PRZEGLĄD KULTUROZNAWCZY
NR 3 (29) 2016, s. 305–317
doi: 10.4467/20843860PK.16.023.6025
www.ejournals.eu/Przeglad-Kulturoznawczy/

Mateusz Felczak

Uniwersytet Jagielloński
Wydział Zarządzania i Komunikacji Społecznej

OBIEKT I KOD. BADANIE GIER WIDEO W PERSPEKTYWIE *SOFTWARE STUDIES* ORAZ REALIZMU SPEKULATYWNEGO

The Object and the Code: Analyzing Video Games through Software Studies and Speculative Realism

Abstract: The article deals with the analysis of the ludic and mechanical aspects of the video games using Levi Bryant's notion of onticology. It expands the Bryant's idea that the being is composed entirely of machines or processes, and treats a video game as a collective of objects that communicate with each other through various codes. The main focus of this approach is to analyze the instances in which the said objects present a certain type of self-alienation, independency and agency that allow for a continuous reconfiguration of the states in which a video game object can be found. The various scripts and their role in altering the communication patterns within the game mechanics are investigated in the context of bots, regarded as both automatic prothesis of the human agents and autonomous interpreters of the games' implicit rules. Cases of speedruns, glitches and particular uses of the basic C++ language syntax are also considered as the examples of events in which the hidden properties of the objects form a collective process of a video game. On such grounds, this paper discusses the possible removal of distinct ontological domains of subject and object in the context of object-oriented philosophy, taking into account the fluid hierarchy of beings and their mutual inaccessibility.

Keywords: object-oriented philosophy, game studies, video games, onticology

Celem tego tekstu jest przyjrzenie się mechanizmom działania gier wideo w kontekście szczególnego ujęcia filozofii przedmiotu – ontikologii Leviego Bryanta, wchodzącej w skład nurtu zwanego realizmem spekulatywnym. Analizowane zagadnienie rozważane będzie w perspektywie studiów nad oprogramowaniem (*software studies*), co motywowane jest chęcią stworzenia mapy zależności pomiędzy informatyczną a ludyczną warstwą poszczególnych fenomenów elektronicznej rozrywki w ujęciu kulturoznawczym. Przedmiotem zainteresowania w sensie szczegółowym

będą relacje, w jakie wchodzą z sobą poszczególne obiekty tworzące grę – zarówno te umieszczone bezpośrednio w kodzie, jak i egzekwujące później zawarte w nim instrukcje. Taka perspektywa otwiera pole analizy sprawczości aktorów ludzkich, jako użytkowników i odbiorców komunikatów zawartych w grze, oraz aktorów nieludzkich, którzy odgrywać mogą takie same role w łańcuchu wymiany informacji podporządkowanym algorytmom, składającym się na grę wideo. Wybór tego właśnie medium wynika nie tylko z konieczności rozpoznania mechanizmów działania najszybciej rozwijającego się obecnie sektora kultury audiowizualnej, ale również z możliwości przetestowania w praktyce połączenia różnych narzędzi analizy i interpretacji: od językowego badania struktury kodu do namysłu nad współdziałaniem w obiegu kultury różnych instancji nadawczo-odbiorczych, wśród których gracze, projektanci gier i kuratorzy platform dystrybucji stanowią jedynie fragment wielopłaszczyznowego systemu wzajemnych zależności. Badanym polem, na którym owe zależności mogą zachodzić, będą dwa szczególne przypadki: działanie skryptów automatyzujących mapowanie urządzeń wejścia oraz praktyki związane ze *speedrunem*, czyli możliwie jak najszybszym ukończeniem danej gry bez uciekania się do oszustw. Choć definicję obu z nich wyznaczają konkretne działania człowieka, to zarówno skrypty, jak i kulturę *speedrunu* będę traktował jako furtki umożliwiające wgląd w to, jakie właściwości posiadają obiekty wchodzące w skład gry rozumianej jako przedmiot. W szerszym kontekście interesuje mnie pytanie o poszczególne jednostki (*unit*) w rozumieniu Iana Bogosta, używającego tego terminu w zastępstwie implikującego materialność obiektu[1]. Innymi słowy artykuł będzie próbą zbadania, czy traktowanie wszystkich możliwych do wyróżnienia składowych gry oraz zachodzących wewnątrz niej relacji jako materialnych jednostek (w ujęciu Bogosta) bądź obiektów (u Bryanta) może pomóc w zorientowanej kulturoznawczo analizie interesującego mnie obszaru studiów nad grami. Choć zdaję sobie sprawę z różnic pomiędzy poszczególnymi stanowiskami badaczy skupionych na przedmiocie, uważam, że dla udanej analizy fenomenu gier przydatny jest (choćby doraźny) sojusz tych różnych od siebie perspektyw filozofii spekulatywnej. Z tego właśnie powodu w dalszej części artykułu będę używał terminów obiekt, jednostka i byt w znaczeniu równoważnym.

