



SPOŁECZNA AKADEMIA NAUK

www.san.edu.pl

Kierunek studiów “Informatyka”

1. Nazwisko i Imię: Kamil Cecherz

Numer albumu: 82739

2. Nazwisko i Imię: Łukasz Bednarski

Numer albumu:

Sprawozdanie z laboratorium przedmiotu

“Języki i paradygmaty programowania”

Prowadzący zajęcia dr inż. Zbigniew Filutowicz

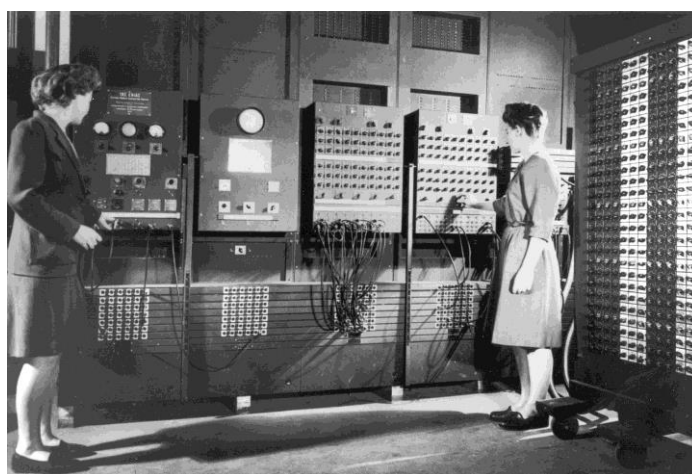
Łódź rok akademicki 2017/2018

Programista wczoraj i dziś.

Dziedzina nauk jaką jest inżynieria komputerowa na przestrzeni lat uległa diametralnym przeobrażeniom. Nie mam na myśli zmian zapoczątkowanych przez przewrotne idee pierwszych rewolucjonistów komputeryzacji, cybernetyki i biznesu; chodzi mi o efekt uboczny tego co stoi u podstaw wszystkiego - zawrotną prędkości rozwoju samego wynalazku.

Komputer, praktyczna realizacja modelu Turinga i Von Neumana, maszyna która potrafi jedynie dodawać:

„...mózg elektronowy, elektroniczna maszyna cyfrowa, maszyna matematyczna) – maszyna elektroniczna przeznaczona do przetwarzania informacji, które da się zapisać w formie ciągu cyfr albo sygnału ciągłego.”¹



Ryc. 1 Komputer ENIAC wraz z obsługą.²

Obecnie dla nikogo, spoza branży IT, definicja ta nie wyda ani sensowna, ani poważna. Nie może być inaczej, dziś o wiele częściej rozmawia się, wielokrotnie zupełnie

¹ <https://pl.wikipedia.org/wiki/Komputer> - definicja komputera, (dn 29.04.2018)

² [https://pl.wikipedia.org/wiki/Komputer#/media/File:Two_women_operating_ENIAC_\(full_resolution\).jpg](https://pl.wikipedia.org/wiki/Komputer#/media/File:Two_women_operating_ENIAC_(full_resolution).jpg)

Unidentified U.S. Army photographer - Image from Historic Computer Images

Two women operating the ENIAC's main control panel while the machine was still located at the Moore School. "U.S. Army Photo" from the archives of the ARL Technical Library. Left: Betty Jennings (Mrs. Bartik), right: Frances Bilas (Mrs. Spence). (dn. 29.04.2018)

nieświadomie, nie komputerze ale o jego używaniu. Zniknęło dawne pojmowanie „liczy więc działa”, myślimy odwrotne „liczę, że zadziała”. Maszyna wymyka się kontroli człowieka albo to on traci zdolność percepcji binarnej. Nie musi być już bowiem matematykiem, wnikliwym znawcą elektroniki by tworzyć wirtualne byty, które zwiemy programami. Czemu to takie ważne?

Z komputerami rozmawia się w sposób intuicyjny, z perspektywy ostatnich dwudziestu lat, niewyobrażalnie łatwo. Moja własna babcia, zupełnie bez pomocy nauczyła się obsługiwać smartfona z system Android, choć nigdy wcześniej nie dotykała komputera. System zainstalowany na urządzeniu znał język jej percepcji. Szybko się więc dogadali.

Pamiętam z zajęć na uczelni, opowieść doktora Zbigniewa Filutowicza³, który próbował wyjaśnić zasady działania komputera, na którym deklarowało/wykonywało się rozkazy z wykorzystaniem przełączników, lamp, taśmy perforowanej. Wydała mi się ona rodem z opowiadań S.Lema „Bajki Robotów”. Nie mogę pamiętać czasów Odry⁴, dziś podobne kolosy stoją w muzeum, ale „prehistoria informatyki” nie jest mitem; maszyny ewoluowały szybciej niż ludzie byli w stanie przewidzieć.

Mogłoby się wydawać, że technologia doprowadziła do „zidiocenia” społeczeństwa, stwierdzenie to powielają wytrawni znawcy świata IT, także programiści. Nie jest to dobre słowo, nie uwzględnia całej gamy, spektrum zmian jakie wniosła epoka cyfryzacji. Komputer pozwala ludziom na bycie „idiotami”: sprawia to łatwość z jaką możemy go obsługiwać ale też, o czym nie można zapominać, łatwość z jaką możemy pisać programy.

Przyszedł czas kiedy śmiało można rozczłonkować dziedzinę ukrytą pod nazwą programowanie: w tym aspekcie będziemy można już zacząć mówić o inżynierii, lingwistyce, analityce i projektowaniu. Skupię się na tym ostatnim, starając się przybliżyć narzędzia ułatwiające przede wszystkim efektywną produkcję aplikacji

³ <https://lodz.san.edu.pl/oferta/studia-i-stopnia/informatyka/o-kierunku-1> (dn. 01.05.2018) Specjalizuje się w zakresie inżynierii oprogramowania, komunikacji człowiek-komputer oraz automatyzacji procesów przemysłowych. Zdobywał doświadczenie w placówkach badawczych i przemysłowych w kraju i zagranicą. Uważa, że informatyka jest polem działania dla doświadczonych entuzjastów, do których zalicza sam siebie oraz dla otwartych i ciekawych świata młodych ludzi.

