Fides, Ratio et Patria. Studia Toruńskie 3/2015
ISSN 2392-1463

Ilona Nowosad*
Wyższa Szkoła Kultury Społecznej i Medialnej w Toruniu

Graficzne wizualizacje dźwięku i zbiorów danych

Wprowadzenie

Stale rosnący nacisk na media zmieniające się w czasie i interakcję z kodem w czasie rzeczywistym oddziałuje na języki i środowiska programowania, a przy ich użyciu z udziałem najnowszych dostępnych technologii wpływa na wyznaczanie różnorodnych szlaków w tworzeniu wirtualnej rzeczywistości. Celem artykułu jest przybliżenie wybranych podejść w tym zakresie, szczególnie tych nacechowanych interaktywnością oraz działaniem w czasie rzeczywistym, bowiem „interaktywność jest najistotniejszą, jak można zasadnie sądzić, właściwością kształtującej się współcześnie cyberkultury (termin obecnie powszechnie stosowany dla określenia kultury ery komputera). Jest ona również jedną z kilku najważniejszych cech rzeczywistości wirtualnej. Następna właściwość rzeczywistości wirtualnej, ściśle związana z poprzednią, to fakt, iż interakcja odbywa się w czasie rzeczywistym" [Kluszczyński, dostęp: 10 VIII 2015].

Live coding

Live coding jest nową praktyką, zwaną też *programowaniem w locie*, *programowaniem w czasie* lub po prostu *programowaniem na żywo*. Głównym wyznacznikiem programowania na żywo jest pisanie kodu programu lub jego części w trakcie jego wykonywania. Taka praktyka programowania jest też określana mianem programowania interaktywnego bądź konwersacyjnego i jest szczególnie użyteczna w przypadkach, gdy problem, który ma być rozwiązany, nie jest z góry jasno wyspecyfikowany.

Od 2003 r. obserwuje się intensywny rozwój grup badawczych skupionych wokół *live coding*, obejmujących muzyków, choreografów, artystów wizualnych, psychologów, techników i inżynierów oprogramowania. Rosnąca społeczność

* **Ilona Nowosad** – doktor nauk matematycznych, specjalność informatyka. Główne zainteresowania naukowe: systemy informatyczne, algorytmy i programowanie generatywne, procesy poznawcze i percepcyjne, interaktywne aplikacje multimedialne, rzeczywistość wirtualna.

badaczy *live coding* zainteresowanych tworzeniem muzyki i wizualizacji na żywo z uwzględnieniem informacji zwrotnej z systemu multimedialnego lub z danych wyjściowych wyznacza nowe kierunki badań i eksperymentów w dziedzinie projektowania narzędzi i języków programowania na żywo. Organizowane są festiwale i konferencje, których ideą są próby znalezienia analogii i relacji między tym, co dzieje się w muzyce, a tym, co można zobaczyć w obrazie, animacji, oraz uzyskania efektu synergii poprzez umiejętne połączenie świata muzyki ze światem obrazu.

Technika programowania na żywo otwiera wiele możliwości dla użytkowników i już znalazła szereg rozmaitych zastosowań. Dotychczas największe korzyści z technik programowania konwersacyjnego dostrzega się w obszarze projektowania dźwięku i kompozycji algorytmicznych, które zyskały na swobodzie i dokładności dzięki implementacji w 2002 r. języka programowania audio o nazwie ChucK, będącego obiektowo zorientowaną pochodną języka C [Wang, 2008] Język ChucK wprowadził nowy model programowania współbieżnego oparty na czasie, umożliwiający dodawanie, usuwanie i modyfikowanie kodu podczas działania programu bez jego zatrzymywania i ponownej kompilacji. Semantyka ChucK wniosła precyzyjną kontrolę w czasie, co zapoczątkowało falę projektów i eksperymentów w interaktywnym pisaniu skryptów, a zarazem zainspirowało do konstruowania nowych języków programowania. ChucK w połączeniu z Audicle, specjalizowanym środowiskiem graficznym, daje możliwość programowania audio na żywo z jednoczesną wizualizacją w czasie rzeczywistym [Wang, Cook, 2004]

Programowanie na żywo aktualnie koncentruje się na tworzeniu mediów cyfrowych opartych na obrazie i dźwięku, często zespolonych intermedialnie oraz prezentowanych przede wszystkim w czasie rzeczywistym. Techniki *live coding* są wykorzystywane w produkcji audiowizualnych elementów instalacji artystycznych, gier komputerowych, a także w postprodukcji filmowej.

W ostatnich latach rodzi się szereg inicjatyw, projektów i grup badawczych zajmujących się interdyscyplinarnym badaniem *live coding*, między innymi TOPLAP [http://toplap.org/, dostęp: 10 VIII 2015], Live Coding Research Network [http://www.livecodenetwork.org/, dostęp: 10 VIII 2015], których celem jest integracja osób i środowisk zajmujących się zgłębianiem już znanych i poszukiwaniami nowych ścieżek dla *live coding*, rozwój programowania na żywo – szczególnie w dziedzinie kreatywnego wyrażania idei przy użyciu komputerów, oraz nowych koncepcji w projektowaniu języków programowania na żywo i dedykowanych zintegrowanych środowisk programistycznych.

Już w latach siedemdziesiątych budowano systemy, które w późniejszych czasach zostały sklasyfikowane jako *live coding*. Thomas DeFanti w 1974 r. opracował autorski system GRASS („The Graphic Symbiosis System") [DeFanti, Thomas,

1976], który określa się mianem interaktywnego interpretowanego języka programowania. GRASS został zaprojektowany z myślą o definiowaniu i manipulowaniu grafiką wektorową. Jedną z właściwości systemu była możliwość wizualizowania muzyki. Grafika w GRASS była programowana na żywo w synchronii z muzyką i wyświetlana wraz z kodem nakładanym na wynikowe wideo. W skład systemu na poziomie sprzętowym wchodził standardowy na tamte czasy komputer PDP-11/45 z wyświetlaczem Vector General 3DR i klawiaturą VT05, tablet, trzydzieści urządzeń stanowiących wejścia analogowe (dialery, potencjometry, joysticki itp.) oraz kilka kanałów wyjść analogowych, które miały za zadanie sterować procesorem obrazu.

