**INSTALL INSTRUCTIONS**

**SECTION 1: DOWNLOADING SOURCE CODE**

GIT INSTALL

1. Install git for your operating system from <https://git-scm.com/downloads>
2. Once git is installed a program called git bash will be available to use or you can use the terminal of your OS to complete the following step.
3. Clone the Tetris repository using the command:

git clone <https://github.com/DCSHGAI/HGAITetris>

DIRECT DOWNLOAD

1. If you don't wish to use git you can always download the zipped files from <https://github.com/DCSHGAI/HGAITetris>

**SECTION 2: INSTALLING PYTHON AND REQUIRED DEPENDENCIES**

PYTHON

Assumptions:

Most of all, this package assumes the user has *some* familiarity with python (i.e. downloading, installing, and running python files on the OS of the user’s choice)

Requirements:

1. This Tetris tool was developed using **Python 3.9.7**
2. Install Python 3 from <https://www.python.org/downloads/> for the development of this program Python 3.9.7 was used.
3. **PyGame** is also required to run the code and a guide to install packages can be found here <https://packaging.python.org/en/latest/tutorials/installing-packages/>

Anaconda Version

1. Anaconda is one potential python package manager and toolkit
2. Anaconda can be at <https://www.anaconda.com/products/distribution>
3. After installing Anaconda, open an Anaconda Prompt, create a virtual development environment with Python 3.9 using the following command:

>> conda create –n name\_of\_your\_environment python==3.9

1. Activate the virtual environment using the command

>> conda activate name\_of\_your\_environment

1. Pip install needed dependencies

>> pip install pygame

>> pip install tensorflow

1. If needed

>> pip install numpy

Optional:

In order to run the sample AI tool that is provided with this package, the following dependencies are required

Tensorflow

Numpy

**SECTION 3: RUN THE GAME**

1. Make sure you are in the folder “Tetris” with files Tetris.py and RunTetrisGames.py

OPTION 1

1. Run the RunTetrisGames.py to spool up multiple games

This can be done

1. In the command prompt with the command: python RunTetrisGames.py
2. Double clicking on the RunTetrisGames.py script
3. Or in the IDE of your choice

OPTION 2

1. Run Tetris.py to run a single instance of the game either in editor or by double clicking on it

**SECTION 4: UNDERSTANDING THE GAME**

The code is set-up to allow multiple variants on the standard Tetris game and to allow users to insert AI controls at their discretion. An example AI has been provided in the file **StateEvaluation.py**

The sample AI has additional dependency requirements of Tensorflow and Numpy, which can be installed (depending on your OS) using the pip install commands

**Tetris.py**

This is the base version of the game.

To play the game one uses the arrow keys to move each piece into position:

*Up Arrow = Rotate Piece*

*Down Arrow = Move Piece Down*

*Left Arrow = Move Piece to the Left*

*Right Arrow = Move Piece to the Right*

*Space Bar = Move Piece All the Way Down*

*Escape = Restart the Game*

In addition, the following keys have been reserved to work with the example AI provided:

*A = Pass Control to the AI (or turn off the AI). This is a toggle switch*

*J = Give Encouragement to the Example AI (used for training)*

*K = Give Discouragement to the Example AI (used for training)*

*Q = Allow the AI to run at its “OWN” speed. This is a toggle switch.*

*G = Will save basic game stats to a csv file*

**Note:** If the AI is enabled and the user manually moves the piece using the arrow keys (i.e. overriding the AIs plan), the AI will automatically be disabled and full control will be given back to the user… not unlike intervening with the vehicle controls (i.e. pushing the brake) when a car is in cruise control.

**RunTetrisGames.py**

This function can be used to read a configuration file and spawn multiple instances of the game (Tetris.py)

Up to 10 instances can be spawned in parallel

In the Windows OS, the keys 0-9 can be used to toggle/switch between games.

**StateEvaluation.py**

This is an example AI provided with the Tetris game. The AI must be trained by the user in order to operate intelligently.

The AI function is called on each loop of the main Tetris.py function.

Therefore, the AI runs in serial with the main Tetris game. The AI \*could\* pause the game with a long planning step. However, this would limit the number of pieces/tetriminos per unit time, which would impact the overall score.

The AI is provided the entire state of the game and the keyboard events

The AI uses a compressed (low dimensional) hard coded feature set and linear neural network to evaluate the utility of each possible final position for the given piece and chooses the final position with the greatest value.

The AI then implements an action tree that moves the piece into that final position.

After each move is completed (and before the NEXT move is completed) the user can give positive reward (encouragement) or negative reward (discouragement) to the AI using the keys J & K (see above). In other words, reward will be assigned to the last *completed* move.

The AI receives no reward from the game

The AI uses the transition model encoded in the file TetrisSym.py to determine where a piece will land, given an action sequence.

The AI updates its model weights EVERY time that it receives a reward

The following keys are used by the example AI:

*S = save model weights to the “model” directory AND save current trials (including reward) to the “data” directory. Models and data are saved with the unique ID for that Tetris instance.*

*L = load all saved data from the “data” directory and train on it*

*This loads each file in the data directory in turn, batches the data, and updates the model using model.fit command*

*B = batch over all current data.*

**StateEvaluationBlank.py (note: MAY NOT BE MERGED YET IN THE CURRENT VERSION)**

If you do not have tensorflow installed and wish to use another package to implement your own AI, StateEvaluationBlank.py is provided for you.

**TetrisSym.py**

This function provides a copy of the Tetris class used for forecasting possible moves