1. Wstępne ustalenia

Zbieżność proponowanych tu pojęć z Latouriańską konceptualizacją aktantów w procesie wytwarzania i weryfikowania informacji nie jest przypadkowa, choć zamiast teorii aktora-sieci w dalszej części pracy posłużę się ujęciami nastawionymi bardziej na analizę procesów niż miejsca poszczególnych sprawczych instancji we

[1] I. Bogost, *Alien Phenomenology, or What It's Like to Be a Thing*, University of Minnesota Press, Minneapolis–London 2012 [wersja e-book], loc. 440.

wzajemnej sieci zależności[2]. W tym celu posłużę się ustaleniami Iana Bogosta, którego zarówno w *Alien Phenomenology*[3], jak i w *Persuasive Games*[4] zajmowały bardziej zagadnienia właściwości różnych obiektów niż naszego dostępu do nich. Nawet wcześniejsza publikacja tego autora, wydana w 2006 roku *Unit Operations*[5], analizowała poszczególne procesy z punktu widzenia tworzących je przedmiotów. Innymi słowy Bogost zdaje się powtarzać za Levim Bryantem, że nie można zredukować obiektów do relacji, w które wchodzą[6], tym bardziej że zaproponowane w *The Democracy of Objects* ujęcie kolektywu wskazuje na konieczność wykonywania za każdym razem wysiłku translacji informacji pomiędzy poszczególnymi obiektami. Bogost – badacz gier – optuje za wykorzystaniem matematycznej metafory Alaina Badiou, według którego poszczególne części zbioru opisują ich wzajemną konfigurację, ale nie reguły swojego zachowania[7]. Wydaje się, że w grach komputerowych tym bardziej ujawnia się pewna alienacja i brak wzajemnego, pełnego dostępu obiektów do siebie, stąd konieczność aplikacji odpowiednich instrukcji postępowania, które z uwagi na określone właściwości przedmiotów podejmują pracę przenoszenia znaczeń między równymi sobie instancjami.

Po tych wstępnych ustaleniach dotyczących obiektów i ich potencjalnych relacji czas zastanowić się głębiej nad kwestią sprawczości. W proponowanym przeze mnie ujęciu ściśle łączy się ona z pojęciem algorytmu rozumianego jako opis metody, za pomocą której maszyna rozwiązuje stawiane przed nią zadania[8]. Pojęcie sprawczości na gruncie badania gier jest szczególnie kłopotliwe do zdefiniowania, jako że zbudowanie typologii zachowań spełniających warunki dobrowolności oraz mierzalnego wpływu na poszczególne elementy gry jest trudne nawet przy ograniczeniu jego użycia do samych tylko graczy. Interesującą propozycję odmiennego rozumienia możliwych aplikacji koncepcji sprawczości w odniesieniu do gier wideo stanowi koncepcja Karen i Joshuy Tanenbaumów[9], którzy w miejsce kojarzonych ze sprawczością kategorii wyboru oraz wolności zaproponowali skupienie na „zaangażowa-

[2] Oprócz samej teorii przedstawionej przez Brunona Latoura (przede wszystkim w *Reassembling the Social: An Introduction to Actor-Network-Theory*, 2005) zob. również opinię polskich badaczy teorii o ANT, np.: „ANT uparcie zajmuje się relacjami, a nie esencjami". K. Abriszewski, *Poznanie, zbiorowość, polityka. Analiza teorii aktora-sieci Bruno Latoura*, Universitas, Kraków 2012, s. 10.

[3] I. Bogost, *Alien Phenomenology…*, *op. cit.*

[4] I. Bogost, *Persuasive Games: The Expressive Power of Videogames*, The MIT Press, Cambridge–London 2010.

[5] I. Bogost, *Unit Operations: An Approach to Videogame Criticism*, The MIT Press, Cambridge–London 2006.

[6] L.R. Bryant, *The Democracy of Objects*, Open Humanities Press, Ann Arbor 2011, s. 26.

[7] I. Bogost, *Alien Phenomenology…*, *op. cit.*, loc. 515.

[8] A. Goffey, *Algorithm*, w: M. Fuller (ed.), *Software Studies: A Lexicon*, The MIT Press, Cambridge–London 2008, s. 15.

[9] K. Tanenbaum, J. Tanenbaum, *Commitment to Meaning: A Reframing of Agency in Games*, „Digital Arts and Culture" 2009, no. 12–15. Wszystkie tłumaczenia, o ile nie podano inaczej, w opracowaniu autora artykułu.

niu/poświęceniu" (*commitment*) w tworzeniu nowych znaczeń jako wyznaczniku pomagającym uczynić pojęcie sprawczości bardziej przydatnym w analizie konkretnych zagadnień związanych z grami wideo. Chciałbym wykorzystać tę koncepcję, rozszerzając jednak kategorię „zaangażowania" na wszystkie obiekty wchodzące potencjalnie w skład kolektywu nazywanego grą. Stąd w dalszej części artykułu sprawczość wszelkich obiektów będzie rozpatrywana w kontekście ich potencjalnej woli wchodzenia w interakcje z innymi obiektami, nie zaś wyboru konkretnej formy bądź wyniku takiego spotkania. W praktyce oznacza to, że świadomie rezygnuję tu z badania sprawczości za pomocą porównania liczby możliwych do osiągnięcia stanów końcowych interakcji. W zamian proponuję podejście, które skupia się na wyrażaniu przez obiekty gotowości do rekonfiguracji swojej pozycji za każdym razem, gdy tworzy się nowy wariant rozgrywki.

