

**Tetris. Instructions Manual**

**Getting Started**

Tetris is played by stacking up the Tetrominos to complete lines. Players earn points by clearing lines, and the objective of the game is to clear as many lines as possible and earn the highest score.



A Tetromino is made up of four blocks arranged next to each other. These are the seven different forms of Tetrominos in the game. Each has its own separate colour and name. (From left to right: I, L, J, T, Z, S, and O)

To run the game, first extract the zip file. Then open Tetris.v2.xlsm and hit the start button.

**Playing the Game**

The Player must manipulate these pieces as they fall down the game board—moving them left and right across the board as well as rotating them to stack them up and fill up the rows of the game board. Upon hitting a piece or the bottom of the game board, the piece will stop falling and lock in place, retaining its form. At this point, the Player will receive a new piece that will proceed to fall down the game board. This repeats until the Player “tops out,” meaning that the pieces were stacked to the top of the game board and the new piece can no longer fall when it is spawned. When the player has topped out, the game is over.

The objective of Tetris is to clear lines. Upon filling a horizontal line in the game board, the line will clear, removing all blocks from that row and shifting the rest of the blocks in play down. This is how the Player earns points. More points are also earned by clearing more lines:

Single 40

Double 100

Triple 300

Tetris 1200

As the Player clears more lines, the game also levels up—increasing the rate at which pieces fall. The game starts at level 1 and a level is gained for every 10 lines that are cleared. As the difficulty level increases, so do the points! See if you can make it to the max level! (12)

**Controls**



The game is played entirely with the keyboard:

**Left and Right Keys** – Move the piece left or right across the board.

**Down Key** – “Soft drop” the piece—dropping it quickly down the board

**Z Key** – Rotate the piece counter clockwise

**Up and X Keys** – both rotate the piece clockwise

**Shif**t – hold the active piece if the option has been enabled. After holding a piece, you must use the swapped piece

**Space** – “Hard drop” the piece—drops the piece as far as it will go and sets it in place

**S** **Key** – Pauses the game

**H** **Key** – Asks the computer for a hint on where to place the next piece if the option is enabled

**The Options Menu**

C:\Users\Timothy\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Controls_Button.fw.pngC:\Users\Timothy\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Level_Select.fw.png



This is the options menu for the game. From here, the Player will select their preferences for the game to increase or decrease its difficulty. The Player can click on what difficulty they would like to start at (keep in mind you will still need to clear 10 x Level lines to progress though) as well as activate other features. The controls button will display the game controls for the Player, and the Start button will begin the game when the Player is ready.

**Holds** – will allow the player to hold a piece and swap it when they want

**Wall Kicks** – an alternate rotation style that will shift the piece if it is spun against a wall and there is no space

