Poetyka w działaniu.

Czas i kod w poezji Johna Cayleya i poetów *Rozdzielczości Chleba*

Mariusz Pisarski

Apelom o poetykę poszerzoną, pojawiającym się od co najmniej dekady w polu refleksji nad współczesną tekstualnością i obiecującym odrodzenie poetyki i kręgu jej zainteresowań, nie trudno o pozytywny odzew. Któż spośród literaturoznawców, których wiek XX dzięki ustanowieniu języka i zagadnień tekstologicznych w centrum badań nad kulturą traktował z przywilejami, nie chciałby utrzymać pozycji silnego gracza w wieku XXI, po zwrocie performatywnym[1] i cyfrowym[2]? Warto jednak zapytać, jaką poetykę poszerzamy? Dla Barretta Wattena jest ona gatunkiem pisarstwa[3]; na polskim i europejskim gruncie traktuje się ją raczej jako dyscyplinę. Między tymi dwoma biegunami istnieje droga środka, której adepci odnajdują poetykę w każdej metodzie i jakiejkolwiek cesze języka, które można użyć w celach retorycznych i estetycznych. W dalszej części tego artykułu będę przekonywał, że za poetykę powinniśmy także uznać sam akt programowania utworu literackiego w celach estetycznych, poznawczych i metarefleksyjnych.

[1] E. Domańska, *Zwrot performatywny we współczesnej humanistyce*, „Teksty Drugie" 2007, nr 5, s. 48-61.

[2] M. Meryl, *F5: Odświeżanie filologii*, „Teksty Drugie" 2014, nr 2, s. 9-20.

[3] B. Watten, *Poetics in the expanded field: textual, visual, digital…*, [w:] *New media poetics. Contexts, technotexts and theories*, red. A. Morris, T. Swiss, London 2006.

Z punktu widzenia teorii i praktyki literatury elektronicznej, dyscyplin, które pojawiły się jako kulturowa konsekwencja rewolucji komputerowej i internetowej[4], perspektywa pierwsza – poetyka jako rodzaj pisarstwa traktującego za swój przedmiot metody i środki produkcji artystycznej – nie potrzebuje ani poszerzeń, ani wzmocnień, ponieważ ma się dobrze. Niemal każda wypowiedź w utworze „rdzennie cyfrowym"[5] naznaczona jest autorefleksyjnością. Wymowną alegorią tej samozwrotności jest popularny komunikat „Hello World!" inaugurujący naukę języków programowania. To radosne zawołanie laika, który wchodzi na ścieżkę informatycznego wtajemniczenia jest powitaniem swojej podwojonej ontologii: bycia tekstem i kodem zarazem, bycia zapisanym i zaprogramowanym. Ślady autorefleksyjności i metatekstowości nosiły wszystkie klasyczne powieści hipertekstowe Michaela Joyce'a, Shelley Jackson i Stuarta Moulthropa. Gdy e-literatura zaczęła wyrywać się z zamkniętych ram wydawniczych i podbijać Internet, tendencja się jedynie nasiliła, czego dowodem *Grammaron* i *Hipertekstualna świadomość* Marka Ameriki (1999) czy *Lexia to Perplexia* Talana Memmota (2002).

O ile zatem poetyka jako gatunek kwitnie w polu e-literatury i domaga się już własnego, historycznego opracowania (które przyjrzałoby się np. powodom przejść rodzajowych na przestrzeni ostatnich dekad od prozy hipertekstowej, poprzez e-poezję, po aplikacje grywalne z elementami animacji i filmu), o tyle poetyka jako dyscyplina, jeśli ma na serio objąć swoim zainteresowaniem praktykę literacką w medium programowalnym, musi nie tylko poszerzyć swój horyzont o te właśnie zjawiska, ale i pogłębić metody o „hermeneutykę interakcji" i „hermeneutykę kodu"[6].

Wydaje się, że ta druga hermeneutyka, a zatem *close-reading*, analiza i interpretacja kodowego podłoża „techstów", jest najbardziej zaniedbana. Mimo iż mija 10 lat od czasu publikacji tomu *New Media Poetics*[7], w którym liczący się literaturoznawcy i komparatyści, a także praktycy i teoretycy e-literatury, budowali podwaliny pod poetykę hybrydycznych, poszerzonych form, to poszerzenie o charakterze najbardziej transformatywnym dla samej poetyki pozostaje terenem, w które krytyk woli się raczej nie zapuszczać, najpewniej z obawy o zakres swych interdyscyplinarnych kompetencji[8]. Tym bardziej cenny w dyskusji jest głos badaczy, którzy owe kompetencje – poety, literaturoznawcy i programisty – łączą. Programatologia (John Cayley), „ekspresyjne przetwarzania danych" (ang. *expressive processing*: Nick Montfort, Noah Wardrip-Fruin), „cyfrowa archeologia tekstu" (Mathew

[4] O literaturze elektronicznej jako polu praktyki artystycznej i teoretycznej zob. m.in. K.N. Hayles, *Literatura elektroniczna: czym jest*, przeł. M. Pisarski, „Techsty" 2007, nr 1 (7), <http://www.techsty.art.pl/magazyn/magazyn7/literatura_elektroniczna_czym_jest_1.html> dostęp: 23.03.2016; S. Rettberg, *Budowanie wspólnoty wokół literatury elektronicznej*, „Teksty Drugie" 2015, nr 3, s. 135-155.

[5] W ten sposób, jako „rdzennie cyfrową" (ang. *born digital*), definiuje literaturę elektroniczną Noah Wardrip-Fruin, Scott Rettberg, Katherine N. Hayles. Zob. m.in. K.N. Hayles, *Literatura elektroniczna*...

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

[7] Także i w Polsce w przeciągu ostatnich 10 lat niemało mówiło się o potrzebie poszerzenia poetyki o media i literackie formy cyfrowe, czego przykładem są m.in. prace Ewy Szczęsnej, Seweryny Wysłouch, Urszuli Pawlickiej, Piotra Mareckiego, Moniki Górskiej-Olesińskiej, Piotra Kubińskiego i moje.