W rozważaniach na temat ontikologicznej interpretacji gier wideo będę przede wszystkim zajmował się obiektami, które skupiają się wokół bądź wewnątrz maszyny skupiającej w sobie procesy komunikacji ustanawiające działanie gry wideo. By użyć terminologii Bryanta[10]: interesuje mnie działanie kolektywu, w którym ludzie należą do zbioru obiektów nazywanego grą wideo. Skoro według jednej z nakreślonych przez Bryanta zasad „płaskiej ontologii" żaden obiekt nie ma bezpośredniego dostępu do innego obiektu[11], tym bardziej interesujące jest badanie procesu komunikacji i wymiany informacji, który musi zachodzić między obiektami, by te mogły tworzyć kolektywy i realizować swoje zadania. Dla Bryanta kluczową postacią w refleksji o takich procesach jest – co dość zaskakujące – Niklas Luhmann, szczególnie w jego poglądach na temat *systemu* jako osobnego i rzeczywistego bytu/obiektu[12]. Perspektywa systemowa dobrze współgra ze współczesnymi praktykami odbioru gier, które jako usieciowione fenomeny nieustannie redefiniują nie tylko definicje nadawcy i odbiorcy, ale także – szerzej – instancji nadającej bądź odbierającej informacje. W ontikologicznej perspektywie Bryanta obiekty mogą funkcjonować poza systemem zależności, który jest ich domyślnym środowiskiem[13], co jest po pierwsze znaczną modyfikacją modelu Luhmannowskiego, a po drugie – otwiera pole do lepszego zrozumienia właściwości obiektów mogących tworzyć system-obiekt zwany grą. Uważam, że śledzenie działania obiektów gry w sytuacjach takiego dobrowolnego „wykluczenia" z naturalnego środowiska zaprojektowanych przez deweloperów scenariuszy rozgrywki jest właściwym momentem obserwacji skupionej na grze-przedmiocie, z jakim rzeczywiście mamy do czynienia we współczesnym krajobrazie obiegu mediów cyfrowych.

[10] L.R. Bryant, *The Democracy...*, *op. cit.*, s. 14.
[11] *Ibidem*, s. 26.
[12] *Ibidem*, s. 137.
[13] *Ibidem*, s. 144.

2. *Speedrun*

Zjawisko *speedrunu* doczekało się na gruncie studiów nad grami stosunkowo niewielu wchodzących z sobą w dialog opracowań, a dla celów niniejszej analizy skupię się na typologicznym ujęciu zjawiska zaproponowanym przez Rainforesta Scully-Blakera[14]. W opublikowanym przez niego na łamach periodyku „Game Studies" artykule pojawia się podział *speedrunów* ze względu na – z punktu widzenia ontikologii Bryanta – zmianę końcowego stanu wchodzących w jego skład obiektów. W przypadku *finesse runs* (rozgrywek finezyjnych) przedmioty pozostają na swoich znarratywizowanych miejscach: ich początkowy układ, zbieżny z zapisanym w kodzie przebiegiem fabuły, pozostaje właściwie nienaruszony. Drugą kategorię stanowią *deconstructive runs* (rozgrywki dekonstrukcyjne), gdzie dochodzi do głosu „przemoc szybkości" Paul Virilio; wyjściowy układ zapisanych w kodzie zależności przyczynowo-skutkowych ulega całkowitej niemal destrukcji, a na jego zgliszczach powstaje nowy byt, który z zaprojektowaną sytuacją kompletnej rozgrywki łączą jedynie dwa punkty: zgodne z wyjściowym projektem twórców początek oraz koniec gry. Dodatkowo w ujęciu kanadyjskiego badacza pojawiają się dwie definicje reguł, według których funkcjonuje *speedrun*. Poza dość standardowo rozumianymi zasadami następstwa treści fabularnych pojawia się tu kategoria reguł *eksplicytnych*, które dotyczą wzajemnego oddziaływania obiektów w grze. Znamienne, że Scully-Blaker ostatecznie charakteryzuje *speedrun* jako „praktykowaną praktykę" (*practiced practice*) oraz „praktykę przestrzenną wewnątrz praktyki przestrzennej". W perspektywie ontikologicznej byłaby to próba dość omownego przyznania prymatu niezbywalnym właściwościom poszczególnych jednostek współtworzących grę (czy też raczej szczególny jej przypadek: obiekt nazywany *speedrunem*) w stosunku do chwilowych i, jak się okazuje, powierzchownych relacji, w jakie wchodzą, tworząc nietrwałe byty poszczególnych wariantów rozgrywki. W tak zarysowanym podziale można też dostrzec związki z myślą Alaina Badiou, dla którego „byt wydarza się w każdej prezentacji", a „sytuacja ontologiczna jest prezentacją prezentacji"[15], to znaczy nie może zaistnieć poza *mnogością*. Typologia Scully-Blakera rozpoznaje różne rodzaje *speedrunu* jako gwałtowne ujawnienie się właściwości (i potencjalnej sprawczości) bytów dotychczas skrytych poza obszarem zbiorów „klasycznych" wersji rozgrywek. Dzieje się tak nawet w przypadku *finesse runs*, obiekty w grze bowiem wchodzą wtedy z sobą w wyidealizowany dialog, umożliwiający jak najszybszą „podróż" po przestrzeni gry w kierunku przewidywanego przez jej twórców zakończenia. Opozycja, którą w praktykach *speedrunów* wyróżnia Scully-Blaker, daje się zatem opisać jako eksploracja dwóch krańcowych postaci zbioru obiektów nazywanych „grą": za-

[14] R. Scully-Blaker, *A Practiced Practice: Speedrunning through Space with de Certeau and Virilio*, „Game Studies" 2014, vol. 14, iss. 1.

