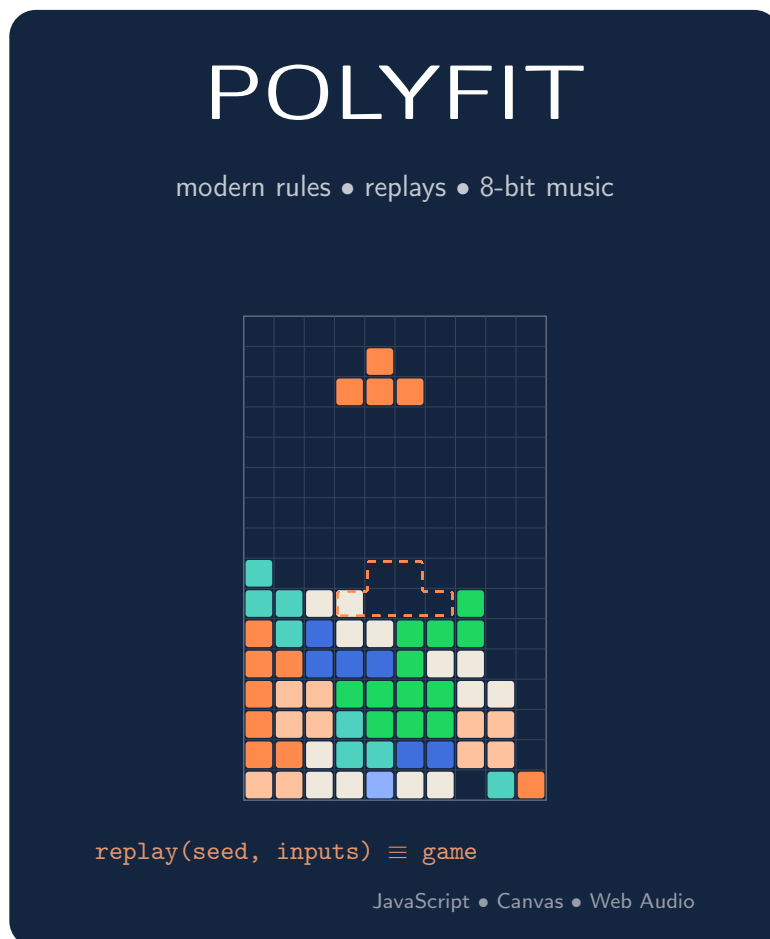




giuseppemuschetta.com



PolyFit: a Deterministic Falling-Blocks Puzzle in Plain JavaScript

Project Report

Giuseppe Muschetta

giuseppemuschetta.com/challenge

2026

Abstract

PolyFit is a falling-blocks puzzle with modern rules, written from scratch for the web in plain JavaScript: no frameworks, no libraries, no build step and no server. A game lasts three minutes and the goal is the highest score. The rules run in an engine kept apart from drawing and input, in fixed steps of 1/60 of a second, with whole-number gravity and a seeded random generator, so that the same seed and the same inputs always produce the same game, to the last point. This property carries the features that would normally need a server. A *challenge link* holds a whole game, the seed and every input with its tick, in the part of a web address after the #, which browsers never send to any server: whoever opens it plays the same pieces with the challenger's game replayed next to theirs, in step, and their own browser computes the score by replaying the game, so that no score can be made up. A Hall of Fame of replayed games and a top 10 kept on each device complete the leaderboards. The interface works with keyboard and touch, in a light and a dark theme, with a mark inside every piece for color-blind players, and the sound effects and an 8-bit arrangement of a folk song of 1861 are synthesized on the spot with the Web Audio API. The project is verified by 62 automated tests, among which an automatic player plays full games while every rule is checked at every tick. A three-minute game replays in about one millisecond, and the author's record game, 27,638 points, replays identically from its 1,362-character link.

Contents

1	Introduction	3
2	The rules of the game	3
2.1	Board, pieces and marks	3
2.2	The order of the pieces	3
2.3	Moving, turning and holding	4
2.4	Gravity and levels	5
2.5	Locking and the end of the game	5
3	Scoring	6
4	Architecture	7
4.1	Modules	7
4.2	The main loop	7
4.3	The modes of the page	8
4.4	Drawing	8
5	Replays and challenge links without a server	9
5.1	Determinism	9
5.2	The link	9
5.3	Opening a link safely	10
5.4	The score comes from the replay	10
5.5	Playing against a replay	10
5.6	Leaderboards	12
5.7	What a replay proves	12
6	The interface	12
6.1	Layouts	12
6.2	Controls	12
6.3	Feedback and comfort	14
6.4	The results	14
7	Sound and music	14
7.1	Sound effects	14
7.2	The music	14

7.3	Keeping time	15
7.4	Mixing	15
8	Testing	16
8.1	The test suite	16
8.2	Invariants at every tick	16
8.3	The automatic player	16
8.4	Beyond the suite	17
9	Results	17
9.1	The record game	17
9.2	Size and speed	17
10	Conclusion	18

1 Introduction

PolyFit was built as the playable part of the author’s website, under the title *The 3-minute challenge*: three minutes on the clock, one score, and the author’s record to beat. The genre is well known, and that is the point: anyone who has played a falling-blocks puzzle should feel at home at once, and the interest of the project lies in how the game is made rather than in new rules. Five goals guided every decision.

- G1. Responsive and fair.** An input must reach the screen within the next frame, and the rules must run at the same speed on every computer, whatever its screen refresh rate.
- G2. Modern rules.** The game follows the conventions of modern games of the genre: the 7-piece bag, the Super Rotation System with wall kicks, hold, a five-piece preview, a ghost piece, a lock delay, T-spins, back-to-back, combos and perfect clears.
- G3. No server.** The website is static: competition between players must work without a database, an account or a network request during play, and still produce scores that cannot be made up.
- G4. Every device and every player.** Keyboard and touch, phones and desktops, light and dark themes, players who do not tell colors apart and players who ask their system for less motion.
- G5. Made from scratch.** Plain JavaScript modules, drawing on a canvas, sounds and music synthesized at run time: no frameworks, no libraries and no audio files. The only files not written for the project are three fonts under the SIL Open Font License, and the melody of the music is a folk song in the public domain.

Section 2 describes the rules and Section 3 the scoring. Section 4 presents the architecture and the main loop, and Section 5 the central idea of the project: deterministic replays, and the challenge links and leaderboards built on them. Section 6 covers the interface, Section 7 the sound and the music, Section 8 the testing and Section 9 the measured results. Section 10 concludes.

2 The rules of the game

2.1 Board, pieces and marks

The board is a grid of 10 columns and 40 rows, stored as an array of rows with $y = 0$ at the top. Only the bottom 20 rows are visible; the hidden rows above them are where new pieces appear, and two of them are drawn above the field so that a piece can be seen arriving. A cell is either empty or holds the letter of the piece that left it, which tells the screen its color.

The seven pieces are made of four squares each (Figure 2). Every piece has its own color, taken from the palette of the website, and a small sign inside each square, a different one for every piece, so that the pieces can be told apart without relying on color. Each piece turns inside a bounding box, 3×3 for J, L, S, T and Z and 4×4 for I and O, and its four rotation states are generated from the spawn state by a quarter turn clockwise, $(x, y) \mapsto (s - 1 - y, x)$ in a box of size s .