[8] W 2011 roku, gdy wydawnictwo Ha!art publikowało powieść hipertekstową *popołudnie, pewna historia* Michaela Joyce'a (której byłem producentem) do recenzji przymierzał się ceniony polski tygodnik. Gdy przysłaliśmy do działu kultura płytę CD-rom z plikami powieści, zadzwonił do nas redaktor z prośbą o przysłanie PDF-a, dzięki któremu mógłby przeczytać powieść w sposób bardziej przyjazny recenzowaniu: tradycyjny, całościowy i nieinteraktywny.

Kirshenbaum), „krytyczne studia na kodem" (Mark Marino) czy „studia nad platformami" (ang. *platform studies*: Nick Montfort, Ian Bogost)[9] to dziedziny praktyki i teorii literatury nowych mediów, które nie zadowalają się metaforami pasma kodowego. W świetle gruntownych analiz wpływu kodu na semantykę, retorykę i poetykę form cyfrowych głośne „działa kodowe" net.artu, sprowadzane często do nieinteraktywnych zderzeń kodu i języka naturalnego, w warstwie werbalnej i wizualnej okazują się powierzchowną, znaną w kulturze od lat, fascynacją językami pobocznymi. W roku 2016 poezja cyfrowa wykonana według takiego przepisu nie ma racji bytu, tymczasem potencjał twórczej manipulacji tekstu przez kod pozostaje niezmierzony.

Proponuję więc wrócić do podstawowych i istniejących już rozróżnień, choć oprzeć je na nowszych, także polskich przykładach. Propozycja wydaje się tym bardziej na miejscu, gdyż praktyka poetycka i programistyczna autora tej typologii skutecznie ją w ciągu następnych lat ugruntowała. John Cayley – kanadyjski poeta, wykładowca na uniwersytecie Browna, pionier poezji cybertekstowej, autor ruchomych, programowalnych wierszy wystąpił w tomie *New Media Poetics* obok Warrena Battena[10] z apelem o kompleksowe rozumienie warstwy kodu w sztuce cyfrowej, poszerzając ówczesne ustalenia co do „dzieł kodowych" (ang. *codework*) Rity Raley, jak i rozumienia roli kodu w literaturze cyfrowej proponowanego przez N. Katherine Hayles wraz z kategorią „migotliwego *signifians*" (ang. *flickering signifier*)[11]. W przypadku pierwszym refleksja nad kodem skupia się – zdaniem Cayleya – głównie na powierzchni tekstu, na którą kod jako artefakt wizualno-językowy wypływa. W przypadku drugim „migotliwość" techstu odnosi się do procesów wewnętrznych, związanych z fizycznym zapleczem wyświetlanego komunikatu, w zasadzie nieistotnym dla czytelnika. W obu sytuacjach rola kodu nie wpływa ani na procesy wytwarzania znaczeń na „scenie" dzieła, czy w jego polu zdarzeń, ani nie wnosi specjalnego wkładu w poetykę cyfrową jako taką.

Anatomia kodu

Cyberteksty i dzieła ergodyczne, a zatem takie, których treść i przebieg są zmienne, wprowadzają do komunikacji literackiej pole zdarzeń[12]. Umiejscowione między nadawcą a odbiorcą czyni ono z dzieła rodzaj dramatycznej rozgrywki, performance'u, którego wynik zależy od kilku czynników i determinowany jest przez wstępne założenia, jakie autor wprowadza do programu sterującego przebiegiem. Ów wynik nie leży bynajmniej w rękach czytelnika. Może

[9] Dyscypliny te zarysowane były m.in. w następujących pozycjach: J. Cayley, *The code is not the text (unless it is the text)*, „Electronic Book Review" 2002, <http://www.electronicbookreview.com/thread/electropoetics/literal> dostęp: 23.03.2016; N. Wardrip-Fruin, *Expressive processing. Digital fictions, computer games, and software studies*, MIT Press, Cambridge-Londyn 2010; M.G. Kirschenbaum, *Mechanisms. New media and the forensic imagination*, MIT Press, Cambridge-Londyn 2008; M. Marino, *Critical Code Studies*, „Electronic Book Review" 2006, <http://www.electronicbookreview.com/thread/electropoetics/codology/>, dostęp: 23.03.2016; N. Montfort, I. Bogost, *Racing the Beam. The Atari Video Computer System*, MIT Press, Cambridge-Londyn 2009.

[10] J. Cayley, *Time code language: new media poetics and programmed signification 307 John Cayley*, [w:] *New Media Poetics…*, s. 307-334.

[11] Zob. R. Raley, *Interferences: [Net.Writing] and the practice of codework*, „Electronic Book Review" 2002, <http://www.electronicbookreview.com/v3/servlet/ebr?command =view_essay&essay_id=rayleyele> dostęp: 23.03.2016; N.K. Hayles, *Print is flat, code is deep: the importance of media-specific analysis*, „Poetics Today" 2004, vol. 25, s. 67-90.

[12] E. Aarseth, *Nonlinearity and literary theory*, [w:] *The New Media Reader*, red. N. Montfort, N. Wardrip-Fruin, Cambridge 2002, s. 762-780.

on oczywiście modyfikować pewne aspekty tekstu na jego interfejsowej i paratekstowej powierzchni, jednak *modus operandi* współczesnych, cyfrowych form poetyckich uprzywilejowuje raczej dopięty w szczegółach scenariusz niż lekturową dowolność. W pisaniu jako „wydarzeniu pisania" (ang. *performance of writing*), a tak rozumie swoją działalność John Cayley, na tę drugą opcję nie ma miejsca. Czytelnik jako aktywny uczestnik może zainicjować pracę kodu, podpowiedzieć mu kierunek lub podstawić materiał, na którym ma ona przebiegać, nie ma jednak swobody poważnego przekształcania materiału. Dzieje się tak zwłaszcza w przypadku utworów, których wyniku nie jest w stanie przewidzieć sam autor, gdyż w grę wchodzą bądź to algorytmy losowe, bądź skrypty pochodzące z zewnątrz[13].