W GRASS posługiwano się dwoma podstawowymi typami prymitywów: poleceniami i obrazami. Polecenia były przechowywane w postaci kodów ASCII, natomiast obrazy były skompilowane do kodu binarnego akceptowanego przez procesor bezpośredniego dostępu do pamięci wyświetlacza Vector General. Urządzenia wejść analogowych podlegały w czasie rzeczywistym stałej kontroli użytkownika, dzięki czemu sprawiały, że system stał się użyteczny jako instrument w pokazach na żywo. Animacje często obejmowały kilka niezależnych elementów zdarzających się symultanicznie. Własność tę zawdzięczały równoległej obsłudze makr, co skutkowało wykonywaniem poleceń z przeplotem. Oryginalne wideo z występu na żywo Thomasa DeFanti, Daniela Sandina i Mimi Shevitza z lat siedemdziesiątych z użyciem systemu GRASS można podziwiać w internecie [https://www.youtube.com/watch?t=302&v=hw9kY85DkfE; Spiral 5 PTL - Dan Sandin, TomDeFanti, and Mimi Shevitz, Live recording of performance,1979, dostęp 10 VIII 2015]

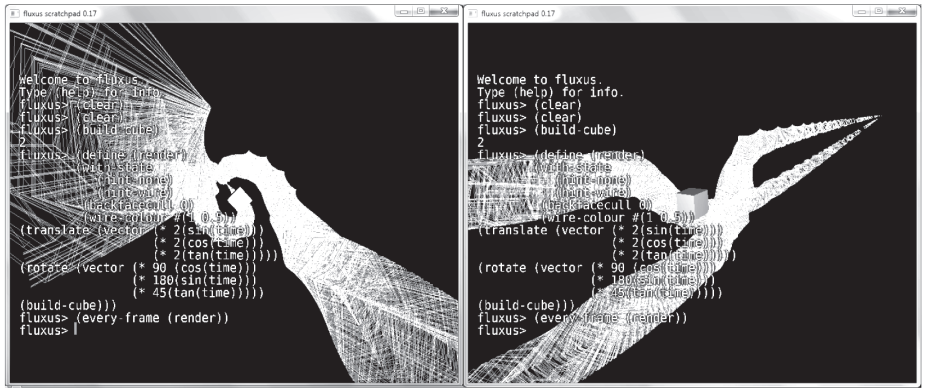
Współcześnie wśród najszerzej stosowanych do celów *live coding* narzędzi i środowisk programistycznych wymienić można: Fluxus, COLT, SuperCollider, Quoth oraz Extempore. Narzędzia te działają według koncepcji, która eliminuje dotychczasową dychotomię towarzyszącą programowaniu w tradycyjnym, statycznym ujęciu, w którym to wyraźnie rozdziela się narzędzie od wytworu, a obszary programu, procesu i zadania są od siebie odseparowane. Program w ujęciu tradycyjnym jest statyczną specyfikacją założeń projektowych, krótko zwaną kodem, proces jest urzeczowieniem programu na konkretnym komputerze, zaś zadanie jest rozumiane jako celowe oddziaływanie na świat rzeczywisty.

W tym kontekście kod programu przed uruchomieniem musiał być całościowo opracowany, a program zatrzymany, zanim możliwe było edytowanie jego kodu. Natomiast naturą algorytmów generatywnych, będących fundamentem *live coding*, jest to, że w trakcie działania, tj. wykonywania ich kodu, są one poddawane modyfikacjom w czasie rzeczywistym, a zmodyfikowany fragment kodu jest interpretowany zaraz po jego napisaniu, jednakże bez zatrzymywania programu.

Ilustrację tego procesu można znaleźć w internecie [https://vimeo.com/51993089; Codelife - glsl live-coding editor test #5, h3xl3r, 2013,dostęp: 10 VIII 2015]. Zatem technika interaktywnego pisania skryptów zapewnia programom ciągłe działanie, również gdy skrypt jest edytowany. W ten sposób programowanie przestaje być odrębną czynnością specyfikowania produktu w oderwaniu od czasu jego uruchomienia, co było charakterystyczne w klasycznym podejściu z użyciem kompilatorów, analizerów, linkerów itp. W konsekwencji rodzi to konieczność przebudowy infrastruktury środowiska uruchomieniowego i programistycznego. Jest to zjawisko znamienne w ostatnich latach i wiąże się z nowym paradygmatem wszechobecnej interaktywności i „bezszwowego wszywania technologii w tkaninę codzienności" [Weiser, dostęp: 10 VIII 2015]. Wobec nowych trendów związanych z intensyfikacją nowoczesnych technologii penetrujących każdą dziedzinę życia, z pogłębiającym się fenomenem *ubicomp* (*ubiquitous computing*), z modną i konieczną hipermedialnością, z wielobodźcowym doświadczaniem artefaktów w chwili ich powstawania i trwania, z wdzierającą się w świat materialny wirtualną i rozszerzoną rzeczywistością wiele społeczności oraz niezależnych badaczy podejmuje trud opracowania i ufundowania na platformie komputerowej nowych ścieżek i sposobów zaspokajania odbiorców na polu tego typu doświadczeń, doznań, sposobów komunikacji, w szczególności doświadczeń multimedialnych w przestrzeni hybrydycznej. Jedno z głównych pytań, jakie rodzi się w tym kontekście, dotyczy tego, jak dalece *live coding* może wzbogacić technologiczne zaangażowanie w szerokiej kulturze. Częściową odpowiedź znaleźć można w poniższej prezentacji możliwości oprogramowania Fluxus[1].

Pierwsza w historii audiowizualna projekcja architektoniczna została zrealizowana przy użyciu między innymi programu Fluxus, co stało się na festiwalu PixelACHE 2005 w Helsinkach. Fluxus jest wieloplatformowym środowiskiem (Windows, Linux, OSX, PS2, Android) zaprojektowanym do użytku w czasie rzeczywistym. Program umożliwia szybkie tworzenie dźwięku i animacji na żywo oraz pozwala zmieniać je w sposób ciągły i elastyczny. Fluxus należy do środowisk grafiki 3D wykorzystywanych do celów szybkiego prototypowania, gier komputerowych oraz edukacyjnych. Bazujący głównie na deklaratywnym języku programowania Scheme, uznawany jest za rozszerzenie języka programowania Racket z rodziny Lisp/Scheme. Posiada charakterystyczny dla *live coding* interfejs, w którym jednocześnie jest prezentowany kod programu i jego wizualny output.

[1] Nazwa *Fluxus* (z łac. *płynący*) nawiązuje do nazwy międzynarodowego ruchu artystycznego w sztuce XX wieku.



