JTetris

The provided JTetris class is a functional tetris player that uses your piece and board classes to do the work. Use the keys "4 5 6" to position and "0" to drop (j k l, n -- are alternatives). The "speed" slider adjusts how fast it goes. You will use a subclass of JTetris that uses an AI brain to auto-play the pieces as they fall. For your assignment, you'll add the much needed adversary feature.

Milestone — Basic Tetris Playing

You need to get your Board and Piece debugged enough that using JTetris to play tetris works. If it's too fast to see what's going on, go back to the mule. Once your piece and board appear bug free, you can try the next step.

JBrainTetris

The Brain interface defines the bestMove() message that computes what it thinks is the best available move for a given piece and board. Brain is an interface although it could have been defined as a class.

LameBrain

The provided LameBrain class is a simple implementation of the Brain interface. Glance at LameBrain.java to see how simple it is. Given a piece, it tries playing the different rotations of that piece in all the columns where it will fit. For each play, it uses a simple rateBoard() method to decide how good the resulting board is — blocks are bad, holes are bad. Board.dropHeight(), place(), and undo() are used by the brain to go through all the board combinations.

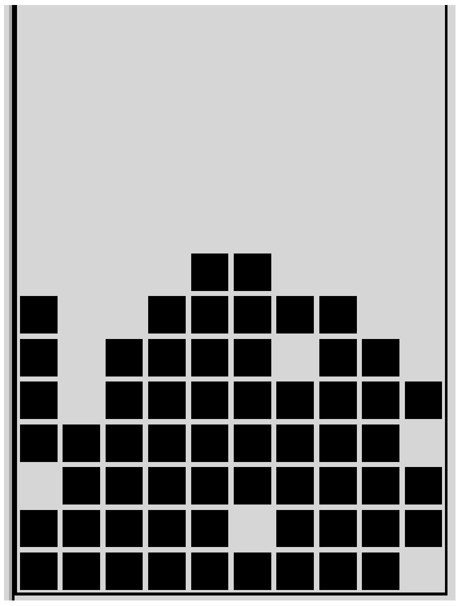
Putting together a better tetris brain is a fascinating algorithmic/AI problem. For this lab you will create your own subclass of DefaultBrain and use the Load Brain button to get JTetris to use it. Two things that the default strategy does not get right are: (a) avoiding creating great tall troughs so only the long skinny one will fit, and (b) accounting for things near the top edge as being more important things that are deeply buried. There's also a problem in tuning the weights. If you really wanted to do this well, you would want to write a separate program to work on optimizing the weights -- that can make a big difference.

To load a brain into Tetris, type in the name of the class into the appropriate box and then hit the load button (this works for adversary too). E.g. to try out my JediMindTrick brain, just make sure it is in the folder with your working Tetris game, and type JediMindTrick into the brain location. Then look on in awe as it plays forever.

It can be sort of mesmerizing to watch the brain play; at least when it's playing well and the speed isn't too fast. Otherwise it can be sort of stressful to watch.

Stress Test

Use test mode (use the arg "test" when running the main method) to force JTetris to use the fixed sequence of 100 pieces (make sure to turn the Brain on). Test mode with the unchanged DefaultBrain and with the pieces array built in the standard order should lead to exactly the following board after the 100 pieces have been played...



This is an extremely rigorous test of your Board. Although there were only 100 different boards with a piece landed on screen, the brain explored thousands of boards that were never on screen. If getColumnHeight() or clearRows() or undo() were wrong once among the thousands of boards, it could change the course of the whole thing. If your Board passes the stress test, it's probably perfect.

If the stress test is not coming out right, you could...

- Look at the JPieceTest output to verify that the pieces are correct in every detail.
- Put in additional sanityCheck() style code: check that clearRows changes the number of blocks by the right number (should be multiple of the board width), check that undo() is really restoring exactly the old state.

Adversary

The brain can not only be used to choose what to do with pieces, it can also be used to choose the next piece!

