Poetics in Action. Time and Code in the Poetry of John Cayley and the Poets of *Rozdzielczość Chleba*

Mariusz Pisarski

Appeals for an expanded poetics, heard in the past decade in discussions of contemporary textuality, always promising a renaissance for poetics and its preoccupations, are quick to find a sympathetic response. Who among literary scholars, a privileged crew in the 20th century due to the position of language and textual problems at the centre of cultural studies, would not wish to hold onto that strong hand in the 21st, after the performative[1] and digital "turns"?[2] Should we not ask, however, which poetics we are expanding? For Barrett Watten, poetics is a writing genre;[3] in the Polish and European contexts, it is treated as a discipline. Between these two poles there is a middle way, whose adepts find poetics in each method and every possible feature of language adaptable for rhetorical or aesthetic purposes. In this article, I will argue that we should also consider the act of programming a literary work for aesthetic, cognitive and meta-reflexive purposes to be a form of poetics.

[1] E. Domańska, "Zwrot performatywny we współczesnej humanistyce" (The Performative Turn in the Contemporary Humanities), *Teksty Drugie* 2007, no. 5, 48-61.

[2] M. Meryl, "F5: Odświeżanie filologii" (F5: The Renewal of Philology), *Teksty Drugie* 2014, no. 2, 9-20.

[3] B. Watten, "Poetics in the Expanded Field: Textual, Visual, Digital . . .," in: *New Media Poetics. Contexts, Technotexts and Theories*, ed. A. Morris, T. Swiss, Cambridge: MIT Press, 2006.

From the perspective of the theory and practice of electronic literature, disciplines which developed as cultural consequences of the computer and internet revolution,[4] the first point of view – of poetics as a kind of writing treating as its object the methods and means of artistic production – does not require expansion or amplification, since it is doing well. Almost every utterance in the work "born digital" is marked by self-reflexivity. A revealing allegory for that self-reflexivity is the popular message "Hello World!" at the beginning of lessons in programming languages. That joyous greeting by the layman entering the path of IT initiation is – at the same time – an invitation into the double ontology of digital text being text and code simultaneously, being both written and programmed. Traces of self-reflexivity and metatextuality featured in the classic hypertext novels of Michael Joyce, Shelley Jackson, and Stuart Moulthrop. When e-literature began to break free of the closed publishing framework and conquer the Internet, that tendency only grew stronger, as proven by Mark Amerika's *Grammaron* and *Hypertextual Consciousness* (1999) or Talan Memmot's *Lexia from Perplexia* (2002).

But while poetics as a genre is thriving in the field of e-literature and claiming its own historical analysis (which would examine, for example, the reasons for generic transitions over recent decades from hypertextual prose through e-poetry to game applications with animation and film elements), poetics as a discipline, if it is seriously going to encompass literary practice in the programming medium in its interests, must not only expand its horizon to include those developments, but also deepen its methods to include a "heremeneutics of interactivity" and a "hermeneutics of code."[5]

It seems that the latter hermeneutics, that is, the close reading, analysis and interpretation of the code substratum of "techsts," has been most neglected. Despite the fact that 10 years have passed since the publication of the book *New Media Poetics*,[6] in which important literary scholars and comparativists, as well as practitioners and theorists of e-literature, built the foundations for a poetics of hybrid, expanded forms, that expansion, potentially deeply transformative, for poetics itself remains a zone into which critics prefer not to venture, no doubt for fear of overstepping their interdisciplinary credentials.[7] So the voices of scholars with credentials in poetry, literary scholarship and computer programming are highly valued in this context. "Programmatology" (John Cayley), "expressive processing" (Nick Montfort, Noah Wardrip-Fruin), "digital text archaeology" (Matthew Kirschenbaum), "critical code studies" (Mark Marino) and

[4] On electronic literature as an artistic and theoretical field, see, among others, K. N. Hayles, "Electronic Literature: What Is It?" at https://eliterature.org/pad/elp.html; Scott Rettberg, "Communitizng Electronic Literature," http://www.digitalhumanities.org/dhq/vol/3/2/000046/000046.html (last accessed 30.03.2016).

[5] R. Simanowski, *Interfictions: von Schreiben im Netz,* Frankfurt: Suhrkamp, 2002, 121, 139.

[6] In Poland, there has also been considerable discussion over the last decade of the need to expand poetics to include media and digital literary forms; contributions to the discussion include the work of Ewa Szczęsna, Seweryna Wysłouch, Urszula Pawlicka, Piotr Marecki, Monika Górska-Olesińska, Piotr Kubiński, and the author of the present work.