Rys.1. Kod i wizualizacja w interfejsie programu Fluxus (opracowanie własne)

Fluxus, oprócz wbudowanego interpretera języka Scheme, zawiera moduły rozszerzające Scheme o polecenia typowe dla grafiki 3D (np. scale, rotate, translate), moduły odpowiedzialne za przetwarzanie strumieni wejść audio oraz dodatkowe moduły służące do kontroli sygnału wideo i dostępu do kamer na żywo (fluxus-engine, fluxus-audio, fluxus-midi, fluxus-osc, fluxus-video). Moduły te mogą być załadowane do dowolnego interpretera MzScheme i uruchomione w dowolnym kontekście OpenGL. Scena 3D jest definiowana za pomocą standardowych narzędzi obecnych w typowych środowiskach grafiki 3D, takich jak: materiał i oświetlenie, teksturowanie, wieloteksturowanie, mipmapping, cubemapping, cieniowanie. Bazą wszystkich informacji o prymitywach budujących scenę są specjalne stałe struktury *pdata (Primitive Data Arrays)*, będące tablicami o stałym rozmiarze. Odgrywają one ważną rolę w zarządzaniu obiektami typu NURBS, systemami cząsteczek, wstęgami czy deformacjami obiektów w czasie modyfikacji sceny 3D na żywo. Dzięki bogatej implementacji powierzchni i brył, wliczając w to też reprezentacje powierzchni wyższego rzędu, takie jak prymitywy *blobby*, środowisko Fluxus umożliwia produkcję bardzo wydajnych i urozmaiconych wizualizacji.

Nieodłącznym elementem środowiska programistycznego współpracującego z dźwiękiem na żywo jest protokół OSC (Open Sound Protocol), który służy do komunikacji i wymiany danych w czasie rzeczywistym pomiędzy instrumentami, syntezatorami, kontrolerami MIDI, komputerami i innymi urządzenia multimedialnymi. Przewagą tego protokołu jest możliwość przesyłania informacji przez sieć komputerową typu LAN oraz internet, a także szybkość transferu danych, która przewyższa uzyskiwaną w powszechnie stosowanym dotąd transferze MIDI. Dzięki komunikacji sieciowej możliwe stało się zestawienie systemu multimedialnego o dość skomplikowanej architekturze, złożonego z różnych urządzeń współpracujących ze sobą we wspólnej przestrzeni interaktywnej. Fluxus daje możliwość odbioru i wysyłania wiadomości poprzez OSC. Dane sczytywane

z portu, pochodzące z podłączonych urządzeń, przekształcane są w parametry do graficznej manipulacji obiektami, animacji i zmian wizualizacji, na przykład ustalenia nowej pozycji, przesunięcia lub rotacji obiektu w scenie 3D.

Działanie Fluxus w połączeniu z wejściem dźwiękowym polega na przetwarzaniu wartości rzeczywistych napływających ze źródła dźwięku, którym może być wejście z karty dźwiękowej lub inna aplikacja generująca dźwięk. Wczytane przy użyciu funkcji *gh* (*get harmonic*) wartości zmiennopozycyjne wykorzystuje się do generowania parametrów, które służą do manipulacji animacją.

Pierwotnym zastosowaniem środowiska Fluxus były aplikacje VJ mapujące dźwięk w światło. Z czasem *live coding* w tym środowisku dostarczył implementację silnika 3D gier komputerowych, który ma tę cechę, że podlega zmianom w trakcie działania gry. Podczas wykonywania skryptu gry w tle w wyniku modyfikacji kodu scena 3D podlega graficznej rozbudowie, podobnie jak ścieżka dźwiękowa, która zmienia się odpowiednio do zmian w kodzie skryptu. Użytkownicy gry mają możliwość interaktywnego kontrolowania jej przebiegu, modyfikując na żywo fragmenty skryptu poprzez tworzenie obiektów i sterowanie nimi. W konsekwencji skrypt w sposób ciągły zmienia się pod wpływem zmian wprowadzanych w środowisku przez użytkownika. Fluxus posłużył także do zaprogramowania prototypu gry w rozszerzonej rzeczywistości. W interaktywnej instalacji muzycznej zaprojektowanej do użytku w klubach i na imprezach zastosowano specjalne markery przytwierdzone do dna naczyń z napojami. Markery w czasie spożywania napoju generowały przestrzenne wizualizacje instrumentów muzycznych, a reagując na zmieniającą się ilość napoju w naczyniu, również dźwięki instrumentów o wysokości odpowiedniej do aktualnego poziomu napoju.

FRP – *Functional Reactive Programming*

Języki programowania z kategorii FRP dają możliwość budowania sceny reagującej na zachowanie użytkownika w czasie poprzez konwersję zachowania na określone zdarzenia. Zachowania i zdarzenia są dwoma fundamentalnymi pojęciami występującymi w FRP. Zachowania są reaktywnymi wartościami zróżnicowanymi w czasie, podczas gdy zdarzenia tworzą sekwencję dowolnie złożonych warunków o charakterze *time-stamped*, czyli pojawiających się w pojedynczym, dyskretnym punkcie w czasie, bez trwania, mogących wnosić dodatkową informację z zewnątrz. Zarówno zachowania, jak i zdarzenia charakteryzują się polimorficznym typem danych, wynikającym chociażby z rozmaitości zmieniających się w czasie mediów (obrazów, wideo, dźwięku, geometrii 3D) oraz specyfiki niesionej informacji, jaką systemy mają za zadanie przechwytywać (kolory, punkty, wektory, liczby, transformacje). Reaktywność należy rozumieć jako sposób, w który zachowanie zmienia się w odpowiedzi na zdarzenie. Dane wejściowe dla

systemów reaktywnych rzadko są z góry znane, jednak napływają w sposób ciągły od momentu uruchomienia programu przez cały czas jego działania. Zadaniem systemu reaktywnego jest wytwarzanie danych wyjściowych w odpowiedzi na wejściowe impulsy w chwili ich zarejestrowania, przeplatając wejście i wyjście. FRP znalazło szereg użytecznych zastosowań w wielu dziedzinach, wliczając grafikę i animację, robotykę, wizualizacje komputerowe czy interfejsy graficzne użytkownika. Wysokie wymagania stawiane systemom reaktywnym wynikają z ich interaktywnego charakteru działania i przetwarzania sygnałów w czasie rzeczywistym. Animacja na żywo powinna wyświetlać się na tyle szybko, by zapewnić płynny ruch obrazu; robot powinien szybko i właściwie reagować na zdarzenia pochodzące ze środowiska, interfejs graficzny użytkownika musi natychmiast dać odpowiedź na komendy użytkownika.