2.2 The order of the pieces

The game never calls `Math.random()` while playing. Every random choice comes from Muberry32, a small 32-bit generator [1], started from a seed that the browser’s secure random source draws at the beginning of each game. Pieces are dealt with the *7-bag*: the seven pieces are put in a bag, the bag is shuffled with the Fisher–Yates algorithm [2], the pieces are dealt one by one and, when the bag is empty, a new one is filled. Two consequences follow directly:



Figure 1: PolyFit in a desktop browser, 34 seconds before the end of the author’s record game: the hold box, the score and the Statistics panel on the left, the board with the falling Z and its ghost in the middle, the next five pieces and the scoring table on the right.

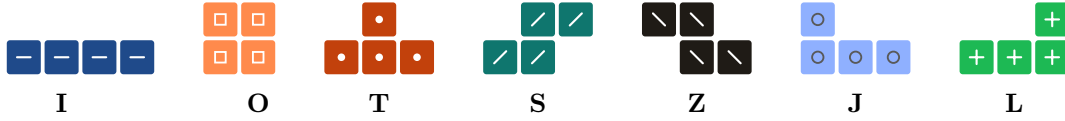


Figure 2: The seven pieces in their spawn state, in the colors of the light theme, with the mark that each square carries: a dash in the I, a square in the O, a dot in the T, a slash in the S, a backslash in the Z, a ring in the J and a cross in the L.

- between two copies of the same piece there are at most 12 other pieces: the worst case is a piece dealt first from one bag and last from the next;
- after n pieces have been dealt, every piece has been dealt either $\lfloor n/7 \rfloor$ or $\lceil n/7 \rceil$ times, because the first $7\lfloor n/7 \rfloor$ pieces are whole bags and the rest are distinct pieces of one bag. The Statistics panel of the game, which counts how many times each piece came into play, therefore never shows two counts that differ by more than one.

2.3 Moving, turning and holding

A new piece appears with the left edge of its box in column 3, counting from zero, and drops one row at once if it can, so that it is already partly in view. A horizontal key moves the piece one column when pressed; held down, it starts to repeat after 10 ticks (about 167 ms) and then moves the piece every 3 ticks (50 ms). When both directions are held, the last one pressed wins. The soft drop makes the piece fall twenty times faster and the hard drop places it at once where the ghost shows. The game handles its own repetition and ignores the operating system’s key repeat, so the timing is identical on every keyboard.

A turn first tries the rotated piece where it is and then, if it overlaps a wall or a block, four alternative offsets, the *kicks*, keeping the first one that fits. The offsets are those of the Super

Rotation System used by modern games of the genre (Table 1); the published tables count rows upward, so the engine flips them once for its board, whose rows grow downward. The O piece does not turn. *Hold* puts the falling piece aside and brings in the one that was set aside before, or the next piece if the hold box is empty; it can be used once per piece, and the held piece returns in its spawn state.

Table 1: The wall kicks: the offsets $(\Delta x, \Delta y)$ tried in order for each turn, with y growing upward as in the published tables. Test 1 is always $(0, 0)$. The other four turns use the same offsets with the opposite sign.

Turn	J, L, S, T, Z				I			
	Test 2	Test 3	Test 4	Test 5	Test 2	Test 3	Test 4	Test 5
$0 \rightarrow R$	$(-1, 0)$	$(-1, +1)$	$(0, -2)$	$(-1, -2)$	$(-2, 0)$	$(+1, 0)$	$(-2, -1)$	$(+1, +2)$
$R \rightarrow 2$	$(+1, 0)$	$(+1, -1)$	$(0, +2)$	$(+1, +2)$	$(-1, 0)$	$(+2, 0)$	$(-1, +2)$	$(+2, -1)$
$2 \rightarrow L$	$(+1, 0)$	$(+1, +1)$	$(0, -2)$	$(+1, -2)$	$(+2, 0)$	$(-1, 0)$	$(+2, +1)$	$(-1, -2)$
$L \rightarrow 0$	$(-1, 0)$	$(-1, -1)$	$(0, +2)$	$(-1, +2)$	$(+1, 0)$	$(-2, 0)$	$(+1, -2)$	$(-2, +1)$

2.4 Gravity and levels

The game starts at level 1 and goes up one level every 10 lines, to level 15. At level ℓ a piece falls one row in

$$t(\ell) = (0.8 - 0.007(\ell - 1))^{\ell-1} \text{ seconds,}$$

the gravity curve of modern games of the genre. Floating-point numbers could round differently on different machines, so the engine never uses them while playing: each level gets an integer gravity $g(\ell) = \text{round}(65536/(60t(\ell)))$, counted in $1/65536$ of a row per tick, which a piece accumulates at every tick and turns into whole rows. Table 2 lists the 15 levels: from one row per second at level 1 the speed grows to more than two rows per tick at level 15.

Table 2: The gravity of the 15 levels: seconds per row from the formula, the integer gravity used by the engine and the resulting speed.

Level	s/row	g (1/65536)	rows/s	Level	s/row	g (1/65536)	rows/s
1	1.000	1,092	1.00	9	0.094	11,634	10.65
2	0.793	1,377	1.26	10	0.064	17,026	15.59
3	0.618	1,768	1.62	11	0.043	25,416	23.27
4	0.473	2,311	2.12	12	0.028	38,709	35.44
5	0.355	3,075	2.82	13	0.018	60,169	55.09
6	0.262	4,169	3.82	14	0.011	95,483	87.42
7	0.190	5,759	5.27	15	0.007	154,742	141.67
8	0.135	8,107	7.42				

2.5 Locking and the end of the game

A piece that rests on the floor or on a block locks after half a second, 30 ticks. A move or a turn on the ground restarts that half second, at most 15 times; after the fifteenth reset the piece locks as soon as it touches. Reaching a row lower than any reached before gives the piece its full delay and its 15 resets back, so that a piece that keeps descending is never forced to lock, while one that only slides or spins in place cannot stall the game forever.

A game ends in one of three ways: after 180 seconds, when the clock runs out (the falling piece then does not count); when a new piece has no room to appear; or when a piece locks entirely above the visible field without clearing a line.

3 Scoring

Points are awarded when a piece locks, by a pure function that receives the move (lines cleared, kind of T-spin, perfect clear, level) and the current streak, and returns the points split by source and the new streak. Table 3 gives the values. Line clears and T-spins are multiplied by the level *before* the clear is counted, so the lines that cause a level up are still paid at the old level.

Table 3: The points of PolyFit. “PolyFit!” is the game’s name for four lines at once.

