Autorská Kniha

Kristian Maňas

Bakalářská práce
2013

Univerzita Tomáše Bati ve Zlíně
Fakulta multimediálních komunikací

PROHLÁŠENÍ AUTORA
BAKALÁŘSKÉ/DIPLOMOVÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské/diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby [1)];
- beru na vědomí, že bakalářská/diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- na moji bakalářskou/diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3 [2)];
- podle § 60 [3)] odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 [3)] odst. 2 a 3 mohu užít své dílo – bakalářskou/diplomovou práci - nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské/diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské/diplomové práce využít ke komerčním účelům.

Ve Zlíně 21.5.2015

Kristian Mináš

.....

Jméno, příjmení, podpis

---

1) zákon č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, § 47b Zveřejňování závěrečných prací:

(1) Vysoká škola nevýdělečně zveřejňuje disertační, diplomové, bakalářské a rigorózní práce, u kterých proběhla obhajoba, včetně posudků oponentů a výsledku obhajoby prostřednictvím databáze kvalifikačních prací, kterou spravuje. Způsob zveřejnění stanoví vnitřní předpis vysoké školy.

(2) Disertační, diplomové, bakalářské a rigorózní práce odevzdané uchazečem k obhajobě musí být též nejméně pět pracovních dnů před konáním obhajoby zveřejněny k nahlížení veřejnosti v místě určeném vnitřním předpisem vysoké školy nebo není-li tak určeno, v místě pracoviště vysoké školy, kde se má konat obhajoba práce. Každý si může ze zveřejněné práce pořizovat na své náklady výpisy, opisy nebo rozmnoženiny.

(3) Platí, že odevzdáním práce autor souhlasí se zveřejněním své práce podle tohoto zákona, bez ohledu na výsledek obhajoby.

2) zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, § 35 odst. 3:

(3) Do práva autorského také nezasahuje škola nebo školské či vzdělávací zařízení, užije-li nikoli za účelem přímého nebo nepřímého hospodářského nebo obchodního prospěchu k výuce nebo k vlastní potřebě dílo vytvořené žákem nebo studentem ke splnění školních nebo studijních povinností vyplývajících z jeho právního vztahu ke škole nebo školskému či vzdělávacího zařízení (školní dílo).

3) zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, § 60 Školní dílo:

(1) Škola nebo školské či vzdělávací zařízení mají za obvyklých podmínek právo na uzavření licenční smlouvy o užití školního díla (§ 35 odst. 3). Odpírá-li autor takového díla udělit svolení bez vážného důvodu, mohou se tyto osoby domáhat nahrazení chybějícího projevu jeho vůle u soudu. Ustanovení § 35 odst. 3 zůstává nedotčeno.

(2) Není-li sjednáno jinak, může autor školního díla své dílo užít či poskytnout jinému licenci, není-li to v rozporu s oprávněnými zájmy školy nebo školského či vzdělávacího zařízení.

(3) Škola nebo školské či vzdělávací zařízení jsou oprávněny požadovat, aby jim autor školního díla z výdělku jím dosaženého v souvislosti s užitím díla či poskytnutím licence podle odstavce 2 přiměřeně přispěl na úhradu nákladů, které na vytvoření díla vynaložily, a to podle okolností až do jejich skutečné výše; přitom se přihlédne k výši výdělku dosaženého školou nebo školským či vzdělávacím zařízením z užití školního díla podle odstavce 1.

Univerzita Tomáše Bati ve Zlíně
Fakulta multimediálních komunikací
Ústav vizuální tvorby
akademický rok: 2012/2013

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Kristian MAŇAS**
Osobní číslo:	**K10126**
Studijní program:	**B8206 Výtvarná umění**
Studijní obor:	**Multimedia a design – Vizuální komunikace**
Forma studia:	**prezenční**
Téma práce:	**Autorská kniha**

Zásady pro vypracování:

1. Rešerše
2. Analýza
3. Stanovení filozofie
4. Vypracování projektu
5. Zhodnocení projektu

Na samostatném nosiči CD-ROM odevzdejte v minimálním počtu 10 kusů obrazovou dokumentaci praktické části závěrečné práce pro využití v publikacích FMK. Formát pro bitmapové podklady: JPEG, barevný prostor RGB, rozlišení 300 dpi, 250 mm delší strana. Formáty pro vektory: AI, EPS, PDF. Loga a texty v křivkách. V samostatném textovém souboru uveďte jméno a příjmení, login do Portálu UTB, obor (ateliér), typ práce, přesný název práce v češtině i v angličtině, rok obhajoby, osobní mail, osobní web, telefon. Přiložte svou osobní fotografii v tiskovém rozlišení.

