UMÁTICA. Revista sobre Creación y Análisis de la Imagen
{ISSN: 2659-5354 // D.L.: MA-1628-2018}

Versos ejecutables y comandos de colores: debates entorno a la poesía código

Executable verses and colourful commands: debates around code poetry

MARÍA GARAY ARRIBA 0009-0004-9934-8572
Universidad de Salamanca (USAL), España.

Resumen

La poesía escrita en lenguajes de programación, llamada poesía código, pertenece a la literatura electrónica. En ella, quien lee se enfrenta al código fuente, no al resultado ejecutado. El debate principal dentro del «codework» gira en torno a su ejecutabilidad: si el poema puede ser interpretado por la máquina y generar un output. Tradicionalmente, se valora más la obra que compila y produce efectos visuales o verbales, pues articula procesos de máquina y lectura humana. Este artículo retoma la discusión a partir de *./code --poetry* (2023) de Daniel Holden y Chris Kerr, que sostiene que toda lectura de pantalla es una lectura superficial, incluso cuando usa lenguajes de programación. Luego se explora su parentesco con la poesía concreta y el movimiento OuLiPo, destacando cómo al usar lenguajes diseñados para órdenes claras, la poesía código interrumpe su funcionalidad y revela la materialidad oculta de esas herramientas.

PALABRAS CLAVE: poesía código, funcionalidad, literatura electrónica, poética materialista, extrañamiento.

Artículo original
Original Article

Correspondencia/
Correspondence
María Garay Arriba
mgarayarriba@usal.es

Financiación/Fundings
Este texto forma parte de los resultados del Proyecto de investigación: ESCON -- Escrituras en contacto. Redes de escritura intermedial en la era de la globalización analógica (1961-1991), PID2024-159610NB-I00, financiado por MICIU/AEI/10.13039/501100011033/FEDER, UE

Received: 27.04.2025
Accepted: 19.11.2025

CÓMO CITAR ESTE TRABAJO / HOW TO CITE THIS PAPER

Garay Arriba, M. (2025). Versos ejecutables y comandos de colores: debates entorno a la poesía código. *Umática. Revista sobre Creación y Análisis de la Imagen, 8.*
https://doi.org/10.24310/Umatica.2025.v8i8.21791

Executable verses and colourful commands: debates around code poetry

María Garay Arriba
University of Salamanca (USAL), Spain.

Abstract

Code poetry, written in programming languages, belongs to electronic literature. In it, the reader faces the source code rather than the executed result. The main debate within "codework" revolves around executability: whether the poem can be interpreted by the machine and generate an output. Traditionally, works that compile and produce visual or verbal effects are valued more, as they engage both machine processes and human reading. This article revisits the discussion through *./code --poetry* (2023) by Daniel Holden and Chris Kerr, arguing that every screen reading is a surface reading, even when using programming languages. It then explores its link with concrete poetry and the OuLiPo movement, highlighting how, by using languages designed for unambiguous commands, code poetry interrupts their usual function and exposes the hidden materiality of these tools.

KEY WORDS: code poetry, functionality, electronic literature, materialist poetics, estrangement.

Summary – Sumario

1. Introducción

Aunque hoy en día casi toda la literatura pasa por procesos de producción (y, a menudo, de recepción) digitales, es muy reducida la producción literaria que se pregunta por esta condición, que la reconoce y apela a ella en su propia forma. Las prácticas literarias que, en el último medio siglo, aproximadamente, han surgido como «nativas» dentro de ordenadores y que son inextricables del medio digital conforman lo que ha venido propiamente a llamarse literatura electrónica. Las obras de literatura electrónica no podrían haber sido creadas ni tienen sentido leídas fuera de los dispositivos en los que han nacido; el hardware, software y la programación son parte íntegra de la experiencia y del significado de cada pieza (Tomasula, 2012, p. 483).

En *Writing Machines* (2002), Katherine Hayles establece varios parámetros necesarios para la crítica de la literatura electrónica. En primer lugar, repara en que tradicionalmente los estudios literarios, con excepciones notables, han relegado la materialidad del texto a un último plano si es que estaba presente siquiera. Afirma que, con la llegada de la literatura electrónica, esta falta de atención a las condiciones materiales del texto ya no es posible: «la forma física del artefacto literario siempre afecta a lo que las palabras (y otros componentes semióticos) significan». No todas las obras literarias interrogan las «tecnologías de inscripción» que las producen, pero, para aquellas que sí lo hacen, Hayles propone el término «tecnotextos», los cuales «movilizan *loops* reflexivos entre su mundo imaginativo y el aparato material» que proporciona presencia física a la creación (Hayles, 2002, p. 25)[1].

Pese a este optimismo hacia la interrogación de las «tecnologías de inscripción», desde sus primeros momentos, la literatura electrónica ha tendido a centrarse en el texto sobre el interfaz. Aprovecha propiedades formales de la tipografía digitalizada en pantalla, la capacidad que el signo adquiere de moverse y cambiar, así como la opción a la participación activa del usuario en el desarrollo del texto y las posibilidades de generación de textos múltiples a partir de un solo programa. Se puede decir que la atención se ha dirigido a jugar con lo que el código y el lenguaje sobre el interfaz pueden *hacer*, más que a lo que *son*. El texto que se escribe raramente es el que se lee. Es poco habitual encontrarse en los medios programables con literatura que, además de hacer uso de estas propiedades, trate de exponer su propio aparataje. La poesía código—poesía escrita en lenguajes de programación o cuya estética textual refiere al código—es de los únicos tipos de literatura electrónica en la que lo escrito por el autor es el mismo texto al que se enfrentará la mirada de potenciales lectores. Las publicaciones más conocidas de poesía código son la colección editada por Ishac Bertran en 2012[2] y el *./code --poetry* (2023) de Daniel Holden y Chris Kerr[3]. Dentro del entorno ibérico,

1 Todas las traducciones de citas al español son a mano de la autora.

2 Bertran, I. (Ed.). (2012). *code {poems}*. Ishac Bertran. https://monoskop.org/File:Code_Poems_2012.pdf

3 Holden, D., & Kerr, C. (2023). *./code—poetry*. Broken Sleep Books and code-poetry.com/home.

cabe destacar que el poemario *A 6000 metros de profundidad* (2023) de Belén García Nieto incluye tanto poemas en lenguaje natural como en código[4].