⁴ <https://aresluna.org/attached/computerhistory/articles/odra> (dn. 29.04.2018) Wrocławskie Zakłady Elektroniczne ELWRO “Okres maszyn cyfrowych typu ODRA” - 2002-2005 Marcin Wichary

Czyste kody.

Istnieje pewien dogmat w informatyce, mówiący o przewadze purystów językowych nad projektantami rozwiązań.

Rygorystyczna dbałość o składnię i semantykę instrukcji wprowadzanych do komputera to istota programowania; komputery jeszcze nie posiadały jeszcze umiejętności empatii, nie potrafią snuć domysłów, dokonywać własnej interpretacji języka. Język ten jest w istocie kodem. Kod jest zapisem informacji w postaci ciągu sygnałów. Komputer zna reguły jego odczytywania, posiada interpreter. Może wykonać tylko takie zadania, które jasno określa jego budowa.

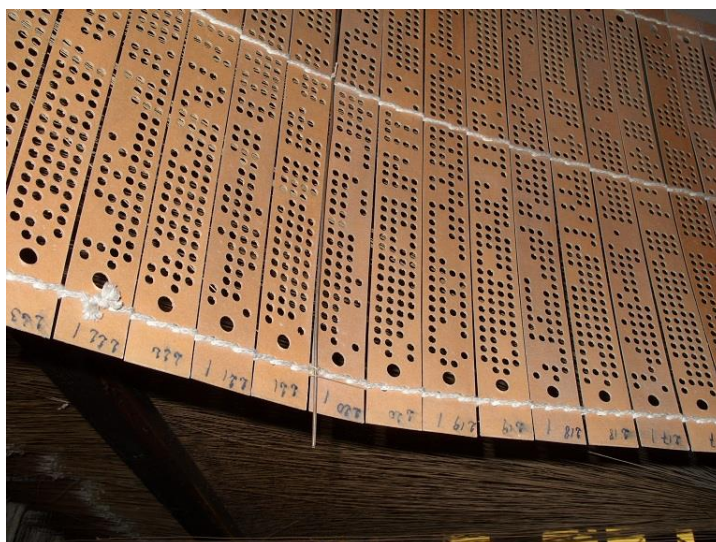
Pamiętam jedną z wizyt w Muzeum Włókiennictwa w Łodzi, dokonałem tam niezwykłego dla mnie odkrycia.

Programiści istnieli jeszcze przed wynalezieniem pierwszego tranzystora. Dowodzi to wykorzystanie przez Joseph'a Marie Jacquard'a kart perforowanych w urządzeniu znanym dziś jako maszyna żakardowa. Mechanizm zautomatyzował produkcję wyrobów włókienniczych, wytwarzanych przy pomocy krosna tkackiego⁵. Żakard to tkanina stworzona poprzez splatanie w sposób krzyżowy nici, na krośnie tkackim, w celu uzyskania konkretnego wzorca. Maszyna działa podobnie jak dzisiejsza drukarka z laserową głowicą, „linijka po linijce” rysuje zdefiniowany wzór poprzez układy pojedynczych nici. Ilustracje działania przedstawia film pt. [Jacquard weaving](#)⁶

Programy dla maszyn tkackich istniały przed stworzeniem komputerów, zostały stworzone w celu uniknięcia powtarzania rutynowych czynności, pisane były bezpośrednio w kodzie dwójkowym.

⁵ Pierwsze krosna tkackie (rozumiane jako złożone maszyny), wykorzystywano w czasach wczesnośredniowiecznych. Należy jednak pamiętać, że tkactwo jako jedno z najstarszych rzemiosł pojawiło się czasie neolitu. Dokładny rys historyczny przedstawia: Jan Głowacki, „Narzędzia i maszyny włókiennicze – w zbiorach Centralnego Muzeum Włókiennictwa”, wyd. Centralne Muzeum Włókiennictwa, Łódź, 2012

⁶ <https://youtu.be/K6NgMNvK52A> (dn. 30.04.2018) - Victoria and Albert Museum, opublikowano (dn 8.10 2015)



Ryc. 2 Karty perforowane Jacquard’a

„(...)Wzory do tkania nanosił Jacquard na karty perforowane, sterujące podnoszeniem nitek osnowy (tworzeniem przesmyku między nitkami). Pierwsze praktyczne zastosowanie kart perforowanych do przetwarzania danych przypisuje się amerykańskiemu wynalazcy o nazwisku Herman Hollerith, który zdecydował się na użycie kart perforowanych Jacquarda do przedstawiania danych zebranych w czasie amerykańskiego spisu ludności w roku 1890, a następnie do odczytywania i zestawiania tych danych przy pomocy automatycznej maszyny.