Move	Points	Notes
Single, Double, Triple	100, 300, 500	× level
PolyFit! (four lines)	800	× level; only the I piece can do it
T-spin with 0, 1, 2, 3 lines	400, 800, 1,200, 1,600	× level
T-spin Mini with 0, 1, 2 lines	100, 200, 400	× level
Back-to-back	×1.5	on a PolyFit! or a T-spin with lines that follows another one with no plain clear in between
Combo	$50 \times c$	× level; c = pieces in a row that clear lines, minus one
Perfect clear	800, 1,200, 1,800, 2,000	× level, for 1–4 lines; 3,200 for a back-to-back PolyFit!
Soft drop, hard drop	1, 2 per row	not multiplied by the level

T-spins. A T-spin is recognized with the *3-corner rule* (Figure 3): the last move of a T piece was a turn, and at least three of the four cells diagonal to its center are filled, walls and floor included. It is a full T-spin if both corners on the side the T points to are filled, or if the turn needed the fifth kick; otherwise it is a T-spin Mini. A Mini cannot clear three lines; should a strange case ever do it, it is scored as a full T-spin.

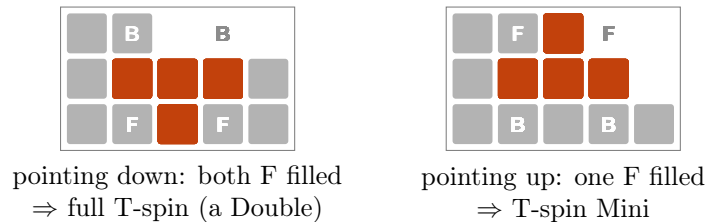


Figure 3: The 3-corner rule, after a T has turned into place. F marks the two corners on the side the T points to, B the two behind its flat side; gray cells are filled. In both cases three of the four corners are filled. On the left both front corners are filled: a full T-spin, here a T-spin Double, since two rows are completed. On the right only one front corner is filled: a T-spin Mini.

Streaks. *Difficult* clears are PolyFit!s and T-spins that clear lines. From the second one in a chain, each is worth half as much again; only a plain Single, Double or Triple breaks the chain, while a piece that clears nothing leaves it as it was. A combo counts the pieces in a row that clear lines and ends at the first piece that clears nothing. The result of every lock also keeps the points of each source apart, lines, PolyFit!s, T-spins, back-to-back, combos, perfect clears and drops, and the results screen shows where the points of a game came from.

As an example, the author’s record game (Section 9) contains six PolyFit!s, worth 800 at level 1, then 1,200 (800 plus a back-to-back bonus of 400, still at level 1), 2,400 at level 2 (1,600 plus 800 of back-to-back), 1,600 at level 2 after a plain clear broke the chain, 3,600 at level 3 (2,400 plus 1,200 of back-to-back) and 3,200 at level 4: the same move is worth four and a half times more when it is chained late in the game.

4 Architecture

4.1 Modules

The code is organized in plain ES modules that the browser loads directly, with no build step. Table 4 lists the folders and Figure 4 how they depend on each other. The essential constraint is that the *engine imports nothing outside its folder and never touches the page*: it has no access to the document, the clock or the network. The same code therefore runs in the page, in the tests, and in the browser of whoever opens a challenge link to replay it.

Table 4: The source code, by folder (lines of JavaScript, comments included).

Folder	Files	Lines	Content
src/engine/	7	800	board, pieces and kicks, random generator and bag, gravity, scoring, the game, the replayer
src/view/	7	1,124	canvas renderer, theme colors, numbers and popups, Statistics panel, results screen, rival, leaderboards
src/audio/	4	485	the shared audio context, sound effects, the music player and the score of the music
src/share/	2	293	challenge links and the Hall of Fame
src/input/	2	161	keyboard and touch
src/	7	847	the main loop and the modes of the page, records, storage, settings, analytics
tests/	10	1,034	test harness, 62 tests, the automatic player

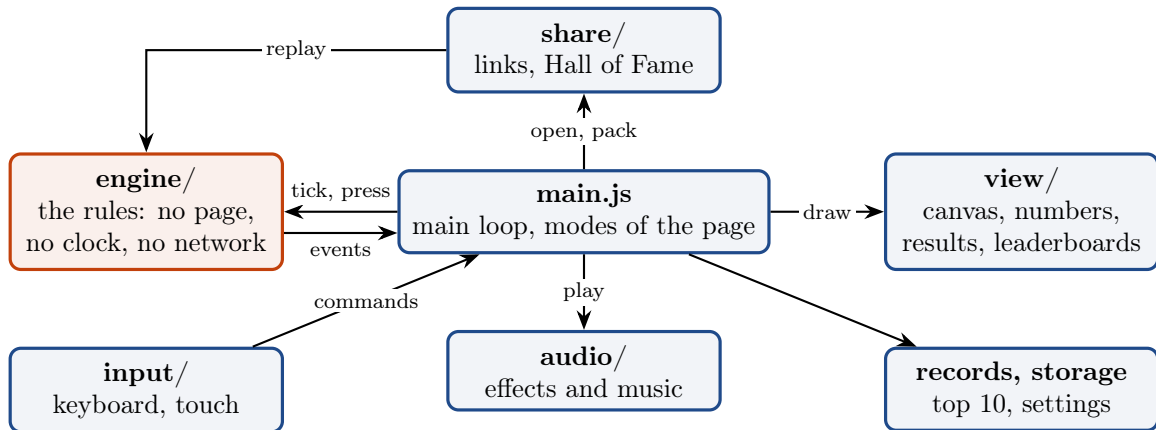


Figure 4: The modules and their dependencies. Only `main.js` knows all the others; the engine is called but never calls out.

4.2 The main loop

The game advances in *ticks* of $1/60$ of a second, whatever the refresh rate of the screen. At every frame, the main loop turns the real time elapsed into whole ticks and keeps the remainder for the next frame [3] (Listing 1). On a 120 Hz display a tick runs every other frame; on a 50 Hz one, some frames run two. After a freeze, for example when the browser stops a background tab, no more than 250 ms are caught up at once. Drawing is done once per frame, after the ticks, so the picture always shows the latest state.

The engine reports what happened as a list of *events*, a move, a turn, a hold, a hard drop, a lock with its lines, points and move name, the end of the game, which the page collects once per frame. The engine therefore never knows how a line clear is shown or heard: the renderer

```
function frame(now) {
  requestAnimationFrame(frame);
  app.pending = Math.min(app.pending + (now - app.lastTime), MAX_CATCH_UP_MS);
  app.lastTime = now;
  while (app.pending >= STEP_MS && game.state === 'playing') {
    game.tick(); // one 1/60 s step of the rules
    app.pending -= STEP_MS;
  }
  react(game.drainEvents(), now); // effects, sounds, popups, screen-reader text
  renderer.draw(game, now);
  hud.update(game);
}
```

Listing 1: The main loop (simplified from `main.js`).

adds a flash, the sound module plays a note, the head-up display shows “T-spin Double” with its points, and a hidden live region tells a screen reader the same.

4.3 The modes of the page

Around the engine, the page moves through a small state machine (Figure 5). A new game starts with a 3-2-1-Go countdown; a pause hides the board, so that the game cannot be studied while the clock is stopped, and resuming shows a short 3-2-1 before play starts again. Leaving the tab or the window pauses the game at once: the clock never runs unseen.