La poesía código se inscribe dentro de un espacio artístico más amplio en el que participan el arte de software y la literatura electrónica, así como toda una serie de prácticas híbridas que emplean procedimientos computacionales o relacionados con la computación. La terminología para denominar lo que es conocido como literatura electrónica y sus subgéneros ha sido un problema constante. El poeta y teórico John Cayley argumenta que mucha de la literatura llamada electrónica guarda escasa relación con esa materialidad de los dispositivos a los que la palabra «electrónica» hace referencia. Por otro lado, apunta que la palabra «digital» con la que a menudo se apellida a la poesía y el arte nacidas en medios programables, no añade nada hoy en día, pues la cultura en la que se desarrollan es una, de por sí, históricamente digital (Cayley, 2018, p. 7). Anteriormente, Cayley ha utilizado el término más específico de «poesía programada y en red» para referirse a su propia producción (Cayley, 2002); y, finalmente, ha elegido asentarse en el término «arte digital de lenguaje» (Cayley, 2018, p. 7). Lo que hay tras estos tumultos terminológicos es una importante pregunta acerca de qué existencia material halla el arte de lenguaje (dentro del cual figura el arte literario como subcampo-institución privilegiado) en los dispositivos programables, y qué expectativas existen tras los nombres que le han sido dados. Esta problemática se reproduce con especial intensidad en la poesía código.

2. El problema de la transparencia

El término más acogido en inglés para la escritura en medios programables atravesada por la estética y el funcionamiento del código fuente (a diferencia de aquella escrita únicamente en lenguaje natural) es «*codework*», acuñado por Alan Sondheim y definido por Rita Raley como «el uso del idiolecto contemporáneo del ordenador y de los procesos computacionales en la escritura experimental en los medios digitales» (Raley, 2002). Bajo esta definición, Raley agrupa toda una serie de obras que incorporan elementos de código, ya sea el texto resultante un objeto ejecutable o no. Tanto «*codework*» como «poesía código» son nombres que dan al lector la impresión de que el texto que va a leer será código. Sin embargo, son muchos más los *codeworks* cuyo texto es un discurso únicamente legible por humanos en el que se han integrado fragmentos de código o elementos referentes a su apariencia que aquellos cuyo texto es, en efecto, código funcional y potencialmente ejecutable.

En las últimas décadas del siglo pasado, la literatura electrónica estaba imbricada con esperanzas políticas basadas en el utopismo digital que se respiraba en los años inmediatamente anteriores a la privatización de las arquitecturas de internet. En 1990 se abrió una fugaz ventana durante la que, para muchos, la red representaba nuevas posibilidades de organización política y sus protocolos parecían prometer el fin del poder corporativo (Turner, 2006, p. 257). Se empezó a cerrar desde que se retiraron las restricciones comerciales en in-

4 García Nieto, B. (2023). *A 6000 metros de profundidad*. La Oveja Roja.

ternet en 1994 hasta que Google empezó a enclaustrar los comunes digitales (proceso cuya culminación Cayley fija en 2004, con el lanzamiento de Gmail; Cayley, 2020). Los valores principales de este utopismo digital, según Cayley, eran la transparencia y la traducibilidad. Así, el uso de código (o cosas que parecen código) para hacer poesía podría satisfacer la aspiración de que el interfaz mostrase lo que hay más allá: «La utopía del *codework* reconoce que los símbolos del lenguaje de superficie (lo que lees en pantalla) son lo "mismo" ... que los símbolos de los lenguajes de programación que guardan, manipulan y muestran el texto que tú lees» (Cayley, 2002).

Los debates acerca de la clasificación del *codework* estaban teniendo lugar en los años más próximos al cambio de siglo. En 2001, Sondheim había delineado una taxonomía con las siguientes categorías: «obras que utilizan los juegos intersintácticos del lenguaje en la superficie», «obras en las cuales el contenido sumergido ha modificado el lenguaje de la superficie», «obras en las cuales el código sumergido es contenido emergente» (citado en Raley, 2002). Cayley publicó su ensayo «The Code is Not the Text (Unless It Is the Text)» en la EBR en 2002 con robustos argumentos a favor de diferenciar tipos de *codework* que afectan a sistemas distintos. A diferencia del aquellos cuyo texto no es código que compila, los poemas cuyos textos son potencialmente ejecutables activan la facultad fundamental del código: ser texto performativo con la capacidad de «alterar el comportamiento de un sistema» (Cayley, 2002). Resulta sorprendente entonces, especialmente en el caso de Cayley, que no se haga referencia alguna en los escritos del momento a la Perl Poetry que Sharon Hopkins y Larry Wall estaban escribiendo desde 1990 y las otras formas emergentes de escritura creativa en lenguajes de programación que aparecieron en torno al cambio de milenio.

