player is shown in the top of the middle section.

Watch AI

This mode is where an AI plays by itself. The game area is displayed on the left side of the window and the stats of the game with details of the organism are displayed on the right side of the window.

*Train AI

*In order to access this option, you must press the "A" key on your keyboard to enable advanced mode and show this button. There are two possible AI training modes: Fast Training and GUI Training. See Saving and Loading AI Files - Loading Population for more details on loading the populations. For more information about the AI algorithm, see Evolutionary Algorithm.

Both Training Modes

There are several features that can be found in both training modes.

Auto-Save to File: After every generation (training, selection, and breeding), the population is automatically saved to the file. As a result, even if the program crashes or is closed, only the most recent generation is lost. The previous generation is still saved to the file and can be loaded again to continue training.

Organism Name: The initial name of the organism is the string representation of a random UUID.

Training Time: The total time that the population has been training for. This includes the amount of time the population has been training in previous sessions.

Save Elite Button: This button saves the fittest organism in the elites of the population to a file and resumes training once saving is complete. See Saving and Loading AI Files - Saving Organism for more details.

GUI Training

This training mode displays the gameplay of the AI as it is training on the left side of the window and the stats of the training on the right side of the window. Each AI is only trained with one game per generation before the population is bred and evolved. This training mode was the initially intended to be the only training mode. However, since the gameplay was displayed, this training mode is quite slow. As a result, a faster Fast Training mode was created.

Fast Training

This training mode does not display the gameplay of the AI and all of the game logic is handled internally, without being displayed. The left side of the window shows the stats of the population with the stats of the fittest elite of the population and the right side of the screen shows the stats of the current training session. This significantly increases the speed of training, allow for many more generations to be trained in the same period of time. Each AI is trained with five games per generation before the generation is bred and evolved.

Exit Game

This closes the window. It is the same as pressing the "X" in the title bar of the window.

Game Logic

- The Tetromino can be moved left, right, down, or dropped.
- When a Tetromino has reached its final resting position, there is a short delay, and then a new Tetromino is spawned at the top.
- When a row is completely filled, it is removed and all the rows above are shifted one row down.
- The game ends when any of the columns in the top row are filled.
- The level of the game increases by one for every 10 rows that are cleared.
- The distance between the resting position of the Tetromino and the top of the game grid is added to the score.
- When a line is cleared, an additional amount multiplied by the number of lines that were cleared at once with the level number is added to the score.
- The Tetrominoes are spawned in random permutations of the seven possible Tetrominoes. This is the algorithm used to spawn Tetrominos that is specified in the Tetris Guideline. Details of this algorithm can be found here: http://tetris.wikia.com/wiki/Random_Generator.

Saving and Loading AI Files

Loading Population

The difference between Fast Training Mode and GUI Training mode is covered here: *Train AI. When this dialog opens, either a previously saved AI population can be loaded from a file or a new AI population can be created. When the first option to use a previous file is selected, you must then select the file that the population is saved in using the file chooser dialog that will pop up. If the second option to create a new population is chosen, the location to save the file must be chosen. The population file selected or loaded must have a file extension of .pop.ser in order to be considered valid.

Loading Organism

When loading an organism, there is an option to select or deselect AI Fast Mode. When this mode is enabled, the AI will immediately drop the Tetromino once it has been placed in the correct horizontal position. When this mode is disabled, the AI will only drop some of the Tetrominoes are they have been moved to their correct horizontal position. Some Tetrominoes will be left to drop slowly by themselves. This mode is designed to mimic an actual human playing the game and is commonly used in Player vs. AI mode so that the AI does not end up losing every quickly. It should be noted that whether this mode is enabled or disabled, it does not affect the score of the AI, it only affects the amount of time it takes for the AI to achieve the score as there is no penalty or bonus or taking more or less time when making a move.

Saving Organism

When saving a dialog, you must select a folder to save the Organism into. The name of the file that the Organism is saved to is determined by the name of the organism, which can be modified by changing the text in the "Name:" text field. When the "Save" button is pressed, the Organism will be written to the file.

Features & Limitations

- The windows of the game are designed to resize automatically according to the size of the screen (80% of the height or width of the screen, depending on which is smaller)
    - This feature has been tested on two screen resolutions:
        1. 1920 x 1080
        2. 1024 x 768
    - The program should be able to resize to any screen resolution above 1024 x 768
- The background music can be muted by simply pressing the "Mute" button in the window
- The AI Population (testPop.pop.ser) included in the project has been trained for a little over an hour
- The AI Organism (alpha.org.ser) included in the project is the fittest organism in the population
    - It has achieved a top score of 22 312 708 during training (my highest score was almost 30 000)

Explanation for Concepts Used

Inheritance

I used inheritance for three purposes in this project - to create a base class that can be extended by other classes, to modify an already existing class to fit my need, and to simplify the process of updating objects every frame.