[7] In 2011, when the publisher Ha!art issued Michael Joyce's hypertext fiction work "Afternoon, a story" (I produced the Polish version) a respected Polish weekly sought to review it. When we sent the CD-rom to their culture department, an editor called us to request a PDF copy, so that he could read the novel in a manner better-suited to reviewing, i.e., traditional, non-fragmented, and non-interactive.

"platform studies" (Nick Montfort, Ian Bogost)[8] are areas in the theory and practice of literature of new media that go beyond metaphors of codec bandwith. In light of thorough analyses of the influence of code on the semantics, rhetoric and poetics of digital forms, the loud "code works" of net.artu, often merely non-interactive mixtures of code and natural language, at the visual and verbal level bespeak a superficial fascination with secondary languages which is nothing new in culture. In 2016, digital poetry created according to that tired recipe has no justification for its existence; at the same time, the potential for creative manipulation of the text through code remains unexplored.

I therefore propose to return to basic, previously existing distinctions, but base them on newer examples, including some Polish ones. The proposal seems all the more apt since the poetic and programming practice of the creator of my typology has effectively strengthened it in successive years. John Cayley, a Canadian poet, lecturer at Brown University, pioneer of electronic poetry, and the author of mobile, programmable poems, was published in *New Media Poetics* alongside Warren Batten[9] with an appeal for a more complete understanding of the layer of code in digital art, expanding then-prevalent concepts of "codework" promulgated by Rita Raley and the understanding of the role of code in digital literature put forward by N. Katherine Hayles together with the category of the "flickering signifier."[10] In the first instance, reflection on code was focused, in Cayley's view, mainly on the surface of the text, influenced by code in its capacity of visual-linguistic artefact. In the second instance, the "flickering" of the "techst" relates to internal processes connected with the physical origin of the message sent, fundamentally unimportant for the reader. In neither situation does the role of code influence either the processes of meaning creation on the work's "stage" or in its event field, or make any particular contribution to digital poetics as such.

The Anatomy of Code

Cybertexts and ergodic works, those whose content and trajectory are variable, have introduced the event field into literary communication.[11] Located in between the sender and receiver, it makes a work into a kind of dramatic game, a performance, the result of which is dependent on several factors and is determined by the tentative guidelines the author puts into the program that sets its course. The result certainly does not lie in the reader's hands. He or she can, of course, modify certain aspects of the text on its interface or paratextual surface, but the *modus operandi* of contemporary digital poetic forms privileges the details

[8] These disciplines were outlined in the following works: J. Cayley, "The code is not the text (unless it is the text)," *Electronic Book Review* 2002, online: http://www.electronicbookreview.com/thread/electropoetics/literal; N. Wardrip-Fruin, *Expressive processing. Digital fictions, Computer games, and software studies*, Cambridge: MIT Press 2010; M. G. Kirschenbaum, *Mechanisms new media and the forensic imagination*, Cambridge: MIT Press 2008; M. Marino, "Critical Code Studies," *Electronic Book Review* 2006, online: http://www.electronicbookreview.com/thread/electropoetics/codology/, N. Montfort, I. Bogost, *Racing the Beam The Atari Video Computer System*, Cambridge: MIT Press 2009.

[9] J. Cayley, "Time Code Language: New Media Poetics and Programmed Signification," in: *New Media Poetics*, 307–334.

[10] See R. Raley, "Interferences: [Net.Writing] and the practice of codework," *Electronic Book Review* 2002, online. http://www.electronicbookreview.com/v3/servlet/ebr?command =view_essay&essay_id=rayleyele; N. Katherine Hayles, "Print is flat, code is deep: the importance of media-specific analysis," *Poetics Today* 2004, volume 25, 67-90.

[11] E. Aarseth, "Nonlinearity and literary theory," in: *The New Media Reader*, eds. N. Montfort and N. Wardrip-Fruin, Cambridge, Massachusetts: MIT Press 2002, 762–780.

packed into the scenario over readerly freedom of choice. In writing as *performance of writing*, which is how John Cayley understands his work, there is no place for the latter option. The reader as an active participant can initiate codework, prompt its direction or enter the material to be worked on, but the freedom to seriously transform the material does not feature here. This is especially true in the case of works whose result cannot be predicted even by the author, since they involve the use of either randomly generated algorithms or externally-based scripts.[12]

