CLASSIC TETRIS

PROJECT PROPOSAL

CS 154 – Abstractions and Paradigms in Programming

Project proposed by the following students:-

Yash Gupta	Mayank Singhal	Piyush Onkar
(160050037)	(160050039)	(160050012)

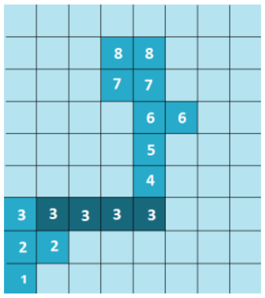
DESCRIPTION:

TETRIS is an old Russian puzzle-type game in which the player has to arrange falling geometric shapes called 'Tetrominoes' in a well. A tetromino is basically a geometric shape consisting of 4 squares connected orthogonally. A random sequence of Tetrominoes fall down the well and the player can move sideways or rotate the Tetrominoes to create horizontal lines of 10 units without any gaps. When such a line is created it disappears and any blocks above it fall down a unit. The objective is to stack as many blocks as possible without letting them stack up to the top by clearing as many lines as possible. In this project we shall attempt to recreate our own racket version of **TETRIS** and also design an A.I. to play it without user input.

OUTLINE OF A SOLUTION:

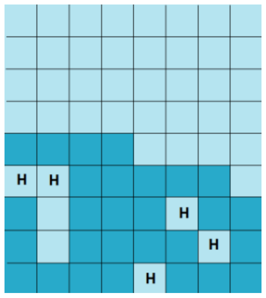
We shall implement a *greedy algorithm* in the A.I., i.e., the AI will select the best possible move at each step of the game. It will do so by considering and analyzing all the possible orientations and positions of the falling tetromino on the current grid. Each possible state will be assigned a 'score' by the AI and the block will be moved to the state with the highest score.

To avoid losing the game, first and foremost we have to avoid placing Tetrominoes in high positions. So for each state, the computer calculates a *height penalty* — for each block the computer adds a number depending on how high it is. Higher height penalties are unfavorable, so we assign negative score for height.

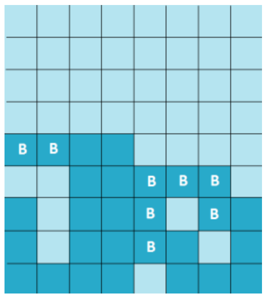


It's also obvious that we should try to get as many clears as possible, so we assign a positive score for clears.

Another key point is to avoid making 'holes' because holes prevent us from making clears. So assign a negative score for holes.



Additionally, blockades make it harder to remove holes so blockades too should be penalized.



Above are the major factors for the AI to decide where to place the Tetrominoes. To fine-tune, we can optionally include other factors: rewarding or penalizing for hugging the wall (edge of the current block touching edge of the wall), hugging the floor (edge of current block touching the floor) and flattening (rewarding if the edge of the current block touches an existing block).

We assign weights to each of the above factors to finally calculate the score for a given state:-

$$\text{Score} = A * (\text{sum of heights}) + B * (\text{number of clears}) + C * (\text{number of holes}) + D * (\text{number of blockades})$$

DISCUSSION:

It is evident that the efficiency of the AI will depend on the values of the weights that we choose. Through trial and error or otherwise, we assign the values of the weights such that the game can last longest without the blocks reaching the top:

- -3.71 for the height multiplier
- -4.79 per hole
- +1.4 per blockade
- -1.87 per clear
- +4.8 for each edge touching another block
- +3.22 for each edge touching the wall
- +3.68 for each edge touching the floor

The height multiplier has a fairly high value, which agrees with intuition. The 'lines cleared' factor seems to be misleading. Intuition suggests against assigning a negative weight for clearing a line, yet the AI will still lines whenever it can: this behavior traces back to the height multiplier. Clearing a line does exactly what it says: removing an entire row of blocks — and removing that many blocks deals a huge blow to the height multiplier. The rest of the values are harder to justify, so let's just say that they *work*.

To implement the game we will be using the teachpack *universe.rkt* from the package *htdp-lib*. The *universe.rkt* teachpack implements and provides the functionality for creating interactive, graphical programs that consist of plain mathematical functions.

It also allows for key events so it is also possible to extend the project such that the user can play against the computer. It's also interesting to introduce a points awarding system based on number of Tetrominoes stacked, and number of lines cleared etc, and maybe introduce the concept of 'levels', as in, the speed of falling Tetrominoes increases after a certain number of lines have been cleared. The game ends at a certain level, say, 5.