All the different game modes (Single Player, Player vs. AI, AI Train) share many common components. These include the layout of the GUI and the logic behind the game with slight modifications between each mode. I placed all the methods and variables that will be common to all game modes inside an abstract class which is extended by the classes for each of the game modes. For example, all my GUI windows share many common components, such as a StatsBox on the side to display the stats of the current game. I created the variable for this StatsBox inside the abstract class, which would be inherited by the classes that extend it for each of the game modes. This allows me to avoid rewriting code and makes it easier for me to change the code for that class as I only have to change the code in the abstract class rather than changing the code in every class for each of the game modes.

The second way I used inheritance in this project is by extending classes that are already present in JavaFX to modify them according to my needs. For example, I create a StatsBox class which extends the VBox class in JavaFX. This allows me to add features I want in the VBox to the StatsBox class such as the spacing between elements, the outline, and the background colour. Instead of creating a VBox and then calling methods on it to modify its appearance every time I created a StatsBox, I simply have to create a StatsBox and the constructor for the StatsBox will automatically modify the appearance of the VBox.

Lastly, I created an Updatable interface which includes one abstract method (onUpdate). This interface is implemented by all classes that need to be updated every frame, allowing them all to be stored in an ArrayList. When it is time to update all the objects, I simply iterate through the ArrayList and call the onUpdate method on each of the objects.

Evolutionary Algorithm

The AI training algorithm I used for this project is an evolutionary algorithm. Essentially, it is designed to mimic the evolution process of a biological population in nature. The population starts off with a number of organisms, whose genes are semi-randomly generated (random within a range). The genes are the values that are used to determine what the best move to make is. Each organism contains a number of genes, each gene with a value. When it is time to make a move, the AI goes through every possible position that the Tetromino can be placed in (there are less than 40), which can be referred to as a move. The value for each of the genes is multiplied with the value of that corresponding state in the game (the height of the tallest column is multiplied with the value of the corresponding gene). The product of all of these genes and states are summed up to generate a score for that move. If that move causes the AI to lose the game, a large penalty is subtracted from the score of that move. The AI then makes the move that produced the state with the highest score. When the game is over, the score of that game is added to the organism's list of scores.

After all the organisms have played a specific number of games (that number is currently at five), the selection and breeding process begins. The fitness of each organism is calculated (the median of all the scores of the games that the organism played this generation) and the organisms are sorted according to their fitness. The fittest organism is added to a list of elite organisms so that even if the breeding produces a group of very unfavourable

genes, some good genes are still saved and will be introduced back into the population. The fittest 50% (this percentage may be changed) of the population will be added to the reproducing pool along with all of the organisms in the elite list. Two organism in this pool are randomly selected to breed and produce a child organism. For each gene, the gene from either parent is selected to be the child gene. There is a chance that a mutation may be introduced to the child gene, which will add or subtract a random value from the value of the gene. This breeding is repeated until enough child organisms are produced for the next generation. When this happens, the entire training and breeding is repeated.

Game Controller

The Game Controller serves as a connection between the player of the game and the game processor, which handles the game logic and inputs to the game. The Game Processor cannot be accessed directly, it can only be accessed through the Game Controller. Any inputs to the Game Processor (moving the Tetromino, controlling the game) has to go through the Game Controller. This prevents tampering with the game logic as the input can be verified by the Game Controller before it is sent to the Game Processor. Additionally, the Game Controller allows control of the Game Processor to be standardized, allowing for a common way to control the game, whether it is controlled by the AI or by a human player.

JUnit & TestFX

The JUnit and TestFX libraries were used to automate testing of the program. I used to them to create separate test cases to test each component of the program. The TestFX library allowed me to create unit tests for the GUI components of the program. This form of testing allowed me to quickly verify that most of the program was still running properly after I make changes to the code.

Serializable

The Serializable interface was used to allow the AI components to be saved to and loaded from a file. This interface allows objects to be serialized and saved to a file and then deserialized and loaded from a file at a later time. The Population and Organism classes implement this interface to allow them to be saved to a file and loaded again at a later time to be trained further or to be used in a game.

Additional Sources Used

http://tetris.wikia.com/wiki/Tetris_Guideline
http://cslibrary.stanford.edu/112/TetrisAssignment.pdf
https://www.youtube.com/watch?v=aeWmdojEJf0
https://www.youtube.com/watch?v=xLHCMMGuN0Q
https://en.wikipedia.org/wiki/Tetris
https://github.com/TestFX/TestFX
https://www.tutorialspoint.com/java/java_serialization.htm