1. Kod jako język

Pierwszym wyszczególnionym przez Cayleya sposobem „pisania kodu" jest kod widziany oczyma profesjonalistów. Jawi się on w tym ujęciu jako specjalny rodzaj języka, który daje się oglądać, czytać i interpretować. W przypadku wiersza Leszka Onaka *bletka z balustrady* (2014) czy generatora *Booms* Piotra Puldziana Płucienniczaka (2016) chodzić będzie o język skryptowy javascript, rozumiany i interpretowany przez przeglądarki internetowe wyświetlające strony WWW, do których skrypty te są dołączane. Składnia języka poddana jest ścisłym regułom, lecz jej stosowanie różni się z realizacji na realizację i poddaje ocenie ze strony innych programistów. W *Booms* mechaniką tekstu sterują cztery pliki javascript. Jeden z nich, booms.js, po otwarciu w programie do edycji, został napisany w 28 linijkach. Między 10. a 12. definiowana jest funkcja „podstawRzeczy"

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

Jak informuje wbudowany komunikat o błędach, w linijce 11. użyto znacznika $ bez jego uprzedniego zdefiniowana. Programista recenzujący kod, lub mający za zadanie jego przetestowanie, musi z takiego komunikatu wyciągnąć pewne wnioski praktyczne, jednakże dla literaturoznawcy czytany w ten sposób kod nie ma specjalnej wartości. Odbiór kodu jako języka nie mówi nam też niczego o poetyce i retoryce dzieła, choć jest w stanie potwierdzać spostrzeżenia poczynione na powierzchni tekstu. Przykładowo, nazwa funkcji podstawRzeczy jest programistycznym dowodem na wariantywność utworu.

2. Kod jako modulator języka (kod nie działa, język działa)

Jeśli w wariancie pierwszym kod czytamy pod powierzchnią tekstu jako jego budulec, w przypadku drugim części kodu jako języka zostaje wywołana na zewnątrz, by wejść w semantyczne relacje z językiem naturalnym. Cayley dodaje, że kod w tym przypadku przestaje działać (jako kod), by stać się częścią komunikatu językowego. Przykładem wiersz *<h1>Depresja</h1>* Leszka Onaka i Łukasza Podgórniego z tomu *wgraa* (2012):

[13]Prezentowane pięciostopniowe rozwarstwienie kodu i tego, jak bywa on rozumiany przez krytyków, ilustruję utworami Leszka Onaka, Piotra Puldziana Płucienniczaka i Łukasza Podgórniego z krakowskiej grupy poetyckiej Rozdzielczość Chleba. Następnie przyjrzę się programom poetyckim z serii *Readers' Project* Johna Cayleya i Daniela C. Howe'a.

```
<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;
?>
```

Ilustracja 1. Wiersz *<h1>Depresja</h1>* Leszka Onaka i Łukasza Podgórniego

Składnia języka php, służącego do budowy dynamicznych stron WWW w ich interakcji z serwerami i bazami danych zostaje w tym wierszu rozbita i zmieszana z polszczyzną w sposób budujący semantyczne napięcie i zachęcający do poszukiwania nowych kontekstów dla obu porządków. Strategia zastosowana przez Podgórniego i Onaka traktuje kod jako idiom, jako obcy żywioł językowy, który po inkrustowaniu nim wiersza zostaje włączony w obiekt poetycki, kwestionując dotychczasowe rozumienie poetyckości i poszerzając horyzonty sztuki słowa o niezbadane jeszcze tereny. Rzecz najnormalniejsza w świecie – mówi Cayley. I rzeczywiście, czym się bowiem różni włączanie do wiersza elementów gwary i folkloru, co czynili romantycy, czy nowych mediów (filmu, radia), co czynili futuryści, od wzbogacania poezji elementami kodu? Problem w tym, że kod jako kod w tym przypadku nie działa, gdyż wpleciony w tkankę języka stracił swoją operacyjną, performatywną moc. Komputer go nie odczyta, a jego pole zdarzeń jest statyczne i ograniczone do wizualno-językowej powierzchni. Z tego powodu Cayley nieprzychylnie ustosunkowuje się do „dzieł kodowych" spod znaku australijskiej artystki net.artowej o Mez Breeze[14], mimo uznania, jakie zdobyła wśród krytyków e-literatury.

Z punktu widzenia, który bierze pod uwagę konteksty przywoływanych subkodów, można się z Cayleyem nie zgodzić. Kod nie jest tylko ornamentem i elementem gry wizualnej wiersza kon-

[14]O pracach Mez Breeze, zwłaszcza jej technice mieszania języka kodu i języka naturalnego „mezagnegele" wypowiadali się w superlatywach m.in. K. Hayles, F. Krammer, R. Raley. Zob. m.in. N.K. Hayles, *Deeper into the machine: learning to speak digital*, „Computers and Composition" 2002, nr 19, 4, s. 371-386.

kretnego, lecz podobnie jak folklor w przypadku romantyków, który wniósł z sobą wyraźne preferencje wersyfikacyjne i rytmiczne stanowiące wyróżnik romantycznej poetyki, czy podobnie jak montaż filmowy i kolaż fotograficzny, które były bezpośrednią inspiracją dla technik przedwojennej awangardy, subkod językowy odnoszący się i cytujący komputerowy program miał potencjał retoryczny i stylistyczny. Część główna wiersza Onaka i Podgórniego „odziedzicza" swą długość od długości pojedynczej deklaracji php i mieści dokładnie w jej ramach. Ich delimiterem są znaczniki otwarcia i zamknięcia programistycznej funkcji („<?php ?>"). Gdyby na podobnej zasadzie oparty był pojedynczy tomik, można by już mówić o aktywnym udziale konwencji kodowej w tworzeniu jednostkowego przekazu, gdyż staje się ona matrycą konwencji wierszowej.