Programowanie reaktywnego środowiska graficznego, jakim jest na przykład audiowizualny interfejs graficzny użytkownika czy środowisko gry komputerowej, koncentruje się na opisie sceny wraz z zachowaniami, które modyfikują ją w czasie. Różnica polega głównie na tym, że w klasycznym podejściu zmuszeni jesteśmy odrębnie opisać scenę, a następnie ją wyanimować. Stosując języki programowania FRP, mamy możliwość opisania sceny z wbudowanymi zachowaniami, których wartości bezpośrednio zależą od czasu. Istnieje wiele sposobów tworzenia i manipulowania zachowaniami, które mogą również reprezentować zmieniającą się informację napływającą z zewnętrznych źródeł danych. Dzięki wbudowanym zdarzeniom zyskuje się kontrolę nad zachowaniem i czasem życia istniejących obiektów sceny oraz tworzeniem nowych obiektów, w tym również możliwość wyzwalania efektownego zjawiska ciągłego namnażania obiektów z jednoczesnym unicestwianiem powstałych wcześniej, naturalnie, by nie dopuścić do spowolnienia działania systemu przy narastającej złożoności obliczeniowej. Manipulacja zachowaniami sprawia, że animacja lub wizualizacja nabiera reaktywnego charakteru.

W 1997 r. opublikowano pierwsze prace z zakresu FRP, w tym też dotyczącą interaktywnych multimedialnych animacji obejmujących audio, wideo, zdjęcia oraz grafiki 2D i 3D [Conal, Hudak, 1997]. Jednym z owoców badań w dziedzinie FRP była implementacja w języku Haskell biblioteki Fran[2] (*Functional Reactive Animation*) służącej do komponowania tychże animacji. Kluczowymi w implementacji Fran są właśnie zachowania i zdarzenia. Zachowania w tym przypadku są wartościami wyliczanymi w ciągłym czasie, z semantycznego punktu widzenia jest to dziedzina czasu ciągłego. Wszelkie zmiany w czasie w sposób automatyczny propagują aktualizację zależnych zachowań, a tym samym zawartość prezentowanej animacji. Dziedzina czasu ciągłego jest szczególnie użyteczna w aplikacjach, które modelują lub wchodzą w interakcję z fizycznym światem.

[2] Fran jest zaliczany do DSL (*Domain Specific Language*) z uwagi na zagnieżdżenie w języku Haskell.

Natomiast zdarzenia generują wartości tylko w określonych momentach, gdyż z natury ich zaistnienie jest zjawiskiem dyskretnym; semantycznie mamy tu do czynienia z dziedziną czasu dyskretnego. Zdarzenia mogą też dostarczać dodatkowych informacji, na przykład o aktualnej pozycji myszy podczas jej ruchu czy wartości przekroczenia ustalonego progu sygnału audio. W zależności od rodzaju zdarzenia i informacji, jaką ono niesie, wizualizacja będzie podlegać odpowiednim zmianom. To wszystko sprawia, że w języku Fran można tworzyć ciągłe animacje zdolne reagować na impulsy zarówno z wewnątrz, jak i z zewnątrz systemu, i wykorzystywać ten język jako narzędzie do wizualizacji dźwięku.

Niestandardowe oprogramowanie audioreaktywne

Istnieje wiele niestandardowych środowisk programistycznych, opracowanych przez artystów informatyków lub przy współpracy artystów z informatykami, których celem jest generowanie form audiowizualnych. Jednak wszyscy korzystają z kilku ustandaryzowanych metod tworzenia wizualnych reprezentacji dźwięku, które w pewnym przybliżeniu można sklasyfikować jako reprezentacja symboliczna, analiza spektralna oraz wskaźniki średniego poziomu częstotliwości. Reprezentacja symboliczna wykorzystuje ustalony zbiór symboli, jak na przykład muzyczny zapis nutowy, notacja graficzna itp. Do wskaźników poziomu należą miernik wartości szczytowej, decybele, wskaźnik wysterowania. Analiza spektralna zazwyczaj korzysta z pomocniczych urządzeń (spektrograf, sonogram) oraz narzędzi matematycznych w postaci transformat Fouriera w celu dekompozycji dźwięku na pewną liczbę sinusoid i ich graficznego wykreślenia.

W 1985 r. Stephen Malinowski, kompozytor i inżynier oprogramowania, przerzucił most pomiędzy nieruchomym zapisem nutowym utworu muzycznego a wizualną warstwą percepcji dźwięków. Jego projekt o nazwie *The Music Animation Machine* (*MAM Player/Viewer*) [http://www.musanim.com/player/, dostęp: 10 VIII 2015], zainspirowany pracami muzyka wizualnego Oscara Fischingera, wykorzystał właściwości formatu plików MIDI i w efekcie stał się narzędziem do odtwarzania i wizualizowania utworów zapisanych w tym formacie. W postaci różnych form graficznych – grafów liniowych, kolorowych kół i innych barwnych kształtów – zaprezentował podstawowe elementy występujące w muzycznych kompozycjach, takie jak: interwały, tonacje, trójdźwięki, sekwencje, harmonie. W implementacji MAM pod swoistymi nazwami: LINES, BALLS, WEDGES, LATTICE, COMPASS, SHAPES, WHEEL, DYAD, YARN, twórca zakodował ich reprezentacje graficzne. W oparciu o przywołaną listę wizualnych reprezentacji oprogramowanie MAM generuje odpowiednio do dźwięków różnorodne formy graficzne w sposób zsynchronizowany z napływającymi danymi wejściowymi z plików MIDI. Oprogramowanie jest również dostosowane do wizualizacji dźwię-