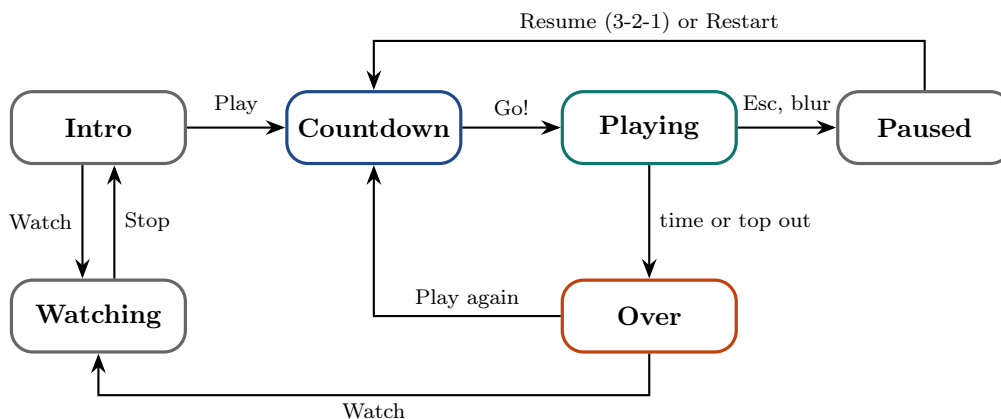


Figure 5: The modes of the page. *Watching* plays a recorded game on the board, at 1×, 2× or 4×.

4.4 Drawing

The board, the hold box, the preview, the rival’s board and the small pieces of the Statistics panel are canvases sized to the screen’s pixel density, so they stay sharp on high-resolution displays. A whole new picture is drawn at every frame. The ghost, where the piece will land, is a dashed outline of its outer edges only. A piece resting on the ground fades by up to 30% while its lock delay runs out, and short effects mark the events: a bright band where a row was cleared (320 ms), a flash on the locked cells (180 ms), a faint trail above a hard-dropped piece (160 ms) and a glowing frame at a level up (900 ms). The top of the field glows when the stack comes within four rows of it.

All colors live in one place, the custom properties of the style sheet, defined once for the light and once for the dark theme. The canvas cannot use CSS directly, so it reads the same properties and reads them again whenever the theme changes; the edge of each square is the piece color

darkened by 22% in the light theme, and the mark inside it is dark on light colors and light on dark ones, chosen by the relative luminance of the color as defined by WCAG [4].

5 Replays and challenge links without a server

5.1 Determinism

A game of PolyFit is fully determined by two things: the seed of its random generator and the list of its inputs, each recorded as a triple (tick, button, pressed or released). Given these, the function `replay(seed, log)` plays the game again and reaches exactly the same board, score, statistics and end. This holds because of five properties of the engine:

1. every random choice comes from the seeded generator, whose arithmetic (`Math.imul` and unsigned 32-bit shifts) is exact in every JavaScript engine;
2. gravity, the lock delay and the key repeat are counted in whole numbers, never in floating point;
3. time exists only as a number of ticks: the engine never reads a clock;
4. an input is recorded with the tick at which it arrives and applied before that tick runs, both in play and in a replay;
5. the frame rate has no influence: the rules see only ticks and inputs.

The tests check this property on full games (Section 8): a game replayed from its seed and inputs ends with the same board, score, number of ticks and statistics, while one changed input gives a different game. Everything in this section rests on it.

5.2 The link

A challenge link is the address of the game page followed by `#c=` and the game itself. The game is packed into bytes as in Figure 6: a format byte, the version of the rules, the game mode, the seed, the day, the player’s name and the inputs. Each input becomes one number, $10\Delta t + k$, where Δt is the number of ticks since the previous input and k a code from 0 to 9: codes 0–5 are the presses and releases of the three buttons that can be held (left, right, soft drop) and codes 6–9 the presses of the other four (hard drop, the two turns, hold). Numbers are written in a variable-length format, seven bits per byte, so most inputs take a single byte. The bytes are then compressed with DEFLATE [5], through the Compression Streams API of the browser [6], and written in the URL-safe variant of Base64 [7]. A browser without compression produces a longer, uncompressed link instead, marked by a different first letter.

The part of an address after `#` is the *fragment*, which is processed by the browser and is not sent to the server when the page is requested [8]. The game therefore travels inside the link without ever reaching the website, its host or its visitor counter, which records only the path of the page. Table 5 gives the sizes. The author’s record game, 831 inputs in three minutes, packs into 1,146 bytes and becomes a link of 1,362 characters; the automatic player of the tests produces three to five times more inputs, but at very regular intervals, which compress much better than a human’s.

Table 5: Sizes of real games packed into links.

Game	Inputs	Packed bytes	Characters after <code>#c=</code>
The author’s record (27,638 points)	831	1,146	1,319
Automatic player, five seeds	2,626 – 3,736	3,021 – 4,287	787 – 1,063

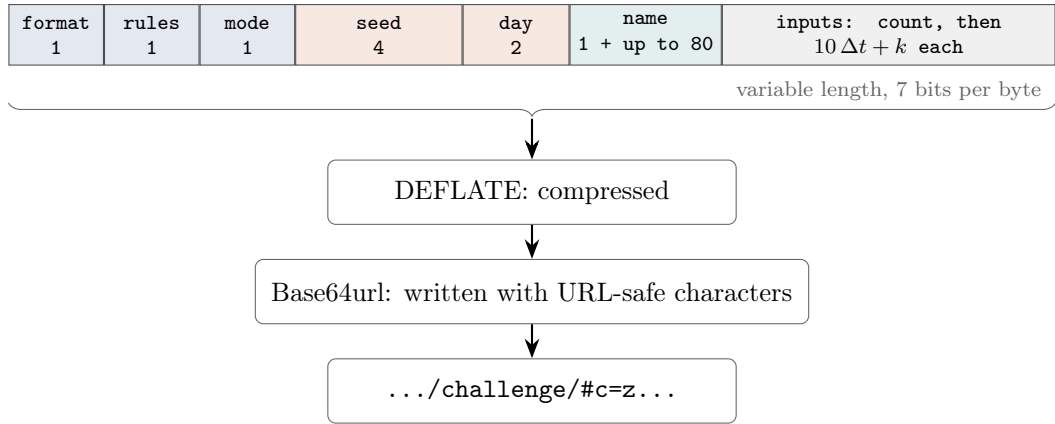


Figure 6: The bytes of a challenge link, with their sizes in bytes, and how they become the text after #c=. The name is at most 20 characters, up to 80 bytes in UTF-8.

5.3 Opening a link safely

A link comes from outside and can be damaged, truncated or crafted. Decoding checks every step and refuses the link with one of three reasons: *damaged*, *old*, when it was made with another version of the rules or another mode, or *browser*, when the browser cannot decompress it. The checks include:

- the decompressed data may not exceed 1 MiB, so that a small link cannot unpack into a huge one;
- at most 30,000 inputs (a frantic player makes about 3,000 in three minutes), no input after the 10,800th tick, and no bytes left over at the end;
- the name is normalized and keeps only letters, digits, spaces and the signs . - _ ', at most 20 characters: no links, no addresses and no invisible characters that could reverse the direction of the text. Every name, in the game and in the leaderboards, is inserted in the page as text, never as HTML.