Jego pierwsze karty były perforowane okrągłymi otworami, ponieważ jego prototypowa maszyna stosowała karty z dziurkami zrobionymi przy użyciu dziurkacza do biletów, stosowanego przez konduktorów w tramwajach(...)”⁷

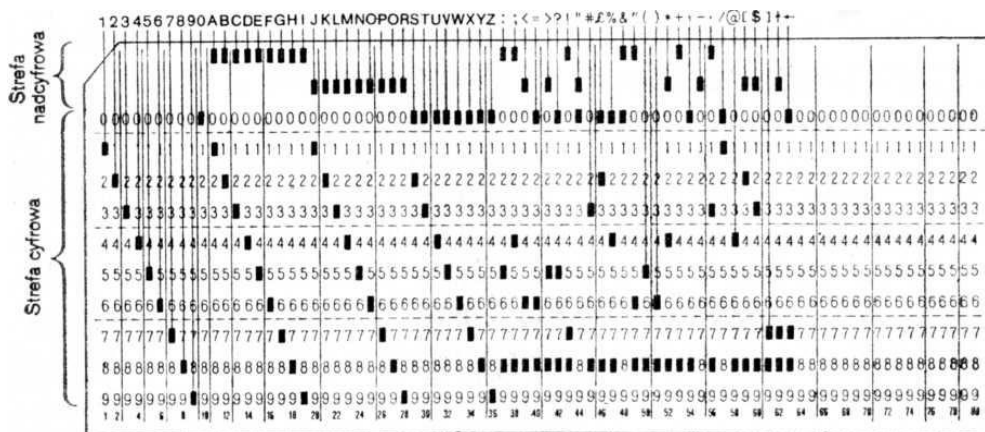
Obsługa maszyn włókienniczych przy użyciu kart perforowanych, bez żadnych warstw pośredniczących: abstrakcji w postaci metajęzyków nie wydaje trudna do zrozumienia. Nośnik danych jednoznacznie definiuje sposób zapisu, w postaci możliwe najczystszy kodu czyli pojedynczych bitów. Wtedy nie używano bramek logicznych jako elementów konstrukcyjnych układów; komputer jeszcze nie istniał.

⁷ <http://pokazywarka.pl/9nz9sy/> (dn. 30.04.2018) - Nr 83 - GAZETKA SPECJALISTYCZNA - archiwalna z 22 sierpnia 2012 roku

Pojawienie się pierwszych języków programowania typu FORTRAN nie sprawiło, że karty perforowane zniknęły; stosowano je jako nośnik danych, formę zapisu danych. Rozkazy zapisano już wtedy w postaci instrukcji przyjaznej programiście⁸:

```
PROGRAM HELLO
WRITE (*,*) 'hello, world'
END
```

Powstało wiele metod zapisu instrukcji na papierowych kartach, uległy one stopniowej standaryzacji. Dochodzimy tu do pewnej konkluzji, nośnik zapisu staje się zarazem meta-warstwą. By napisać program niezbędna była informacja o tym w jaki sposób go zapisujemy.



Rys. 3 Karta 80-kolumnowa z wydziurkowanymi znakami w kodzie maszynowym Odra-1300⁹

Jednym z pierwszych środowisk programistycznych była dziurkarka i kartki papieru. Systemu mimo pozorowanej anachroniczności cechuje trafność i oszczędność zastosowanych rozwiązań. Ciekawi analogowa forma zapisu; nośnik papier, może zostać bez trudu odczytany przez człowieka także w przypadku uszkodzeń. Dysk magnetyczny już tych zalet nie posiada.

Karta perforowana pozwala na zapis rozkazów dla komputera w formie abstrakcji w postaci języka wyższego poziomu, jak i bezpośrednio w kodzie dwójkowym; informacja jest

⁸ <https://pl.wikipedia.org/wiki/Fortran> (dn. 30.04.2018) Hello world, kod dla F77

⁹ https://pl.wikipedia.org/wiki/Karta_dziurkowana#/media/File:Karta_dziurkowana-80kolumn_Odra1300.jpg (dn. 30.04.2018) Oryginalnym przesyłającym był Dr Grzegorz Kaczmarczyk z polskiej Wikipedii

podana wprost programiście, który może ją odczytać swoimi oczyma.

Symulacje działania stanowiska do dziurkowania kart przedstawia strona internetowa: [Masswerk](http://www.masswerk.at/keypunch/)¹⁰ Możemy na niej zapisać karty w językach takich jak np. COBOL, FORTRAN.

Środowisko wizualne.

Trudno dziś wyobrazić sobie pisanie programów bez użycia monitora, a jednak pierwsze powstały bez reprezentacji graficznych na ekranie. Przeciętni użytkownicy komputerów nie używają do deklarowania rozkazów dla komputera języków w postaci liter i symboli, dla nich to czarna skrzynka. Programiści zaś nie korzystają bezpośrednio z kodu maszynowego do zapisu, gdyż jest bardzo pracochłonne i nieefektywne. Nie znaczy jednak to, że instrukcji zupełnie nie da się pisać bezpośrednio w kodzie dwójkowym; we współczesnych maszynach jest to niemal niemożliwe ale przecież istniały przed narodzeniem się elektroniki, mechaniczne konstrukcje dla których taki zapis nie wydawał się niczym kuriozalnym. Dziurkowane taśmy świetnie nadają się do zapisu muzyki.

Tym sposobem dochodzimy do przewrotnego wniosku: wszyscy użytkownicy dzisiejszych komputerów osobistych stali się programistami. Program to zestaw rozkazów dla maszyny, operacje wykonywane na rejestrach procesora i pamięci. Program to także instrukcje dla środowiska, systemu który wykonuje operacje niższego poziomu za użytkownika. Układy elektroniczne nie rozumieją użytkowników, widzą de facto dwa stany: niski i wysoki, ale ówczesne programy potrafią już nawiązać kontakt. Nauczyliśmy komputery reguł odpowiadających za poprawną komunikację. Trudno jest z komputerem rozmawiać przy pomocy słów, ale nic nie stoi na przeszkodzie by używać języka gestów: wszystkim tym zajmuje się interfejs graficzny.