3. Kod jako tekst czytalny i performatywny (kod działa, język nie działa)

Tytuł wiersza *<h1>depresja</h1>* Onaka i Podgórniego jest jednocześnie nagłówkiem języka html, i to działającym! Jeśli wpiszemy go w plik html, to przeglądarka internetowa wyświetli słowo „depresja" dużą czcionką i pogrubieniem. Zabieg ten nie tylko zatem przywołuje kod, ale sam w sobie jest – najprostszą z możliwych – cząstką działającego kodu. Ten rodzaj obecności kodu w dziele Cayley umieszcza w trzeciej kategorii, nie tak popularnej jak jej poprzedniczka i reprezentowanej na przykład przez tzw. *perl poetry* – poezję pisaną krótkimi komendami języka perl, które dzięki temu, iż język angielski jest budulcem subkodów programowania, powstałe wiersze daje się zarówno wykonać na komputerze, jak i czytać. Choć ta ostatnia czynność odbywa się z pogwałceniem gramatyki, spójności zdaniowej i koherencji segmentów.

4. Kod jako kodowanie

Czytanie kodu z naciskiem na jego transformatywny potencjał w obrębie materialnego podłoża tekstu cyfrowego przynależy do kategorii czwartej i reprezentuje go między innymi rozumienie kodu przez Katherine N. Hayles. Kod wykonuje tu głębinową pracę przetwarzania sygnałów z jednego systemu sygnifikacji, począwszy od fizycznego ciągu przerwań w przepływie prądu, które przekładają się na zera i jedynki, a skończywszy na kodowaniu alfabetu języka komunikatu tak, by na ekranie wyświetliły się odpowiednie znaki diakrytyczne. Tak ujęty kod wskazuje na ontologię przekazu cyfrowego, ale jednocześnie, podobnie jak kod jako język (w rozumieniu pierwszym), sytuuje się w rejestrze dla czytelnika nieistotnym. Można by nazwać ów rejestr pasmem nowej, cyfrowej trywialności, odnosząc się do dawnego rozróżnienia Espena Aarsetha na lekturowe czynności trywialne i nietrywialne, kiedy to w lekturze książkowej działaniem trywialnym jest przewracanie kartek[15].

5. Kod jako programowanie

Żadne z wyżej wymienionych spojrzeń na kod nie podnosi go do rangi prawdziwie aktywnego podmiotu sprawczego, oddziaływającego nie tyle na sam akt komunikacji (wyświetlanie się tekstu, interakcja) czy przywoływanie kontekstów socjokulturowych (kod jako przywoływana konwencja i konwencja przywoływania). Czyni to dopiero spojrzenie piąte, w świetle którego kod to przede wszystkim programowanie, zestaw metod, które przebiegają w czasie i wytwarzają tekst zgodnie z określonymi „na wyjściu", po stronie autora, regułami, i które pozwalają

[15]Cayley pisze: „oczywiście dobrze być świadomym tego, że za sprawą kodu tekst w mediach cyfrowym stanowi spiętrzenie warstw, które „migocą" pod tekstem widocznym na ekranie, jednak raczej nie będzie nas obchodzić – jeśli zamiarem naszym jest interpretacja – czy tekst, który czytam, jest zakodowany jako ASCII czy jako Unicode. J. Cayley, *Time code language*..., s. 314.

czytelnikowi obserwować rezultaty takiego programowania jako efekty poetyckie. To dzięki tym aspektom w twórczości Cayleya, ale także poetów Rozdzielczości Chleba, kod komputerowy staje się narzędziem aktywności na tekście, motorem eksploracji granic wypowiedzi w nowym medium, innymi słowy – poetyką w działaniu.

W *Booms* Piotra Puldziana Płucienniczaka rola kodu jest w miarę prosta: w równych odstępach czasu wyświetlić nową zawartość wersów, opierając się na podmianie wyrazów w obrębie ustalonej konstrukcji zdaniowo-wierszowej ze wskazanych zbiorów słów. Kod wyświetla też w tytule liczbę odsłon generatora w danej sesji lekturowej.

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.

Utwór Płucienniczaka rozgrywa się w czasie. Scena tekstu, jak nazywa ją Cayley, nie jest rozbudowana, a działanie kodu przebiega według prostego scenariusza. Niemniej jednak dzięki kodowi wprowadzony został do wiersza kluczowy aspekt temporalny, którego poetycki efekt jest nie do (efektywnego) odtworzenia w medium papierowym. Wiersz ten trzeba czytać jako serię. W postaci jednostkowej kopii, wydrukowanej w tomiku czy przysłanej jako załącznik e-mejlem *Booms* byłby zwykłym wierszem krytykującym konsumpcjonizmu „cyfrowego stylu życia" w obliczu globalnego terroryzmu. Podmiot liryczny jawi się jako osoba nie mogąca, nie chcąca i nie potrafiąca sytuacjom opisywanym w tle zapobiec. Zapewniona przez kod temporalizacja zmienia wymowę całości radykalnie. Po 10 minutach spędzonych przed ekranem czytelnik zaobserwuje, że wariantom wiersza nie ma końca. Co więcej, każdy z nich jest liczony, a liczba ta pojawia się w tytule. Po godzinie, po dwóch, opisywane bomby nadal wybuchają. Podczas gdy kochający gry komputerowe i chipsy narrator zajmuje się coraz to innymi, banalnymi czynnościami, liczba w tytule zaczyna nabierać przerażających rozmiarów. Jej ciężar gatunkowy przekracza w końcu nieokreśloną, być może indywidualną dla każdego, masę krytyczną i zaczyna przechylać szalę całego wiersza z nihilizmu na krzyk rozpaczy i współczucia. Jedynym wyjściem czytelnika, paradoksalnie, jest zamknięcie okna przeglądarki, by nie widzieć owej przeraźliwie rosnącej liczby, która w realnym, pozatekstowym świecie przekłada się na liczbę ofiar zamachów.