Rozsah bakalářské práce:	**viz. Zásady pro vypracování**
Rozsah příloh:	**viz. Zásady pro vypracování**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

1. PEARSON, Matt. Generative Art. Manning Publications 2011
2. REAS, Casey, MCWILLIAMS, Chandler. Form+Code in Design, Art, and Architecture. Princeton Architectural Press 2010
3. REAS, Casey, FRY, Ben. Processing: A Programming Handbook for Visual Designers and Artists. The MIT Press 2007
4. FRY, Ben. Visualizing Data: Exploring and Explaining Data with the Processing Environment. O'Reilly Media 2008
5. BRINGHURST, Robert. The Elements of Typographic Style. Hartley and Marks Publishers 2004
6. MULLER-BROCKMANN, Josef. Grid Systems in Graphic Design. Ram Publications 1996

Vedoucí bakalářské práce:	**Mgr. Richard Vodička** Ústav vizuální tvorby
Datum zadání bakalářské práce:	**5. prosince 2012**
Termín odevzdání bakalářské práce:	**17. května 2013**

Ve Zlíně dne 5. prosince 2012

doc. MgA. Jana Janíková, ArtD.
děkanka

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta multimediálních komunikací

M. A. Vladimír Kovařík
ředitel ústavu

ABSTRAKT

Svět digitálních technologií designérům přináší nové možnosti a způsoby práce. Cílem této práce je vysvětlit, v čem je programování v designu důležité, a ukázat postupný vývoj digitálních nástrojů využívaných v umělecké praxi. Praktická část se zaměřuje na průzkum současně nejvyužívanějších nástrojů a typových příkladů z praxe. Projektová část poté popisuje filozofii za vznikem autorské generované knihy Náhodný_sešit a technologické obtíže s projektem spojené.

Klíčová slova: kreativní programování, Processing, generativní design, interaktivita, pirátství, open source, vizuální komunikace, grafický design.

ABSTRACT

World of digital technologies brings new possiblities to the designers. The goal of this work is to explain why is programming important in design and show gradual development of digital tools used in artistic practice. Practical part of the work focuses to survey the most frequently used tools and typical examples from practice. Project part then explains philosophy behind birth of authors book called Náhodný_sešit (Random_copybook) and technological troubles concerning the project.

Keywords: creative programming, Processing, generative design, interactivity, piracy, open source, visual communication, graphic design.

Rád bych poděloval vedoucímu práce panu Mgr. Richardu Vodičkovi a oponentovi MgA. Kryštofu Peškovi za odborné vedení při zpracovávání bakalářské práce.

OBSAH

ÚVOD **8**
I TEORETICKÁ ČÁST **9**
1 POČÍTAČE A ČLOVĚK **10**
1.1 Design a programování 10
2 POČÁTKY POČÍTAČOVÉHO OBRAZU **12**
2.1 Ben F. Laposky 12
2.2 Počítač SAGE 13
2.3 Sketchpad 14
2.4 Od programátorů k umělcům 14
3 HISTORIE KREATIVNÍHO KÓDU **15**
3.1 Aesthetics + Computation 16
3.2 Kreativní programování 16
3.3 Současné digitální nástroje versus kreativní programování 16
3.4 Kde počítač exceluje 17
3.4.1 Repetice a modularita 18
3.4.2 Parametrizace 18
3.4.3 Simulace 19
3.4.4 Co počítače nedokážou 19
3.5 Využití kreativního programování 19
4 PŘÍKLADY A VŮDČÍ OSOBNOSTI **21**
4.1 Osobnosti a studia 21
4.1.1 Robert Hodgin (Flight404) 21
4.1.2 Nervous systems 21
4.1.3 Marius Watz 21
4.1.4 Casey Reas 22
4.1.5 Onformative 22
4.1.6 Field 22
4.1.7 Universal Everything 22
4.1.8 V Squared labs 22
4.1.9 Matt Pearson 23
4.1.10 Daniel Shiffman 23
4.1.11 Jer Thorp 23
4.2 Typové příklady kreativního programování 24
4.2.1 MIT Media Lab identity 24
4.2.2 House of cards 24
4.2.3 Amon Tobin ISAM 25
4.2.4 Solar Sinter 26
4.2.5 The Carp and The Seagull 27
4.2.6 Project Cascade 27
4.3 Důležité portály a komunity 28

4.3.1 Creative Applications [creativeapplications.net] 28
4.3.2 Visualizing [visualizing.org] 28
4.3.3 Create digital motion [createdigitalmotion.com] 28
4.3.4 Creative coding podcast [creativecodingpodcast.com] 28
4.3.5 Resonate festival [resonate.io] 29
4.3.6 Open processing [openprocessing.org] 29
4.3.7 The Creators Project [thecreatorsproject.vice.com] 29