The version of the rules travels in every link: a future change of the engine must raise it, so that an old link is refused instead of replaying into a different game.

5.4 The score comes from the replay

A link contains no score. The browser that opens it decodes the inputs and replays the game to compute its score, then shows “Verified: replayed in your browser, move by move”. A score can be raised only by changing the inputs, and then the replay gives whatever those inputs really score. Every game also has a *fingerprint*: the 32-bit FNV-1a hash [9] of the rules, the mode, the seed and the inputs, written as eight hexadecimal digits. The name and the day are left out, so two copies of the same game carrying different names have the same fingerprint, and the Hall of Fame keeps only the first.

5.5 Playing against a replay

Whoever opens a link plays the same pieces, in the same order, as the challenger, whose game is replayed next to the board in a small panel, in step with the player: at every frame, the **Replayer** advances the recorded game to the same tick as the player’s, applying its inputs as they come. Under the score, a split time shows the gap at that moment, “−785 vs my record” in Figure 7, like the intermediate times of a race. Without a challenge the same mechanism replays the player’s own best game, kept on the device, and the split time reads “vs your best”. *Watch* uses it once more to play any recorded game on the board, at one, two or four times the speed.

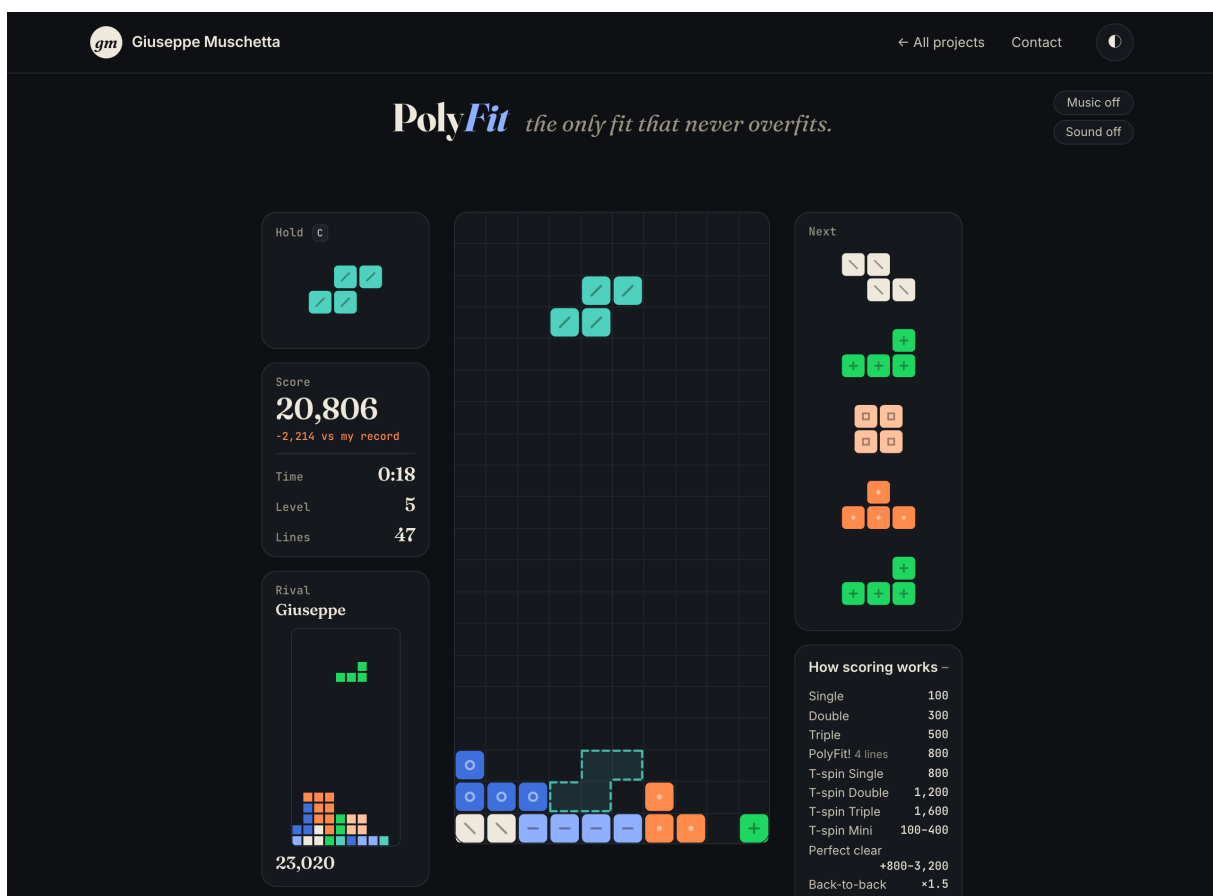


Figure 7: A race against the author’s record game, in the dark theme, 18 seconds before the end. The panel on the left replays the record game in step with the player, on the same pieces; under the score, the gap at that moment.

5.6 Leaderboards

Two leaderboards need no server. The *top 10 on this device* works like the high-score table of an arcade cabinet: the best ten games of the browser with the name typed at the end, each stored with its packed game so that it can be watched again. The *Hall of Fame* lists the players who beat the author’s record and sent him their link: the list of links is part of the source code, and every visitor’s browser replays each entry when the page opens, pausing whenever a game is running so that no replay ever takes time away from play, and sorts them by the replayed score. The author’s own record is a link too, behind the *Race my record game* and *Watch my record game* buttons of the start screen.

5.7 What a replay proves

A replay proves that a listed score is exactly what its inputs produce under the rules of the game: no score can be typed, edited or inflated. It cannot prove that a human produced the inputs. A program that plays the game can make a valid link, and the automatic player of the tests scores up to 307,192 points in three minutes, more than eleven times the author’s record. This is the reason why the Hall of Fame is curated by hand: the replay rules out made-up scores, and a person rules out games that no human could have played. An open global leaderboard would need a server and would face the same question; the decision to keep PolyFit entirely static was taken knowing this limit.

6 The interface

6.1 Layouts

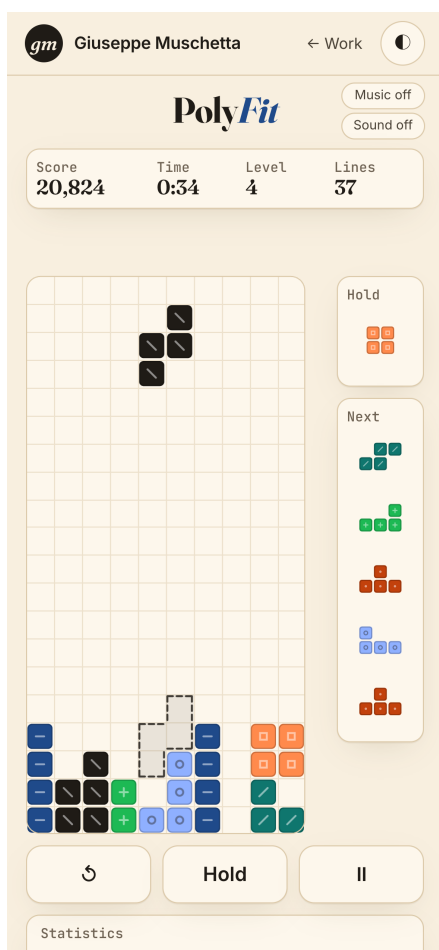
On a desktop the page has three columns (Figure 1): hold, score and Statistics on the left, the board in the middle, the preview and the scoring table on the right. The board is as tall as the window allows, between 440 and 780 pixels, and narrow enough to leave room for the side panels; its cell size, and with it the size of every panel drawn on canvas, follows from that height. On phones (Figure 8a) the numbers become a strip above the board, hold and preview a narrow column on its right, and the Statistics a strip under the touch buttons. The layout was checked at twelve window sizes, from 320×640 to 1440×900 , for overflow, text that wraps badly and a board that does not fit.