En los espacios entre los que se compartía la Perl Poetry durante aquellos años, se aceptaban todo tipo de poemas, ejecutables y no ejecutables, pero parecía haber un consenso acerca del especial valor de los que producían *output* por su mayor dificultad de composición[5]. Recientemente, Daniel Holden y Chris Kerr desarrollaron una tipología que escoge no utilizar criterios jerárquicos de superficialidad y profundidad, sino una matriz en torno a la performabilidad del texto; es decir, su interpretabilidad. Comprende ocho tipos de *codework*, cada uno con una combinación distinta de respuestas a las siguientes preguntas: ¿el texto es interpretable por una máquina? ¿Es interpretable por un ser humano? Y, por último, al ser interpretado por la máquina, ¿el texto produce un *output* además de sí mismo? Al organizarlo en torno a las posibilidades de interpretación, están también jugando con la palabra «*performance*», que en inglés se utiliza tanto para referirse a una actuación (interpretación teatral, musical, audiovisual, etc.), como para hablar del rendimiento de una máquina, aquí enfocado a si la máquina puede sacarle rendimiento al *codework* (Holden y Kerr, 2023, p. 171).

5 Ver resultados del concurso de haikus en Perl en https://history.perl.org/CHI/, o lo que dice Hopkins en «Camels and Needles» sobre las composiciones de Craig Counterman (Hopkins, 1992).

Holden, programador, y Kerr, poeta, compusieron esta tipología a propósito de su propio proyecto colaborativo código *.code/ --poetry* (https://code-poetry.com/)[6]. Consiste en una serie de poemas código, cada uno escrito en un lenguaje distinto. Todos producen un *output*, en general en forma de un objeto visual (y a menudo verbal) animado. Es el trabajo más completo que se ha realizado a día de hoy en cuanto al despliegue de todas las propiedades de la escritura en lenguajes de programación. A diferencia de la mayor parte de poesía código, existe libremente accesible en la web[7], donde se ven el código fuente y su *output* lado a lado, activando así las lecturas recíprocas de ambas partes que, inseparables, constituyen el poema al completo. Cada poema, por supuesto constreñido por el vocabulario del lenguaje en el que está escrito, también hace esfuerzos por reflejar el carácter de dicho lenguaje. y no se espera que, por trabajar con código, la interpretabilidad por parte de la máquina sea una condición necesaria del texto.

En el libro de Holden y Kerr, el código a veces «es el texto» y a veces no. Hay poemas en los cuales la parte interpretable por humanos está escrita en forma de comentario (por definición, no interpretable por la máquina) dentro del programa mientras que, en otros, la integración del discurso interpretable por humanos con el interpretable por la máquina es total. Aun así, las partes de código no interpretables verbalmente por humanos (sin demasiada alfabetización computacional) queda reintegrada en la apelación al lector humano a través de sus propiedades visuales. No por hacer poemas ejecutables los autores han dejado de prestar atención al texto sobre el interfaz: utilizan el resaltado de sintaxis para contribuir a la legibilidad y a la construcción de significado, lo cual también hace referencia a las formas visuales más convencionales en el mundo de la programación (como son los modos oscuros de pantalla, el resaltado de sintaxis útil, o el tipo de letra monoespaciado en el que cada carácter ocupa la misma longitud horizontal) (Holden y Kerr, 2023). El texto oscila entre proporcionar información a la máquina, al humano y a ambos.

En parte respondiendo a «The Code is Not the Text», Raley escribe un artículo también en la EBR titulado «Interferences: [Net.Writing] and the Practice of Codework» (2002), donde afirma que «el ímpetu del arte contemporáneo con código» es «revelar códigos, para hacer visible al público el mecanismo de producción» (Raley, 2006). Raley está sugiriendo que este arte de lenguaje nos da acceso a la maquinaria interna del medio, como si la transparencia que apuntaba Cayley como valor utópico encontrase su impulso en el *codework*. Sondheim hace una alusión similar al decir que el código sumergido se vuelve contenido emergente en estas prácticas literarias.

La implicación mayor del problema del valor utópico de la traducibilidad que señalaba Cayley en el *codework* no es que la transparencia sea engañosa (o que no sea transparencia real), sino que parecería que, al traer el código a superficie, también se traspasa a superficie

6 No se reproducen aquí imágenes de poemas código porque se considera que el lugar apropiado para interactuar con ellos es su propio lugar de residencia, donde están en funcionamiento, con lo que animamos al lector a acudir a la web directamente.

7 Ver lista de publicaciones de poesía código en el apéndice en este artículo.

o se extiende esa capacidad para afectar al comportamiento del sistema con simplemente indicar su existencia. Si el texto no es código ejecutable y ejecutado, esa capacidad no es traducida a superficie, sólo señalada. Que los lenguajes de programación estén concebidos para crear textos destinados a ser ejecutados es la posibilidad más prometedora aportada por los medios programables. Por ello afirma que para que el código funcione como generador, «debe, típicamente, ser una parte distintiva del sistema textual global; debe ser posible recompilar los códigos como procedimientos operativos, como aspectos de una práctica textual de arte en vivo» (Cayley, 2002).

Sin embargo, aun en los casos de *codework* en los que el código existe como procedimiento operativo, como los poemas de Holden y Kerr, la superficie sigue sin estar exponiendo algo que forme parte de la arquitectura de los medios programables ni de la red. Una comprensión materialista de la poesía código comienza por complicar la relación entre lo textual y lo visual entendiendo que es, de partida, una forma de literatura de superficie. Lo que se lee es lo que sucede en pantalla y, por tanto, nunca deja de ser una representación efímera suspendida momentáneamente en el cristal líquido de un dispositivo. Lo que transcurre debajo es un palimpsesto de inscripciones y procesos que van traduciéndose desde el nivel eléctrico (que se reduce a circuito abierto o cerrado) (Tenen, 2017a). El hecho de que aparezca el código fuente presentado al lado de su *output* produce una sensación de inmediatez que no da cuenta de la serie de traducciones que suceden cuando un código es compilado: pasa por programas que lo compilan, es decir, lo traducen a lenguaje de máquina, para que pueda ser interpretado, siendo finalmente traducido al formato del *output* para que se pueda enfrentar a ello el humano. De todo este proceso sólo accedemos a un lenguaje que media entre aquel comprensible por humanos y el de la máquina, el principio y el final del proceso, que ya estaban predestinados a ser los únicos instantes visibles externamente. Cuando se dice que el codework revela algo o que expone el funcionamiento secreto que están inscrito en los códigos que estructuran la era digital, es sólo cierto en tanto que muestra lenguajes que no suelen ser vistos, pero no los procesos que componen la textualidad electrónica.