II PRAKTICKÁ ČÁST 30

5 PROGRAMOVACÍ JAZYKY A NÁSTROJE PRO KREATIVNÍ PROGRAMOVÁNÍ 31

5.0.1 Vysokoúrovňový programovcí jazyk 31
5.0.2 Důležité otázky při volbě programovacího jazyka 32

5.1 Základní rozdělení a přístupy 32

5.2 Textové jazyky 32

5.2.1 ActionScript 32
5.2.2 JavaScript 33
5.2.3 Python 33
5.2.4 Java a Processing 34
5.2.5 C++ 34
5.2.6 OpenFrameworks [www.openframeworks.cc] 34
5.2.7 Cinder [libcinder.org] 34
5.2.8 Polycode [polycode.org] 35

5.3 Vizuální programovací jazyky a rozhraní 35

5.3.1 Nodebox [nodebox.net] 35
5.3.2 Quartz Composer 35
5.3.3 Max/MSP/Jitter [cycling74.com/products/max/] 35
5.3.4 PureData [puredata.info] 35
5.3.5 vvvv [vvvv.org] 35
5.3.6 TouchDesigner [derivative.ca] 36

6 SEZNAMOVÁNÍ SE S KREATIVNÍM PROGRAMOVÁNÍM 37

6.1 Užitečné zdroje 37

6.1.1 The Nature Of Code [natureofcode.com, vimeo.com/shiffman] 37
6.1.2 MOOC [edx.org, udacity.com, coursera.org] 37
6.1.3 MIT open-courseware [ocw.mit.edu] 37
6.1.4 Khan Academy [khanacademy.org] 37
6.1.5 Code Academy [codecademy.com] 37

6.2 Doporučené knihy k jazyku Processing 38

6.2.1 Processing: A Programming Handbook 38
6.2.2 Learning Processing 38
6.2.3 Generative Design 38

III PROJEKTOVÁ ČÁST 39

7 PROJEKT: NÁHODNÝ_SEŠIT 40

7.1 Pozice knihy v naší době 40

7.2 Open source 40

7.3 Pirátství 41
7.4 Do it yourself 41
7.5 Technologie za projektem 41
7.5.1 Problematické části 41
7.6 Software jako kniha 42
ZÁVĚR 43
BIBLIOGRAFIE 44
SEZNAM OBRÁZKŮ 46
SEZNAM PŘÍLOH 47

ÚVOD

Tato práce se zabývá využitím programování ve vizuální komunikaci a designu. Rozebírá, jak může systematické a velmi racionální myšlení nutné v programátorské praxi pomoci designérům najít nová řešení a potencionálně jim rozšířit jejich obzory. Zabývá se rozborem současných digitálních nástrojů a programovacích jazyků vhodných k designérské činnosti. Dále se ohlíží za využitím programování v minulosti a jak se kreativní programování postupně dostalo z umělecké praxe do designu. Reálné příklady využití jsou z rukou těch nejlepších současných designérů využívajících programovaní v jejich práci. Rozebrána je také praktická zkušenost a to, jak obecně začít s programováním v umělecké praxi co nejefektivněji. Projektová část bakalářské práce se pak zabýva tvorbou knihy jménem Náhodný_sešit a myšlenkami za jejím vznikem.

I. TEORETICKÁ ČÁST

1 POČÍTAČE A ČLOVĚK

Počítače se za krátkou dobu staly naší neodmyslitelnou součástí. Ať už jde o větší počítače používané k práci, mobilní telefony zásadní pro naší komunikaci, nebo herní konzole, kterými si mnoho z nás zpříjemňuje svůj volný čas, všichni začínáme býti obklopení výpočetní technikou. Co je ovšem zajímavé, je to, že není tolik lidí, kteří by doopravdy dokázali počítač používat ve své nejpřirozenější poloze. Myslím, že jsme částečně v přechodovém období, je to vidět na tom, jak dnešní software funguje. Většinou se totiž snaží emulovat reálné postupy a snaží se přenést objekty z reálného světa do toho digitálního. Stále více lidí se učí využívat počítač naplno v jeho přirozenosti. Abstrahování je zde velice důležité, začínáme si uvědomovat, že pokud chceme něco vybarvit červenou barvou, nepotřebujeme na to přece plechovku s barvou. Tato abstrakce pomalu přichází k designérům, kteří navrhují stále lepší, efektivnější a hezčí postupy, jak ovládát výpočetní techniku kolem nás. Na vyhození emailu už dávno není potřeba ikona koše, stačí ho přetáhnout z obrazovky pryč. Interakci navrhují designéři a na to, aby ji mohli navrhovat, nutně potřebují… kód. Neprogramátorům přijde kód jako něco záhadného a velice složitého, někdy až zastrašujícího. Programování obecně spadá do kategorie věcí pro nepopulární inteligenty nebo mají lidé představu, že naučit se programovat trvá tak dlouho, že je to zbytečné vůbec zkoušet. Na druhou stranu většina čistých stávajících programátorů na kódu nevidí vůbec nic kreativního. Já myslím, že pro mnoho lidí může existovat i hybridní cesta. Cesta mezi pravou a levou stranou hemisféry. Cesta, která je v dnešním specializovaném světe ne úplně populární. Zajímalo by mě, co v tom je, že donedávna byl racionálnější přístup v umění a obzvlášť v designu trochu shazován, vždyť nejeden velký mistr minulosti přinesl důležité vědecké objevy. I ten, kdo neakceptuje programování jako možné umělecké vyjádření, určitě musí akceptovat racionalitu a možnosti, které programování přináší do sféry designu.