Sonet niezachodzący Leszka Onaka stawia na temporalność w równie mocnym stopniu i na skalę wykraczającą poza ramy pojedynczego autorstwa. Napisanym w php skrypt przeszukuje nagłówki wiadomości z polskich portali informacyjnych i następnie układa je w czternaście linijek sonetu. Gdy czytelnik naciśnie przycisk „generuj nowy sonet", program ponownie wysyła kwerendę i wiersz uaktualnia się o nową zawartość w linijkach i tytule[16]. Konsekwencje pracy kodu widoczne są nie tylko na poziomie lokalnym. O ile utwór Płucienniczaka można sobie wyobrazić

[16]Na temporalny aspekt tego utworu stawiała nacisk Urszula Pawlicka: „warto zwrócić uwagę na nieustanną dezaktualizację przywoływanego linka do konkretnego wygenerowanego sonetu – za każdym razem będzie bowiem odsyłał do nowego utworu, bazującego na najświeższych newsach". U. Pawlicka, *Polska poezja cybernetyczna*, Kraków 2012, s. 115.

w formie tradycyjnej, jako wielotomowy wydruk wszystkich kombinacji, jakie wytwarza generator, o tyle dzieło Onaka takiej wstecznej „retromediacji" poddać się nie jest w stanie. Czas, jaki upływa na scenie tekstu, stapia się jeszcze ściślej z czasem rzeczywistym czytelnika. Z każdą minutą wiersz będzie się zmieniał w zależności od tego, o czym piszą informacyjne portale.

Program Onaka nawiązuje fundamentalną dla istnienia wiersza relację z programami zewnętrznymi, zaproszonymi do swoistej pracy zespołowej „na żywo". Podmiotami tej kolaboratywnej aktywności nie są jednak konkretni ludzie, lecz połączone w sieci komputery i działające na nich programy, których zadaniem jest wypełnianie treścią kanałów informacyjnych RSS. W tle *Sonetu niezachodzącego* doświadczamy zatem sił sprawczych o charakterze zdecydowanie postludzkim. Rola autora sprowadza się do projektanta i kuratora pola zdarzeń. Jeśli technologia RSS zawiedzie, albo wybrane portale, dzięki którym dostarczane są zwroty i zdania, przestaną istnieć, to „nieustająco ść" sonetu albo staje pod znakiem zapytania, albo nabiera dodatkowego znaczenia: o tego typu wiersz, i kod, poeta musi nieustannie dbać niczym ogrodnik, doglądający swoich roślin.

Perigram jako tekst generatywny, sieciowy i postludzki

Poetyka, jak przypomina Watten, żywi się tekstami eksperymentalnymi i radykalnymi. Jako praktyk (wydawca, producent) literatury elektronicznej nie zetknąłem się w ostatnich latach z dziełem bardziej złożonym i radykalnym niż seria programów poetyckich przygotowanych przez Johna Cayleya i Daniela C. Howe'a w ramach cyklu *Reader's project* (2010–2016)[17]. Poszczególne realizacje tego długoletniego projektu skupiają w sobie całą gamę złożoności, którą wnosi do poezji i do poetyki kod w działaniu właściwym, a zatem programowalnym i przekształcającym zawartość wiersza, jego styl, retorykę i kontekst.

„Readers" w nazwie projektu to określenie nader mylące. Nie chodzi bowiem ani o „czytelników" ani o „czytniki" (w sensie nośniki), lecz o pojedyncze algorytmy uważnie skanujące tekst źródłowy i generujące na jego podstawie tekst pochodny[18]. Reguły, według których taka robotyczna lektura się odbywa, oparte są na automacie komórkowym *Gra w życie* (ang. *The Game of Life*) brytyjskiego matematyka Johna Conwaya[19], z tą różnicą, że funkcję komórek w „grze w czytanie" Howe'a i Cayleya pełnią słowa, a ruch odbywa się nie na nieskończonej planszy, lecz na płaszczyźnie wirtualnej strony książkowej, a zatem na obszarze warunkowanym konwencjami czytania i materialnymi uwarunkowaniami nośników tekstu w kulturze[20]. Przestrzeń matrycy wypełniona jest przez tekst źródłowy. Zaprogramowany czytnik przebiega przez tekst, uaktywnia „żywą" komórkę słowną, a zostawia za sobą „martwą". Podobnie jak w *Grze w życie*, gdzie aktywna komórka ma swoich sąsiadów, aktywny obszar komórki słownej, jej typograficzne sąsiedztwo, składa się potencjalnie z ośmiu okalających ją słów, z mniejszej liczby, gdy słowa na krańcach (kierunki: pn.-wsch., pd.-wsch., pn.-zach., pd.-zach.

[17]Zob. *Readers Project*, <http://thereadersproject.org> dostęp: 23.03.2016.

[18]Określam je tutaj – mimo wszystko – terminem „czytników". Szerzej o projekcie obaj autorzy piszą w: J. Cayley, D.C Howe, *The Readers Project: procedural agents and literary vectors*, „Leonardo" 2011, vol. 1, 43, s. 317-324.

[19]Zob. m.in. P. Coveney, R. Highfield, *Granice złożoności. Poszukiwania porządku w chaotycznym świecie*, Warszawa 1997, s. 131-132.

[20]Cayley i Howe przypominają o arbitralności takiego wyboru i jego związku z konwencjami panującymi w Europie i USA.