1. Code as a Language

The first kind of "code writing" that Cayley discusses is code as seen through the eyes of professionals. In this view, it appears as a special kind of language that can be looked at, read and interpreted. In the case of Leszek Onak's poem "bletka z balustrady" (joint from the balustrade, 2014) or Piotr Puldzian Płucienniczak's generator "Booms" (2016), the language is the script language javascript, understood and interpreted by internet browsers displaying websites that connect with java scripts. The syntax of that language is subject to strict rules, but its usage varies with each use and is submitted to evaluation by other programmers. In "Booms," the text's mechanics are controlled by four javascript files. One of them, booms.js, when opened in the editing program,, proves to be written in 28 lines. Between lines 10 and 12 the function "podstawRzeczy" (basiThing) is defined.

```
function podstawRzeczy(n) {
 $(".v1").html(rzeczy[n][0]);
 $(".v3").html(rzeczy[n][1]);
```

["Rzecz" meaning "thing" and "podstaw" being the root or genitive plural of "podstawa," meaning "base, fundament."] As the built-in error message tells us, in line 11 the dollar sign is used without prior definition. A programmer reviewing code or tasked with testing it must draw certain practical conclusions from such a message, but to a literary scholar, the code read on the page like this does not have any particular value. The reception of code as language does not tell us anything about the poetics or rhetoric of a work, though it can confirm observations made at the surface level of the text. For example, the name of the function "podstawRzeczy" is a programming proof of the work's variability.

2. Code as Language Modulator (Language Operates, Not Code)

If, in the first version, we read code under the surface of the text as the material out of which it is constructed, in the second part, code as language is drawn outward to enter into semantic relationships with natural language. Cayley adds that code in this case ceases to function (as code) in order to become part of a linguistic message. In the example of Leszek Onak and Łukasz Podgórny's poem "<h1>Depresja</h1>" (<h1>Depression</h1>) from the book *wgraa* (instaall, 2012):

[12]I will illustrate the five-degree stratification of layers of code presented and the ways it has been understood by critics using works by Leszek Onak, Piotr Puldzian Płucienniczak, and Łukasz Podgórny from the Kraków poetry group *Rozdzielczość Chleba*. I will then examine poetic programs from John Cayley and Daniel C. Howe's series *Readers' Project*.

```
<h1>depresja</h1>
<?php
include(“personal.conf”);
$tresc_zapytania = “SELECT gęstość FROM spacja
                    WHERE ciasteczka = ‘zablokowane’”;
$zapytanie = mysql_query($tresc_zapytania);
while ($row = mysql_fetch_assoc($zapytanie)) {
$unique = $row[‘gęstość’];
$wynik = str_replace(““,”<br>”, $unique);
echo $wynik;
?>
```

Illustration no. 1. Poem *<h1>depresja</h1>* by Leszek Onak and Łukasz Podgórny

The syntax of the php language, used to build dynamic web pages in their interaction with servers and databases, is broken up in this poem and mixed with Polish so as to build semantic tension and encourage the reader to look for new contexts for both orders. The strategy applied by Podgórny and Onak treats code as an idiom, as a foreign linguistic element, which, when it encrusts a poem, is joined to the poetic object, putting into question the previous understanding of what poeticism means and expanding the horizons of verbal art to encompass new, unexplored territory. This is the most normal thing in the world, according to Cayley. And in fact, what is the difference between including in a poem elements of local dialect or folklore, as the Romantics did, or elements of new media (film, radio), as the Futurists did, and enhancing poetry with elements of code? The problem is that code in this case does not function as code, since when woven into the fabric of language it has lost its operational, performative power. A computer will not read it, and its event field is static and constricted to the visual/linguistic surface. That is why Cayley looks disapprovingly on the code works of Australian net.art writer Mez Breeze,[13] despite the recognition she has obtained from critics of e-literature.

[13]Mez Breeze's work, especially her “mezagnegele” technique of mixing code language and natural language, was praised in superlative terms by critics including K. Hayles, Florian Krammer, R. Raley. See e.g. N. K. Hayles, “Deeper Into the Machine: Learning to Speak Digital,” *Computers and Composition* 2002, no. 19, 4, 371-386.

From a point of view that takes into account the contexts of the subcodes cited, Cayley's view can be challenged. Code is not merely an ornament or element in the visual play of concrete poetry, but, like folklore in the case of the Romantics, which brought with it clear prosodic and rhythmic preferences, constituting a distinguishing feature of Romantic poetics, or like film montage and photographic collage, which provided direct inspiration for the techniques of the pre-war avant-garde, a linguistic subcode relating to and used for reading a computer program has rhetorical and stylistic potential. Part of Onak and Podgórny's most important poem "inherits" its length from the length of a single php declaration and fits directly into that declaration's framework, which is delimited by the opening and closing marks of a programming function („<?php ?>). If a book were based on a similar principle, we could talk about active participation by the conventions of code in the creation of a single text, inasmuch as it here becomes the matrix of poetic convention.