1.1 Design a programování

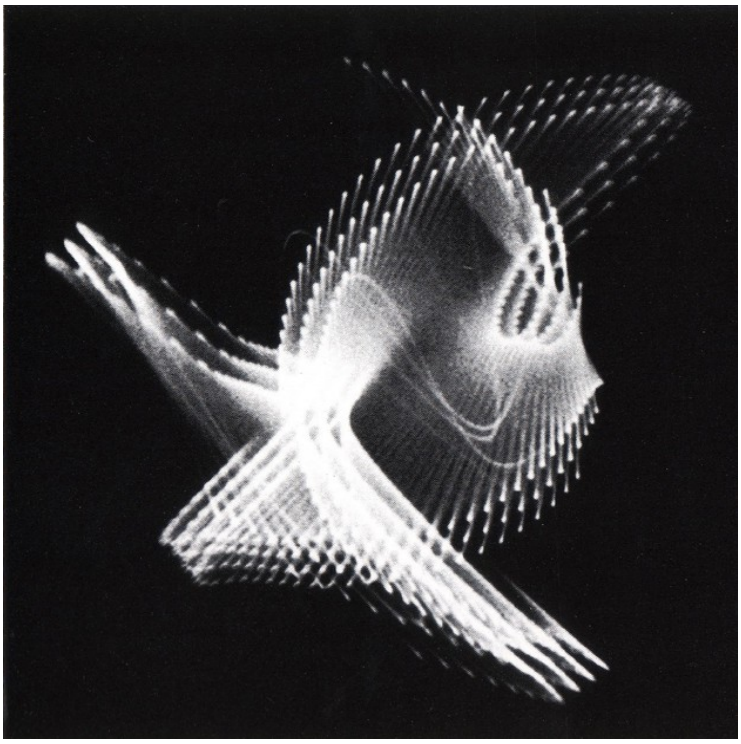
Softwaru na tvorbu digitálních obrázku je celá řada a každý z nich má svoje zaměření a specifičtější vlastnosti, jakými jsou například typy filtrů/efektů, se kterými můžeme upravovat obraz. Designéři se ženou za stále novějším softwarem, pluginy a efekty, protože tím rozšiřují svoje možnosti práce. Software je vnímán jako nedotknutelný a také si hodně lidí myslí, že jeho správnými kombinacemi dojdeme k žádanému výsledku. Každý designér si časem uvědomí limity stávajících digitálních nástrojů, filtry za sebou zanechávají svoji digitální stopu, kterou lze dobře vystopovat. Software, který používáme, dramaticky ovlivňuje výsledek a přístup naší práce. Ale co takhle si udělat vlastní filtr, nebo

rovnou celý grafický program? Najednou se sám nástroj může stát řešením. Grafický design se silně opírá o styl, jednotnou vizualitu, vytvoření originálního efektu (filtru) je velmi efektivní cestou při tvorbě značky. Není to ovšem jen návrh uživatelské interakce a tvorba grafických nástrojů, k čemu je kód dobrý. Programování prostupuje do drtivé většiny oborů od umění po farmářství a to z jednoho prostého důvodu: počítače jsou neskutečně rychlé a zvládají zpracovávat obrovská kvanta dat. Pro digitální design je programování v podstatě definitivním nástrojem. Všechny programy, se kterými designér pracuje, jsou jen nástavbou programování, zbůsobem, jak mu práci ulehčit. Vše, co kdy někdo vytvořil se softwarem, lze vytvořit pomocí programovacího jazyka (jakéhokoliv - vše, co je vypočítatelné, lze provést všemi programovacími jazyky), jen to může být náročné a pekelně komplikované. Tím se dostávám k tomu, co je podle mě na programování pro design tím nejdůležitějším. Nejde o dělání prototypů aplikací, o vizualizace dat, které by jinak bylo nemožné udělat ručně, ovládání hardwaru, nebo o animaci milionu objektů najednou. Jde především o svobodu. Svobodu mysli, odpoutání se od limitů softwaru, který již existuje. Vše je najednou možné, je to jen otázka jak dlouho to bude trvat. Tato svoboda vede k novému světu nápadů a inspirace. Programování není o matematice, jak si asi většina lidí myslí. Programování je především o správném popsání problému a schopnosti rozložit jeho řešení na menší části. Dovednosti, které jsou pro designéra k nezaplacení. Design byl přece vždy o spojení technologie, racionality a emocí. Mnoho věcí naznačuje tomu, že přichází doba, kdy si mnoho designérů osvojí programování a začne ho naplno využívat ve své práci, protože programování ještě nikdy nebylo přístupnější.