Podejście to niesie pewne ograniczenia, interfejs graficzny trzeba pierw zbudować; jego realizacja w formie wizualnej, rozbudowa jest możliwa do pewnego poziomu. Wszystko to niesie koszty i nakład pracy, a co za tym idzie zapotrzebowanie na zasoby sprzętowe. Najczęściej więc łączy się interfejs graficzny ten zupełnie obrazkowy z tekstowym

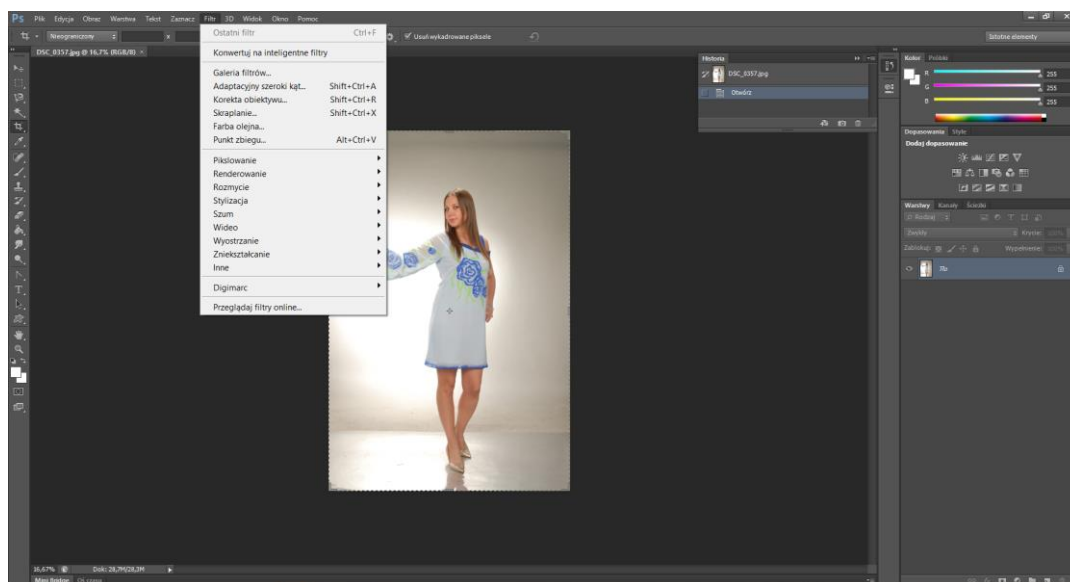
¹⁰ <http://www.masswerk.at/keypunch/> (dn. 01.05.2018) Strona posiada dokumentację oraz ciekawą wizualizację przedstawiającą działanie dziurkarek.

umożliwiającym ściśle określenie konkretnych poleceń. Dziś nawet przeciętny użytkownik może napisać skrypt wsadowy jak i pozostać przy manipulacji ikonami oraz okienkami.

Wolę obrazki.

Powstało pewne nieporozumienie, podział na grafików i programistów. Grafika komputerowa jest wykładana na uczelniach zarówno technicznych jak i artystycznych; stąd często wnosi się przekonanie, że grafik rozumiany artysta już nie musi już znać działania sprzętu. O ile prawdą jest, programiści raczej nie rozwijają się w dziedzinie sztuki, projektowania, to drugie stwierdzenie jest błędne.

Posłużę się przykładem dr Agnieszki Siwochy¹¹ cyt. „*Gdy ja studiowałam filtry graficzne do zdjęć realizowane były pomocy skryptów pisanych ręcznie...*” Słowa te zapadły w pamięć, gdyż sam przymierzałem się do zawodu grafika komputerowego. Wróciłem myślami do najbardziej znanego mi programu graficznego Adobe Photoshop starając uzmysłwić sobie ile linijek kodu: grafik, mechanizmów, modeli, zaimplementowanych algorytmów, mechanizmów kontroli błędów, znajduje się pod interfejsem tej niesamowicie rozbudowanej aplikacji.



Rys 4. Zrzut ekranu aplikacji Adobe Photoshop CS6

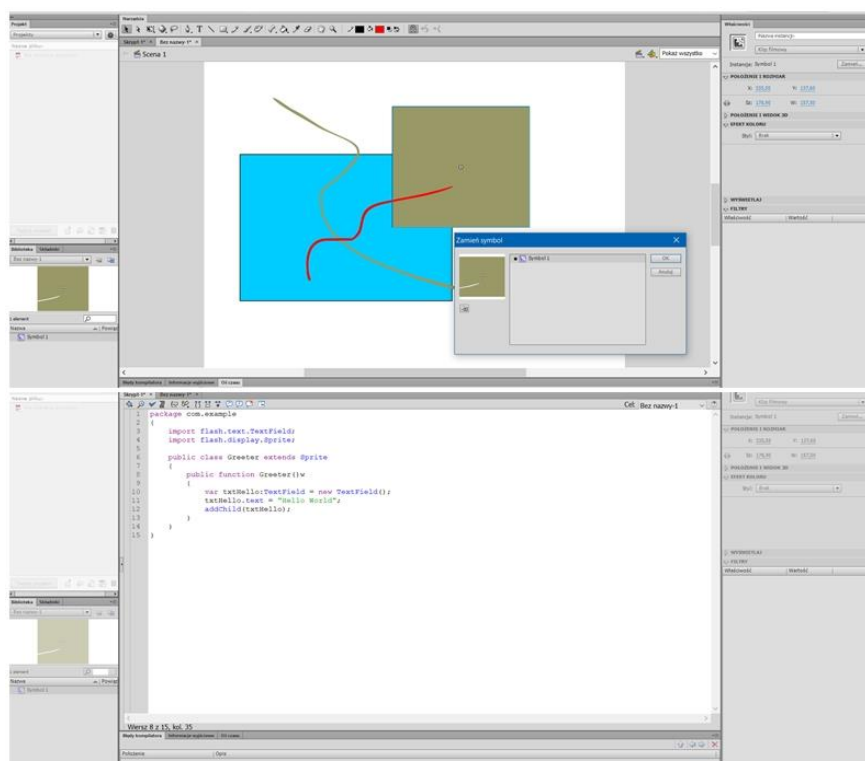
¹¹ <https://lodz.san.edu.pl/oferta/studia-i-stopnia/informatyka/o-kierunku-1> (dn. 01.05.2018) Autorka publikacji naukowych na temat wykorzystania algorytmów interpolacji fraktalnych do kodowania obrazów cyfrowych. Specjalizuje się w grafice komputerowej i multimediami, posiada tytuły Adobe Certified Expert (ACE) z programów Adobe Photoshop, Illustrator, InDesign i Dreamweaver.

