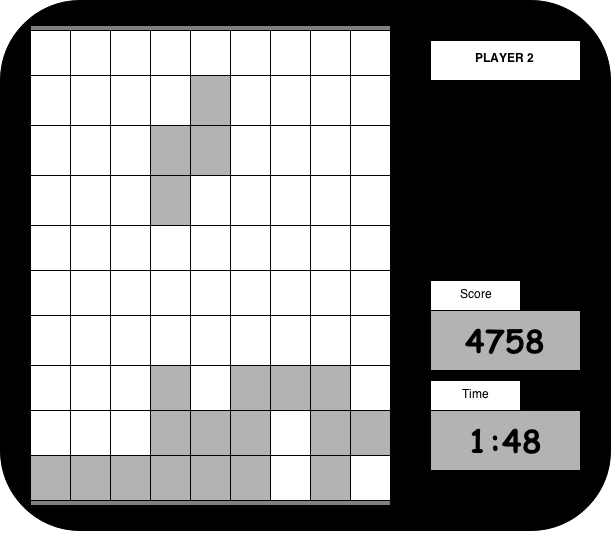
Detailed Game Specifications:  
Tetris

Course: COMP 2659, Winter 2015  
Instructor: Paul Pospisil  
Author(s): James MacIsaac  
Last Modified: 18/01/2015

# 1. General Game Overview

A one or two player game where the player turns various shaped blocks which constantly drop from the top of the play area, fall at a given pace (based on difficulty level), and then lock in to place when they hit the bottom of the play area, or make contact with another block already in position. The user can eliminate blocks if they can make them be continuously adjacent from the left side to the right, all on a single level. Points are awarded for blocks dropped, and for block elimination. The two player version would be a sequential game where one player would go first, then the other would go second. The scores would then be compared between the two.

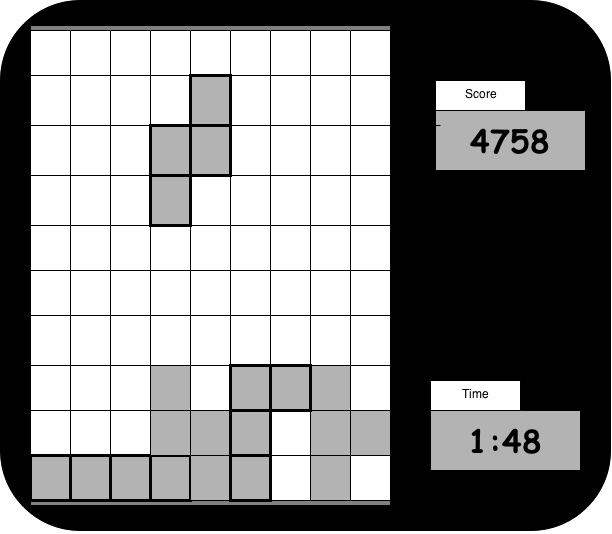


# Core 1-Player Details:

## Overview

One at a time, randomly generated blocks of varying shapes will drop from the top of the play area. The player will be given a set of controls to allow them to turn and shift the blocks as they fall from the top towards the bottom of the screen. The objective is for the user to position these blocks, as they fall, in a fashion as to complete lines of blocks from the left side to the right side of the play area, on one single level. Completing this objective awards points to be compounded into a score. Once the score is awarded, this completed line of blocks gets removed from the play area, and all that is found above gets shifted down one level. The user has finished playing only once the blocks stack in such a way that the block being added at the top can no longer fit on the screen. Time is kept track of to be seen alongside a player’s score, if found in future reference.

1-Player in-game view:



## Objectives and Rules

The start state of the game include an empty play area (no blocks present). The user will be prompted to hit a key to begin. Once this has been achieved, the game time will start, the first block will start appearing from the top of the screen, and the user will have to begin using strategy to place the blocks in a way that will allow them to complete lines across the screen to gain points.

Objective:

The user will be trying to maximize their score and minimize their time simultaneously to achieve the best that they can. A Score category system may be implemented. This would have a few simple guidelines:

* A Score above 3000 points with any time limit awards the player a bronze medal!
* A Score above 6000 points in under 5 minutes awards the player a silver medal!
* A score above 9000 points in under 7 minutes awards the player a gold medal!

(These are subject to change according to difficulty when game is implemented)

This system will allow players to strive to achieve a higher score in a shorter time limit.

Rules:

The rules behind the game are fairly simple. The user gets to control the movement of a randomly generated, slowly falling block. This includes commands to rotate (not to be confused with flipping!) the block in either direction, and a command to speed the falling process of the block towards the bottom of the screen. Once contact is made with either the bottom border of the play area or another block, this block is in its final position. A short time delay between when contact is made with an obstacle and when the block is considered to be in its final position must be allowed to make last moment rotations of the block, if any are available. There are 7 unique block shapes that can be generated. Each has 4 possible orientations (interchangeable with the rotation mechanic). More information on these shapes is found in the Objects Section.