ku w czasie rzeczywistym w odpowiedzi na dane wejściowe MIDI przesyłane na żywo. Barwne i urozmaicone kształty pojawiają się i przesuwają horyzontalnie z prawej do lewej strony okna programu zgodnie z tempem utworu oraz linią czasu; przeszłość jest z lewej strony okna, a przyszłość – z prawej, dając przestrzenno-czasowy wgląd w muzykę. Tony są rozłożone w osi pionowej – wyższe nuty są wizualizowane w stronę górnej części okna programu, zaś niższe ku dołowi; każdemu instrumentowi występującemu w zapisie nutowym przypisany jest inny kształt w wizualizacji. Warto nadmienić, że program umożliwia wielobarwną wizualizację 2D, a także – dzięki specjalnym technikom wykorzystującym właściwości chromostereoskopii – w wizualizacji możliwe jest uzyskanie wrażenia głębi 3D stanowiącej metaforę rozmieszczenia dźwięków w przestrzeni. Program daje swoistą transkrypcję notacji nutowej, która wymaga wielu lat studiów i praktyki, niezrozumiałej dla wielu odbiorców dzieł muzycznych. Wizualizacja utworów pozwala odbiorcom, którzy nie znają języka muzyki, w sposób intuicyjny rozumieć taki zapis i niesie głębsze doświadczenie poznawcze w odbiorze dzieła. W tym znaczeniu program został uznany za cenne narzędzie edukacyjne w dziedzinie muzyki, a w ostatnich latach jest intensywnie eksploatowany podczas koncertów oraz prezentowany na konferencjach i wystawach na całym świecie. Jedną z prezentacji na konferencji TEDx w Amsterdamie w 2013 r. poprowadzoną przez Étienne'a Abelina, muzyka i dyrygenta współpracującego z twórcą oprogramowania, oraz fragmenty koncertu z festiwalu *Sounds of Childhood*, który odbył się w 2013 r. w Holon w Izraelu, można obejrzeć w internecie[https://www.youtube.com/watch?t=492&v=A-OsEyne8eQ; Colouring music with the Music Animation Machine: Etienne Abelin at TEDx Amsterdam, 2013 dostęp: 10 VIII 2015; https://www.youtube.com/watch?t=160&v=KwcfzORcZmA; Back To The Future, Music Animation Machine Children's Concert in Holon, Israel, 2013 dostęp: 10 VIII 2015].

Narzędzie MAM zostało opracowane wyłącznie na platformę Windows, a ostatnią aktualizację przeszło 19 sierpnia 2006 r., jednak wciąż jest wykorzystywane do rozmaitych projektów audiowizualnych, czego przykładem jest interaktywna aplikacja mobilna *Biophilia* [http://biophiliaeducational.org/, dostęp: 10 VIII 2015] znanej piosenkarki i kompozytorki muzyki pop Björk we współpracy z deweloperem aplikacji i artystą interaktywnym Scottem Snibbem, opracowana na urządzenia z systemem iOS i Android [https://www.youtube.com/watch?t=37&v=dikvJM__zA4/ Biophilia – kompilacja fragmentów aplikacji mobilnej, 2011 dostęp: 10 VIII 2015] Aplikacja została pomyślana między innymi jako interaktywna wizualizacja do albumu muzycznego, którego każdy utwór stał się w aplikacji swoistą trójwymiarową galaktyczną grą komputerową. *Biophilia* jest zarazem projektem edukacyjnym angażującym naukowców, artystów, nauczycieli, studentów na każdym szczeblu naukowym, który w innowacyjny sposób łączy muzykę z nowymi technologiami informatycznymi, informacyjnymi i naukami przyrodniczymi.

Systemy wizualizacji dźwięku w czasie rzeczywistym powstają w oparciu o różne języki programowania i środowiska graficzne oraz współpracują z różnorodnymi urządzeniami peryferyjnymi i przetwornikami sygnałów. Emanacja kultury komputerowej sprawiła, że technologie cyfrowe stały się instrumentami w twórczości wielu artystów sztuk audiowizualnych czy hipermedialnych.

Stworzone przez artystów badaczy działających pod pseudonimami Abstract Birds oraz Quayola oprogramowanie o nazwie Partitura jest przykładem systemu generującego w czasie rzeczywistym grafikę 3D, która wizualizuje dźwięk. Sama nazwa oprogramowania podkreśla jego charakter, co było głównym zamysłem projektu. Oprogramowanie zostało napisane głównie przy użyciu narzędzia VVVV [http://vvvv.org/, dostęp: 10 VIII 2015], przy współpracy z dedykowaną zmodyfikowaną wtyczką do analizy audio Max4Live. Oprogramowanie VVVV jest hybrydowym wizualno-tekstowym środowiskiem do tworzenia aplikacji medialnych i programowania na żywo, które umożliwia kontrolę audio i grafiki w czasie rzeczywistym rozmaitych fizycznych urządzeń, takich jak odtwarzacze DVD, kontrolery MIDI czy sensory odległości, oraz symultaniczną interakcję z wieloma użytkownikami. Główną charakterystyką systemu jest jego horyzontalna struktura liniowa nawiązująca do poziomego zapisu nutowego utworu; widać tu pewną analogię do MAM. Wzdłuż tego liniowego środowiska rozmaite abstrakcyjne formy graficzne powstają i ewoluują w czasie wraz z wybrzmiewaniem utworu. System zdaje się tłumaczyć dźwięki na formy wizualne. Scena 3D w Partiturze zapełnia się rozmaitymi typami geometrii, strukturami kości opartymi na prymitywach typu *spline*, kojarzonymi z odpowiednimi modułami systemów cząsteczkowych czy siatek wielokątowych. Ewoluujący abstrakcyjny krajobraz generowany przez Partiturę ściśle odpowiada muzycznej strukturze, analizie audio, a także manualnym gestom przesyłanym do wejścia systemu przez OSC. Każdy z tych modułów posiada wiele parametrów podlegających kontroli, które można podłączyć do OSC jako wejścia z analizy audio lub kontrolera MIDI. Analiza audio obejmuje między innymi amplitudę, detekcję stuknięć, taktów i tonacji, FFT, skalę Mel (logarytmicznie uprzestrzennioną psychoakustyczną reprezentację amplitudy w sygnale audio).

Na bazie Partitury wyewoluował programistyczny instrument do oglądania dźwięków na żywo o nazwie Dedalo. Oprogramowanie to jest kolekcją silników i narzędzi do generowania, mapowania i wymiany danych pomiędzy serią modułów graficznych i silnikiem renderującym. Całe oprogramowanie podzielono na dwie grupy (grafika oraz system) i działa również na dwóch odrębnych maszynach komunikujących się poprzez sieć oraz współpracujących dodatkowo z kilkoma iPadami, które służą do kontroli parametrów. Maszyna główna (*manager*) stanowi repozytorium statusów wszystkich parametrów, kontroluje analizę audio i jest odpowiedzialna za dane wyjściowe. Drugi komputer (*renderer*) ma za zadanie

odwzorowanie stanów parametrów w specyficzne wartości modułów graficznych, kontrolę wirtualnego oświetlenia, materiałów i cieni. Wszystkie grafiki w systemie są generowane przez GPU z użyciem DirectX11.