[15] A. Badiou, *Byt i zdarzenie*, przeł. P. Pieniążek, Wydawnictwo Uniwersytetu Jagiellońskiego, Kraków 2010, s. 37.

równo przemoc *deconstructive runs*, jak i nienaturalna perfekcja *finesse runs* uwalniają inne właściwości „growych" bytów.

Speedrun jest przy tym tylko jedną z form tak zwanej rozgrywki emergentnej, czyli takiej, która powstaje w wyniku indywidualnych działań graczy w obrębie możliwych (choć niekoniecznie dozwolonych czy przewidzianych przez twórców) mechanik interakcji z grą. W tym polu wymienić można wiele innych praktyk, zawsze jednak kluczowe jest w nich wykorzystywanie inwencji oraz tworzenie pola kreacji nowych znaczeń bez względu na to, czy w interakcje wchodzą z sobą poszczególne instrukcje kodu, czy sami gracze. Emergencja jest przestrzenią negocjacji: czy to w przypadku ustalania metareguł w sieciowych grach wieloosobowych (*League of Legends*, *Counter-Strike*), czy też kreatywnego wykorzystania dostępnych narzędzi cyfrowej autoekspresji (*Minecraft*, *Terraria*). By zakończyła się sukcesem, potrzebne jest zbudowanie komunikacji pomiędzy elementami gry – nawet błąd czy *glitch* też jest w tym kontekście pozytywnym rezultatem jakiejś wypływającej z emergentnego potencjału gry akcji.

By móc bliżej przyjrzeć się strukturze *speedrunu*, należy zarysować towarzyszący mu kontekst związany z elementami będącymi przedmiotem zainteresowania *software studies*. W klasycznym paradygmacie programowania obiektowego, który jest często wykorzystywany przy tworzeniu gier na platformy PC oraz niektóre konsole, „obiektami" nazywa się miejsca przechowywania danych. Jako przykład zasad funkcjonowania takich szczególnych obszarów pamięci komputera można podać te dotyczące jednego z najpopularniejszych języków używanych przy pisaniu kodu źródłowego: C++. W tym paradygmacie dostęp do obiektu warunkuje jego nazwa – posiadający ją obiekt staje się zmienną, którą charakteryzuje określony typ, decydujący o rodzaju danych, jakie można w niej zapisać, oraz operacji, które można na niej wykonać[16]. Egzekucją kodu zajmuje się specjalny program nazywany kompilatorem, który musi za każdym razem dokonać interpretacji zawartych w kodzie źródłowym poleceń i ich translacji na zrozumiały dla komputerów kod maszynowy (wyglądający z punktu widzenia przeciętnego człowieka jak ciąg zer i jedynek). Dla naszych rozważań istotne będą dwa fakty wynikające ze specyfiki składni języków programowania obiektowego: konieczność każdorazowego przypisywania danych do określonych zmiennych oraz pewna arbitralność (i nieprzewidywalność) działań kompilatora. Szczególnie interesującym aspektem omówionych tu zjawisk jest ich wymiar temporalny. Zasady doraźnego współdziałania poszczególnych obiektów zawartych w kodzie są teoretycznie wyznaczane przez reguły składni danego języka programowania, lecz już ich właściwości wcale nie muszą się podporządkowywać logicznym regułom języków naturalnych. Dla kompilatora działanie

[16] B. Stroustrup, *Programowanie. Teoria i praktyka z wykorzystaniem C++*, Helion, Gliwice 2010, s. 80–82.

***int* wiek_gracza = –10**

jest instrukcją poprawną i zostanie wykonane, ponieważ –10 jest liczbą naturalną (takie obsługuje typ zmiennej *int*), a nazwa obiektu „wiek_gracza" również zawiera jedynie poprawny łańcuch literałów znakowych. Ten prosty przykład ilustruje specyfikę kompilatora, którego poprawne działanie (umożliwienie interakcji między obiektami) nie zawsze pozostaje w zgodzie z przekonaniami wyznawanymi przez inne obiekty wchodzące doraźnie w skład obiektu-gry (w tym ludzi). Wiek gracza zostanie w powyższym wypadku zapisany jako wartość ujemna i tak też będzie funkcjonował w środowisku gry. Co ważne, tego typu „nielogiczne" działania dość często pojawiają się w kodzie gier nawet wysoko budżetowych tytułów. Niekoniecznie świadczą one o błędach – czasem programistom po prostu łatwiej rozwiązać dany problem z zastosowaniem obiektów wymykających się prostej logicznej typologii. W opisywanym paradygmacie programowania możliwe są też przypadki nieintuicyjnego dziedziczenia właściwości obiektów przez inne, które choć z punktu widzenia zaprojektowanego przez twórców porządku logiczno-fabularnego należą do zupełnie innych kategorii (na przykład ściany i bohaterowie niezależni), to mogą dysponować szeregiem wspólnych właściwości, potencjalnie generujących zaskakujące niekiedy rezultaty. Analiza praktyk *speedrunu* pomaga ujawniać takie zależności, bo gracze należący do tej społeczności nie tylko uznają „tylko jedno prawo – prawo kodu", ale też aktywnie poszukują i wykorzystują owe „wycofane" czy „ukryte" właściwości obiektów w grze do osiągnięcia pożądanego przez siebie celu.