2 POČÁTKY POČÍTAČOVÉHO OBRAZU

2.1 Ben F. Laposky

Design se úzce propojen s vizuálním vjemem, proto asi není překvapením, že využití počítačů v designu, je přímo závislé na vývoji počítačové grafiky. První, kdo hledal krásu na obrazovce, byl na začátku padesátých let Ben F. Laposky. Jeho počiny nejsou sice ještě plně digitální, protože využíval analogových osciloskopů, nicméně osciloskop používá k zobrazování svých klasickou CRT obrazovku. Laposky viděl v abstrakcích na osciloskopu určitou krásu, kterou pak fotil a vystavoval v galeriích.



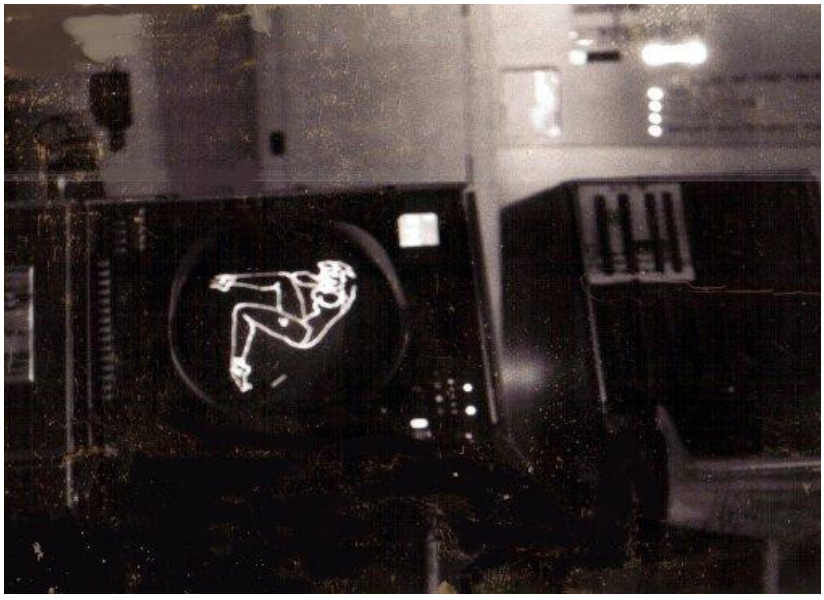
Obr. 1. Obrázek oscilogramu od B. Laposky

I přes to, že jsou jeho obrazy generovány pomocí doma tvořených obvodů a ne kódu, vypadají jako by je generoval někdo dnes za pomocí programování. Ben Laposky vytváří nový druh vizuality, který je velmi podobný tomu, co je později typické pro počítače a digitální umění[1].

1. ARTNER, A.G., 2006. *Shining Light on Work of Electronics Pioneer: Space-Age Modernism seen in Ben F. Laposky's Images*. Washington, United States, Washington: , Sep 07, ProQuest Central.

2.2 Počítač SAGE

Nejdůležitější institucí, která se zajistila o vznik počítačové grafiky, je Massachusetts Institute of Technology (dále MIT). MIT v padesátých letech spolu s IBM dělala největší výzkum v poli informačních technologií. Výzkum, který byl a pořád pokračuje na MIT je zcela zásadní pro vývoj počítačů a obzvlášť programovacích jazyků. Je také zásadní pro moji bakalářskou práci, která by bez programovacího jazyku, který pochází právě z MIT (Processing), nemohla vzniknout. Prvním počítačem s živě interaktivním výstupem byl Whirlwind. Vyroben výzkumníky z MIT, jednalo se o počítač který pomocí radaru monitoroval letadla a jako tečky zobrazoval jejich pozici na obrazovce. Whirlwind byl finančně velmi náročný a americká vláda se ho v rámci škrtů chtěla zbavit, nakonec jeho technologii adaptovala americká armáda a s notnými vylepšeními vznikl projekt SAGE (Semi-Automatic Ground Environment). SAGE nebyl jen počítač, byl to celý americký monitorovací systém skládající z 23 vzájemně sítí propojených bunkrů, rozkládajících se napříč celými Spojenými státy americkými. Přinesl velké pokroky na poli telekomunikačních technologií, radarů, počítačů i programovacích jazyků. V jádře všech center byly vylepšené počítače Whirlwind, navržené MIT a vyráběné IBM. Vylepšení se týkala především příchodu jednoduché vektorové grafiky a také prvního ukazovacího zařízení nazývaného "light pen", kterým mohli operátoři signalizovat a vybírat možné trajektorie letadel. A byl to právě SAGE, kde s největší pravděpodobností vznikl první počítačový obrázek.