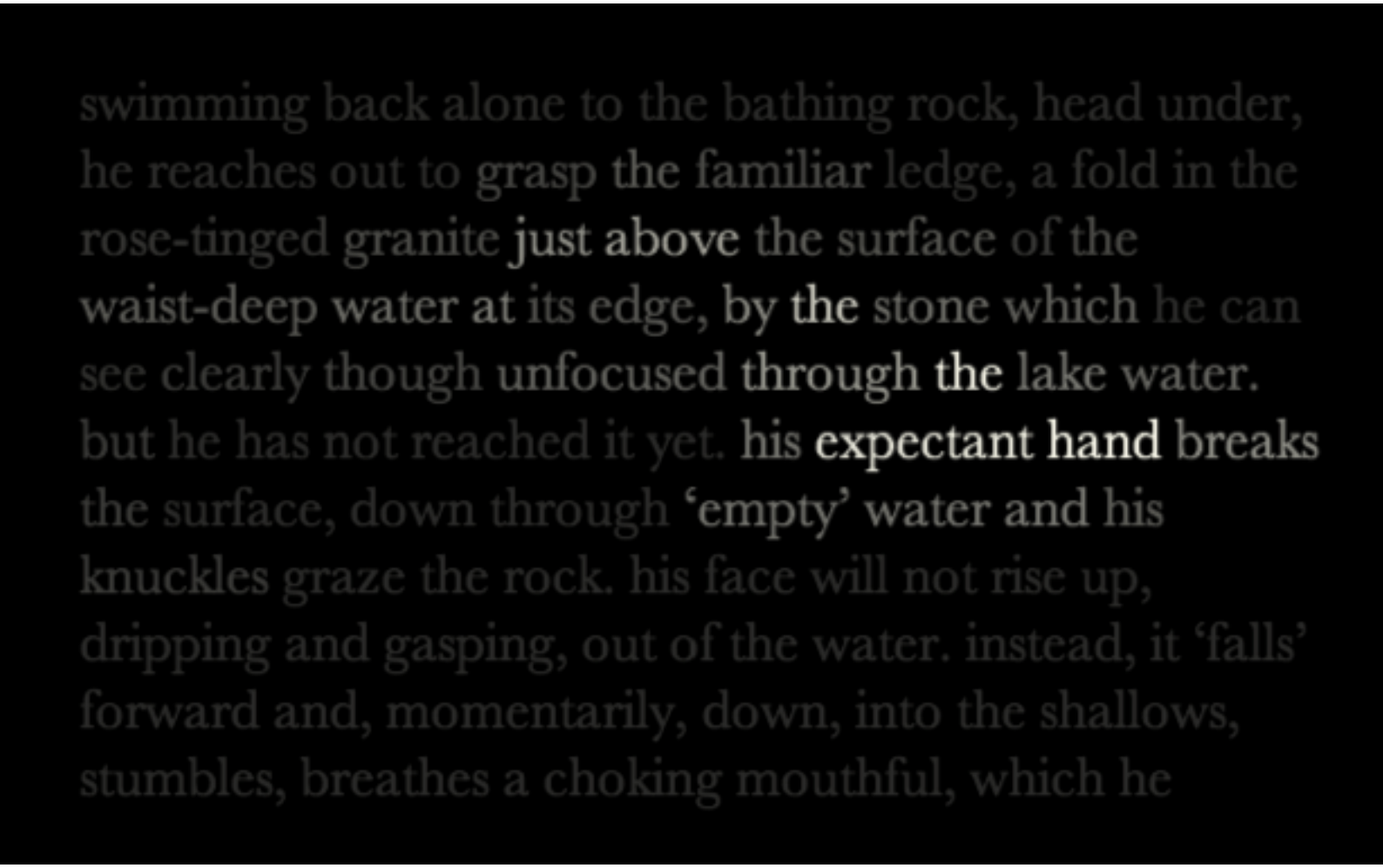
nie zazębiają się graficznie ze słowem aktywnym w centrum. Czytnikiem, na którym Cayley i Howe demonstrują swój system, jest perigram. Biegnie on od lewej do prawej, lecz jego lektura nie jest do końca linearna, gdyż program obiera za swój cel nie sąsiada w tym samym wersie, lecz słowo leżące na jego prawym górnym lub prawym dolnym krańcu. Porusza się on zatem do przodu, ale może zmieniać kurs na osi góra – dół i w obrębie obszaru o powierzchni rozciągniętej na około 20 słów. Reguły ruchu perigramu określone są następująco:

> W trakcie przemieszczania się po tekście, czytnik perigramowy zapamiętuje każde z poprzednio uaktywnionych słów, a następnie sprawdza jako potencjalne następne słowo swoje sąsiedztwo w prawym górnym i prawym dolnym rogu. Jeśli jest w stanie ustalić, że zestaw tych trzech słów tworzy (poprzednie, aktualne i potencjalnie następne, w tej kolejności) frazę, której frekwencja przekracza pewien określony próg (np. bywała ona użyta wcześniej w języku naturalnym), to wówczas jego ścieżka lektury może odbić w inną stronę, w rezultacie generując tekst alternatywny, perigramatyczny[21].

System czytania/generowania tekstu robi się w tym miejscu coraz ciekawszy. Otóż kod perigramu steruje nie tylko jego przebiegiem po typograficznej powierzchni tekstu źródłowego (czasem jest to tekst Samuela Becketta, czasem poetycka proza Cayleya). Równolegle wysyła on zapytania do przeglądarek internetowych, z kwerendą, której zawartością jest potencjalna fraza montowana „w locie" przez perigram. Jeśli generuje ona długą listę wyników wyszukiwania, perigram rezygnuje z niej na rzecz frazy alternatywnej, o niższej frekwencji w listach wyników Google, Bing czy Yahoo. Cayley i Howe, a ściślej napisany przez Howe'a program Rita, sprawdzają zatem poetycką oryginalność sekwencji werbalnych, które wyświetlać ma perigram. Ów test oryginalności odbywa się w obrębie największego, globalnego zasobu językowego, jakim jest zarówno zawartość Internetu, jak i zapytania użytkowników przeglądarek wpisywane w okienka zapytań[22]. Ta „wszechnica" (ang. *commons*), jak nazywa ją Cayley, jest olbrzymim, dynamicznym słownikiem, powiększającym swoje zasoby z minuty na minutę i z godziny na godzinę. W rezultacie perigram z dnia na dzień może zmienić swój tok lektury/pisania, co zostało przez autorów udokumentowane przy okazji wystaw, na których podłączone do Internetu iPady z perigramami potrafiły generować inny tekst nawet wtedy, gdy fraza inicjująca ich pracę (o jej wpisanie proszeni byli zwiedzający) pozostawała taka sama i nie zmieniał się tekst źródłowy. Działo się to nie dlatego, że miliony użytkowników anglojęzycznego Internetu zaczęło nagle używać rzadkich, poetyckich zestawień słownych, lecz za sprawą autokorekty algorytmu Google, który w ponownym przebiegu po tym samym tekście napotkał jako wynik wyszukiwania swoją własną frazę (którą Cayley umieścił w Internecie, i która zdążyła zostać zaindeksowana przez roboty wyszukujące!).