3. Code as Readable and Performative Text (Code Functions, Not Language)

The title of Onak and Podgórny's poem "<h1>depresja</h1>" is simultaneously the name of an html language—a functioning one! If we enter it into an html file, our web browser will display the word "depresja" (depression) in large bold letters. The manoeuvre is not only activated by code, but is itself a particle (the simplest possible one) of active code. This kind of presence of code in a literary work is placed by Cayley into a third category, less popular than the preceding one and represented by, for example, perl poetry – i.e., poetry written using short commands in the perl language, which, since English is the constructive material of programming subcodes, produces poems that can be performed on the computer as well as read. The latter activity, however, confronts the reader with some transgressions against grammar, sentence structure, and segment coherence.

4. Code as Coding

Reading code with an emphasis on its transformative potential in the sphere of the digital text's material substratum belongs to the fourth category and is represented in Katherine N. Hayles's understanding of code, among others. Code here performs the fundamental work of transforming signals from one system of signification into another, beginning with the physical sequence of breaks in the passage of current and ending with the codification of the alphabet of the language of communication in such a way that appropriate diacritical marks are displayed on the screen. Code thus grasped points us toward the ontology of the digital utterance, but simultaneously, like code as language, is situated in a register not essential for the reader. We might call this register the bandwidth of a new digital triviality, referring to Espen Aarseth's old distinction between trivial and non-trivial reading procedures, where in reading a book, turning the pages is a trivial procedure.[14]

5. Code as Programming

None of the views of code mentioned above raises it to the rank of a truly active causative subject, affecting the act of communication itself (displaying text, interaction) or the invocation

[14]Cayley writes: "Although we can be made aware that the codes of digital media make the words we read on screen flicker beneath it, we do not really care – for the purposes of interpretation – whether the text we read is encoded as extended ASCII or Unicode." J. Cayley, "Time Code Language," 314.

of sociocultural contexts (code as invoked convention or the convention of invoking). That is only accomplished by a fifth view, presenting code as primarily programming, a set of methods that play out in time and create a text in keeping with rules defined "at the outset" by the author, and which allow the reader to observe the results of such programming as poetic effects. It is through these aspects of Cayley's work, and also that of the poets of *Rozdzielczość Chleba*, that computer code becomes a tool of active engagement with the text, a motor of exploration of the boundaries of the utterance in a new medium, in other words – poetics in action.

In Piotr Puldzian Płucienniczak's "Booms," the role of code is relatively simple: at regular time intervals, it displays new content in the poem's lines, based on an alternation of words taken from designated word groups in a set poem-sentence structure. Code also makes the title display the number of uses the generator has undergone in a given reading session.

boom #2037
grałem w minecrafta, kiedy pierwszy samobójca
wysadzał się przed meczetem w chanakin.
grałem w minecrafta, kiedy drugi samobójca
wysadzał się przed meczetem w chanakin.
budowałem portal, żeby wejść do piekła.

(i was playing minecraft, when the first suicide bomber / blew himself up in front of the mosque in khanaqin. / i was playing minecraft, when the second suicide bomber / blew himself up in front of the mosque in khanaqin. / i was building a portal so i could enter hell.)

Płucienniczak's poem plays out in time. The stage of the text, as Cayley calls it, is not complex, and code performs its function according to a simple scenario. Nonetheless, it is code that introduces into the poem its crucial temporal aspect, the poetic result of which cannot (effectively) be reproduced in the paper version. The poem needs to be read as a series. In the form of a single-use copy, printed in a book or sent as an e-mail attachment, "Booms" becomes an ordinary poem criticizing the consumerism of the "digital lifestyle" in the face of global terrorism. The lyrical persona appears to be an individual unable, unwilling, and powerless to prevent the situations described in the background. The temporalization permitted by code, however, radically changes the accent of the whole poem. After 10 minutes spent in front of the screen, the reader observes that there is truly no end to the possible variants of the poem being generated. What is more, each of them is counted, and that number appears in the title. After an hour or two, the bombs in the poem are still exploding. Whereas the chips-munching, game-loving narrator shifts to a myriad of other banal activities, the titular number begins to reach dizzying heights. Its specific weight rises above some undefined critical mass, perhaps different for each individual reader, and begins to tip the scale of the whole poem from nihilism toward a cry of despair and empathy. The reader's only escape, paradoxically, is to close the browser window, so as to stop seeing this grimly rising number which translates, in the real, extratextual world, into the number of bomb victims.