6.2 Controls

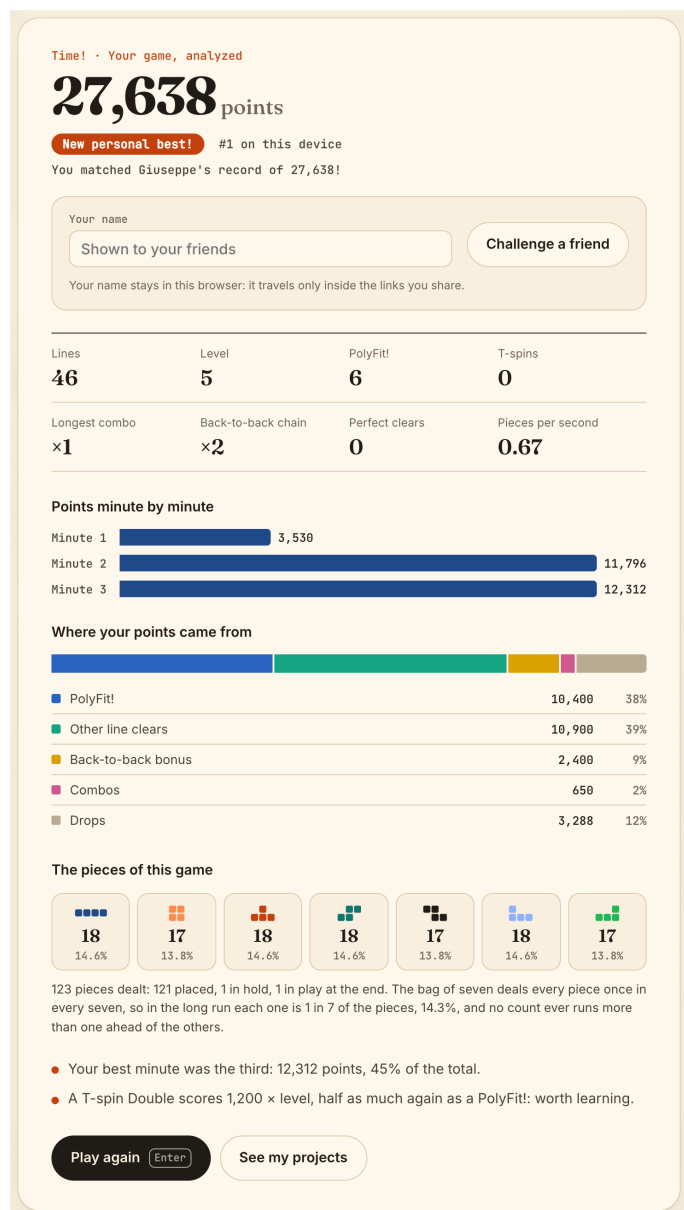
The keyboard is read by *physical* key (`KeyboardEvent.code`), so the controls stay in the same place on every keyboard layout: the arrows move, soft-drop and turn, Space hard-drops, Z and X turn left and right, C or Shift hold, Esc or P pause, M switches the sound and N the music. While a name is being typed, these keys are letters again. On touch screens the board itself is the controller (Table 6), with three buttons under it for the left turn, hold and pause. Gestures are told apart by distance, duration and speed; the mouse is left to keyboard players.

Table 6: The touch gestures and their thresholds.

Gesture	Action	Recognized when
Drag sideways	move	one column for every 0.9 cells of finger travel
Tap	turn right	the finger moves less than 12 px and lifts within 300 ms
Slow drag down	soft drop	downward speed under 0.5 px/ms, while the finger stays down
Flick down	hard drop	a short gesture (under 250 ms) or a fast end (over 0.9 px/ms)
Flick up	hold	the same, upward



(a) On a phone, at the same moment as Figure 1.



(b) The results of the author's record game.

Figure 8: The phone layout and the results screen.

6.3 Feedback and comfort

Named moves pop up over the board, “PolyFit!”, “T-spin Double”, “Back-to-back $\times 1.5$ ”, “Combo $\times 3$ ”, with their points, at most two at a time. The score counts up quickly instead of jumping, the clock pulses in the last ten seconds, and the Statistics panel lights for half a second the count of the piece just dealt. Every visual cue has a non-visual or non-color counterpart: the marks inside the pieces (which can be switched off), the live region for screen readers, and charts on the results screen whose values are also written as text. When the system asks for less motion, the score no longer counts up and the drop trails disappear. The music and sound buttons are 24 pixels tall on phones, the minimum target size of WCAG 2.2 [4], and both start switched off: a visitor who opens the page at the office never hears anything unexpected.

6.4 The results

At the end of a game, *Your game, analyzed* (Figure 8b) shows the score and how it stands against the player’s best, the device’s top 10 and the author’s record; a box to type a name and make a challenge link; the main numbers of the game, including the pieces per second; the points minute by minute; where the points came from, as a bar and as a table; the pieces of the game, how many of each shape came into play and its share of all the pieces dealt (with the 7-bag, 1 in 7 in the long run); and two remarks: the best minute, and a tip chosen from what the game lacked, for example the value of a T-spin Double for a player who never made one.

7 Sound and music

7.1 Sound effects

Every sound is synthesized when it plays, with the Web Audio API [10]: an oscillator, an envelope with a quick attack and a short fade, and sometimes a sliding pitch. Moving, turning, holding, locking, hard-dropping, clearing lines, levelling up, the countdown and the end of the game each have their own short sound. The notes are taken from a pentatonic scale on C5, so that sounds that overlap are never out of tune with each other: a line clear plays one rising note per line, brighter for a T-spin, and a PolyFit! adds a chord. A compressor keeps several sounds at once from getting too loud. Browsers allow a page to play audio only after a click or a key press, so the sound and the music share one audio context, created on first use and woken by the first click or key press.

7.2 The music

The background music is *Korobeiniki*, a Russian folk song on a poem by Nikolay Nekrasov of 1861 [11], which is in the public domain, in an 8-bit arrangement written for PolyFit. The melody is played twice, followed by a second part of its own, eight bars on the chords F, C, G, Am, F, C, then Dm and E, with lighter drums, which leads back to the beginning: 24 bars in all, 96 beats at 150 beats per minute, 38.4 seconds, played in a loop. The score is written in the source as plain text, one word per note (E5 B4/.5 C5/.5 D5...), and compiled at start-up into 516 timed events.