**Hints** – will allow the player to hit H and receive a suggestion from the computer (but at a point cost)

**Ghost** – will place a ghost piece in the position where the piece will be hard dropped

**The Game Screen**

**6**

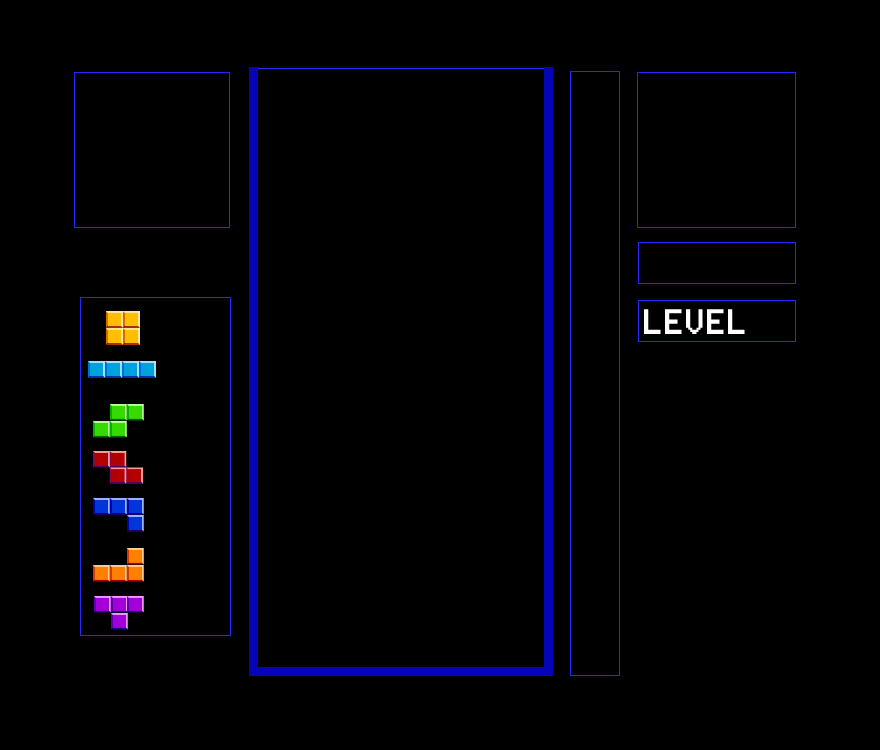
**5**

**4**

**3**

**2**

**1**



This is the screen where the game is played.

**[1]** – Game Board. This is where the Player will stack the pieces and clear lines

**[2]** – Next Piece. This displays the next piece that the Player will receive

**[3]** – Points. This displays the current points

**[4] –** Level. This displays the Player’s current level, the bar on the side will show the progress to next level

**[5]** – Piece Counter. This keeps track of how many piece of each type have been used

**[6]** – Held Piece. This shows the Player’s currently held piece (if any)

See if you can’t get to level 12!

Enjoy!

**Tetris. Technical Note**

This technical note will take you through the logic and coding behind the game including:

1. Setting up the game board for play
2. Random piece generation
3. Piece movement and player input
4. The game timer
5. Hint calculation

**Setting up the game board**

The Grid class.

* This class corresponds to a single picture box on the user form and contains a numerical value as well as a colour value for tracking the blocks that pass through it
* An array is created of these Grids. 10 wide x 20 long. This represents the game board and is displayed on the user form.

The Block class

* This class represents a single block in the Tetromino. It has a colour and a position in relation to its other blocks

The Tetromino class

* This class represents a whole Tetromino and uses 4 Blocks to make it up.
* The class has an x and y location on the game board as well as a specific form

On starting the game, the array of Grid objects is initiated and displayed on the user form. Then the first Tetromino is generated and displayed on the game board, ready for the player.

**Random piece generation**

The system uses a “bag” of 7 Tetrominos. One of each Tetromino is placed in the bag, and then the system chooses at random from this bag. The system picks a number and checks to see if the Tetromino at that location has been selected—if it is still there, it picks out the Tetromino and removes it from the bag. If the Tetromino has already been selected then the system will generate another random number, and do this up to four times. If a piece still cannot be found, then the system will just use the Tetromino anyways. Upon emptying the bag, the bag is then refilled with one of each Tetromino again. This method of *randomization* allows the user to get a fairly even mixture of pieces and prevents long runs of the same piece—which could be perceived as unfair.

**Game Timer**

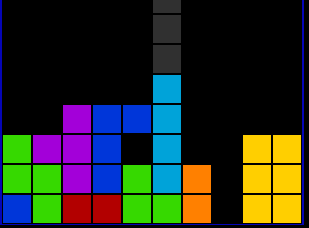
The game timer is a timer class that was imported from the Windows API. This is because VBA doesn’t offer a timer function that goes off in intervals of less than a second and the timer is what handles the inputs from the user as well as the automated movement of the pieces. The timer is set to go off every 16 milliseconds; however, it realistically only ticks every 30 milliseconds because of the speed limitations of VBA. Upon each timer *tick*, the updateGame() subroutine is called.

The updateGame() subroutine manages the game and inputs from the user. It checks to see if it is time to automatically drop the piece down a line through a counter variable. If the piece cannot be moved any further, it will set the piece in place and generate a new piece for the user. The timer also checks for user inputs. This is done through a second checkKeys() sub that looks at what key is being pressed and performs the related action. Lastly, the timer will check to see if the game is over. If a newly generated piece overlaps one of the already set pieces, this sets off a flag to the game timer letting it know that user has *topped out* and to bring the player to the game over screen.

