Tetris Demo ReadMe

Table of Contents

Introduction

This is a document highlighting some of the features of the Tetris demo. This document was written by Rahul Chandan for the Dynamic Programming course (ECE271C) offered at the University of California, Santa Barbara and taught by Prof. Jason Marden.

Code

You should not have to modify very many of the files below, but an understanding of how this demo works is critical to using it effectively. The only files you should have to modify are:

- `tetrisMain.m` (the game parameters)
- `tetrisMyPlayDemo.m` (only if you want to modify the plotting/reporting)
- `myPlay.m` (your strategy)

Another file to familiarize yourself with is `nextBoard.m`, which we will discuss in the `myPlay.m` section.

Code Structure

The Tetris demo has the following code structure:

```
% tetrisMain.m
%   > tetrisBuild.m
%   > tetrisMyPlayDemo.m
%       > myPlay.m
%           > nextBoard.m
%           > boardHeight.m
```

tetrisMain.m

Here, the game parameters are intialized and the demo is instantiated.

All the game parameters are stored in the `tData` structure. The parameters are as follows:

- `Deterministic:` a flag for if future pieces are generated randomly or in a determinstic cycle. if `Determinstic=0`, the next piece will be generated randomly. If `Determinstic=1,2` or `3`, the first piece will be that number and will cycle through 1->2->3->1->... etc.
- `nMaxPieces`: the number of pieces to run during each stage. Just select a really large number if you'd like it to run until failure.
- `nEpisodes`: the number of times to run the game of Tetris. Each game ends if `nMaxPieces` pieces have been played, or if the player loses.
- `buildStates:` a flag that indicates whether the code should generate the state map (all possible states of the game) before the game is played. If `buildStates = 1`, the state map is generated, else, if `buildStates = 0`, the game will not generate the state map. In either case, the code generates the valid move set at every time-step.
- `morePieces`: a flag that indicates whether the 's'-shaped pieces should be added. Can be ignored.
- `GameSize`: the dimensions of the game board, stored as `[height, width]`. Make sure you leave room for some overshoot. As a rule-of-thumb, the board height should be `rowCap` plus the maximum height of any of your pieces.
- `RowCap`: if the height of the stack exceeds `RowCap` the current game ends.
- `TimeDelay`: the amount of time (in seconds) between frames in the game animation.
- `SelectMethod`: Not used in the demo.
- `Pieces`: A cell of the valid pieces in the game, which must be labelled as `1:length(Pieces)`.
- `numRots`: An array of length `length(Pieces)` with the number of unique rotations each piece has. Note that the indeces correspond between `Pieces` and `numRots`.
- `Pcolor`: the colours of the pieces, can be ignored.
- `moves`: a list of the possible moves for every given piece. The code assumes that the piece will fall vertically from the top, and not be strategically placed sideways into gaps.
- `flatBoards`, `boards`, `stateMap`: unique to when `buildStates = 1`. `flatBoards` stores the various possible board states as binary strings, `boards` stores their graphical representations, `stateMap` contains all the possible states (board, piece).
- `S_Sounds`: a flag that indicates whether the code should beep on every row deletion, and gong when the game ends (`1` indicates sounds on, `0` indicates sounds off).
- `S_Plot`: a flag that indicates whether the game animation should be shown (`1` indicates animate, `0` indicates no animation). If `S_Plot = 0`, the game will not print the scores for every round, unless lines `105` and `110` are commented out in `tetrisMyPlayDemo.m`.
- `startPiece`: the first piece to be played.

tetrisBuild.m

If `buildStates = 1`, generates and returns `flatBoards`, `boards`, `stateMap` and `moves`. Else, if `buildStates = 0`, only generates and returns `moves`. See the definitions of these parameters in the previous subsection.

tetrisMyPlayDemo.m

Plays `nEpisodes` games using the strategy encoded in `myPlay.m`. The game ends when `nMaxPieces` pieces have been successfully stacked, or the stack height exceeds `RowCap`.

If you would like to write your strategy in a different function, just redefine the vale `myFun` to a string with the name of your function, i.e. if your strategy function is named `myStrategy`, then define `myFun = 'myStrategy'`.

myPlay.m

This is the function that determines what the next move should be depending on the current board and the valid moves. This is where the majority of your added code will be implemented. Feel free to add your own functions and implement them in `myPlay.m`. The function receives the current board (`board`) and piece (`pieceNum`), and returns the move to be made.

- `board` is a matrix of size `RowCap x width` with a `0` representing an empty space, and a `1` representing a space occupied by a piece.
- `pieceNum` is the index of the current piece to be played. Recall that this is sufficient information to get the list of next possible moves (via `DATA.moves{pieceNum}`).

```
% Example: board and pieceNum for a game of Tetris with
% DATA.GameSize = [9,6] and DATA.RowCap = 6

board = zeros(6,6,'logical')
```

```
board = 6×6 logical array
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0
```

```
pieceNum = 4
```

```
pieceNum = 4
```

```
% From tetrisMain.m: Pieces{4} = [1 1 1 1]
```

For every board state '`board`' and move '`move`' in `DATA.moves{pieceNum}`, one can run `nextBoard(board,move)` to retrieve the next board state. `nextBoard` returns a new board '`newBoard`' and the number of rows eliminated '`score`'. The default `myPlay.m` file implements the strategy that minimizes the height of the next board.

IPlay.m

This function allows the user to choose where the next piece is dropped. By using the left and right arrow keys, you can cycle between the available moves (stored in `DATA.moves{pieceNum}`) and choose where the piece will drop by selecting the down arrow. Try to see if you can do better than the policy you coded!

To switch between `myPlay.m` and `IPlay.m`, toggle the string `myFun` between `'myPlay'` and `'IPlay'` in line 4 of the `tetrisMyPlayDemo.m` function.

boardHeight.m

The function `boardHeight` returns the height of each column of the input board.

Troubleshoot

- If the code is taking a long time to initialize, try setting `tData.buildStates` to `0` in `tetrisMain.m`, and make sure your board size is of reasonable size.
- If you are not seeing feedback regarding the score achieved and number of pieces played, try commenting out lines `105` and `110` in `tetrisMyPlayDemo.m`.