Leszek Onak's "Sonet niezachodzący" (Never Setting Sonnet) uses temporality with similar force and on a scale transcending its particular individual author. Written in php, the poem's script searches the headlines of Polish news sites and then arranges them into the fourteen lines of a sonnet. When the reader clicks on “Generate New Sonnet,” the program re-sends its query and the poem becomes updated with new content in its lines and title.[15] The results of the code work are not only visible at the local level, however. If Płucienniczak’s poem cited above can be imagined in traditional print form, as a multi-volume printout of all possible combinations that the generator can produce, Onak’s work could not possibly be subjected to any such kind of obscurantist retromediation. The time that elapses on the stage of the text becomes even more closely fused with real time as experienced by the reader. The poem changes with every minute that passes, depending on what information the news portals are relaying.

Fundamental to Onak’s poem is the way his program engages with other external programs, inviting them to a kind of “live” online collaboration. The subjects participating in this active collaboration are not actual people, however, but computers connected through the web and the programs operating in them, whose task is to supply content to RSS information channels. Our experience of this sonnet is thus imbued with causative forces of a decidedly post-human character. The author’s role is reduced to that of planner and curator of the event field. Should the RSS technology fail, or the selected portals that provide the phrases and sentences used cease to exist, the indefatigability of the sonnet’s re-production will either be suspended indefinitely or will acquire another layer of meaning: this kind of poem and this kind of code require constant care from the poet, who becomes like a gardener looking after his plants.

Perigram as Generative, Internet, and Post-Human Text

As Watten points out, poetics feeds off of experimental, radical texts. As a practitioner (a publisher and producer) of electronic literature, I have not encountered a more complex and radical work in recent years than the series of poetic programs prepared by John Cayley and Daniel C. Howe that constitute the cycle *The Readers Project* (2010–2016).[16] Individual installments of this long-term project contain the whole gamut of complexity that code in its proper function brings to poetry and poetics: the programmable function of transforming a poem’s content, style, rhetoric, and context.

The word “Readers” in the title of the project is a bit misleading. It refers not to those who read, nor to tools for reading (conveyors of texts), but to particular algorithms that carefully scan the source text and generate a secondary text derived from it.[17] The rules that govern this robotic reading are based on the cellular automaton *The Game of Life* devised by the British mathematician John Conway,[18] with the difference that in Howe and Cayley’s “game of reading,” the role of cells is taken over by words, and the game plays out not in an infinite orthogonal grid, but on the surface of a virtual page of a book, and thus an area conditioned by the

[15]Urszula Pawlicka highlights the work’s temporal aspect thus: “we should observe that any particular generated sonnet at the cited link immediately retreats into obsolescence, since each time the link directs us to a new work, basing it on the latest news […].” U. Pawlicka, *Polska poezja cybernetyczna* (Polish Cybernetic Poetry), Kraków 2012, 115.

[16]See *The Readers Project* online: http://thereadersproject.org.

[17]For a broader discussion of the project by both authors, see: J. Cayley, D. C Howe, “*The Readers Project*: Procedural Agents and Literary Vectors,” *Leonardo* 2011, vol. 1, 43, 317–324.

[18]See for example, P. Coveney and R. Highfield, *Frontiers of Complexity: The Search for Order in a Chaotic World*, New York: Fawcett Columbine, 1996, 94-96.

conventions of reading. [19] The space of the matrix is filled by a source text. The programmed reader runs through the text, activates a "live" verbal cell, and leaves behind it a "dead" one. As in *The Game of Life*, where an active cell has its neighbours, the active area of a verbal cell, its typographic vicinity, potentially consists of eight surrounding words, or a smaller number if the words at the edges (directions: N-E, S-E, N-W, S-W) do not graphically infect the active word in the middle. The reader that Cayley and Howe use to demonstrate their system is called a **perigram**. It runs from left to right, but its reading is not entirely linear, since the program adopts as its goal not a neighbouring word from the same line, but the word in the upper or lower right corner. It thus moves forward but can switch course on the up-down axis and within an area extending to about 20 words. The rules of movement for the perigram are defined as follows:

> As the Perigram Reader moves through a text, it remembers each previously read word and checks its NE and SE neighbors as potential next words. If it finds that a combination of these three words (previous, current, and potential next, in order) constitutes a phrase with a frequency above a certain threshold (i.e., it has been used previously in natural language to some extent) then its reading path may diverge, effectively also generating an alternative text that is, as it were, perigrammatic […].[20]