The arrangement has four voices, like the sound chips of 8-bit handheld consoles (Figure 9):

- the **melody**, on a pulse wave with a duty cycle of 50%, with a slight vibrato on long notes;
- a **second voice**, a quieter pulse wave with a duty cycle of 12.5%, that follows the melody a third below on the scale of the chord underneath: A natural minor, or A harmonic minor (with G sharp) over the E chord;
- the **bass**, a triangle wave that alternates every half beat between the root of the chord and its octave;

- the **drums**: a kick made of a triangle wave falling from 170 to 42 Hz in 90 ms, and a snare and a hi-hat made of short bursts of noise, filtered around 1.9 kHz and above 7 kHz.

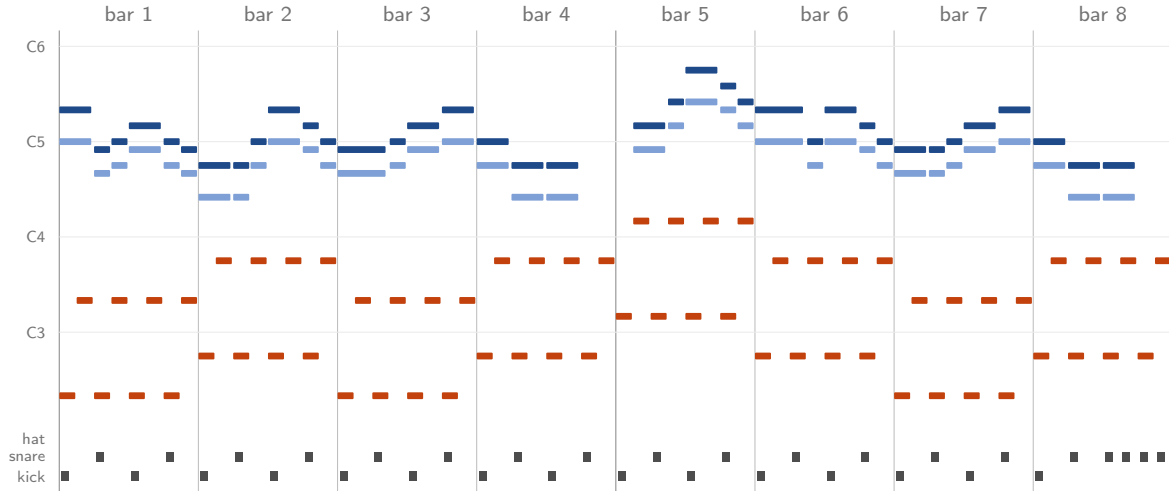


Figure 9: The first eight bars of the music: the melody (dark blue), the second voice a third below (light blue), the bass jumping by octaves (orange) and the drums. Each rectangle is a note, its length the note's length.

A pulse wave with duty cycle d , high for a fraction d of each period, has the Fourier series

$$p(t) = d + \sum_{k \geq 1} \left(a_k \cos 2\pi k f t + b_k \sin 2\pi k f t \right), \quad a_k = \frac{\sin 2\pi k d}{k\pi}, \quad b_k = \frac{1 - \cos 2\pi k d}{k\pi},$$

and the music builds each pulse wave from its first 47 harmonics as a `PeriodicWave`. The noise comes from a 15-bit linear-feedback shift register, the way the old chips produced it: at every step the two lowest bits are combined by exclusive or, the register shifts right and the result enters at bit 14; stepped about 16,000 times per second, the lowest bit gives the samples $+1$ or -1 .

7.3 Keeping time

Notes are not played when a frame happens to run, which would make them stutter whenever a frame is late, but scheduled on the audio clock about 0.2 seconds ahead: at every frame, the player sends to the audio system the notes that start in the next 0.2 seconds, with their exact start times. A change of tempo moves a pair (beat, time) to the present and computes the following notes from there. The music plays only while the clock of the game runs: it stops with a 20 ms fade at a pause, remembers where it stopped and resumes there; a new game starts it again from the first bar; and in the last 30 seconds it plays 15% faster, to add some urgency.

7.4 Mixing

The level of the music was set by ear with the author, who found the first one too loud against the sound effects. The final level was then measured in the browser with an analyser node: over three seconds of play, the music has a median RMS level of 0.041, while the sound effects of moving, turning and hard-dropping reach 0.090 (95th percentile of 20 ms windows). The effects stand about 6.8 dB above the music, so they always come through.

8 Testing

8.1 The test suite

The tests run in the browser, from `tests/index.html`, with a harness of 49 lines written for the project: `test()` registers a test, three assertions (`ok`, `equal`, `deepEqual`) throw an error when a check fails, and `run()` executes everything and reports. The tests import the very same modules as the game. Table 7 lists the 62 tests by file; the whole suite, including the long games, runs in about 3.3 seconds.

Table 7: The 62 automated tests.

File	Tests	What they check
<code>engine.test.js</code>	13	the generator (same seed, same numbers; values in $[0, 1)$), the bag (every group of seven holds each piece once), the rotation states and kicks, line clears, walls and floor, the gravity formula and the levels
<code>game.test.js</code>	20	spawning, falling at level 1, key repeat after 10 ticks then every 3, the last direction wins, soft and hard drop points, lock delay and its 15 resets, hold, piece counts, a Single, a PolyFit! then back-to-back, a real T-spin Double turned into its slot, the 3-corner rule, perfect clear, scoring at the level before a level up, the three ways a game ends
<code>scoring.test.js</code>	10	every row of Table 3, back-to-back chains, combos, perfect clears, points by source, move names
<code>share.test.js</code>	11	a game comes back from its link identical; the score comes from the replay; one changed input gives another fingerprint; the fingerprint ignores name and day; link size; damaged and foreign links refused; names cleaned; the <code>Replayer</code> in step with <code>replay()</code> at every tick; the top 10
<code>music.test.js</code>	4	reading the score, no note across a bar line, chords as long as the melody, the third below, the compiled song in time order
<code>stress.test.js</code>	4	ten games of the automatic player and three of random key mashing, with every rule checked at every tick; replays identical; a changed input changes the game

8.2 Invariants at every tick

The long tests do not check a final result only. After every single tick of ten games played by the automatic player and three games of random key mashing, they check that: the board still has 40 rows of 10 cells; no full row was left on the board; the falling piece never overlaps the stack; the score equals the sum of its parts by source; the score never decreases; the level matches the number of lines; every piece in play was counted exactly once by the Statistics (dealt = locked + falling + held); and the counts of the seven pieces never differ by more than one.

8.3 The automatic player

The automatic player tries every rotation and column for the current piece and for the one in hold, drops it in its head, and scores the board that would result with six features introduced by Pierre Dellacherie, landing height, rows cleared, row transitions, column transitions, holes and wells, weighted with the values published by Islam El-Ashi in 2011. It then plays its choice through the real inputs, hold, turns, moves and a hard drop, so it exercises the same code paths as a human. Thinking 20 ticks before every piece, it scores between 143,212 and 307,192 points in three minutes on the five seeds of the tests (Table 8). It plays for safety, clearing lines one at a time without PolyFit!s or T-spins, and its points come from reaching the highest levels, where every clear is multiplied by 14 or 15; two of its games end by top out at those levels, where pieces fall more than two rows per tick while it is still thinking.