3. Los orígenes de la poesía código: convergencia de linajes.

Dos trayectorias principales convergen en la aparición de los primeros poemas código. Por un lado, la literatura lúdica de vanguardia ya venía anunciando y probando el uso de lenguajes artificiales para la creación literaria, especialmente el Ouvroir de Littèrature Potentielle (OuLiPo), fundado en 1960 en Francia. Por otro, desde la década de 1950, se puede observar el desarrollo de una conciencia de la programación como práctica escritural.

A lo largo de la segunda mitad del siglo XX, ya se había formado un cuerpo considerable de literatura escrita con la asistencia de ordenadores e incluso generada por ellos. En su cronología de obras, C.T. Funkhouser registra variedad de casos de «poesía digital prehistórica» (antes de la llegada de la WWW) escrita para ser leída tanto en dispositivos electrónicos como en papel (Funkhouser, 2007, pp. xix–xxiv). Entre ellas encontramos generadores de poe-

mas, programas que recombinaban otras obras para crear textos nuevos, y muchas otras piezas que hacen uso de la pantalla y su programabilidad de distintas maneras. No obstante, y como también señala Hopkins, apenas hubo entre ellos intentos de desarrollar poesía legible por humanos escrita en lenguajes de programación ya existentes (Hopkins, 1992, p. 1). Según parece, la Perl Poetry constituye el primer caso de poesía propiamente escrita en lenguajes de programación, es decir, cuyo texto sea código funcional y compile.

Cuenta Maurice Black que, a mediados del siglo pasado, la labor de la programación empezó a diferenciarse progresivamente de las matemáticas y la ingeniería para acabar comprendiéndose a sí misma como escritura, lo que fomentó que el «intercambio desinhibido y sin regulación de código» se convirtiera en un principio fundamental de la ética de su cultura comunitaria (Black, 2001, pp. 14–15). En ello había también un deseo de que fuera reconocida y admirada la belleza de la forma en la que algo estaba escrito, que fuera un código que hubiera encontrado maneras extraordinarias, elegantes y poco farragosas de cumplir su función (lo cual era parte del significado principal de la palabra «*hacker*» en sus inicios). El aspecto del código es un factor importante en las consideraciones de la programación, y los criterios de funcionalidad suelen ir de la mano de otros estéticos internos por los cuales una mejor funcionalidad se considera también más satisfactoria estéticamente. Al contrario, también es común que se comparta código ofuscado, que ha sido creativamente alterado para interrumpir su legibilidad, normalmente con el propósito de proteger propiedad intelectual.

Estos procesos derivaron en que muchos programadores empezaran a concebir su herramienta de trabajo como una creativa. Hopkins, Wall, Counterman y demás participantes en los foros y concursos en los que se compartía la Perl Poetry no eran artistas buscando nuevas técnicas, sino trabajadores del código cuya íntima relación con el material lingüístico de la programación los llevó a empujar las fronteras de su actividad escritural. Escribir poesía se presenta como consecuencia lógica e intuitiva del uso habitual del lenguaje, sea este natural o artificial. Después, el caso más reciente y desarrollado de poesía código ha sido producido por un programador (Holden) y un poeta (Kerr) que han colaborado en hacer su libro. Si en esta poesía vemos una forma vanguardista digital, no es una vanguardia que haya querido implementarse a sí misma en la esfera del mundo digital, sino una derivación lúdica del trabajo del programador que ha llegado a adquirir conciencia de forma vanguardista. La figura del hacker se actualiza en ella como análoga a la del poeta inspirado[8].

8 Se encuentra aquí una contradicción en la relación que guarda la poesía código del tipo que ocupa a este artículo—sintácticamente funcional y potencialmente ejecutable—con la vanguardia literaria. Por un lado, como han defendido otros teóricos (ver, por ejemplo, Szabolscsi, 1972), las vanguardias son movimientos con proyectos estético-políticos definidos, lo cual no se puede decir de este tipo de poesía que nace como forma lúdica entre trabajadores del código. Además, las publicaciones de poesía código reproducen modelos tradicionales de autor-obra e intentan integrarse, a través de su impresión en papel, en los circuitos de mercado literario (ver apéndice de obras). Por otro lado, es justamente la deriva de la dedicación laboral en dedicación creativa la que les acerca a los modelos de artistas que reivindicaban ciertos movimientos vanguardistas de corte socialista en los que se desmantelaba la categoría de artista para concebir todo trabajo como trabajo creativo, o, en clave benjaminiana, todo autor como productor y toda producción como creativa a través de una relación íntima con los medios de producción (Benjamin, 2015).

Escribir en cualquier lengua significa atenerse a una serie de normas y convenciones. Las lenguas naturales son moldeables, flexibles, y no dejan de lograr comunicar aun cuando esas normas son desviadas o incumplidas; permiten un margen razonable para dislate e inventiva. Por el contrario, las normas de un lenguaje de programación han de ser cumplidas con exactitud si el texto espera llevar a cabo su función, sin margen para ambigüedades o comandos inexactos.

Ya en 1962, miembros del OuLiPo habían propuesto que se escribiesen poemas utilizando lenguajes de programación[9]. En 1968, Noël Arnaud publicó *Poèmes Algol*, un libro de protopoemas-código compuestos utilizando únicamente las palabras clave del lenguaje de programación ALGOL, diseñado en 1958 y que, aunque ya no está casi en uso, es del que derivan muchos de los lenguajes más extendidos hoy en día. Esta quizá sea la relación más inmediata con el posterior fenómeno del *codework*. Varios de los escritores del OuLiPo también comparten con los programadores poetas el ser personas educadas en el campo de las ciencias deseosas de implementar principios matemáticos a la creación literaria. Siendo el OuLiPo primeramente un laboratorio (*ouvroir*, a veces traducido por «obrador»), se reconoce el mismo modo de operar que Cayley describe en referencia a sus propios poemas-código: «experimentos con lenguaje que aspiran a convertirse en algo que diga algo» (Cayley, 2002).