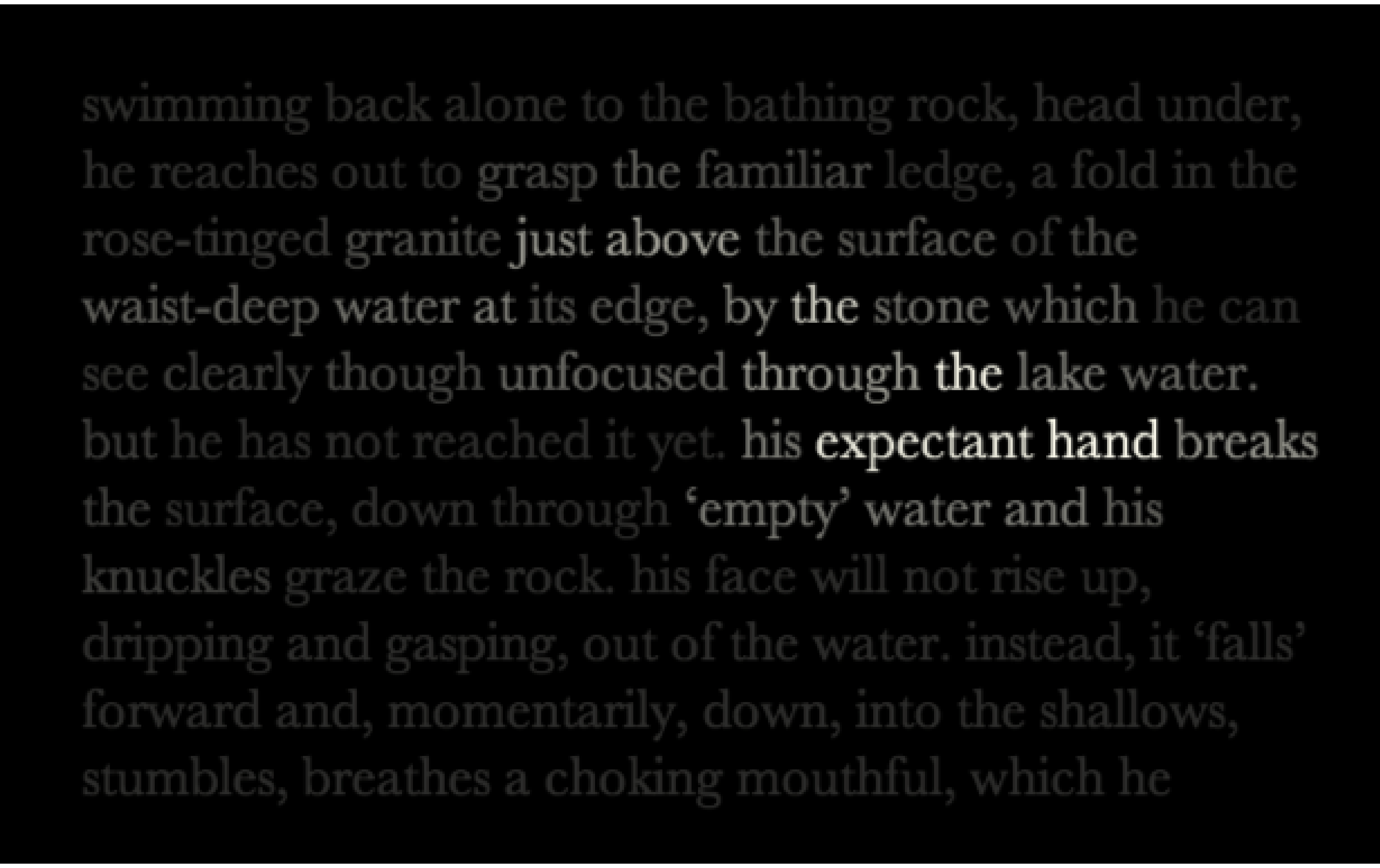
The system of the text's reading/generation becomes increasingly interesting. In fact the perigram's code not only controls its movement over the surface of the source text (which is sometimes a text by Samuel Beckett and sometimes poetic prose by Cayley). It also sends inquiries to internet browsers with a query whose contents consist of a potential phrase assembled "on the wing" by the perigram. If it generates a long list of search results, the perigram chooses an alternate phrase with a lower frequency on the lists of search results at Google, Bing and Yahoo. Cayley and Howe, or more precisely, the Rita program designed by Howe, thus check the poetic originality of verbal sequences to be displayed by the perigram. This test of originality takes place within the largest possible global storehouse, represented by the contents of the Internet and the inquiries of browser users inscribed in the search window.[21] The "commons," as Cayley calls it, is a gigantic, dynamic dictionary, increasing its resources minute by minute and hour by hour. As a result, the perigram can change its flow of reading/writing from day to day, as documented by the authors during exhibitions at which perigrams were able to generate different text even when the phrase initiating their work (written by visitors to the exhibition, upon request) was the same and the source text remained unchanged. This happened, not because the English-language internet's millions of users suddenly began using rare, poetic combinations of words, but because of the Google algorithm's auto-correct function, which in the course of

[19]Cayley and Howe remind us of the arbitrary nature of such a choice, and its connection with prevailing convention in Europe and the US.