Duet muzyków wizualnych Abstract Birds swoje prace twórcze koncentruje na symultanicznym generowaniu abstrakcyjnych kompozycji graficznych i muzyki, korzystając z synestetycznych związków pomiędzy obrazami i dźwiękami. Celem ich badań i pracy jest materiał artystyczny, który synergicznie splata język muzyczny z językiem wizualnym. Wykorzystując nowoczesną technologię cyfrową, poddają analizie i przetwarzaniu dźwięki pochodzące z elektronicznych i akustycznych instrumentów muzycznych, cyfrowych syntezatorów itp. Proces ten przebiega w czasie rzeczywistym, a w jego wyniku renderowane są wirtualne formy wizualne reagujące na dźwięk, będące odpowiedzią na określone jego częstotliwości. Te dynamiczne formy graficzne czynią możliwym ujrzenie dźwięku. Widać to zwłaszcza wówczas, gdy zdaje się panować cisza. Percepcja ludzkiego ucha jest znacznie ograniczona; zwykle jest różna u różnych ludzi. Nie dość, że daleko nam do rejestracji słuchem pełnego spektrum świata dźwięków, to same różnice osobnicze na tym polu dowodzą, jak bardzo możemy być okradani przez nasze narządy słuchu. Nawet przy statystycznie sprawnym aparacie słuchowym na podstawie doświadczenia wizualizacji dźwięku odbiorca wydobywa większe bogactwo informacji o dźwięku niż z samego odsłuchu. Dlatego interaktywne systemy audiowizualne stają się instrumentem niwelującym te niedoskonałości, wzmacniającym percepcję świata dźwięków przez ludzkie ciało w odmienny sposób. Można również powiedzieć, że obiektywizują odbiór audio. Nie powinno zaskakiwać, że tego typu instrumenty znajdują zastosowanie w audiofonologii, gdzie służą do wizualizacji sygnału mowy i wspierają proces wczesnej diagnozy rozwoju oralnego, a jednocześnie okazują się niezwykle pomocne dla osób z poważnymi dysfunkcjami aparatu słuchowego, gdyż wspierają ich komunikację z otoczeniem [Zielińska, 2002]. Powstają specjalne interaktywne systemy audiowizualne wspomagające osoby głuche i niedosłyszące, by pomóc im w reagowaniu na dźwięki poprzez ich przekład na użyteczną wizualną informację, którą mogliby zinterpretować w czasie rzeczywistym. W projekcie Micka Griersona z Goldsmiths College w Londynie [Grierson, 2008] dźwięk jest konwertowany w czasie rzeczywistym w spektrogram zapisywany w jednowymiarowej tablicy, który następnie jest modyfikowany tak, by graficznie przedstawiał zbiór dwuwymiarowych koncentrycznych pierścieni. Nieliniowy układ tych pierścieni lepiej odzwierciedla ludzką percepcję słyszalnych częstotliwości. Pożądane rozmieszczenie pierścieni uzyskuje się dzięki zastosowaniu funkcji logarytmicznej w wizualizacji spektrum szybkiej transformaty Fouriera (FFT). Do tego celu wykorzystuje się mapowanie informacji na trójwymiarową sferę w OpenGL. Taki wybór wizualizacji dźwięku znajduje swoje uzasadnienie w neurologii, w której zidentyfikowano pewną grupę

wizualnych wzorców będących w bezpośrednim matematycznym związku ze strukturami kory wzrokowej. System Lumisonic [http://www.soundandmusic.org/projects/, dostęp: 10 VIII 2015] opracowany przez Griersona zapewnia użytkownikom interaktywną komunikację dwukierunkową, czyli przekład dźwięku na obraz, manipulację obrazem oraz jego przekład z powrotem na sygnał dźwiękowy w czasie rzeczywistym bez zauważalnych opóźnień. Dostępna na iPhone wersja oprogramowania, dzięki wbudowanemu mikrofonowi, stwarza użytkownikom możliwość wizualizacji dźwięku w każdym miejscu.

Spektrograficzne trójwymiarowe wizualizacje dźwięku opierają się na analizie widmowej sygnału audio. Zadaniem spektrografu jest konwersja fal dźwiękowych w spektrogram dźwiękowy. Wśród wielu audiowizualnych prac artysty badacza i programisty Paula Prudence'a, pokazujących sposób, w jaki dźwięk, przestrzeń i formy geometryczne mogą być ze sobą zespolone w wizualno-muzyczne doświadczenie, znaleźć można uzyskiwane w czasie rzeczywistym uprzestrzennienie analizy częstotliwościowej dźwięku szybką transformatą Fouriera (FFT) w postaci rozszerzających się i przesuwających w trójwymiarowej przestrzeni promienistych form graficznych. Na podstawie informacji o częstotliwości dźwięku płynącego z urządzeń wejściowych generowane są kolory i długości pojedynczych segmentów graficznych, a powstające z nich promieniste obręcze poruszają się w przestrzeni, reagując na każdy takt muzyki. Generatywny system wizualizacji FFR (Fast Fourier radials) Paula Prudence'a został opracowany w środowisku VVVV na potrzeby spektaklu audiowizualnego na żywo na konferencji Hactronic w Bostonie w 2007 r. i udoskonalony rok później. Jego dokumentację zdjęciową można oglądać w internecie [https://www.flickr.com/photos/transphormetic/sets/72157606705986957/ FFR – Paul Prudence; dostęp: 18 XII 2015].

Processing [https://processing.org/, dostęp: 10 VIII 2015] jest zaimplementowanym w Javie językiem programowania i zintegrowanym środowiskiem programistycznym stosowanym do celów sztuki wizualnej, dźwiękowej i interaktywnej. Processing stał się głównym środowiskiem programistycznym dla instalacji *3D Soundclash*, która odbyła się w lutym 2010 r. w Loading Bay of the Royal Albert Hall w Londynie [https://www.flickr.com/photos/fliegerhorst/sets/72157623915950906/with/4545638898/ Soundclash 3D dostęp: 18 XII 2015]. Generatywna aplikacja czasu rzeczywistego zaimplementowana na potrzeby tego wydarzenia wyświetlała na pięciu zestawionych w linii ekranach audioreaktywną wizualizację. W osobnej aplikacji *Editor* napisanej na bazie modelu EMF (*Eclipse Modeling Framework*) zespół obsługujący system sterował listą parametrów zapisywanych do pliku XML. W programie *Editor* zostały ponadto stworzone animowane GLSL shadery oraz około 40 różnych teksturowanych scen 3D autorstwa Quayoli. Główna aplikacja była dodatkowo sterowana myszą i klawiaturą we współpracy z kamerą PeasyCam, będącą specjalną biblioteką

środowiska Processing. Dźwięk był obsługiwany za pomocą zewnętrznej biblioteki audio Minim korzystającej z JavaSound API i spreparowanej do użytku ze środowiskiem Processing. Do głównych zadań Minim należały detekcja taktów oraz uruchomienie FFT na strumieniu danych audio w celu wygenerowania spektrum częstotliwości.