Obr. 2. Prvni rafický obrázek na počítači SAGE

Jedná se o překreslenou siluetu pin-up girl z magazínu Esquire, která se používala jako diagnostický test zobrazování vektorové grafiky. Není to jen první příklad počítačové grafiky, ale dost možná také první digitální pornografie. SAGE byl světově jeden z nejambicioz-

nějších počítačových projektů vůbec, pracovalo na něm najednou okolo 800 programátorů a vyžadoval výrobní sílu mnoha velkých amerických společností. Byl to také velký úspěch, SAGE předešel očekávání a poslední Whirlwind počítače byly vyřazeny až v roce 1983, více než dvacet let po jejich spuštění. Není tedy divu, že lidé, co pracovali na SAGE, se později zasloužili o mnoho dalších zásadních pokroků na poli počítačových techonolgii. Například tým, který pracoval na síťovém propojení SAGE základen, později vytvořil známý ARPANET, který se stal základem pro internet.

2.3 Sketchpad

Posledním významným projektem na poli počítačové grafiky, o kterém se chci zmínit, je počítač Sketchpad. Sketchpad stvořil Ivan Sutherland jako svoji doktorskou práci na MIT. Jednalo se opět o CRT obrazovku, na kterou se dalo kreslit pomocí speciálního ukazovacího zařízení. Sutherlandův výzkum byl natolik přelomový, že jeho práce je dodnes v srdci každého počítače. Avšak asi ještě důležitější je jeho vliv na to, jak lidé začali používat počítač. Stvořil první GUI (grafické uživatelské rozhraní). Pomocí přepínačů na ovládacím panelu mohl uživatel ovlivňovat to, jak bude tužka kreslit. Sketchpad umožnil uživatelům kreslit obrazce na obrazovce, aníž by před tím museli napsat kód, který by tyto obrazce stvořil. Objekty se následně daly zvětšovat, otáčet nebo posouvat. Sketchpad je dost možná také prvním exemplářem výskytu fuknce Undo (zpět). Sketchpad byl daleko více než hrubou interpretací kreslení na papír v digitálním světe. Byla to od základu jiný přístup k designu a ovládání počítačů.

2.4 Od programátorů k umělcům

Vývoj rychle pokračoval, obzvlášť na poli 3D grafiky. Grafická rozhraní se stále zdokonalovala až do dnešního momentu, kdy uživatel vůbec nepotřebuje umět programovat, aby mohl ovládat počítač. Vyvíjely se i programovací jazyky, začaly vznikat tzv. vysokoúrovňové programovací jazyky, které jsou přístupnější a lze s nimi psát kód daleko rychleji (to vše na úkor efektivity/rychlosti). A díky tomu se možná tak trochu kruh uzavírá a stále více lidí se obrací zpet k počátkům počítačů. Tedy programování.

3 HISTORIE KREATIVNÍHO KÓDU

Programovací jazyky byly navrženy vědci pro vědce. Tomu taky odpovídala filozofie a využití počátečních programovacích jazyků. Dostat vizuální výstup z běžných programovacích bylo komplikované a vyžadovalo to vysoké technologické znalosti, které trvá roky se naučit. Přitom cíle grafika, designéra nebo architekta jsou dost odlišné od toho, co chce dosáhnout vědec. Postupem času se situace začala trochu měnit. Někteří vědci začali vyvíjet systémy ke generování grafického výstupu (například k vizuální interpretaci dat), někteří dokonce spolupracovali se stávajícími umělci a začali experimentovat. V roce 1963 Kenneth C. Knowlton vytvořil BEFLIX, specializovaný program na tvorbu animací. Použil ho k tvorbě prvních počítačem generovaných filmů, které tvořil ve spolupráci s umělci Stanem VanDerBeekem a Lilian F. Schwartzovou[1]. Dalším filmem vytvořeným čistě pomocí počítače byl film Permutations z roku 1966 od Johna Whitney staršího. K tvorbě Permutations bylo využito knihovny GRAF vyvinuté v IBM. GRAF i BEFLIX jsou postaveny na programovacím jazyku Fortran. Z těchto a jiných počátečních experimentů se začal tvořit zájem o programovací jazyky navržené k tvorbě umění. Tato energie se postupně přenesla až do dnešní doby, kdy je o programování v umění velký zájem. Boom v programování mezi "neprogramátory" přinesl Apple v roce 1987. Šlo o programovací rozhraní jménem HyperTalk. HyperTalk byl velmi podobný normální angličtině a jeho jednoduchost použití způsobila, že to byl velmi oblíbený nástroj u zvídavých dětí (a odrazový můstek pro mnoho dnešních předních programátorů). Avšak prvním programovacím jazykem hojně využívaným designéry a umělci bylo Lingo z roku 1988. Jazyk speciálně vyvinutý pro Adobe Director (tehdy Macromedia Director), v mnohém se inspiroval HyperTalkem a designéři ho využívali, protože to byl jeden z hlavních nástrojů k tvorbě multimediálních projektů, CD a video-kiosků. S nástupem internetu se měl Adobe Director stát hlavním přenašečem interaktivity na webu, ale jak se později ukázalo, jeho jádro a soubory byly na tehdejší rychlosti internetu moc veliké, tak se velké oblibě dostal Flash, jenž je znatelně kompaktnější. Adobe Flash (dříve Macromedia Flash a ještě dříve FutureSplash Animator) s jeho programovacím jazykem ActionScript byl jedním z nejoblíbenějších nástrojů na zpracování multimédií a obecně na kreativní programování vůbec. Flash je dnes na ústupu, hlavně díky nástupu HTML5 a vzniku alternativních nástrojů (které jsou zadarmo), nicméně zasloužil se o velký rozvoj kreativního programování.