[20]J. Cayley and D. C. Howe, "The Readers Project: Procedural Agents and Literary Vectors," 319–320.

[21]The use of the word prompts of the type that Google and other search engines present to users "on the fly" while they are typing in search forms the basis of another side-project of The Readers Project: the conceptual book *How It Is In Common Tongues*, which involves the rewriting of the content of Beckett's novel How It Is by means of (manually) filtering it through the reserves available to internet search engines. Cayley and Howe would write in 3-4 successive words from Beckett into the browser and choose the results with the longest series of words not found in a quotation from Beckett's text on the internet but from original utterances found on websites. Next, each phrase found this way in the commons was placed in the new book with a footnote giving the source of the cited passage. See J. Cayley and D.C Howe, *How It Is in Common Tongues (The Readers Project: Common Tongues)*, Providence 2012.

a repeat run through the same text encountered its own phrase as a search result (a phrase Cayley had planted in the internet, and which had already been indexed by search robots!).



swimming back alone to the bathing rock, head under, he reaches out to grasp the familiar ledge, a fold in the rose-tinged granite just above the surface of the waist-deep water at its edge, by the stone which he can see clearly though unfocused through the lake water. but he has not reached it yet. his expectant hand breaks the surface, down through 'empty' water and his knuckles graze the rock. his face will not rise up, dripping and gasping, out of the water. instead, it 'falls' forward and, momentarily, down, into the shallows, stumbles, breathes a choking mouthful, which he

Illustration no. 2. Perigram reader in action

The question then arises, who is the author of the poem that takes shape before our eyes? Is it the poet, the programmer, the author of the source text, the internet search engine, or the million active internet users? Each of these players has a part in producing the work. What status in the field of communication do we assign to the digital reader? The ontology of this reading robot is already subject to multiplication, since in reading the text, the program simultaneously creates it. We should observe at the same time a significant gap between the "active reader," the beloved figure of 1990s critics enthusiastic about new media, from the "active digital reader." *The Readers Project* asks another important question – about the condition of the modernist *episteme* in the digital context. Linguistic innovation and poetic rebellion in the form of writing that is, in short, taken from Google but against Google, the search for originality in the global store of English language material, is more a continuation than a negation of modernism. The texts do not even feature postmodern riffs on hackneyed chords in the style of Talan Memmot's "Lexia to Perplexia." Something is not quite right here.

Michael Joyce, the pioneer in literary hypertext, defines himself as an "ultramodernist." Jessica Pressman even speaks of a broader tendency, a whole current of "digital modernism," born on the wave of the technological electrification of the text.[22] Cayley and Howe are adding their own chapter to that movement, simultaneously situating themselves in the avant-garde of contemporary scholarly methodology in the humanities. In *The Readers Project* and in the perigram reader itself we find elements of operations relating to *big data*, elements of data-mining and crowdsourcing – digital tools of the humanities. At the same time, however, Johna Cayley reminds us that his work on this project is nothing other than "the visualization of poetics" and "visual poetics."

The Surface and Depths of the Digital Work

In galleries and conference presentations by Cayley, the perigram reader and its reading/writing are presented as a form of palimpsest, where the visually harmonized matrix of the source text is the field of events activated by the perigram. In some presentations, the work takes on the form of an attractive, dynamic acrostic. Behind the aesthetically impressive, mobile map of letters, words, and phrases chosen by the computer in astonishing but evidently monitored sequences of words[23] is hidden a highly organized cybertext. Poetics expanded to the extent of merely including visual, film, or sound elements, or even causative activity from the reader, would not encompass it. Unless deepened to include a hermeneutics of code, here only mentioned, poetics would at best be capable of analysing works not fundamentally different from– for example– richly illustrated children's books with interactive features (such as *lift-the-flap* books) or sound (buttons, keys, the ability to produce or record sound). The programmable aspect, essential to Cayley and Howe's work, would, observed through the lens of a poetics not geared toward code, be little more than an invisible substratum of the remediation of the various orders with which the perigram at the surface level of its text engages in dialogue (animation, visuality, books), where certain features of these orders are amplified, multiplied or expanded.

The programmatology represented by Cayley and manifested in the Polish context by Rozdzielczość Chleba, a direction in poetic reflection exploring the active participation of code written by the author in the production of a work's meanings, represents a poetics of new media in a more consistent and representative form than many works hailed as groundbreaking.[24]

Perigram, Hypertext and the Future of Digital Poetics

Compared with the challenges posed to traditional poetics by Cayley and Howe's programmable reader and its interface with the commons, hypertext – at the center of critics' and authors' attention in the 1990s, when the digital revolution conquered educational centers

[22]J. Pressman, *Digital Modernism*, Oxford 2014.