By zilustrować przedstawione w tym podrozdziale teorie, posłużę się przykładem *speedrunu* gry *Baldur's Gate II: Cienie Amn* (BioWare 2000). Jest to tytuł uznawany za jeden z kanonicznych w rozwoju gatunku *computer Role-Playing Game* (cRPG), łączący niebanalną fabułę ze stosunkowo skomplikowaną mechaniką rozgrywki, wzorowaną na drugiej edycji zasad systemu *Advanced Dungeons & Dragons*. Modelem będzie zapis aktualnego rekordu świata, który – stosując podział Scully-Blakera – można zakwalifikować do kategorii *deconstructive runs*.

Druga odsłona sagi *Wrót Baldura* w roli protagonisty obsadza bohaterkę lub bohatera, który z pomocą zebranej drużyny powoli odkrywa swoje boskie dziedzictwo i wiążące się z nim zobowiązania. Oś fabularną tytułu wspomaga następujące w pierwszych godzinach rozgrywki tajemnicze porwanie siostry stworzonej przez gracza postaci – wydarzenie to ma stanowić dodatkową motywację do podążania określoną narracyjną ścieżką, prowadzącą w konsekwencji do zaprojektowanego przez twórców epickiego finału. Kluczowa scena zilustrowana jest nieinteraktywnym przerywnikiem, w którym podczas długiej sekwencji starcia z *nemezis* głównego bohatera następuje zawiązanie właściwej akcji i przedstawienie nadrzędnego celu całej gry. W *speedrunie* siostra postaci gracza nigdy jednak nie zostaje uprowadzona. Jeśli gracz w odpowiednim momencie podejmie decyzję o wyrzuceniu siostry z drużyny, będzie ona starała się zainicjować dialog z protagonistą – jeśli jej się to powiedzie, przerywnik zostanie pominięty (pozwalając na zaoszczędzenie kilkudziesięciu cen-

nych sekund), porwanie nie nastąpi, a gracz nieodwracalnie zaburzy fabularny ciąg rozgrywki. W opisywanej sytuacji kluczowy jest brak możliwych do wyróżnienia warunków powodzenia takiego zabiegu – prawdopodobnie losowość tego faktu każdorazowo określa arbitralnie ustalana przez poszczególne obiekty w kodzie kolejność zadań egzekwowanych przez silnik gry. Dodatkową trudność stanowi przypisanie wszystkich wymienionych tu akcji (inicjacja dialogu, start *cutscenki*) do tego samego przedziału czasowego (jednej klatki) w grze[17]. Inną możliwością płynącą z wykorzystania opisywanego tu *glitcha* jest późniejsze posłużenie się postacią siostry jako swoistą „żywą tarczą". Brak przerywnika powoduje permanentne zapisanie w odpowiednim slocie (dedykowanym miejscu) pamięci wersji obiektu siostry z początkowego stadium gry. Ze względu na ryzyko nieodwracalnego utknięcia graczy w razie wczesnej śmierci „siostry 1", odpowiedzialny za nią obiekt cechuje między innymi nieśmiertelność. „Siostra 1", choć na poziomie interfejsu graficznego wyglądająca dokładnie tak samo jak „siostra 2", towarzyszy zatem drużynie na długo po tym, kiedy według „standardowych" reguł gry powinna zostać zastąpiona przez swoją drugą, pozbawioną specjalnych właściwości wersję.

Kolejną sytuacją ujawniania za pośrednictwem *speedrunu* ukrytych właściwości obiektów w grze jest wykorzystywanie przez postaci z drużyny przedmiotów, których nie mogłyby użyć według normalnych reguł mechaniki. Przykładem może być tu podmiana (podczas możliwej do zainicjowania z poziomu interfejsu pauzy) w ekwipunku mikstury na hełm, który pozwala na rzucenie potężnego czaru. Zamiana miejscami hełmu i mikstury po użyciu akcji jej „wypicia" powoduje po wznowieniu gry rzucenie zaklęcia – skrypty odpowiedzialne za kolejkowanie akcji odczytują bowiem polecenia z ostatnio aktywowanych w interfejsie obiektów bez sprawdzania, czy według nadrzędnych reguł mechaniki jest to dozwolone.