1REAS, Casey a Chandler MCWILLIAMS. *Form code in design, art and architecture: guide to computational aesthetics*. New York: Princeton Architectural Press, 2010, 176 s. ISBN 978-1-56898-937-2.

3.1 Aesthetics + Computation

Aesthetics + Computation bylo jméno známého výzkumného kroužku na MIT Media Lab[1] který začal v roce 1996. Bylo to zde, kde studenti Ben Fry a Casey Reas pod vedením Johna Maedy vytvořili programovací jazyk Processing[2]. Práce Johna Maedy stojí na pomezí designu a technologie. Sám začínal jako počítačový vědec, vystudoval MIT, později se probudil jeho zájem o grafický design a estetiku. V Japonsku vystudoval Tsukuba University se zaměřením na design. Jeho práce je zásadní, jako jeden z prvních měl kuráž propojovat grafický design s programováním a tím položil základy kreativního programování v designu. Dost pravděpodobně napomohl popularizaci pojmu kreativní programování, díky svojí knize Creative Code[3] z roku 2004. Později se stal vůdčí osobností a obličejem hnutí, které propaguje propojení programování a designu. Není tedy překvapením, že to bylo právě zde na MIT, škole která stvořila to jak dnešní počítače vypadají a fungují, pod Johnem Madeou, kde vznikl Processing. Díky Processingu dnes kreativní programování v designu zažívá velký rozkvět a nemalou zásluhu na tom všem má John Madea.

3.2 Kreativní programování

Kreativní programování není žádný oficiální pojem, technicky se neliší od programování normálního. Je to obecně přijmaný název, který naznačuje, že jde o programování využívané v umělecké praxi. Nejčastěji se jedná o interaktivní instalace a video, rozsáhlé projekce na budovy, projekty počítačové vize nebo interaktivní zvukové instalace. Další využití názvu „kreativní programovaní“ se týká projektů, kde nejde o program samotný, ale jen o jeho výstup. U takovýchto projektů je technická i formální kvalita na druhém místě, prioritou je jen výstup. Uživatelské rozhraní nemusí existovat a v tuto chvíli se textový editor vlastně stává uživatelským rozhraním a kód je hlavním přenašečem nápadů a kreativity.

3.3 Současné digitální nástroje versus kreativní programování

"Počítač je intelektuální a aktivní partner, který pokud je naplno využíván, dokáže vytvořit naprosto nové druhy umění a nové estetické zážitky."[4]

V digitálním umění a designu obecně dominuje software od jedné firmy… Adobe. Fotografové se neobejdou bez Adobe Photoshop a Adobe Lightroom, grafici nestvoří knihu bez Adobe Indesign, ilustrátoři připravují svoje ilustrace na tisk pomocí Adobe Illustrator,

1. MIT Media Lab. [online]. [cit. 2013-05-03]. Dostupné z: http://www.media.mit.edu/
2. GREENBERG, Ira. *Processing: creative coding and computational art*. New York: Distributed to the book trade worldwide by Springer-Verlag, c2007, s. 35. ISBN 15-905-9617-X.
3. MAEDA, John. Creative code. New York, N.Y.: Thames, 2004, 239 p. ISBN 05-002-8517-9
4. REICHARDT, Jasia. *Cybernetics, art, and ideas*. Greenwich, Conn.: New York Graphic Society, 1971, s. 143. ISBN 0821204319.