Grafik nie musi znać składni i budowy języków programowania, by programować. Programując wykorzystuje filtry, tworzy modele 3D, kadruje zdjęcia, korzysta z narzędzi do automatycznej selekcji itd. Przykład ten może wydać się mało klarowny, a zaprezentowane przeze mnie porównanie umowne. Photoshop to narzędzie, jedna z wielu **napisanych** przez kogoś aplikacji. Kalkulator to też narzędzie, liczenie to na nim wymaga wykonywania kolejno odpowiednich rozkazów/operacji w celu uzyskania wyniku. Kalkulatory te fizyczne, jak i te w postaci okienkowych programów są obsługiwane przez konkretne **języki**. Można się w tym momencie zastanowić kto ma przewagę: osoby programujące chipy, osoby programujące aplikacje, czy osoby programujące w ściśle określonym środowisku danego programu.

Istnieją aplikacje z bogatym interfejsem graficznym, w których użytkownik nigdy nie zobaczy kodu, są też takie gdzie środowisko do kodowania, definiowania instrukcji w danym języku, jest zintegrowane z obszarem wizualnym aplikacji.

Do tej grupy należy Adobe Flash z wbudowanym językiem ActionScript¹² (ActionScript 3.0 to przykład nowoczesnego imperatywnego języka, który stawia na obiektowość). Język rozszerza interfejs graficzny. Program, którego przeznaczeniem jest tworzenie m.in. animacji, gier znajdzie zastosowanie w rękach typowego koderka jak i osoby z uzdolnieniami artystycznymi, bez dogłębnej znajomości paradygmatów języków programistycznych. Pozwala w sposób graficzny definiować zachowania obiektów w postaci prymitywów, które można podporządkować do schematów, które znajdują odzwierciedlenie w instrukcjach definiowanych tekstowo. Wielokrotnie przyspiesza to pracę nad budową wizualnej części tworzonych w nim klipów, animacji, gier, stron www.

¹² Obecnie technologia Flash jest na "wymarcu": Firma Adobe wskazuje na technologie HTML5, WebGL i WebAssembly jako możliwe zamienniki dla Flasha. Do tworzenia animacji internetowych poleca zaś programy Animate CC i Premiere Pro [...] Inny gigant technologiczny - Microsoft - poinformował, że do końca 2020 roku Flash zostanie usunięty z systemu Windows. Google podało zaś, że tylko 17 proc. użytkowników przeglądarki Chrome odwiedza na co dzień witryny korzystające z Flasha. Chrome ma przestać wspierać tę technologię także do 2020 roku. (<https://businessinsider.com.pl/technologie/adobe-oglasza-koniec-technologiei-flasha/m0tkh3z>, dn. 08.05.2018). "Stało się. Adobe ogłasza koniec Flasha" - Kif Leswing 26.07.2017



Rys 5. Zrzut ekranu aplikacji Adobe Flash CS6: pogląd kodu i edytor wizualny grafiki wektorowej.

Flash to środowisko do tworzenia aplikacji wizualnych, środowisko do pracy nad kodem stawia jego drugi filar. Nie można jednak zastąpić nim narzędzi wspierających pisanie programów głównie w postaci tekstu. Te często oferują zaawansowane funkcje w postaci testów jakościowych/wydajnościowych, szablonów, kontroli błędów itd.

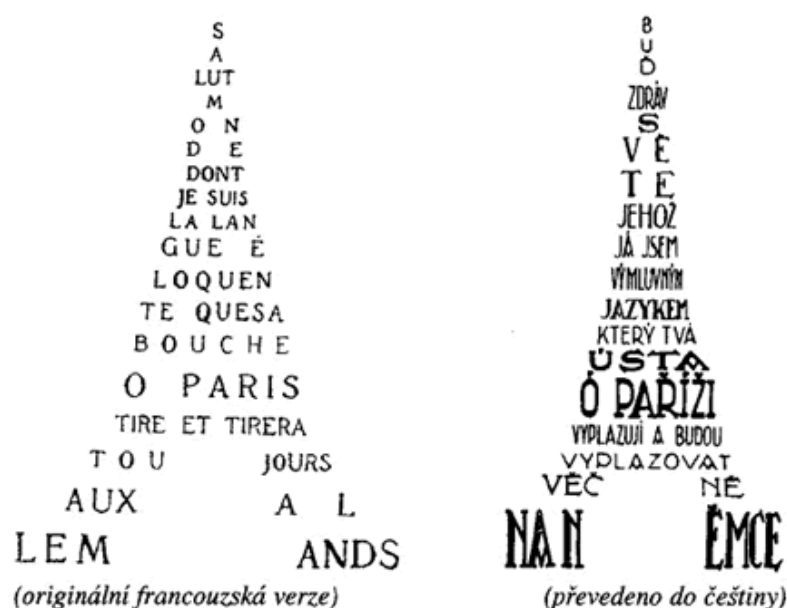
Programy nadal muszą być pisane ręcznie, trudno się bez tego obejść. Nic nie stoi jednak na przeszkodzie by tekst uczynić bardziej wizualnym, estetycznym kolorowym pobudzając przy tym pamięć i prawą półkulę.

Typografia programisty.

Klasyczna informacja zakodowana w formie tekstu, jest szybko interpretowana przez człowieka. Tak zwykliśmy zapisywać książki. Mózg nie problemu z szybkim przetwarzaniem długich ciągów liter, które niemal intuicyjnie łączy w słowa, a te z kolei w zdania. Za ich pomocą potrafi tworzyć obrazy i skojarzenia w pamięci.