W 2012 r. na festiwalu sztuki nowych mediów Ars Electronica w Linzu publiczność miała unikalną sposobność nie tylko usłyszeć, lecz też zobaczyć muzykę fal mózgowych. Monitorowanie i odczytywanie fizjologicznych parametrów organizmu jest możliwe dzięki wykorzystywanym w medycynie technologiom, takim jak: EEG, RM, EKG, EMG, MMG. Niewidoczna wewnętrzna aktywność ludzkiego ciała stanowi źródło danych wejściowych sterujących na żywo oprogramowaniem generującym dźwięk i obraz. Biointerfejsy, będąc cennym źródłem danych dla kompozycji algorytmicznych, coraz częściej stają się modułami złożonych systemów wykorzystywanych w dynamicznych interaktywnych widowiskach audiowizualnych. Prezentowany przez portugalskiego artystę i programistę João Martinho Moura projekt Super Collider Shape, reprezentujący sztukę hybrydyczną, bazował w większości na dźwiękach szumu nagranych za pomocą autorskiego oprogramowania opracowanego w dwóch środowiskach: Super Collider i Processing, połączonych ze sobą protokołem OSC. System był zintegrowany z interfejsem EEG, który na żywo rejestrował i wykonywał analizę sygnałów mózgowych, a następnie skonwertowane do plików dźwiękowych dane przesyłał siecią radiową do głównego oprogramowania wizualizującego. Dodatkowym i jedynym rzeczywistym instrumentem muzycznym był fortepian, którego dźwięki były również rejestrowane i wizualizowane na żywo. Minimalistyczny audiowizualny przekład dźwięku powstał na bazie algorytmów generatywnych wykorzystujących zasady samoorganizacji w celu wyszukania optymalnych układów przestrzennych.

Rys. 2. Kadr z wizualizacji Super Collider Shape, João Martinho Moura [http://jmartinho.net/super-collider-shape/]

Chromostereoskopia

Chromostereoskopia jest techniką symulacji wrażenia głębi w obrazie 2D. W odróżnieniu od tradycyjnej techniki symulacji 3D chromostereoskopia nie wymaga współgrania dwóch obrazów, gdyż informacja jest dekodowana z pojedynczego obrazu w oparciu o mikrooptykę i barwy. Jeśli stosuje się tu dwa obrazy, to drugi ma za zadanie skorygować jasność obrazu. W procesie kodowania efektu 3D w obrazie najistotniejszym czynnikiem jest właściwy dobór palety barw, co jest uzależnione od wyboru tła. Zwykle ustala się tło w kolorze czarnym lub ciemnoniebieskim. Wówczas paleta barw RGB dobrze sprawdza się w uzyskiwaniu efektu głębi. Na białym tle stosuje się paletę CMY barw komplementarnych do barw modelu dla czarnego tła, jednak obrazy z białym tłem mogą okazać się dość nienaturalne. Trójwymiarowy zbiór danych jest rzutowany na wybraną płaszczyznę, a pozostały, trzeci wymiar służy do odwzorowania barw na piksele reprezentujące zrzutowane punkty danych. W ten sposób głębia zostaje zakodowana w płaskim obrazie za pomocą barw ułożonych w liniowym porządku. Efekt głębi można oglądać przez specjalne okulary ChromaDepth™3D z podwójnym układem dyspersyjnym. Warstwa o wysokiej dyspersji ma za zadanie zdekomponować wpadające światło na składowe jednobarwne, a niskodyspersyjna ma przeciwdziałać kątowemu odchyleniu spowodowanemu przez pierwszą. W uproszczeniu można powiedzieć, że obraz 3D jest uzyskiwany z obrazu 2D poprzez wydobycie na pierwszy plan barwy czerwonej, na ostatni niebieskiej i posortowanie pozostałych barw zgodnie z ich pozycją w spektrum elektromagnetycznym. Wielką zaletą tej techniki jest to, że otrzymany obraz nie traci swoich walorów i jest czytelny również wtedy, gdy jest oglądany bez dedykowanych okularów. Jednak należy zauważyć, że w obrazie uzyskanym tą techniką nie sposób zachować naturalne barwy.

Technika chromostereoskopii sprawdza się nie tylko w wizualizacji dźwięku, o czym wspominaliśmy wcześniej, lecz także w wizualizacji trójwymiarowych zbiorów danych naukowych. Ilustracją zastosowania chromostereoskopii w wizualizacji muzycznej może być wygenerowana w MAM graficzna reprezentacja kompozycji chóralnej Tallisa, *Spem in Alium*, złożonej z czterdziestu głosów podzielonych na osiem chórów po pięć głosów w każdym. Nutom każdego chóru są przyporządkowane elipsy w określonym kolorze i rozmiarze. Pierwszy chór reprezentowany jest przez małe czerwone elipsy na pierwszym planie, ostatni chór przez duże niebieskie elipsy zupełnie z tyłu. W wizualizacji komputerowej audio utrzymana jest struktura liniowa wzdłuż osi czasu, zaś przestrzenność wyraża ideę polifonii i rezonansu.