[21] J. Cayley, D.C Howe, *The Readers Project*, s. 319-320.

[22] Wykorzystywanie słów podpowiedzi, jakie Google i inne wyszukiarki prezentują użytkownikom w locie, w takcie wpisywania kwerendy, złożyło się na inny podprojekt *Readers Project*. Była nim konceptualna książka *How it is in common tongues*, która polegała na przepisaniu treści opowiadania *How it is* Samuela Becketta po (ręcznym) przefiltrowaniu jej przez zasoby dostępne z poziomu wyszukiwarki internetowej. Cayley i Howe wpisywali po trzy do czterech kolejnych słów z Becketta w przeglądarkę i wybierali wyniki o najdłuższym ciągu słów, które nie pochodziły z tekstu Becketta cytowanego gdzieś w Internecie, lecz z oryginalnych wypowiedzi odnajdywanych na stronach WWW. Następnie tak odnalezioną we „wszechnicy" frazę umieszczali w nowej książce z przypisem, w którym podawali źródło cytowanej sekwencji. Zob. J. Cayley, D.C Howe, *How It Is in Common Tongues (The Readers Project: Common Tongues)*, Providence 2012.



Ilustracja 2. Czytnik perigramowy w działaniu

Powstaje pytanie, kim jest autor powstającego na żywo wiersza? Czy jest to poeta programista, autor tekstu źródłowego, wyszukiwarka internetowa czy milion aktywnych użytkowników Internetu – każdy z tych graczy ma swój współudział w dziele. Jaki statut w polu komunikacji przyznać czytnikowi? Ontologia tego lekturowego robota i tak jest już zwielokrotniona, gdyż program, czytając tekst jednocześnie go tworzy. Warto przy okazji dostrzec, jak duża przepaść dzieli „aktywnego czytelnika" – ulubioną figurę entuzjastycznie nastawionej do nowych mediów krytyki lat dziewięćdziesiątych od „aktywnego czytnika". *Readers Project* stawia jeszcze jedno ważne pytanie – o kondycję modernistycznego *episteme* w środowisku cyfrowym. Językowa innowacyjność i poetycka rewolta w postaci pisania, mówiąc w skrócie, z Google, lecz przeciwko Google, poszukiwanie oryginalności w globalnych zasobach angielszczyzny, to raczej kontynuacja niż zaprzeczenie modernizmu. Nie są to nawet postmodernistyczne riffy na ogranych akordach w stylu *Lexia from perplexia* Talana Memmota. Coś jest na rzeczy. Michael Joyce – pionier hipertekstu literackiego określa siebie mianem „ultramodernisty". Jessica Pressman mówi nawet o szerszej tendencji, całym nurcie „cyfrowego modernizmu", który zrodził się na fali technologicznego naelektryzowania tekstu[23]. Cayley i Howe dopisują pod tą banderą swój własny rozdział, jednocześnie sytuując się w awangardzie współczesnych metod badawczych humanistyki. W projekcie *Readers Project* i w samym

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

czytniku perigramowym odnajdujemy bowiem elementy działań na *big data*, elementy *data-miningu* i *crowdsourcingu* – narzędzi cyfrowej humanistyki. Jednocześnie jednak John Cayley przypomina, że działalność w ramach tego projektu nie jest niczym innym niż „wizualizacją poetyki" i „poetyką wizualną".

Powierzchnia i głębia dzieła cyfrowego

Czytnik perigramowy i jego lektura/pisanie prezentowany jest w galeriach i na wystąpieniach konferencyjnych Cayleya jako palimpsest, gdzie stonowana wizualnie matryca tekstu źródłowego to pole zdarzeń uruchamianych przez perigram. Podczas niektórych wystąpień praca przybiera formę atrakcyjnego dynamicznego mezostychu. Za tą estetycznie niebanalną, ruchomą mapą liter, słów i fraz, dobieranych przez komputer w zaskakujące, lecz wyczuwalnie kontrolowane ciągi wyrazowe[24], kryje się wysoce zorganizowany cybertekst. Poetyka poszerzona jedynie o element wizualny, filmowy, dźwiękowy czy nawet o działania sprawcze po stronie czytelnika nie byłaby w stanie go objąć. Nie pogłębiona o hermeneutykę kodu, tutaj jedynie przywołaną, byłaby zdolna co najwyżej analizować dzieła nie różniące się zasadniczo od – na przykład – bogato ilustrowanej książki dla dzieci z elementami interakcji (reprezentowanej np. przez książki typu *lift-the-flap*) czy dźwięku (przyciski, klawisze, możliwość odtwarzania i nagrywania głosu). Aspekt programowalny, kluczowy dla dzieła Cayleya i Howe'a, obserwowany pod lupą poetyki pozbawionej wyczulenia na kod, byłby zaledwie niewidocznym podłożem pola remediacji wszelkich porządków, z którymi perigram wchodzi na powierzchni tekstu w dialog (animacja, wizualność, książkowość), gdzie pewne cechy tych porządków podlegają amplifikacji, zwielokrotnieniu i poszerzeniu.

Reprezentowana przez Cayleya, a na polskim gruncie realizowana przez Rozdzielczość Chleba programatologia, a zatem kierunek autorskiej refleksji badający aktywne uczestnictwo napisanego przez autora kodu w produkcji znaczeń dzieła, reprezentuje poetykę nowych mediów w sposób bardziej konsekwentny i reprezentatywny niż niejedno dzieło okrzyknięte mianem przełomowego[25].