- When the adversary is on it create a random number between 1 and 99. If the random number is greater than the slider, then the piece is chosen randomly as usual. But if the random value is less, the mischief begins. In that case the "adversary" gets to pick the next piece. When the piece is chosen at random, we setText() the status to "ok", otherwise set it to "*ok*". (We don't want the feedback to be too obvious so the roommate test below can work nicely.) It the slider is 0 (all the way left), the adversary should never intervene.
- The adversary is implemented with a little JBrainTetris code that uses the brain to do the work. It loops through the pieces array. For each piece, it asks the brain what it thinks the best move is. It remembers the piece that yielded the move with the worst (largest) score. When it has figured out which is the worst piece — the piece for which the best possible move is bad, then that's the piece the player gets! "Oooh tough break, another dog.

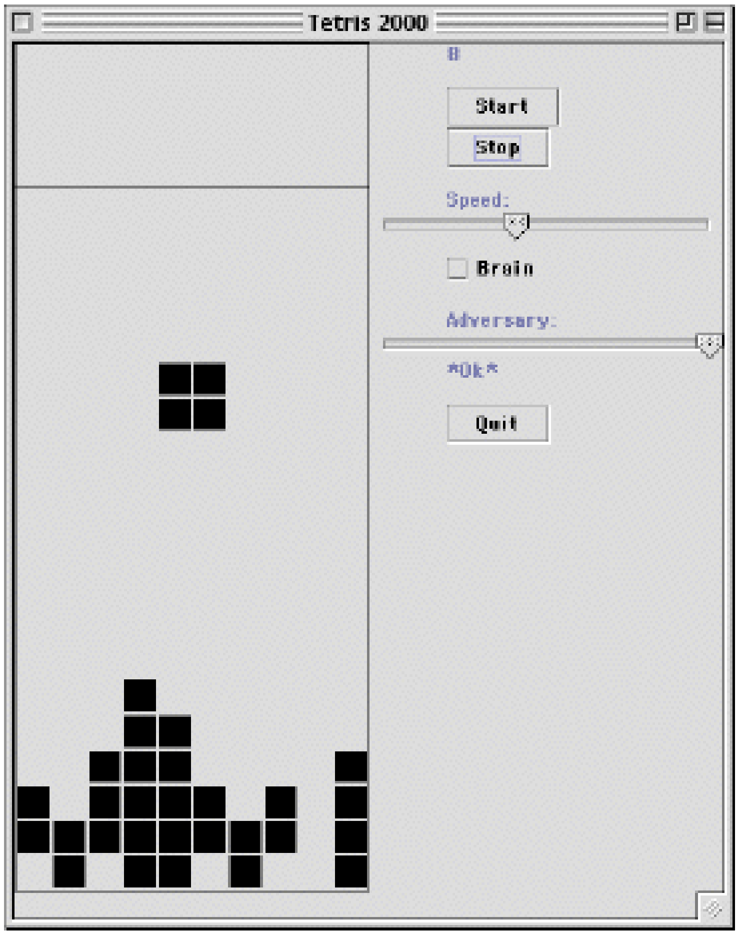
Gosh that's too bad. I'm sure the long skinny one will be along real soon."

- Little does the brain realize the bizarre context where it will be used — just the way modular code is supposed to work. Also notice how vital the speed of Board is. There are about 25 rotations for each piece on a board, so the adversary needs to be able to evaluate 7*25=175 boards in the tiny pause after a piece has landed and the next piece is chosen "at random". That's why the place()/undo() system has to be so fast. Row clearing will be rare in all that, but we need to be able race through the placements.
- It's absolutely vital that once you have your code working, you go test it on your roommate or other innocent person. Leave the adversary set to around 40% or so, and leave the speed nice and slow. "Hey Bob, I understand you're pretty good at Tetris. Could you test this for me? It's still pretty slow, so I'm sure you'll have no problem with it."
- For ironic enjoyment, have the brain play the adversary.

Deliverables

Aside from a well tested Board class, your main task is to write a good brain. As my test of this, I'll try your brain out against other people's brain classes (including my own). Since this isn't really a test of programming ability, I want to see that you have put some honest to goodness thought into this project.

Ahhhh -- good old adversary -- always able to find the perfect piece for an occasion...


Tetris 2000
8
Start
Stop
Speed:
Brain
Adversary:
Ok
Quit