Table 8: Three-minute games of the automatic player (20 ticks of thinking before every piece).

Seed	Score	Lines	Level	Pieces	End
11	143,212	139	14	383	top out
22	290,572	214	15	539	time
33	278,308	209	15	539	time
44	307,192	215	15	539	time
55	187,360	165	15	446	top out

8.4 Beyond the suite

During development the page was also checked in a headless browser: the layout at twelve window sizes, the security policy of the website, which forbids inline scripts (the small script that applies the saved theme before the page is drawn is a separate file for this reason), and the levels of the music and the effects, measured with an analyser node (Section 7). When the game is copied into the website, the build opens every challenge link of the Hall of Fame and the record game, and stops if one is damaged or listed twice.

9 Results

9.1 The record game

The author’s record, 27,638 points, was played on 26 October 2026 and is published as a link, so anyone can race it or watch it. Table 9 summarizes it and Figure 10 shows how the score grew. In 180 seconds, 121 pieces were placed, 0.67 per second, with 831 inputs: 575 presses, among which 188 right turns and no left turn at all, and 256 releases. Plain line clears brought 39% of the points and PolyFit!s 38%. The level multiplies every clear, so the score grew faster as the game went on: 3,530 points in the first minute, 11,796 in the second and 12,312 in the third, 45% of the total, where a single PolyFit! at level 4, at 144 seconds, was worth 3,200 points. The Statistics panel ended with 17 or 18 of each piece: 123 pieces dealt, the 121 placed plus the one in hold and the one falling when the time ran out.

Table 9: The record game, replayed from its link.

The game		Where the points came from	
Score	27,638	Line clears	10,900 (39%)
Lines, final level	46, 5	PolyFit!	10,400 (38%)
Singles, Doubles, Triples, PolyFit!	6, 5, 2, 6	Back-to-back	2,400 (9%)
Longest combo, back-to-back chain	×1, ×2	Combos	650 (2%)
Pieces, per second	121, 0.67	Drops	3,288 (12%)
Inputs, holds	831, 9		

9.2 Size and speed

The page of the game is made of 31 files, 183 KB in all and about 64 KB compressed, with no dependencies: the fonts are those of the website. The JavaScript of the game is 3,776 lines and the tests 1,034 more. Replaying a whole three-minute game, as the browser does for every link it opens and for every entry of the Hall of Fame, takes 1.0 ms (median of 20 replays of the record game in Chrome 156 on an Apple M5 Max laptop): a Hall of Fame of a hundred games replays in about a tenth of a second.

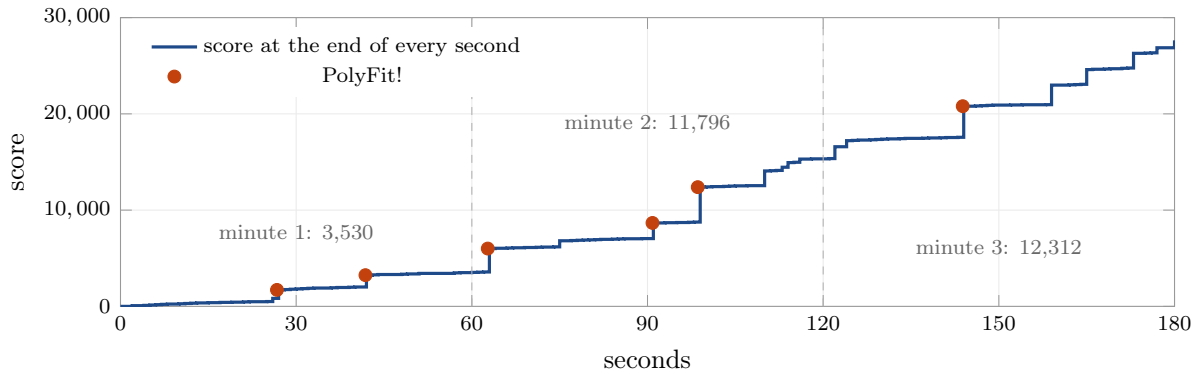


Figure 10: The score of the record game over time. The steps are line clears; the orange dots mark the six PolyFit!s.

10 Conclusion

PolyFit shows that a static page can offer what usually needs a server. Making the engine deterministic, with integer arithmetic, a seeded generator and inputs recorded by tick, turned every game into a small piece of data that can travel inside a link, be replayed anywhere and be compared with any other game on the same pieces. On that single property rest the challenge links, the rival replayed in step, the split times, Watch, the device’s top 10 and a Hall of Fame whose scores cannot be made up. The same property made the game testable in depth: an automatic player can play thousands of moves in seconds while every rule is checked at every tick, and every game it plays must come back identical from its seed and inputs.

The limits are known and accepted. A replay proves that a score is legal, not that a human played it, so the Hall of Fame stays curated; and without a server every leaderboard is either local or curated. Several extensions fit the same design and need no server: a *daily seed*, the same pieces for everyone on the same day, so that scores of different players become directly comparable; an opponent of selectable strength, built on the automatic player that already exists; medals for the moves of a game; and a sprint mode, 40 lines in the shortest time.

References

- [1] T. Ettinger. *Mulberry32*, a 32-bit pseudo-random number generator, 2017.
- [2] D. E. Knuth. *The Art of Computer Programming*, Volume 2: Seminumerical Algorithms, 3rd edition, Section 3.4.2, Algorithm P (shuffling). Addison-Wesley, 1997.
- [3] G. Fiedler. *Fix Your Timestep!* Gaffer On Games, 2004. https://gafferongames.com/post/fix_your_timestep/.
- [4] W3C. *Web Content Accessibility Guidelines (WCAG) 2.2*, W3C Recommendation, 2023: relative luminance; Success Criterion 2.5.8, Target Size (Minimum).
- [5] P. Deutsch. *DEFLATE Compressed Data Format Specification version 1.3*, RFC 1951, 1996.
- [6] WHATWG. *Compression Streams*, Living Standard. <https://compression.spec.whatwg.org/>.
- [7] S. Josefsson. *The Base16, Base32, and Base64 Data Encodings*, RFC 4648, Section 5: Base 64 Encoding with URL and Filename Safe Alphabet, 2006.
- [8] T. Berners-Lee, R. Fielding and L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*, RFC 3986, Section 3.5: Fragment, 2005.
- [9] G. Fowler, L. C. Noll, K.-P. Vo, D. Eastlake and T. Hansen. *The FNV Non-Cryptographic Hash Algorithm*, IETF Internet-Draft draft-eastlake-fnv.
- [10] W3C. *Web Audio API*, W3C Recommendation, 2021.
- [11] N. A. Nekrasov. *Korobeiniki* (The Peddlers), poem, 1861; set to music as a Russian folk song.