Podsumowując te przykłady, można zauważyć, że każdy *speedrun* proponuje nową rekonfigurację obiektów należących do zbioru-obiektu nazywanego „grą". Potwierdzają to relacje samych graczy komentujących w sieci nagrania wyjątkowo udanych prób: „Oglądanie tego *speedrunu* sprawia mi radość, ponieważ pokazuje coś, czego wcześniej nie widziałem. Wiele rzeczy zostaje tu odkryte"[18]. Należy przy tym zauważyć, że nie wszystkie interakcje między obiektami podczas *speedrunu* są możliwe do jednoznacznego określenia, a ich potencjalne właściwości zmieniają się z rozgrywki na rozgrywkę.

Spośród niewielu naukowców zajmujących się badaniem podstawy informatycznej gier ciekawą interpretację obiektów w nim obecnych zaproponowała Ea Christina Willumsen. Przy okazji analizy niezależnej gry *Passage* zwróciła ona uwagę na problemy z lokalizacją intencji autora w przypadku gry wideo, proponując w miejsce indywidualnych odczytań polegających na interpretacji rozgrywki analizę opartą na

[17] Por. http://speeddemosarchive.com/BaldursGate2.html (data dostępu: 7.08.2016).

[18] Komentarz użytkownika ZerglingWasteland pod nagraniem *speedrunu* gry *Baldur's Gate II: Cienie Amn* zamieszczonym w serwisie YouTube, https://www.youtube.com/watch?v=9kfBOWax2yk (data dostępu: 7.08.2016).

„zrozumieniu obiektu gry, który jest aktualizowany tylko, kiedy zrealizowany zostaje zapisany w niej kod”[19]. Gdyby zastosować podejście skandynawskiej badaczki do przypadku *speedrunu Baldur's Gate II*, można dojść do analogicznego w stosunku do *Passage* wniosku, że „kod źródłowy może zawierać w sobie procesy, które nie muszą być widoczne w jego wykonanych już przez maszynę partiach”[20]. Oznaczałoby to, że odczytanie rozgrywki jako tekstu mającego autora i z góry określony scenariusz fabularny jest tylko jedną z możliwych konfiguracji obiektów gry definiujących w dodatku swoje właściwości nie na podstawie działań gracza, ale doraźnych aktualizacji wzajemnych relacji. Podobna, holistyczna perspektywa analizy gry jako systemu znalazła już rozwinięcie u niektórych badaczy kręgu *game studies*. Kristine Jørgensen w swojej publikacji o interfejsach cyfrowych światów pisze na przykład o całym (widocznym dla gracza) świecie przedstawionym rozgrywki jako o specyficznym interfejsie formalnego systemu reguł, które definiują zasady danej gry[21].

Przypadek *speedrunu* wymaga odwołania do jeszcze innego elementu ontikologicznej teorii autora *The Democracy of Objects*. Istotne jest tu rozróżnienie na system i środowisko, którego za Luhmannem używa Bryant. Powiada on, że to system wyznacza swoje własne granice w obrębie danego środowiska[22]. System-gra zachowuje się podobnie: analogicznie do opisywanego przez Bryanta pierwotniaka[23], sama decyduje, czy wchodzi w relacje z innymi obiektami (na przykład ludźmi), nie zawsze są one bowiem ściśle wyznaczane przez matematyczne granice interakcji zawarte w kodzie źródłowym. Opieranie się *speedrunów* na trudnych do powtórzenia *glitchach*[24] ujawnia szczególną cechę gry rozumianej jako system-obiekt: jej opór przed wchodzeniem w relacje na warunkach niezgodnych z jej wewnętrzną strukturą organizacji. Poparciem tej obserwacji są setki, a niekiedy i tysiące, powtórzeń jednej sekwencji gry przez *speedrunnerów* chcących wywołać określone zachowanie innych obiektów w grze. Trudność polega też na konieczności uwzględnienia czynnika losowego – reakcje i powiązania między poszczególnymi bytami w grze bardzo często określa zakres prawdopodobieństwa ich wystąpienia, stąd pobicie rekordów szybkości (a więc osiągnięcie celu *speedrunu*) jest wynikową umiejętności manualnych, wiedzy graczy oraz losowego (przynajmniej z ludzkiego punktu widzenia) układu obiektów w czasie, w którym odbywa się *speedrun*.

[19] E.Ch. Willumsen, *Source Code and Formal Analysis: A Hermeneutic Reading of Passage*, Proceedings of 1st International Joint Conference of DiGRA and FDG, Dundee 2016, s. 4.

[20] *Ibidem*, s. 9.

[21] K. Jørgensen, *Gameworld Interfaces*, The MIT Press, Cambridge–London 2013, s. 4.

[22] L.R. Bryant, *The Democracy...*, *op. cit.*, s. 146.

[23] *Ibidem*, s. 147.

[24] W miejsce klasycznej definicji *glitcha* jako błędu w przepływie informacji wolę używać tego pojęcia w znaczeniu zaskakującego (trudnego, niemożliwego do przewidzenia) zjawiska wynikającego z doraźnego odczytania kodu gry.