W klasycznym słowie pisanym panuje porządek, który ułatwia zarówno czytanie jak i pisanie. Jest to na tyle oczywiste, że nawet nie zdajemy sobie z tego sprawy. Tekst, posiada swoje charakterystyczne uporządkowanie, jest przekazywany przy pomocy wielu wygląków i stylów będących formą ekspresji piszącego.

Istnieje wiele form sztuki polegających na zabawie układem wyrazów; jedną z nich jest wiersz wizualny. Słowa takiego utworu zostają rozbite, wyrwane ze swojego pierwotnego porządku; by móc przemówić nowym rodzajem informacji. Ciągi znaków układem, krojem, wielkością, zagęszczeniem kolorem tworzą dodatkowe przekazy wizualne.



Rys 6. Reprodukcia kaligramu Apolinnaire'a¹³

Ekspresja w formie wyrażania słów na piśmie, ma fundamentalne znaczenie dla informatyki. Za stylem przekazu tekstu, jego wyglądem, stoi nowy sposób rozumienia i interpretacji, który znalazł zastosowanie w teorii informacji. Wynalazek, tak oczywisty jak hipertekst, bierze swój początek w teorii literatury. Każda strona internetowa składa się z wielu odnośników, linków przenoszących w inne miejsce tego samego zasobu lub do zupełnie innego.

¹³ <http://www.techsty.art.pl/magazyn2/artykuly/garncarek2.html> (dn. 08.05.2018) - Magda Garncarek „Animacje Słowa”

„[...] Liberatura, od łacińskiego liber - książka, oznacza taki rodzaj literatury, dla której tworzywem jest nie tylko słowo, ale również jego fizyczny nośnik - pismo, ilustracja, fotografia, cała książka. Przykładem może być wydanie "Cent milieux miliard du poems" Raymonda Quenau, gdzie "pojedynczy sonet wydrukowany był na każdej prawej kartce książki, a jego wersy pocięte na papierowe paski" tak że można było je dowolnie kombinować.

[...] Z pojęciem liberatury łączy się współcześnie pojęcie hipertekstualności. Wtedy literatura jako swojego nośnika używa ekranu komputera, natomiast jej eksperymentem formalnym jest użycie hipertekstu - linku jako zasady konstrukcyjnej. Ma to przede wszystkim na celu zaburzenie linearności utworu literackiego.”¹⁴

Współczesny programista korzysta z dwóch wynalazków, których teoretyczny model został stworzony nim w ogóle zbudowano pierwsze komputery:

- Formatowania i kolorowania instrukcji zapisywanych w tekstach programów.
- Intellisense¹⁵, systemu podpowiedzi w postaci gotowych fraz pojawiających się podczas pisania kodu.

Porównanie programowania do poezji wydaje się kuriozalne; trudno jednak zaprzeczyć istnienia wizualnej fasady, za którą stoi pisanie programów. W programowaniu najmniej chodzi o piękno, przepych a przede wszystkim o jasność deklarowanych instrukcji. Komputer działa linearnie, nie rozumie metafor, nie obchodzi go styl kodu ani jego formatowanie. Kod musi ładny dla samego koderza; w taki sposób by mógł on go łatwo i szybko modyfikować.

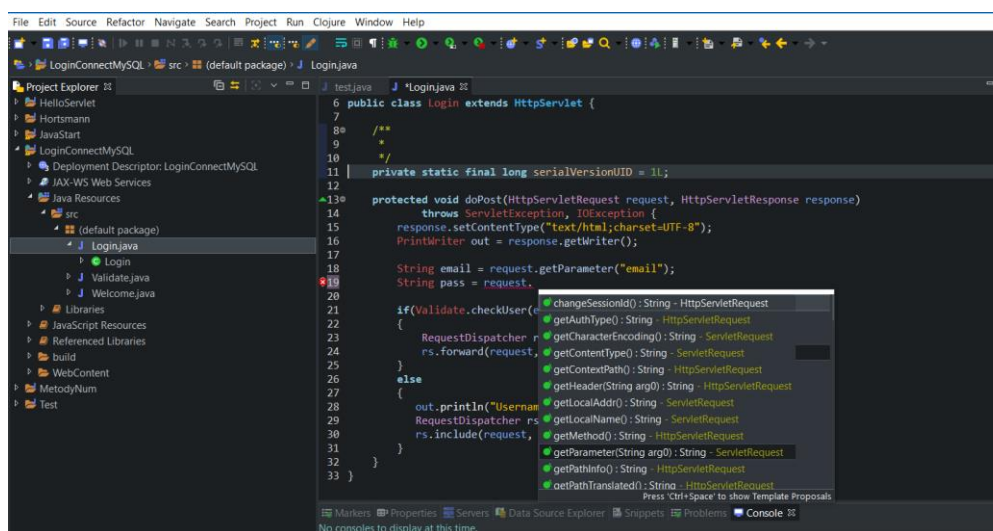
Człowiek jest istotą omylną, a niektóre języki takie jak np. Java posiadają tysiące narzędziowych klas i bibliotek. Nie da się, a nawet nie opłaca pamiętać nazwy dokładnie

¹⁴ Tekst, prezentacja dr hab. Prof. UAM Beaty Gromawadzkiej - <http://www.staff.amu.edu.pl/~beatagr/> „Rola Słowa i Obrazu w Społeczeństwie Cyfrowym” www.staff.amu.edu.pl/~beatagr/hipertekst.pptx (dn. 08.05.2018)

¹⁵ IntelliSense został po raz pierwszy wprowadzony jako dodatek do wydanego w 1996 Visual Basic 5.0 Control Creation Edition, gdzie był zasadniczo publicznie dostępnym prototypem dla Visual Basic 5.0. Pomimo początkowego testowania na zintegrowanym środowisku programistycznym Visual Basic, IntelliSense został szybko włączony do Visual FoxPro i Visual C++ w Visual Studio 97. <https://pl.wikipedia.org/wiki/IntelliSense> (dn. 08.05.2018)

każdej funkcji. Każdą najdrobniejszą rzecz można sprawdzić korzystając z książek to jednak rodzi konieczność długich poszukiwań, wertowania stron. Intellisense stanowi idealną pomoc. Mechanizm podpowiedzi rozszerza mechanizm hipertekstu, użytkownik nie skacze bezładnie po fragmentach informacji. Program podpowiada instrukcje w zależności od kontekstu, domyśla się co programista chce wpisać.