Para el OuLiPo, el elemento de la norma debía ser elevado a principio o, mejor dicho, a juego, imponiendo exigencias ineludibles a su propio uso del lenguaje. Algunas de las constricciones formales más famosas que los escritores del OuLiPo emplean como molde potenciador de la escritura son sencillas, como los famosos lipogramas o las sustituciones (re-escribir textos que ya existían sustituyendo cada sustantivo uno que le suceda a este cierto número de entradas de diccionario después). Otras conllevan más complicados sistemas a seguir, como los poemas eulerianos o las queninas. En tanto que es una práctica regida por las restricciones, la poesía código es de fuerte descendencia oulipiana, pero hay una diferencia entre la procedencia de las restricciones de unas y de otras: donde los escritores del OuLiPo elegían qué norma aplicar a un texto, a los poemas código se les impone un sistema complejo de reglas verbales y matemáticas con la elección del lenguaje en el que están escritos. El lenguaje elegido para un poema-código actúa como formato y molde; porta consigo una serie de especificidades que hacen que crear un poema sea todo un logro si el código ha de ser correcto sintácticamente y, además, generar un *output* que participe activamente del poema en sí.

Holden y Kerr se declaran a sí mismos descendientes de la poesía concreta aludiendo a la faceta de la visualidad del signo, la cual trabajan a conciencia (Holden y Kerr, 2023b). Cada uno de sus programas-poema está preparado visualmente según lo permite cada lenguaje: algunos ignoran los espacios entre expresiones, facilitando así que el texto sea maquetado a placer; el subrayado de sintaxis y los colores asignados a cada símbolo responden o figuran los objetos a los que hacen mención; e, incluso, en algunos escritos en lenguajes esotéricos

9 Funkhouser registra la fundación del OuLiPo como uno de los eventos en su cronología de poesía digital prehistórica (Funkhouser, 2007, p. xix).

como Piet o Ante, el código en sí es, a su vez, representación pictórica[10]. Por otro lado, varios de los *outputs* de los poemas de *./code --poetry* son animaciones compuestas de signos de puntuación y textos que evolucionan visualmente hacia el caos. Al destacar la visión como parte de la relación con el código, pretenden aludir a que la visión de lo científico, lo tecnológico del software, no deja de estar anclada culturalmente (Holden y Kerr, 2023, p. 168).

De hecho, según Philippe Bootz, escritor de poesía programada desde 1977, la poesía concreta brasileña ya producía obras con carácter de tecnotexto, tal y como lo describe Hayles. Las ubica, en particular, en la vertiente encarnada por el movimiento poema/proceso y los poemas semióticos del artista Wlademir Dias-Pino (Bootz, 2007). En Brasil, la poesía concreta surgió con una intención de crítica radical a los medios de comunicación, y quienes integraban el movimiento poema/proceso entendían su poesía como una herramienta de «humanismo funcional para las masas» (Dias-Pino, 1967). Fue una preocupación política la que llevó al movimiento poema/proceso a tener una posición casi anti-lenguaje, y a realizar obras que cuestionaran sus propias condiciones y en las que se pudiera «ver la estructura/leer el proceso», condición necesaria del tecnotexto y clara portadora de la intención de democratizar el acceso al funcionamiento oculto del lenguaje, que simboliza formas enajenantes de control.

La poesía código escrita por programadores-poetas no comparte la motivación ni las enardecidas propuestas políticas del movimiento brasileño (o, si lo hace, se hallan altamente diluidas), sino más bien, y desligada, la faceta de organización visual del código. El problema del compromiso político por parte de movimientos literarios formalistas, que es terreno de debate desde que el nacimiento de la teoría literaria como disciplina con el formalismo ruso a principios del siglo XX, se reproduce en el campo del *codework*.

4. Extrañar la herramienta

El *codework* producido por artistas como Mez Breeze, Jodi o Sondheim—en cuyos textos el código no es funcional o cuyo código «no es el texto», según Cayley—, se plantea como parte de un programa estético-político radical centrado en el uso desviado del lenguaje. Sus estrategias y creaciones tienden a estar vinculadas a cuestiones de identidad, colonialismo, género, autoría, la construcción ideológica de la tecnología y su influencia en las relaciones humanas. Mez Breeze, creadora del lenguaje mezangelle (que mezcla lenguaje informal con palabras y sintaxis de lenguajes de programación), escribe y difunde textos que hacen que «tanto los lenguajes maestros del código como del inglés, se vuelvan disfuncionales, no-ejecutables e ilegibles en gran medida», siguiendo «objetivos feministas y anticoloniales de crear estructuras de conocimiento y sentidos alternativos» (mezangelle, n.d.). En estas prácticas se actualiza lo que Cayley llama la estética de «revelar código», pues el valor utópico de la transparencia ya no está solamente asociado a la visibilidad de lenguajes que no suelen emerger a superficie, sino que también se ponen en evidencia las convenciones y códigos

10 Ver «blinds.ante» (https://code-poetry.com/blinds) y «piet bark.png» (https://code-poetry.com/bark)

sociales y políticos que están inscritos en la configuración del mundo de la programación, su sintaxis y su semántica.

