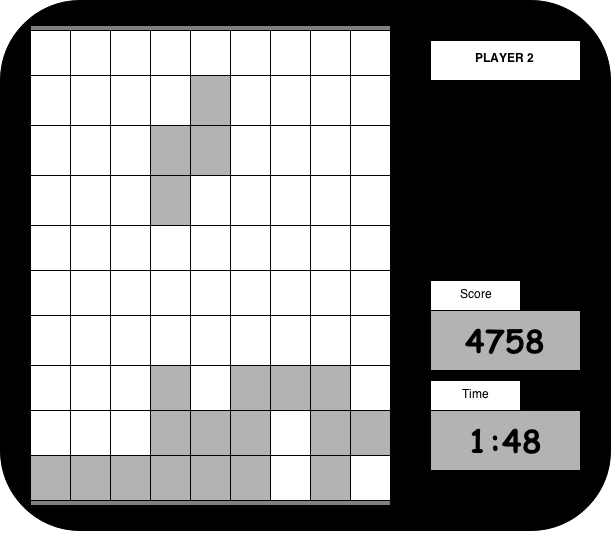
Detailed Game Specifications:  
Tetris

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.

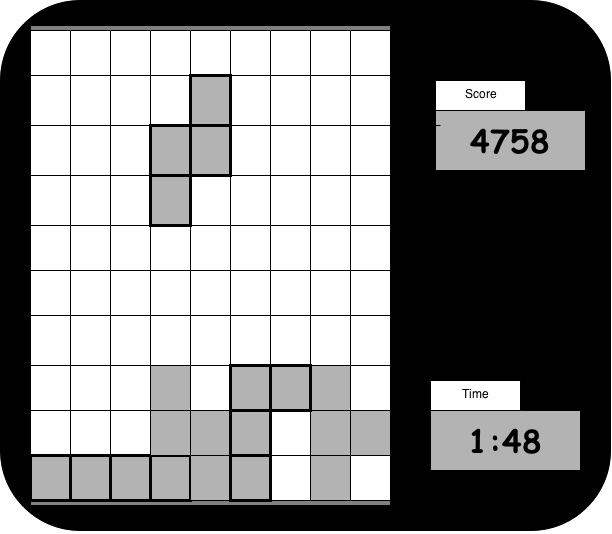


# Core 1-Player Details:

## 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 |  |

[Note: for some games, it may not be appropriate or realistic to include images for all object types. However, this is recommended if possible. Simple bitmap-style images (e.g. 8×8, 4×4, 16×16, etc.) are best for many game types. There are free tools to help you construct such images.]

[Note: for some games a particular object may have >1 associated image, depending on its current state (e.g. Mario walking left vs. Mario walking right).]

## Physics

[Description of game world’s physical laws goes here if applicable, e.g. regarding gravity, object velocity/acceleration/momentum, collision detection, rebounding, etc.]

## Asynchronous (Input) Events

[Note: keyboard input is required of all games. Additional mouse input is optional for core game play.]

|  |  |  |
| --- | --- | --- |
| Event Name | Triggering Input Event | Description |
| [E.g. “teleport request”] | [E.g. ‘T’ key is depressed] | […]  [Note: the description should include a summary of the effect on the relevant objects and their properties.] |
|  |  |  |
|  |  |  |
|  |  |  |

## 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 |
| [E.g. ship moves] | [E.g. every 1/10th of a second.] | [E.g. ship moves forward *speed* pixels (where *speed* is a property of the ship object)] |
|  |  |  |
|  |  |  |
|  |  |  |

## 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 collides] | [E.g. ship has moved and ship bounding box intersects with obstacle bounding box] | […] |
|  |  |  |
|  |  |  |
|  |  |  |

## Hypothetical Gaming Session

[Narration of typical example session goes here.]

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

[A detailed description of the core 2-player version goes here. Include a description of all differences between this and the core 1-player version (e.g. rule and game-play modifications, differences in graphics, etc.]

[2-player sample screenshot goes here.]

# 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.]