Scoring:

Scoring is summed up throughout a game. Points are awarded each time 1 to 4 completed horizontal lines of blocks are completed. Scoring works as follows:

* 1 Line: 40 points
* 2 Lines: 100 points
* 3 Lines: 300 points
* 4 Lines: 1200 points

Once the player has stacked blocks to the top of the screen such that the next block generated can no longer be fully visible on the screen, the game is over, and we will be shown the player’s final score, game time, and a congratulatory message if any medals are to be awarded. In a potential additional implementation of a leaderboard database, each player would pick a username under which to play. The highest scores would then be kept and made available to anyone else playing the game.

## Objects

|  |  |  |  |
| --- | --- | --- | --- |
| Object | Properties | Behaviours | Graphical Image |
| Square Block | * Start position:   Centered at the top of the play area.   * End position:   User controlled during the game. | * Falls from the top of the screen towards the bottom at a standard rate, unless told to accelerate by the user. * Rotating this object has no effect since it is a square. | C:\Users\James\Downloads\SquareBlock.png |
| Zig-Zag Block (right) | * Start position:   Centered at the top of the play area.   * End position:   User controlled during the game. | * Falls from the top of the screen towards the bottom at a standard rate, unless told to accelerate by the user. * Two possible orientations are available, chosen and interchanged by user input | C:\Users\James\Downloads\SquareBlock (1).png |
| Zig-Zag Block (left) | * Start position:   Centered at the top of the play area.   * End position:   User controlled during the game. | * Falls from the top of the screen towards the bottom at a standard rate, unless told to accelerate by the user. * Two possible orientations are available, chosen and interchanged by user input | C:\Users\James\Downloads\SquareBlock (2).png |
| Straight Block  Corner Block (left)  Corner Block (right)  T-Block | * Start position:   Centered at the top of the play area.   * End position:   User controlled during the game.   * Start position:   Centered at the top of the play area.   * End position:   User controlled during the game.   * Start position:   Centered at the top of the play area.   * End position:   User controlled during the game.   * Start position:   Centered at the top of the play area.   * End position:   User controlled during the game. | * Falls from the top of the screen towards the bottom at a standard rate, unless told to accelerate by the user. * Two possible orientations are available, chosen and interchanged by user input * Falls from the top of the screen towards the bottom at a standard rate, unless told to accelerate by the user. * Four possible orientations are available, chosen and interchanged by the user in-game * Falls from the top of the screen towards the bottom at a standard rate, unless told to accelerate by the user. * Four possible orientations are available, chosen and interchanged by the user in-game * Falls from the top of the screen towards the bottom at a standard rate, unless told to accelerate by the user.   Four possible orientations are available, chosen and interchanged by the user in-game | Macintosh HD:Users:James:Downloads:SquareBlock (1).png  Macintosh HD:Users:James:Downloads:SquareBlock (2).png  Macintosh HD:Users:James:Downloads:SquareBlock (3).png  Macintosh HD:Users:James:Downloads:SquareBlock (4).png |
| Cell | * A single cell in the play area * Can either be filled with a portion of a game block or empty at any given time in the game | * Remains fixed in its position for the entirety of the game * User actions will either fill or empty the cell. | Macintosh HD:Users:James:Downloads:SquareBlock (5).png |
| Play area | * The entire set of cells for gameplay * Set is ten cells wide and 18 cells high * Depictions have less to simplify depiction | * Fixed grid of cells in which all of the gameplay happens * Cell interaction must be ran through this object | Macintosh HD:Users:James:Downloads:Tetris diagram (2).png |
| Score | * The score object keeps track of the players progress gradually throughout the game * Updated whenever the player accumulates points | * The score will remain fixed in position in the user interface and must be updated immediately following a scoring move by the player |  |
| Time | * The time object keeps track of how long the game has been running * Time is halted and recorded once the game is over | * The time object will advance itself from when the user starts the game until the user causes the game to enter the ‘loss state’ |  |

## Physics

The only physics behind the game idea that need to be visited are:

* Play area edge collision
* Block collision
* Constant movement pace of the blocks

The scheme I am going to propose to accomplish these three aspects will be to set up a cell coordinate system ( 10 x 18 grid -> 0 through 9 for horizontal cell rows, A through R for Vertical cell columns). This will facilitate making the play area collision (reject anything outside the grid) and the block collision (“current block in cells C1,C2,C3,C4 has collided with a filled cell at cell B3 -> set the current block’s position as it’s final position and generate a new current!”).

As for the movement pace of the blocks, a one row per second rate of descent should keep the game simple enough that the user can play but not too long that no enjoyment can be had. The user however can always speed the descent of the block by hitting the down key.