Korzystając z języka programowania IDL (*Interactive Data Language*) przeznaczonego do analizy i wizualizacji danych, można uzyskać wysokiej jakości re-

prezentacje graficzne złożonych danych numerycznych. Moduły graficzne silnika IDL umożliwiają szybkie tworzenie graficznych wizualizacji od dwuwymiarowych plotów i map do złożonych interaktywnych aplikacji 3D. Przykładami mogą być dynamiczna wizualizacja trójwymiarowych struktur pola magnetycznego czy interaktywna mapa powstała na bazie analiz zebranych danych geoprzestrzennych Wielkiego Kanionu, które są dostępne w internecie[http://www2.warwick.ac.uk/fac/sci/physics/research/cfsa/people/erwin/research/3d/pictures/sunmovie.gif; SunModel – Chromostereograficzny model 3D Słońca; http://www.arcgis.com/home/webmap/viewer.html?webmap=160bdeefa0f9487f81e3162ca98bfe18; ArcGIS - Over the Edge 3D: Death in Grand Canyon, dostęp: 10 VIII 2015]

Nurtujące pytanie postawione w artykule odnośnie do *live coding*, którego kontekst można śmiało rozszerzyć na wszystkie omawiane w artykule zagadnienia, znajduje odpowiedź w następującej myśli: „technologie mediów interaktywnych tworzą paradygmat określający w wielkim stopniu przebieg współczesnych praktyk społecznych, oddziałujący nie tylko na sztukę i badania naukowe, ale także na procesy komunikowania, organizację pracy i edukacji, metody reklamy i marketingu, formy zabawy i rozrywki, wkraczając tym samym niezwykle głęboko w sferę najszerzej pojmowanej codzienności" [Kluszczyński, dostęp 10 VIII 2015].

Słowa kluczowe: *wizualizacje dźwięku, audiowizualizacje, interaktywne systemy audiowizualne, live coding, programowanie na żywo, programowanie reaktywne, audioreaktywne wizualizacje, 3D sztuki hipermedialne, biointerfejsy, przestrzeń hybrydyczna, chromostereoskopia, wizualizacja zbiorów danych 3D.*

Summary

Sound and Datasets graphical visualizations

The article presents various approaches in the field of sound visualization based on hi-tech audiovisual systems and specialized software targeted at generating diverse graphical forms responding to sound in real-time. Concepts of interactive audiovisual systems, sound-reactive programming software and immersive environments refer to synergy of sound, visuals and gestures. They explore relationships between musical structures and abstract predesigned shapes rendered in real-time. The overview of chromo-stereoscopic 3D visualizing technic and its applications is also given.

Keywords: *sound visualizations, audiovisualizations, interactive audiovisual systems, live coding, reactive programming, sound-reactive 3D visualization, hypermedial arts, biointerfaces, hybridic space, chromo-stereoscopy, 3D datasets visualizations.*

Bibliografia

Opracowania

Conal E., Hudak P., *Functional Reactive Animation.*(PDF). ICFP '97 Proceedings of the second ACM SIGPLAN international conference on Functional programming1997, http://conal.net/papers/icfp97/icfp97.pdf.

DeFanti Thomas A., *The digital component of the circle graphics habitat,* National Computer Conference 1976, http://excelsior.biosci.ohio-state.edu/~carlson/history/PDFs/cgh-defanti.pdf.

Grierson M., *Making music with images: Interactive audiovisual performance systems for the deaf* (PDF). Proc. 7th ICDVRAT with ArtAbilitation, Maia, Portugal 2008, http://doc.gold.ac.uk/~mus02mg/wp-content/uploads/m-grierson-icdvrat_artabilitation_2008_submissiondoc.pdf .

Kluszczyński R., Społeczeństwo informacyjne – Cyberkultura – Sztuka multimediów, Kraków 2001.

Kluszczyński R., *Sztuka interaktywna. Od dzieła-instrumentu do interaktywnego spektaklu*, Warszawa 2010.

Verwichte E., Galsgaard K., *On the visualization of 3d datasets*, *Solar Physics*, 1998**.**

Weiser M., *The computer for the 21st century*, "Scientific American" 1991, vol. 265, no. 3, http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.

Wang G., *The ChucK Audio Programming Language: A Strongly-timed and On-the-fly Environ/mentality* (Ph.D.). Princeton University 2008.

Wang G., Cook P., *The Audicle: A context-sensitive, on-the-fly audio programming environ/mentality* (PDF). In Proceedings of the International Computer Music Conference 2004, http://soundlab.cs.princeton.edu/publications/audicle_icmc2004.pdf.

Zielińska J., *Badanie głosu u małych dzieci z wadą słuchu za pomocą specjalistycznej aparatury* (PDF). „Audiofonologia" 2002, t. XXI.

Internet

http://art.runme.org/1107861145-2780-0/livecoding.pdf.

http://chuck.cs.princeton.edu/doc/language.

http://doc.gold.ac.uk/~mus02mg/wp-content/uploads/drm-grierson-icmc-submissioncr4p-age-2007.pdf .

http://www.arcgis.com/home/webmap/viewer.html?webmap=160bdeefa0f9487f81e-3162ca98bfe18; ArcGIS - Over the Edge 3D: Death in Grand Canyon .

http://www.etienneabelin.com/#!music-animation-machine-live/cj0k.

http://www.medialarts.pl/download/skrypty/Estetyka-sztuki-nowych-mediow.pdf.

http://www.musanim.com/player/MAMPlayerUserGuide.pdf.

http://www.pawfal.org/fluxus/files/fluxus-documentation-en.pdf.

http://www.strangeloop.co.uk/Dr.%20M.Grierson%20-%20Audiovisual%20Composition%20Thesis.pdf.

http://www2.warwick.ac.uk/fac/sci/physics/research/cfsa/people/erwin/research/3d/pictures/sunmovie.gif; SunModel – Chromostereograficzny model 3D Słońca.

https://vimeo.com/51993089; Codelife - glsl live-coding editor test #5, h3xl3r, 2013.

https://www.youtube.com/watch?t=302&v=hw9kY85DkfE; Spiral 5 PTL - Dan Sandin, Tom DeFanti, and Mimi Shevitz, Live recording of performance,1979.

https://vimeo.com/73671956; Excerpts - Audiovisual concert of Quayola and Natan Sinigaglia live with Mira Calix, 2013.

https://www.youtube.com/watch?t=160&v=KwcfzORcZmA; Back To The Future, Music Animation Machine Children's Concert in Holon, Israel, 2013.

https://www.youtube.com/watch?t=37&v=dikvJM__zA4; Biophilia – kompilacja fragmentów aplikacji mobilnej, 2011.

https://www.youtube.com/watch?t=492&v=A-OsEyne8eQ; Colouring music with the

Music Animation Machine: Etienne Abelin at TEDx Amsterdam, 2013.

http://www.quayola.com/partitura-ligeti; Partitura – Sonate for Viola Solo, Ligeti, Live at Nemo Festival, Paris 2012, Quayola.

https://www.flickr.com/photos/transphormetic/sets/72157606705986957/ FFR – Paul Prudence.

https://www.flickr.com/photos/fliegerhorst/sets/72157623915950906/with/4545638898; Soundclash 3D.