Współczesne IDE pilnują by programista się nie pomylił, dając mu sugestie, układają kod by wydawał się czytelny, pozwalają przy pomocy wbudowanych funkcji wklejać gotowe fragmenty instrukcji. Programista bez narzędzi zdaje się być hydraulikiem bez klucza; nie da się zaprzeczyć, że jest to programowanie wizualne.



Rys 7. Zrzut ekranu, podgląd działania środowiska Eclipse wspierającego Intellisense

Programowanie z klocków.

Istnieje zapotrzebowanie na aplikacje, rynek pełen jest konsumentów. Coraz więcej z nich chce się wejść do wirtualnego świata by realizować w nim potrzebę zabawy. Komputer operujący obrazami przyciąga, stymuluje człowieka. Ludzie uwielbiają gry od tych banalnych z cukierkową grafiką polegających na przesuwaniu bloczków, po te rozbudowane, zaawansowane z własnymi wirtualnymi światami.

Gry nie byłyby tak łatwo dostępne gdyby nie programowanie wizualne. Stworzenie pełnowartościowego silnika gry wymaga setki tysięcy linii kodu, znajomości fizyki, optyki, matematyki. Wymaga zaangażowania całego zespołu nieprzeciętnych koderów. Jednakże do

stworzenia samej gry opartej o silnik wystarczy kreatywność i elementarna znajomość zasad programowania. Każdy dziś może robić własne gry.

Pierwsze edytory gier pamiętam jeszcze z czasów 8-bitowych konsolowych gier wideo na Pegasus¹⁶. Nie były one wyposażone w mechanizmy pozwalające na kontrolę zdarzeń, pozwalały na edycję poziomów ale dawały pełni możliwości manipulacji mechanizmami programu. Nazwałbym to „deklaratywnym programowaniem wizualnym”.



Rys 8. Gra „Tank 1990”¹⁷

Jedną z bardziej znanych gier, na które trwonilem czas jako dziecko jest gra Battle City, która doczekała wielu klonów i modyfikacji. Oferuje ona prosty tryb konstrukcji, gdzie można określić zawartość każdego z bloczka na planszy jak rodzaj przeciwników. Tworzenie poziomów przypomina w dużej mierze układankę.

Obecnie wiele gier oddaje w ręce użytkowników, edytory które teraz należałoby nazwać środowiskami programistycznymi, gdyż pozwalają one na sterowanie dowolnymi elementami środowiska świata programu.

¹⁶ <https://pl.wikipedia.org/wiki/Pegasus> konsola jest polskim klonem japońskiej konsoli Famicom. (dn. 09.05.2018)

¹⁷ http://bezsensopedia.wikia.com/wiki/Tank_1990 oraz https://pl.wikipedia.org/wiki/Battle_City (dn. 09.05.2018)

Battle City – gra zręcznościowa typu akcja wyprodukowana i wydana na konsole Famicom w 1985 r. przez japońską firmę Namco. Jest to kolejna część gry Tank Battalion z 1980 roku. Battle City jest również znane pod nazwą Tank 1990 oraz wydane pięć lat później po grze Tank Battalion przez tego samego producenta.

Powstały gry, które same w sobie są edytorami światów. To znaczy, że uzależniają rozwój rozgrywki od kreatywności użytkownika. Przykładem takiego programu a zarazem multiplatformy jest Minecraft.

Gra daje nieograniczone możliwości tworzenia, dlatego stała się ona ewenementem na rynku światowym i źródłem inspiracji ludzi z całego świata. Minecraft wraz z całym dostępnym zestawem pluginów, tworzy programistyczny ekosystem, którego celem jest zabawa poprzez budowę bytów wirtualnych z sześciennych bloczków. Sama rozgrywka, którą oferuje program jest nieskomplikowana; w zasadzie przypomina te które oferują MMORPG. Do gry dochodzi jednak system narzędzi, który pozwala na nadawanie wizualizacjom z sześciennych bloków całej gamy zachowań. W efekcie zabawa programem może trwać tak długo aż użytkownikowi nie braknie pomysłów.

Nie tylko mnie do tej zadziwia mnogość zastosowań gry w edukacji, powstała już publikacja pt. „Nauka programowania z Minecraftem” autorstwa Craig Richardson’a. Jak piszę autor we wstępie do swej publikacji:

„[...]W książce Nauk programowania z Minecraftem zrobicie to wszystko i jeszcze więcej dzięki sile Pythona, bezpłatnego języka używanego przez miliony zawodowców i początkujących programistów! Zaczynamy od krótkich, prostych lekcji Pythona, a następnie wykorzystujemy nasze nowe umiejętności do modyfikowania Minecrafta, aby osiągać natychmiastowe i robiące wrażenie wyniki. Nauczycie się, jak dostosowywać Minecraft, aby tworzyć mini-gry[...]”¹⁸

Odkrywamy, że gra staje się poligonem do testowania języka Python. Autor chce pokazać jak ważne jest wizualizowanie własnych działań. Odchodzi od suchych definicji, regulek które muszą być znane, by dobrze móc programować.