Las preocupaciones de la estética de «revelar código» se pueden rastrear hasta el análisis literario materialista de los formalistas rusos. En el lenguaje prosaico o habla cotidiana, los objetos, dice Shklovski, pasan desapercibidos, identificados solamente por sus rasgos iniciales, «como dentro de un paquete; sabemos que él existe a través del lugar que ocupa, pero no vemos más que su superficie» (Shklovski en Jakobson et. al, 1970, p. 59). Prosigue afirmando que, automatizada, la vida se convierte en nada, y que el arte sirve precisamente para facilitar que vuelva a ser percibida y sea, de nuevo, vida. Por influencia de y afinidad con el estructuralismo, los formalistas rusos llevan a cabo un acercamiento a la lingüística para asemejar su análisis literario al quehacer científico, rechazando las «teorías ideológicas del arte» (Eichenbaum en Jakobson et. al, p. 25). En *Plain Text* (2019), Dennis Tenen explica que Shklovski entendía que desnaturalizar aquellos mecanismos que habían perdido su «poder evocativo» al integrarse y automatizarse en usos habituales, como las metáforas integradas en el habla cotidiana, significaba «abrirlos en canal» (Tenen, 2019, p. 61). Si los formalistas rusos ponen su atención en «los mecanismos de producción de sentido», las artificiosas reglas del juego literario, para ponerlos a disposición del público (Tenen, 2019, p. 59-61); los oulipianos, rigurosamente formalistas, realizan obras en las cuales el artificio se muestra como premisa; y la poesía código produce textos que se presentan como aperturas en canal de su propio aparataje.

Como jugadores profesionales del puzle del lenguaje, es conocida la autodefinición jocosa de los oulipianos como «ratas que construyen ellas mismas un laberinto del cual se proponen salir» (ver, por ejemplo, Bénabou, 2016, p. 9; o Leong, 2017, p. 100.). Si el formalismo ruso ya había recibido críticas por su desinterés por cuestiones sociológicas, políticas y estéticas; por la misma razón, la literatura oulipiana ha sido calificada negativamente como una dedicación a las acrobacias difíciles y vacuas. Michael Leong recoge varias de las críticas que aún se le hacen al movimiento hoy en día, como que meramente consiste en «apolíticos trucos de salón de escasa consecuencia» en espacios únicamente masculinos o que, como apunta Christian Bök, los intereses poéticos del grupo parecen banales, en tanto que «sus miembros parecen disfrutar coqueteando con las cajas de cambio de géneros literarios obsoletos (...) atrincherando su prestigio canónico» (citado en Leong, 2017, p. 102). Aun siendo estos reproches importantes y productivos, es delicado desechar en su totalidad el programa del OuLiPo y del papel de la restricción como mero juego formal inconsecuente. Según los oulipianos, la escritura bajo constricción es la manera paradójica de liberar el sistema del lenguaje al obligarlo «a salir de su funcionamiento rutinario» (Benabou, 2016, p. 13). Todo habla y escritura se somete a ciertas normas y convenciones que no siempre se conocen. Para ellos, el artefacto de la obra literaria es más libre cuando las restricciones no vienen impuestas por convención interiorizada y automatizada, sino que se eligen por voluntad propia y con intención, lo cual, además, abre las puertas a potenciales realidades otras (Morales Benito, 2021, p. 152).

La preocupación por un programa estético-político parece no ser de gran importancia en la actividad poética de los poetas-programadores. Esta ausencia es contradictoria con el hecho de que en la poesía código ejecutable se satisfacen las demandas creadas por la tradición de la estética de «revelar código», especialmente reflejada en los *codeworks* no ejecutables, de mostrar el aparataje. En la poesía código cuyo texto es código funcional y que compila, hay un truco preexistente al evento artístico que sí queda revelado al público, quienes, por medio del poema, reciben, de manera anómala y excepcional (porque se trata de código), la misma información material que quien escribe. Y esto sucede en un ámbito en el que el aparataje está más escondido que nunca.

La pantalla gestiona nuestras interacciones con el ordenador a través de metáforas que imitan al mundo en el que nos movemos diariamente y que hacen que el uso del ordenador sea más fácil de aprender y más eficiente: escritorios, carpetas, papeleras, ventanas, cortar y pegar. Esta simulación aleja al usuario profano de la comprensión de las formas reales de lo que sucede cuando realiza una acción. A nivel de almacenamiento, el vocabulario cambia a uno de control: comandos, funciones, ejecución, iniciación, etc. (Tenen, 2019, p. 24). En nuestro encuentro con los ordenadores, procesos e ideas complejas se sustituyen por acciones abreviadas. La visión de Shklovski de los objetos sólo vistos por su superficie encuentra aquí su reflejo literal.

En la poesía código, las palabras clave o palabras reservadas de un lenguaje de programación, que, en su contexto habitual, se observaban como invariables por la máquina, ahora conllevan también un sentido semántico y connotativo dentro del poema. Esta forma tan laboriosa de hacer y leer poesía procede con un *ethos* contrario a la eficiencia que esperamos del uso de una máquina. La funcionalidad de los lenguajes de programación, creados para asegurar que la productividad de la máquina sea lo más fluida posible, es desviada hacia un uso que produce muy poco pero cuyo proceso es rico en significados y afecta bidireccionalmente al humano y a la máquina. No es sólo el medio el que determina la forma que puede tomar el artefacto literario, sino que la contextualización de los signos transforma el medio en el que están inscritos y desvía el funcionamiento de esta herramienta. Explicar la metáfora o interrumpirla saca a relucir las normas que gobiernan un medio, por ello insiste Tenen en que una poética materialista de la computación es necesaria para poder «consentir o, al contrario, resistir los elementos de una estructura impuesta» (Tenen, 2019, p. 41).