[23]Since a perigram moves through a surface of about 20 words from the active word at a given moment, a clear semantic relation is preserved with the words that constitute the resulting phrases in the derivative text.

[24]Talan Memmott's "Lexia to Perplexia," discussed by Barret Watten in the book *New Media Poetics*, is a parody of hypertext and cyberculture discourse, typical for the period in which it was written (2002), when authors of second-generation digital literature took a critical stance toward the authors of the first generation. Its status as a breakthrough work is accorded mainly for historical reasons. For new media poetics, however, John Cayley's work has had much greater significance.

in the rich countries of the West, appears a fairly conventional form of writing, maintaining as it does strong links to the book paradigm.[25] From today's perspective, we must say that as a form of text that branches out and functions on command, hypertext constituted not so much a break with as a remediation of print. The stories of Michael Joyce (*afternoon, a story*; *Twilight, a symphony*), which came closest to fulfilling the demand, voiced in manifestoes, for the creation of a work that would change each time we read it – not in the sense of the text's interpretation, but in the sense of the very substance, quantity, and sequence of the narrative material's appearance on the screen in successive reading sessions – compared to the generative and web-based poetics of the perigram, looks amazingly static. Even when enhanced with a system of conditional links, that stasis, engaging as it is for the reader, is still deprived of significant programming, which puts hypertext in the same group as the earlier-mentioned hybrid, interactive children's books, since its text remains fixed and final in its definition and is not in a position to expand or contract. The strategies of critical reflection hitherto applied to digital works, formulated in the Polish context in frequent attempts to outline the poetics of new media, are somewhat inadequate and must be expanded and deepened.

The deepening process must also concern the code and web aspects of the text. For if we agree that the electronic text is formed from several equally important and interconnected layers (material, code, text, and operation), then perigrams, redefine the scope of the layer of code, differentiate various segments of the material domain (hardware, platforms, distribution systems) and revolutionize the very concept of text, because textual and programmatological phenomena that are taking place on its surface originate neither from the author, nor the reader, nor from the text itself.

[25]This is the condition diagnosed by the authors of recent studies on the subject. See Alice Bell, *The Possible Worlds of Hypertext Fiction*, London 2010; J. Baetens, F. Truyen, "Hypertext revisited," *Leonardo* 2013, vol. 46, no. 5.

KEYWORDS

generative poetics

HYPERTEXT

code poetry

ABSTRACT:
The purpose of the article is to broaden readers' understanding of how code is used in digital literary forms. Although digital poetics in the Polish context seems relatively established, the code aspect of works, particularly when we consider works in which a computer program becomes an active, causative subject beyond the full control of author and reader, requires some additional clarification. Using examples from Polish electronic literature, the article recapitulates the typology of code formulated by John Cayley, a pioneer in digital poetry; next, it examines a series of works by Cayley and Howe in which programmed "readers" – supplied with a source text and linguistic resources indexed by Google – are sent on a special mission in search of poetic originality. Three main theses are formulated: programming is a new kind of poetics in action; code in temporal or internet texts attains the status of an autonomous actor, situated in between text, author, and reader, and maintaining contact with other programs on the web; hypertext as a primary paradigm of digital textuality turns out to be a transitional form, from the point of view of the practices of Cayley and Polish cybernetic poets, much closer to the print paradigm, that was originally acknowledged.

d i g i t a l s e m i o t i c s

e-literature

poetic generators

Note on the Author:
Mariusz Pisarski is a scholar and publisher of electronic literature. He is the author of the book *Xanadu. Hipertekstowe przemiany prozy* (Xanadu, Hypertext Transformations of Prose, Kraków 2013), editor of the magazine *Techsty* and the multimedia department of the publisher *Ha!art*. A translator of digital poetry and prose, he has written hypertext adaptations of literary classics (*Rękopisu znalezionego w Saragossie* [The Saragossa Manuscript], 2012; an internet adaptation of Bruno Schulz's short stories entitled *Bałwochwał* [Idolater], 2013). In 2011, he was nominated for the Ted Nelson Award by the American IT association ACM (Hypertext 2011). His doctoral thesis on hypertext (defended at UAM, under Prof. Bogusław Bakuła) received First Place in a National Cultural Center contest. He is a member of the Electronic Literature Organization and the recipient of the SAIA stipend from the Institute of World Literature of the Slovak Academy of Sciences. He is an associate of the Laboratory of Intersemiotic and Intermedia Research at the Institution of Polish Studies at the University of Warsaw. |