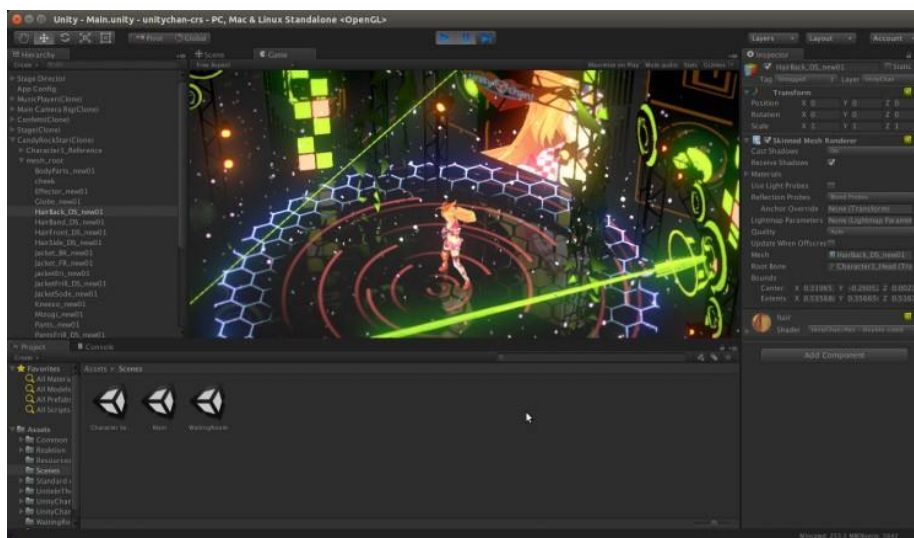
Trudno obecnie jednoznacznie określić czym właściwie jest „Minecraft”, gdyż nie przypomina on zwyczajnej gry. Ja porównałbym go do Kodu Game Lab. Aplikacja firmy Microsoft pozwala, w sposób niezwykle intuicyjny, na tworzenie gier. Nie można jej włożyć

¹⁸ <https://ksiegarnia.pwn.pl/Nauka-programowania-z-Minecraftem,498008305,p.html> Nauka programowania z Minecraftem (dn. 09.05.2018)

do jednego worka z systemami takimi jak Unity, choć też jest to silnik do gier. Kodu Game Lab posiada dużo więcej wizualnych uproszczeń, jest to program, który stylem interfejsu użytkownika nie przypomina fachowego narzędzia. Wydaje się dużo bardziej trywialny.



Rys 9. Zrzut ekranu środowisko Kodu Game Lab



Rys 10. środowisko Unity¹⁹

Oczywiście spektrum zastosowań i możliwości obu platform jest zróżnicowane, ten przykład doskonale pokazuje jak odmienne może być podejście do tworzenia gier. Trudno odmówić twórcom Kodu Game Lab wyobraźni, w programie nawet zdarzenia dodajemy z

¹⁹ <https://www.omgubuntu.co.uk/2015/08/unity3d-linux-build> „Unity At Last: Unity 3D Games Editor Is Now Available on Linux” - Joey Sneddon (dn 09.05.2018)

pomocą śmiesznych ikonek; osobiście spodziewałem się edytora tekstowego celem zapisu funkcji.

Podsumowanie.

Komputery nie uczyniły z nas „idiotów”. Choć ich obsługa stała się bardziej przyjazna i intuicyjna nadal wszystko zależy od ludzkiej kreatywności. Komputer nadal potrafi tylko liczyć, zapisywać i zwracać ciągi danych w postaci bitów. To oprogramowanie a tym samym sami ludzie nadali definicji programowania nowe znaczenie. Maszyna nie nauczyła się rozumieć człowieka, to on sam wypracował lepsze sposoby komunikacji z nią samą.

Wraz z narodzinami interfejsu graficznego, weszliśmy w świat programowania wizualnego. Programy z materialnych nośników przeszły na ekrany, na nich się rodzą i umierają. Dziś trudno sobie wyobrazić tworzenie programów bez narzędzi ale niektórym jeszcze trudniej deklaratywnie ich w narzędziach takich jak Kodu Game Lab, które nie pozostają suchej nitki na zatwardziały purystach, mistrzach czystego kodu.

Czystego kodu nie ma, przestał istnieć w raz z pojawieniem assemblera a potem języków wyższego poziomu. Stoimy za kolejnymi warstwami abstrakcji, nie tworzymy programów od zera; zawsze jest jakieś narzędzie. Narzędzia są jak fale postępu w cywilizacji ludzkiej, nowe opierają się o dawniejsze.

Puryści kodu nadal są potrzebni zwracają oni uwagę na wadliwość aplikacji tworzonych w pośpiechu by zaspokoić popyt konsumentów. Według dr inż. Konrada Grzanka²⁰ mniejsza popularności LISP'ów²¹ na rzecz języków obiektowych jest spowodowana tym, że te pierwsze wymagają dużego stopnia zaawansowania w dziedzinach para-informatycznych, a rynek potrzebuje każdego nowego programistę.

²⁰<https://lodz.san.edu.pl/oferta/studia-i-stopnia/informatyka/o-kierunku-1> (dn. 09.05.2018) Programista specjalizujący się w programowaniu funkcyjnym, praktyk z wieloletnim doświadczeniem w branży IT, przy tym – nauczyciel akademicki i naukowiec posługujący się metodami machine learning w badaniach związanych z oceną jakości oprogramowania.

²¹https://jcubic.pl/jakub-jankiewicz/lisp_tutorial.php (dn. 09.05.2018) „Kurs Programowania w Języku Common Lisp” – Jakub Jakubcewicz

Lisp jest to skrót od LISt Processing. Został opracowany przez Johna McCarthy'ego w latach 50-tych. Został zainspirowany "Rachunkiem Lambda" Alonzo Churcha, od tej nazwy pochodzi charakterystyczne dla lisp'a wyrażenie lambda, które tworzy nową funkcję anonimową. Jest drugim najstarszym językiem, który jest nadal w użyciu.

Programowanie graficzne to szereg wielu uproszeń, możemy tworzyć programy mniej lub bardziej świadomie, ale przy obecnym etapie rozwoju komputeryzacji będzie to zawsze ścisła współpraca z narzędziami. Nasza praca w wielu aspektach ulega stopniowej automatyzacji, także myślenie.