3. Skrypty

Jeszcze ciekawsze – również z przyjętej tu perspektywy podejścia spekulatywnego – wydaje się odwrócenie opisanego dotychczas modelu badawczego. Zamiast śledzenia sposobów, w jaki kod warunkuje odbiór tekstów kultury, warto się przyjrzeć procesom powstawania i działania programów, które mają emulować zachowania graczy. Przedmiotem części dalszej analizy będą fragmenty kodu[25] pochodzącego ze skryptów opracowanych przez samych fanów, którzy motywowani są chęcią jak najlepszej optymalizacji swoich działań w grze. Omawiane tu skrypty mają charakter nieinwazyjny – nie zmieniają bezpośrednio kodu źródłowego produktu, lecz umożliwiają nadpisanie interfejsu taktylnego przez zautomatyzowany interfejs sterowany algorytmami. Co znamienne, ta szczególna praktyka dobrowolnej automatyzacji dokonuje się przez wykorzystanie składni języka AHK, który charakteryzuje się dużą kompaktowością kodu (poszczególne instrukcje zajmują potencjalnie mniej miejsca niż w językach takich jak Java czy C++) i łatwością do zaimplementowania w środowisku wielu systemów operacyjnych. Podany niżej przykład jest fanowską wersją bota (programu sterującego poczynaniami awatara gracza) w *Diablo III: Reaper of Souls* (Blizzard 2014), współpracującym ze środowiskiem Auto Hot Key, które umożliwia automatyzację danych przesyłanych od urządzeń wejścia (takich jak mysz czy klawiatura) na zmapowany przestrzennie interfejs gry.

```
CheckLoot(LootLegend=true ,LootRares=true ,Lootmagic=true ,Lootgems=true)
	{
	RanSleep(500,2000)
	Loop, 3
		{
		RanSleep(1000,2000)
		send, {ALT}
		Mousemove, 150,150
		If (LootLegend = true)
			{
			PixelSearch, Set_x, Set_Y, 200, 180, 600, 450, 0x00bb00, 8, fast ; set loot
			if !ErrorLevel
				{
				Click, %Set_x%, %Set_y%
				tooltip, Set found!, 0,0
				RanSleep(2000, 3000)
				}
			Sleep, 200
			Mousemove, 150,150
			PixelSearch, Legen_x, Legen_y, 200, 180, 600, 450, 0x2d5ba9, 2, fast ; legendary loot
			if !ErrorLevel
				{
				Click, %Legen_x%, %Legen_y%
				tooltip, legendary found!, 0,0
				RanSleep(2000, 3000)
				}
			}
```

[25] Całość skryptu dostępna jest pod adresem: https://autohotkey.com/board/topic/83114-diablo-iii-sarkoth-bot-script-800x600/ (data dostępu: 7.08.2016).

Powyższy fragment odpowiada za działanie pętli, która sprawdza, do jakiej kategorii należą znajdowane w grze przedmioty. Można wyróżnić w nim parę interesujących z punktu widzenia ontikologii faktów: na przykład hierarchiczny układ powiązanych z konkretnymi instrukcjami obiektów (wycinków umieszczonych w nawiasach {}) jest w istocie jedynie wskazówką odnoszącą się do realizacji poszczególnych zadań w czasie trwania gry. Rzeczywista kolejność wykonywania poleceń warunkowana jest przez czynnik losowy, jakim jest pojawianie się obiektów (w tym przypadku przedmiotów) generowanych przez silnik gry. Egzekucja skryptu odbywa się na zasadzie współpracy jego fragmentów – jeśli znaleziony przedmiot zakwalifikowany zostanie jako legendarny, uruchomiona zostaje dodatkowa procedura kategoryzująca go jako element zestawu bądź „zwykły" złoty przedmiot.

W składni kodu szczególną uwagę zwraca słowo *Sleep*, którego wartość odpowiada okresowi bezruchu awatara. Stanowi to szczególne zabezpieczenie przed wykryciem działania skryptu przez działające w imieniu Blizzarda – właściciela franczyzy *Diablo* – programy na bieżąco monitujące działalność szczególnie aktywnych obiektów w grze (domyślnie służą one zbieraniu informacji o strategiach rozgrywki stosowanych przez samych graczy). W przypadku wykrycia zbyt efektywnego wchodzenia w interakcje z innymi obiektami program zwany *Wardenem* (Strażnikiem) wysyła do serwerów informację o potencjalnie podejrzanym elemencie systemu, co może doprowadzić do zawieszenia konta użytkownika korzystającego z zabronionych przez Blizzarda skryptów. Choć implementacje AHK – jak już wspominałem – nie ingerują w kod źródłowy gry, właściciele serwerów *Diablo* ewidentnie nie życzą sobie aktantów podszywających się pod aktorów ludzkich.