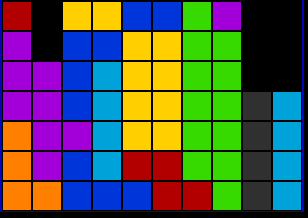
**Hint Calculation**

To calculate hints, the program tests all possible moves of the Tetromino—10 horizontal tiles with 4 rotations each—and then scores each of these moves. The scoring of the moves is based off of three factors, each with a different priority.

The first of these factors is whether or not the piece creates any new closed-off holes on the game board. This occurs when a piece blocks off an empty space on the board, forcing the player to clear all the lines above it before that line can be cleared. This factor is given the heaviest weight when scoring the move and the hint will avoid giving moves that create holes at all costs. To calculate the holes, the system first runs through the board and counts all the existing holes, then makes the move and recounts the holes. The more holes the move creates, the lower the score.

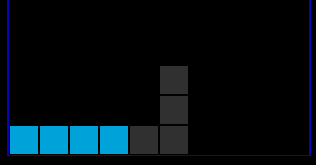
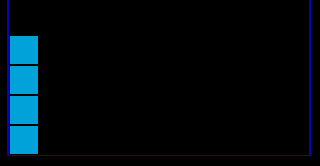


The next factor the algorithm looks at is how many lines the move can clear. This is fairly simple, where the score is linearly increased depending on how many lines are cleared. The scoring is also set up so that only when clearing 3 or more lines will its contribution surpass the holes in the board. Calculating this score is straight-forward—the system makes the move and then counts how many lines are filled.



Clears 4 Lines

The last factor calculates how the piece fills the game board and prioritizes filling the lower lines over higher ones. The logic behind this is because the optimal way to play Tetris is by filling the lower lines as much as possible and clearing them from the bottom to the top. To reflect this, the hint calculation makes the move and then sums up how the board is filled up with a multiplier for each line. This multiplier increases the further down on the game board the line is.



>

The result of these three factors is a hint that can accurately pick out a good move for the player to make—simply because the *optimal* move is highly subjective to the player and their playstyle.

**Tetris. VBA Documentation**

**Game Module –** manages the game menus and inputs from the user

initializeGame(l As int, r As Boolean, h As Boolean, s As Boolean, hint As Boolean) – initializes the tetris game and settings based on the users input. L is the starting level, r is the rotation style (false = classic, true = with wall kicks), h is whether holds are enabled, s is whether piece shadows are enabled, and hint is whether hints are enabled.

terminateGame() – terminates the game timer and unloads the game form

checkKeys() – checks what key is being pressed by the user and performs the appropriate action

updateGame() – updates the game board. Checks if the right amount of time has passed before the piece moves down, checks for key presses, and checks if the game is over.

updateLeaderboard() – sorts the leaderboard in ascending order

**Tetris Module –** includes all of the functions needed for playing the game

setGameBoard() – sets up the worksheet as a grid, puts the tetris board on it, and clears the board array

setSequence() – sets the array of next pieces

checkSequence() – checks if the array of next pieces is used up

genNext() – generates the next piece in the game by randomly selecting one from the array, if the piece has already been taken, it selects another piece (retries up to four times before selecting a taken piece)

move(t as Tetromino, x, y as int) – will move the given Tetromino in a direction passed through the parameters.

spin(CounterClock as Boolean) – will rotate the Tetromino and bump it left or right if it hits out of bounds

spinClassic(CounterClock as boolean) – rotates the piece without any wall kicks

canMove(t as Tetromino, x, y as int) as Boolean – will check if the Tetromino can move in a direction or if it is blocked and return true or false

hardDrop(t as Tetromino) – will drop the given Tetromino as far as it will go and set it in place

updateCount() – updates the piece counter

setActive() – will set the active piece to the next piece, reset its position, and generate a new random piece

setHold() – will hold the active piece and either swap with the existing hold or go to the next piece

checkLines() as Boolean – will check if any lines are complete and return true or false

clearLines() – will clear all the completed lines and shift the lines down, should also increase the points with a multiplier based on how many lines were cleared at once

checkGameOver() as Boolean – checks if any pieces are hitting the top and will return true if that’s the case

setPiece() – will set a piece in place, check if any lines are clear, and check if the player has lost

shiftDown(index as int) – will shift the lines of the board down past a given row.