Perigram, hipertekst i przyszłość poetyki cyfrowej

W zestawieniu z wyzwaniami, jakie tradycyjnej poetyce stawia programowalny i podłączony do „wszechnicy" językowej czytnik Cayleya/Howe'a, hipertekst – będący w centrum uwagi krytyków i autorów w latach dziewięćdziesiątych, gdy cyfrowa rewolucja podbijała ośrodki edukacyjne w bogatych krajach Zachodu – jawi się jako pisarstwo dość konwencjonalne i mocno jeszcze przywiązane do paradygmatu książkowego[26]. Z dzisiejszej perspektywy należy powiedzieć, że jako tekst rozgałęziający się i działający na żądanie, hipertekst stanowił nie tyle

[24]Jako że perigram porusza się w obszarze o powierzchni około 20 słów od słowa aktywnego w danym momencie, w tekście pochodnym zachowane zostaje wyraźne pokrewieństwo semantyczne wyrazów, które składają się na wynikowe frazy.

[25]Omawiany przez Barreta Wattena w tomie *New Media Poetics* utwór *Lexia from Perplexia* Talana Memmotta to parodia hipertekstu i dyskursu o cyberkulturze, charakterystyczna dla czasu, w którym powstała (2002), kiedy to autorzy tzw. drugiej generacji literatury cyfrowej krytycznie odnosili się do autorów generacji pierwszej. Jej przełomowość ma charakter głównie historyczny. Dla poetyki nowych mediów dużo bardziej zanczęca okazała się jednak działalność Johna Cayleya.

[26]Taką też kondycję tego gatunku diagnozują dziś autorzy nowszych opracowań tematu. Zob. m.in. A. Bell, *The Possible Worlds of Hypertext Fiction*, London 2010; J. Baetens, F. Truyen, *Hypertext revisited*, „Leonardo" 2013, vol. 46, nr 5.

zerwanie, co remediację druku. Powieści Michaela Joyce'a (*popołudnie, pewna historia*; *Zmrok. Symfonia*), które były najbliżej wypełnienia manifestowego postulatu stworzenia dzieła, które zmienia się za każdym razem, gdy je czytamy – nie w sensie interpretacji tekstu, lecz w sensie samej substancji, ilości i kolejności, z jaką materiał powieściowy pojawia się na ekranie w kolejnych sesjach lekturowych – w świetle generatywnej i sieciowej poetyki perigramowej okazują się zadziwiająco statyczne. Nawet przy wzbogaceniu o system linków warunkowych statyczność ta, co prawda angażująca czytelnika, lecz pozbawiona znaczącego programowania, stawia hiperteksty w tej samej grupie, co wspomniana hybrydyczna i interaktywna książka dla dzieci, gdyż jego tekst pozostaje określony raz na zawsze i nie jest w stanie się ani poszerzyć, ani uszczuplić. Dotychczasowe strategie refleksji krytycznej nad dziełem cyfrowym, postulowane na polskim gruncie w niejednym wystąpieniu zarysowującym poetykę nowych mediów, raczej nie wystarczają i muszą zostać poszerzone i pogłębione.

Pogłębienie to dotyczyć musi aspektu kodowego i sieciowego tekstu. Jeśli bowiem zgodzimy się, że tekst elektroniczny złożony jest z warstwy materii, kodu, tekstu i działania, to perigramy, na nowo określając zakres tej drugiej, podają w wątpliwość ustalenia dotyczące tej pierwszej (hardware, platformy, systemy dystrybucji), wydzielając w jej obrębie kolejne segmenty, i rewolucjonizują tę trzecią. Działanie, do którego na niej dochodzi, płynie bowiem w tym przypadku od instancji nietożsamej ani z autorem, ani z czytelnikiem, ani z tekstem.

SŁOWA KLUCZOWE:

generatory poetyckie

HIPERTEKST

poezja kodu

Abstrakt:
Celem artykułu jest poszerzenie rozumienia kodu w cyfrowych formach literackich. Choć poetyka cyfrowa na polskim gruncie zdaje się ugruntowywać, to kodowy aspekt dzieła, zwłaszcza gdy przedmiotem refleksji są utwory, w których program komputerowy zamienia się w czynny podmiot sprawczy, poza pełną kontrolą autora i czytelnika, domaga się dodatkowych rozróżnień. Opierając się na przykładach polskiej literatury elektronicznej, zrekapitulowana zostaje typologia kodu według Johna Cayleya – pioniera poezji cyfrowej, a następnie przybliżona zostaje seria prac Cayleya/Howe'a, w ramach których zaprogramowane „czytniki" na podstawie tekstu źródłowego i zasobów językowych indeksowanych przez Google, wysyłane są na misję specjalną: poszukiwanie oryginalności poetyckiej. W pracy sformułowano trzy tezy: programowanie jest nowym rodzajem poetyki w działaniu; kod w tekście temporalnym i sieciowym osiąga statut autonomicznego aktora, sytuującego się pomiędzy tekstem, autorem i czytelnikiem, i będącego w kontakcie z innymi programami w sieci; hipertekst jako pierwotny paradygmat tekstualności cyfrowej okazuje się formą przejściową, z punktu widzenia praktyki Cayleya i polskich poetów cybernetycznych, bliższą paradygmatowi druku.

s e m i o t y k a c y f r o w a

e-literatura

poetyka generatywna

Nota o Autorze:
Mariusz Pisarski – badacz i wydawca literatury elektronicznej. Autor książki *Xanadu. Hipertekstowe przemiany prozy* (Kraków 2013), redaktor pisma „Techsty" oraz linii multimedialnej wydawnictwa Ha!art. Tłumacz prozy i poezji cyfrowej, autor hipertekstowych adaptacji klasyki literackiej (*Rękopisu znalezionego w Saragossie*, 2012; sieciowej adaptacji opowiadań Brunona Schulza *Bałwochwał*, 2013). W 2011 nominowany do nagrody Theodora Holma Nelsona przez Amerykańskie Stowarzyszenie Informatyczne ACM (Hypertext 2011). Jego praca doktorska o hipertekście (UAM, prof. Bogusław Bakuła) otrzymała I wyróżnienie w konkursie Narodowego Centrum Kultury. Członek Electronic Literature Organization. Stypendysta SAIA w Instytucie Literatury Powszechnej Słowackiej Akademii Nauk. Współpracownik Pracowni Badań Intersemiotycznych i Intermedialnych w Instytucie Filologii Polskiej Uniwersytetu Warszawskiego. |