5. Conclusiones

Al trazar algunos de los linajes que juntan caudales en la poesía código, se ha procurado integrar su curso en la historia de la poesía vanguardista y la literatura lúdica del siglo XX. Faltaría, por otro lado, profundizar en la narrativa que da cuenta de cómo las hibridaciones entre la computación, lo visual y lo literario no son tanto producto de la iniciativa del arte software sino una parte constitutiva de las historias tanto de la literatura como de la computación desde el nacimiento de esta última. Maurice Black ha realizado estudios acerca de esta

trayectoria, defendiendo la construcción de visiones que entiendan las realidades materiales de la computación. Pretende así alejar la narración de la historia de la computación de los diversos relatos utópicos y distópicos edificados «como análogo teórico a la ciencia ficción» que las humanidades han formado alrededor del ciberespacio, utilizándolo como plataforma donde llevar a cabo la «resolución ideal» de sus preocupaciones políticas sin una comprensión real de la constitución de la máquina (Black, 2002, p. 16). Tenen insiste, de manera similar, en el análisis materialista de las inscripciones digitales, así como en el alfabetismo computacional, ya que la pantalla es una simulación metafórica de procesos ocultos y «las simulaciones codifican estructuras políticas que no debieran ser naturalizadas, no sea que sucumbamos a la complacencia del determinismo tecnológico» (Tenen, 2017, p. 26).

La poesía código lleva a cabo una importante labor de des(en)cubrimiento y extrañamiento de herramientas lingüísticas que gestionan tantas partes omnipresentes del mundo digitalizado. Por medio de estas, se emiten como inmediatos y visuales procesos complejos basados en el lenguaje, las matemáticas y la abstracción de la materialidad al cálculo; a través de los cuales los comandos quedan distantes de su impacto. No es banal el potencial que tiene el uso literario del código para activar espacios poéticos, y, por tanto, subversivos, de los lenguajes que organizan las arquitecturas del ciberespacio. Aun así, parece que el poder transformador que podría ejecutar esta vertiente del uso creativo de los lenguajes de programación queda aturdido o atrofiado por su por su propio aislamiento como actividad lúdica[11]. La pregunta permanece y hace eco a aquellas críticas que ya despertaba el análisis literario del formalismo ruso: ¿a servicio de qué están estas técnicas?

Poner la atención sobre los mecanismos de producción de sentido poco logra si las conclusiones no son puestas a trabajar para alterar el mismo medio en el que existen. Ello requeriría que fuesen reintegradas en el panorama social, político y estético del cual fueron extraídas y extrañadas para su descontaminada observación. En el caso de la «literaturización[12]» de la programación, el potencial reside en su posterior reinserción en las arquitecturas algorítmicas de la red, donde podríamos imaginar la poesía ejecutable como herramienta de transformación de los sistemas textuales que componen la esfera digital. Si ahora se recuerdan los argumentos de Cayley sobre la diferencia entre el *codework* en el cual el código es el texto (que compila) y aquellos en los que no, se puede añadir una hipotética distinción entre el *codework* aislado como obra artística que comienza y acaba en sí misma con aquella que entra a formar parte de otras estructuras. No se puede decir que el trabajo creativo con código afecte al funcionamiento del sistema textual global, como da a entender la exigencia de Cayley, a menos que pase a estar en circulación.

11 Esta problemática también pertenece a debates más amplios sobre el doble filo de la actividad lúdica como espacio, por un lado, con potencial subversivo al ser uno que no responde a las normas habituales; y por otro, limitado por ser una actividad aislada del transcurso de la vida cotidiana.

12 Eichenbaum define «literaturidad» como «lo que hace de una obra dada una obra literaria» (Eichenbaum en Jakobson et. Al, 1970, p. 26).

A diferencia de mucha literatura electrónica que suele quedarse en circuitos cerrados de producción y recepción dentro de los espacios académicos que la estudian bajo riesgo de que la investigación acabe generando sus propios objetos de análisis (Cramer, 2016, p. 18), las prácticas en las que el código fuente existe en superficie se disfrutan en otros contextos. Donde más vida adquiere parece ser en el ámbito de la performance audiovisual. Ya existen eventos de poesía SLAM código, así como la práctica extendida del *live coding* (programación de música en vivo), donde hay actuaciones integran la escritura poética en el código con el que programan la música, que aparece proyectado para el público. También ha habido casos, algunos famosos como el virus ILOVEYOU, en los que se han escrito programas víricos con cierta actitud poética que sí han circulado en red. Queda para otra ocasión la consideración de estas vertientes de la poesía código en las cuales activa su potencial performativo tanto en vivo como aquellos en los que se hace partícipe de la vida textual de las redes (humanas, eléctricas, cibernéticas). Cabe pensar que es en esos casos donde esta forma de arte-lenguaje puede encontrar públicos más amplios y receptivos, así como reincorporar el carácter crítico que otros *codeworks* han podido mantener para con los modelos hegemónicos recurrentes de autor y obra.

6. Apéndice de publicaciones

Como parte de esta aproximación a la investigación en poesía código, incluyo un pequeño apéndice de obras de este subgénero de la poesía digital publicadas en los últimos quince años con doble intención: por un lado, darlas a conocer; por otro, mostrar que, salvo el libro y la web de Holden y Kerr que ha sido mi principal referente aquí, casi todas se represan en el hecho de estar escritas en lenguajes de programación sin intentos de poner en juego las posibilidades visuales, cinéticas y de ejecutabilidad aportadas por el código. La mayoría son poesía ejecutable pero que no produce un output. De las que sí lo producen, rara vez se publica el *output* junto al poema. Finalmente, casi todas existen como publicaciones estáticas en papel o formato PDF y no en sus entornos nativos en los que se podría activar su potencial funcionalidad.

- *code {poems}* (2012), editado y publicado por Ishac Bertran; re-publicado por Impremta Badia (2018). Antología de poemas código.
- Tres fascículos de *Code Poetry* que produjo Shawn Lawson con las producciones de poesía código del alumnado de su asignatura Art & Code & Interactivity en el Rensselaer Polytechnic Institute. (2016-2017) [https://www.shawnlawson.com/portfolio/code-poetry-book/]
- «amazon» (2019): poema código de Eugenio Tisselli publicado en su página web [https://www.motorhueso.net/wuwei/amazon/]
- code::art journal (2019-) editado por Sy Brand [https://code-art.xyz/]
- *Code Poems: 2010-2019* (2020) colección de poemas de Francisco Apirile publicada en papel por Post-Asemic Press.