drawBoard() – draws the existing board

getHint() – gets a hint for the active piece

printBoard() – prints the board values onto the spreadsheet (for tracing and debugging)

drawPiece() – draws the active piece

clearPiece() – clears the active piece

drawNext(), clearNext(), drawHold(), clearHold(), drawShadow(), clearShadow(), drawHint(), clearHint()

**HintCalculation module –** Manages the functions for calculating and finding the hints

findHint(form as int, row as int, Hint as Tetromino) – sets the given Tetromino to the best hint location according to the algorithm

scorePosition(Hint as Tetromino, testBoard() as int) as double – returns a score based on the current position of the given Tetromino on the board

checkboard(a() as int, t as Tetromino) as Boolean – checks to see if the current position of the Tetromino overlaps another piece on the board (a).

checkHoles(a() as int, t as Tetromino) as Double – returns how many new holes the piece will create in the game board (a) if it is placed in its current position (a hole being a gap surrounded by placed blocks)

findTops(tops() as int, a() as int) – finds how many blocks have been placed in each column on the board (a) and fills an array (tops) with these values

countHoles(tops() as int, a() as int) as Double – returns how many holes are currently in a game board (a)

clearedLines(a() as int) as Double – returns how many lines are currently filled in a game board (a)

checkFill(a() as int) as Double – checks how filled the lines are and scores this with lower lines valued at more than higher lines.

**Tetrimino class** - an object representing a Tetromino on the game board

Properties:

Form – an integer corresponding to its shape (0 = O, 1 = I, 2 = S, 3 = Z, 4 = L, 5 = J, 6 = T)

Name – a string corresponding to its shape

x – the piece's x coordinate on the worksheet

y – the piece's y coordinate on the worksheet

i – The piece’s first index in an array

j – the piece’s second index in an array

bBound – the piece's bottom coordinate in relation to its registration point (0,0)

tBound – the piece’s top coordinate

rBound – the piece's rightmost coordinate in relation to its registration point

xBound – the piece's leftmost coordinate in relation to its registration point

Functions:

private setForm() – sets the piece's form according to its form property

init(Form as int) – initializes the piece given an integer for its form

draw() – will draw the piece at its x and y coordinate on the sheet

clear() – will clear the piece from the sheet

hintDraw() – draws the piece as an outline

hintClear() – clears the piece outline

toArray(a() as Grid) – will write the piece into a Grid array (adjusts the value

property)

toIntArray(a() as int) – will write the piece into a numerical array

clearArray(a() as Grid) – clears the piece from a Grid array

clearIntArray(a() as int) – clears the piece from a numerical array

spin – rotates the piece clockwise

spinC – rotates the piece counter clockwise

colourOverride(c as Long) – overrides the piece’s default colour given a new one

**TetrominoImage class** – an object that displays the images of the Tetrominos onto the game form

Properties:

Form – the integer corresponding to the piece’s shape

Name – the string corresponding to the piece’s shape

Functions:

Draw(PictureBox as variant) – draws the Tetromino into a picture box on the game form

Clear(PictureBox as variant) – clears the Tetromino from a picture box

**Block class** - an object that represents a single block in a Tetromino

Properties:

x – the block's x coordinate in relation to a registration block

y – the block's y coordinate in relation to a registration block

colour – the block's colour

Functions:

init(x as int, y as int, colour as long) – initializes the block at an x, y coordinate and sets its colour

draw(x as int, y as int) – draws the block given the location of a registration block

clear (x as int, y as int) – clears the block

hintDraw(x as int, y as int) – draws the block as an outline at the given location

hintClear(x as int, y as int) – clears the block outline

**Grid class –** an object representing one of the squares on the game grid, creates a picture box on the game form.

Properties:

Value – the integer value of the board. 1 = has piece, 0 = empty

Image – The colour of the piece filling the game board

Functions:

setBorder (c as long) – sets the border colour of the grid square

clearBorder() – clears the border colour of the grid square

init(i as int, j as int) – initializes the game grid, setting the value to 0 and the image to the background colour of the board. The i and j is where the picture box will be drawn on the game from.

**Timer** – a timer class that controls the game – updates 60 times per second.

Functions:

startTimer() – starts the timer

Class\_Terminate() – upon termination of the class, will stop the timer.