## Asynchronous (Input) Events

|  |  |  |
| --- | --- | --- |
| Event Name | Triggering Input Event | Description |
| Shift block | < and > arrow keys | The right and left arrow keys will instantaneously shift the current block, retaining orientation of the block, right or left one cell, respectively. This can be used as often as the user would like. |
| Rotate block | ˄ arrow key | This command will shift the orientation of the current block 90 degrees in a clockwise fashion, as to allow the user to place the blocks in an organized fashion. This can be used as often as the user would like. |
| Accelerate block | ˅ arrow key | This command will increase the rate at which the current block will descend towards the bottom of the screen. This allows the user to speed up a block already in position and proceed to the next block. The rate of acceleration will be 1 click to move 1 cell (not counted in the 1 cell per second rate of descent). |
| Update score | The user has successfully scored points by completely filling a horizontal line with blocks | This event will call for the user score to be resumed. The old score must be added to the newly scored points to generate the current total score. This will update the score object in the UI |
| Update grid | A player scoring | Once the score has been updated, the grid must first get rid of the row(s) of cells that were completed and shift all cells in their final position above the completed cell downwards by exactly one cell. This will be the new grid that the user is to play with. |

NOTE: score updating and grid updating must be completed virtually instantly one after another to allow the game to continue without a pause.

## Synchronous (Timed) Events

[Note: on the Atari ST, one easy option will be to make use of a 70 Hz timer (i.e. 70 ticks per second). So, it may be easiest to base timed events on multiples of 1/70th of a second.]

[Note: the trigger for a synchronous event is typically based on a clock. But, some synchronous events are also triggered by the occurrence of other synchronous events. E.g. a ship may move forward 1 pixel every 1/10th of a second, but it might as a result collide with an obstacle – these are two distinct events, one triggered directly by clock ticks and the other conditionally triggered by the original movement.]

|  |  |  |
| --- | --- | --- |
| Event Name | Trigger Timing | Description |
| Block descends | Every 70/70th of a second (subject to change according to feel of the implementation from a user’s perspective) | Current block descends one block maintaining its current orientation and shape |
| Block collision | When a block descent results in a collision | Once the collision is detected between the two cells, the user will be given 70/70th of a second to shift the block and orient the block ( if the surrounding grid schema allows rotation freely without other immediate collisions) to allow the user a final positioning phase for this block. |
|  |  |  |
|  |  |  |

## Condition-Based (Cascaded) Events

[Note: some events may trigger other events, conditionally. This is called event “cascading”. The triggering event(s) may themselves be asynchronous, synchronous, or other condition-based events.]

|  |  |  |
| --- | --- | --- |
| Event Name | Triggering Condition | Description |
|  | [E.g. ship has moved and ship bounding box intersects with obstacle bounding box] | […] |
| Starting the game | The game start is triggered when the enter key is hit | This will cause the game to initialize (created the grid, make it visible to the user, start the time, and generate the first block). The first block will then begin the synchronous descent event and user control will be granted for that block. |
| Final block isolation | Block is in final position on the grid | When the final position timeout ends ( after 70/70th seconds, the user loses control of the current block, all of the blocks current cell locations must become filled cells on the grid, allowing the cells to cause collisions and be used in point scoring routines |
| Block generation | When time starts OR a new block is needed | When time starts, the game will generate a random block with a random possible orientation that will be centrally positioned at the top of the screen and begin to descend towards the bottom at a fixed rate. The user will be granted control with their 4 commands of the block. This must also happen upon the loss of control of a block that has reached and successfully been positioned by a collision. |
| Ending the game | When a newly generated block is not able to display its entire shape onto the screen via natural descention | Once this event has occurred, the game time must end and be recorded at its final state, the game score must be recorded, and both must be displayed to the user for contemplation. |

## Hypothetical Gaming Session

* User has started the game by hitting the enter key
* Current block is a zig-zag (left) block with horizontal orientation
* Block descends one cell
* Block descends one cell
* User shifts the block to the right 2 spaces
* Block descends one cell
* User shifts the orientation of the block one time
* Block descends one cell
* User shifts the block to the right one cell
* …
* Block components at cell [X,Y] have had a collision with the bottom boundary of the game board at position [X,Y-1].
* Block final position timeout has commenced.
* User has shifted the block one cell to the left
* Block final position timeout has completed, block is now in final position. User control has been terminated on the block
* Block cells have been changed to filled cells on the board
* Current block is a Corner block (right) with orientation 1
* Block descends one cell.
* ……………………………………
* Current block is a straight block with vertical orientation
* Block attempted to descend one block but a collision was detected. Block cannot be fully represented on the screen. Game is over
* Record time, score
* Display stats to user

# 3. Game Play Details for Core 2-Player Version

The two player modification of the game will be selected from a menu when the game starts up. The users will then go one after the other. Each player plays as long as they can until the game ends for them. Once both players have ended their individual turns, the game will display scores and announce who has won the game.

The 2 player version is identical to the one player version in terms of playstyle.

The user interface will appear identical to the 1 player version apart from a player name displayed in the top right of the window.

# Macintosh HD:Users:James:Downloads:Tetris diagram.png

# 4. Sound Effects

|  |  |  |
| --- | --- | --- |
| Sound Effect Name | Brief Description | Event which Triggers Playback |
| [E.g. ship explosion] | [E.g. white noise with a fast attack and slow decay] | [E.g. ship collides] |
|  |  |  |
|  |  |  |
|  |  |  |

[Optional: brief description of background music. Music will be required by assignment 3.]

# 5. Additional Features (Time Permitting)

[Add descriptions here.]