Warto w tym miejscu dla jasności wprowadzić rozróżnienie na programy sterujące awatarami gracza w świecie przedstawionym gry (nazywane często botami) oraz programy mające naśladować „idealne" zachowania graczy w obrębie całego środowiska gry – oprócz samej rozgrywki, również na poziomie zarządzania interfejsem. Różnica wynika z zakorzenienia w konkretnym, wizualnie wyróżnionym obiekcie (awatarze) w przypadku botów, a emulowaniu na poziomie interfejsu działań samej instancji gracza (np. poprzez automatyzację ruchów muszki odpowiedzialnych za wybór opcji w menu itp.). Prezentowany wyżej fragment przyporządkować można do pierwszej z wymienionych tu kategorii, choć równie interesującym obiektem badań byłby skrypt automatyzujący działania prowadzone w ramach funkcjonującego przez pierwsze miesiące po premierze *Diablo III* domu aukcyjnego[26]. Jako ciekawostkę można tu dodać, że analogiczne do użytego w przypadku gry Blizzarda skrypty analizują i nadzorują również przebieg transakcji giełdowych przeprowadzanych bynajmniej nie w realiach cyfrowych światów, lecz globalnego obiegu kapitału. Zasady działania i charakter warunków przeprowadzania transakcji kupna/sprzedaży warunkowane są przez bardzo podobne, łatwe do modyfikacji i zaimplementowania

[26] Jego fragmenty wciąż dostępne są pod adresem: http://diablo3story.blogspot.com/2014/07/a-diablo-3-story.html (data dostępu: 7.08.2016).

zmienne, różni się tylko przedmiot operacji: zamiast magicznych artefaktów, spekulacjom podlegają papiery wartościowe oraz akcje notowanych na giełdzie firm. W omawianym tu przypadku „klasycznego" bota (emulującego działania gracza w środowisku gry, nie zaś interfejsie serwisu aukcyjnego) istotny jest fakt, że sam gracz też jest obiektem – nie tylko jako osoba, na której akcje domyślnie zorientowana jest gra, ale także jako obiekt posiadający swoje konkretne umiejscowienie w kodzie. Co znamienne, miejsce awatara zastępują tu koordynaty pozycji myszy i polecenia wykonania poszczególnych kliknięć, oznaczane w kodzie jako *mouse-move* z dodaniem określonych wartości. Domyślny obiekt „gracza" zostaje zatem za pomocą skryptu pomyślnie rozbity (lub, z innej perspektywy, zdekonstruowany) na szereg precyzyjnie wyliczonych współrzędnych, które uruchamiane w odpowiednim czasie wystarczą do zemulowania jego cyfrowej obecności w systemie gry.

4. Podsumowanie

Przedstawiona w niniejszym artykule analiza pozwala na postawienie kilku końcowych tez, które – jak mniemam – uzasadniają przydatność aplikacji ontikologii oraz opartej na podejściu obiektowym analizy kodu gry i mogących jej towarzyszyć paratekstualnych skryptów. Grę wideo tworzą równorzędne obiekty, zarówno te obecne w kodzie (np. instrukcje czy operatory) i przezeń definiowane (byty w przestrzeni gry: awatary, broń, elementy cyfrowego środowiska), jak i pozostające poza nim (np. gracze, skrypty automatyzujące informacje z urządzeń wejścia). Każda rozgrywka traktowana jako obiekt-system bliska jest pojęciu wydarzenia w rozumieniu Badiou – następuje w niej rekonfiguracja budujących ją obiektów, które porozumiewają się między sobą, zachowując jednak rezerwuar niewykorzystanych w danym wycinku czasu potencjalności. Nawet proste instrukcje mogą przekształcić się w *glitche*, a omawiane przeze mnie w artykule przykłady grania emergentnego (*speedrun*) i wykorzystanie zewnętrznych skryptów mogą być interpretowane jako sytuacje inicjujące konfigurację mającą potencjał wywrotowy, wynikający z partykularnych połączeń i translacji wybranych przez obiekty porcji informacji.

Bibliografia

Abriszewski K., *Poznanie, zbiorowość, polityka. Analiza teorii aktora-sieci Bruno Latoura*, Universitas, Kraków 2012.

Badiou A., *Byt i zdarzenie*, przeł. P. Pieniążek, Wydawnictwo Uniwersytetu Jagiellońskiego, Kraków 2010.

Bogost I., *Alien Phenomenology, or What It's Like to Be a Thing*, University of Minnesota Press, Minneapolis–London 2012 [wersja e-book].

Bogost I., *Persuasive Games: The Expressive Power of Videogames*, The MIT Press, Cambridge–London 2010.

Bogost I., *Unit Operations: An Approach to Videogame Criticism*, The MIT Press, Cambridge, London 2006.
Bryant L.R., *The Democracy of Objects*, Open Humanities Press, Ann Arbor 2011.
Goffey A., *Algorithm*, w: M. Fuller (ed.), *Software Studies: A Lexicon*, The MIT Press, Cambridge–London 2008.
Jørgensen K., *Gameworld Interfaces*, The MIT Press, Cambridge–London 2013.
Latour B., *Reassembling the Social: An Introduction to Actor-Network-Theory*, Oxford University Press, New York 2005.
Scully-Blaker R., *A Practiced Practice: Speedrunning through Space with de Certeau and Virilio*, „Game Studies" 2014, vol. 14, iss. 1.
Stroustrup B., *Programowanie. Teoria i praktyka z wykorzystaniem C++*, Helion, Gliwice 2010.
Tanenbaum K., Tanenbaum J., *Commitment to Meaning: A Reframing of Agency in Games*, „Digital Arts and Culture" 2009, no. 12–15.
Willumsen E.Ch., *Source Code and Formal Analysis: A Hermeneutic Reading of Passage*, Proceedings of 1st International Joint Conference of DiGRA and FDG, Dundee 2016.