– Algunos poemas código Belén García Nieto se pueden encontrar online publicados en entrevistas o en el artículo que le dedica Encarna Alonso Valero, así como en su poemario *A 6000 metros de profundidad* (2023).
– *.code/ --poetry*, libro publicado online (https://code-poetry.com/) y en papel por broken sleep books (2023)

(Varias de estas publicaciones también las citan Holden y Kerr como inspiración para su proyecto.)

Bibliografía

Alonso Valero, E. (2019). Aproximación a la poesía escrita en lenguajes de programación (sobre Belén García Nieto). *Signa: Revista de la Asociación Española de Semiótica, 28*, 331-349. https://doi.org/10.5944/signa.vol28.2019.25055

Apirile, F. (2020). *Code Poems: 2010-2019*. Post-Asemic Press.

Arnaud, N. (1968). *Poèmes Algol*. Temps meles.

Bénabou, M. (2016). Introducción. Cincuenta siglos del OuLiPo, más seis. En *OuLiPo. Ejercicios de literatura potencial* (pp. 9-21). Caja Negra.

Benjamin, W. (2015). *El autor como productor*. Casimiro.

Bertran, I. (Ed.). (2012). *Code {poems}*. Ishac Bertran. https://monoskop.org/File:Code_Poems_2012.pdf

Black, M. J. (2002). *The Art of Code*. University of Pennsylvania.

Bootz, P. (2006). Digital Poetry: From Cybertext to Programmed Forms. *Leonardo Electronic Almanac, 14, n. 5-6*(New Media Poetry and Poetic Special Issue). https://elmcip.net/node/2176

Cayley, J. (2002). The Code is not the Text (Unless It Is the Text). *electronic book review*. https://electronicbookreview.com/essay/the-code-is-not-the-text-unless-it-is-the-text/

Cayley, J. (2018). *Grammalepsy: Essays on Digital Language Art*. Bloomsbury.

Chun, W. (2008). On "Sourcery," or Code as Fetish. *Configurations, 16*(3), 299-324. https://doi.org/10.1353/con.0.0064

Cramer, F. (2016). Post-digital literary studies. *Estudos Literários Digitais 1*, 4(1).

Funkhouser, C. T. (2007). *Prehistoric Digital Poetry: An Archaeology of forms, 1959-1995*. The University of Alabama Press.

Galloway, A. R. (2006). Language Wants To Be Overlooked: On Software and Ideology. *Journal of Visual Culture, 5*(3), 315-331.

Glazier, L. P. (2002). *Digital Poetics: The Making of E-Poetries*. University of Alabama Press.

Hayles, K. (2002). *Writing Machines*. MIT Press.

Hayles, K. (2008). *Electronic Literature: New Horizons for the Literary*. University of Notre Dame Press.

Holden, D., & Kerr, C. (2023a). *./code—Poetry*. Broken Sleep Books and code-poetry.com/home.

Holden, D., & Kerr, C. (2023b). Optimizing Code for Performance: Reading ./code—Poetry. En *Poetry and Contemporary Visual Culture / Lyrik und zeitgenössische Visuelle Kultur* (Vol. 3, pp. 167-184). de Gruyter.

Hopkins, S. (1992). *Camels and Needles: Computer Poetry Meets the Perl Programming Language*. Usenix Winter 1992 Technical Conference. https://www.are.na/block/25733863

Jakobson, R., Tinianov, Eichenbaum, Brik, Shklovski, Vinogradov, Tomashevski, & Propp. (1970). *Teoría de la literatura de los formalistas rusos* (T. Tzvetan, Ed.; A. M. Nethol, Trad.). Siglo XXI.

Leong, M. (2017). Oulipo, Foulipo, Noulipo: The Gendered Politics of Literary Constraints. En *The Oulipo* (pp. 100–130). Verbivoracious Press.

Mezangelle, Mez Breeze (1994–presente). (s. f.). En *Net Art Anthology*. Rhizome. https://anthology.rhizome.org/mez-breeze

Miranda, A. (s. f.). Manifesto Proposição-67. *Antonio Miranda*. http://www.antoniomiranda.com.br/poesia_visual/manifesto_proposicao.html

Morales Benito, L. (2021). *El arte del funámbulo. Juego, 'Patafísica y OuLiPo, aproximaciones teóricas y equilibrismos literarios*. Universidad Iberoamericana de Puebla.

Raley, R. (2002). Interferences: [Net.Writing] and the Practice of Codework. *electronic book review*. https://electronicbookreview.com/essay/interferences-net-writing-and-the-practice-of-codework/

Raley, R. (2006). Code.surface || Code.depth. *Dichtung Digital. Journal für Kunst und Kultur digitaler Medien, 36*(8), 1–24. http://dx.doi.org/10.25969/mediarep/17695

Szabolscsi, M. (1972). La «vanguardia» literaria y artística como fenómeno internacional. *Casa de las Américas, 74*, 4–17.

Tenen, D. (2017). *Plain Text: The Poetics of Computation*. Stanford University Press.

Tenen, D. Y. (2017). Laminate Text: The Strata of Digital Inscription. *Amodern, 7*. https://amodern.net/article/laminate-text/

Tomasula, S. (2012). Code Poetry and New-Media Literature. En *The Routledge Companion to Experimental Literature* (pp. 483–496). Routledge.

Turner, F. (2006). *From Counterculture to Cyberculture: Stewart Brand, the Whole Earth Network, and the Rise of Digital Utopianism*. University of Chicago Press.