motion design nejde dělat bez Adobe After Effects a Adobe začíná pomalu dominovat na poli střihu nebo třeba psaní scénářů. Adobe je standard a v podstatě neexistují alternativy, pokud ano, tak jsou skoro vždy jen levnější imitací, která je pozadu. Adobe si vybudovalo skálopevnou pozici a stalo se definitivní odpovědí na všechny designérské potřeby. Tato odpověď je standardizovaná a marketing nás přesvědčil o tom, že s jejich softwarem jde udělat jakýkoliv představitelný vizuální vjem. Teoreticky to nebude daleko od pravdy, výstupem je vždy jen velké pole pixelů a pokud s nimi mohu jednotlivě manipulovat, mohu dosáhnout všech možných řešení. Nicméně software sám o sobě inspiruje a vede k novým postupům, stejně jako určitá technika a technologie u umění tradičního. Pokud by všichni malíři měli stejná plátna, barvy a štětce, svět umění by byl ochuzen o spoustu šťastných náhod a přístupů, které by se jinak stěží mohly stát. Ilustrovat to, jak software hluboce ovlivňuje výstup a přístup naší práce, lze každý rok a půl při příchodu nových verzí softwaru od Adobe. Komunita tvůrců očekává nové schopnosti jejich softwaru s velkým nadšením. Z nových možností se téměř okamžitě stanou trendy a tak to jde každý vývojový cyklus. Pro příklad, do nedávna bylo složité zpracovávat 3D grafiku s produkty Adobe, s nástupem nových verzí tento nedostatek Adobe napravuje a s tím přichází vlna nesmyslného nadužívání 3D. Někdy dokonce funguje software proti nám, zkuste například nasázet poházený, nepřesný text v Adobe InDesign. Je paradoxní, jak moc velkou námahu museli designéři v minulosti vyvinout, aby měli opravdu přesně vyrovnanou typografii, dnes musíme vyvinout velkou námahu, abychom typografii vyrovnanou neměli. Tak moc námi použitý software ovlivňuje naši práci. Programování je cesta velmi odlišná, žádný software neexistuje, takže není od čeho se odrazit. Odlišné programovací jazyky mají sice svoje specifika a knihovny (předpřipravené části programu), ty mohou ovlivnit výsledek naší práce, ale vliv je oproti tradičnímu softwaru naprosto marginální. Jde totiž o elementární schopnosti jako přečíst určitý formát nebo udělat linii z bodu A do bodu B. Pokud chceme v dnešní digitální době přinést novou nebo unikátní vizi, je potřeba, aby se designéři a umělci dostali za limity existujícího softwaru. Tradiční software jsou obecné nástroje navržené k výrobě očekávaných a standardních výstupů. Když už využíváme software pro naši práci, proč se zastavit a limitovat vizí a schopnostmi softwarové společnosti nebo programátora? Abychom mohli jít dále za tyto limity, je potřeba upravit existující aplikace pomocí programování nebo si napsat celý vlastní software.

3.4 Kde počítač exceluje

Počítače jsou neuvěřitelně rychlé. Dokáží najednou zpracovávat obrovské množství dat a tím nám ulehčit práci. Opakování, modularita, simulace a parametrizace jsou nejsilnějšími

důvody, proč zvažovat použití počítačů v designu.

3.4.1 Repetice a modularita

Dnešní průměrný počítač zvládá přes 2 miliony operací za vteřinu, to z něj dělá skvělý nástroj pro tvorbu repetice a opakujících se, jinak časově náročných prací. Krása modularity programování je v tom, že pokud program rozdělíme na menší samostatně fungující podčásti, můžeme kód znova využívat jinde[1]. Není náhodou, že animace byla jedním z prvních uměleckých oborů, kde se počítače ukázaly býti velmi užitečnými. V animaci má každá vteřina 24 snímků, takže udělat 5 minutovou animaci znamená vytvořit 7200 odlišných obrázků, kde jsou změny mezi jednotlivými obrázky minimální. Pokud jsme schopni počítači popsat, jak by měl pohyb animace vypadat, tak za nás odvede mnoho zdlouhavé práce. Když to vezmeme o krok dále, můžeme se pokusit stvořit jednoduchý fyzikální simulátor, který bude počítači popisovat změny za nás. Síla modularity přichází ve chvíli, kdy zmíněný fyzikální simulátor navrhneme jako oddělený podprogram, protože jej následně můžeme využít ve všech našich animacích.

3.4.2 Parametrizace

Parametrizace[2] nastává ve chvíli, kdy designér přestane přemýšlet o jednotlivém objektu, ale o nekonečném poli možností. Vytváří populaci jednotlivých návrhů, které splňují dané požadavky a určitým způsobem se chovají. To je dáno parametry. Mezi vytvořenými možnostmi poté designér hledá tu nejlepší. Bežným parametrem může být například maximální cena. Pro příklad, designér může prozkoumávat to, jak fungují různé barevné palety při návrhu loga. V tomhle případě jsou barevné částí loga parametrem a seznam možných barevných palet rozmezím možností. Jak zvyšujeme počet parametrů, zvyšuje se také počet možných výsledků. Tento způsob návrhu se může zdát být zbytečně komplikovaný, ale většina designéru jej aplikuje přirozeně, jen ne využitím počítače a ne v takové míře. Tím, že si definujeme parametry a soustředíme návrh kolem jejich vzájemných vazeb, zajistíme celkovou soustředěnost na hlavní cíle. Pokud se do procesu zapojí i počítač, designér je schopný vygenerovat velmi rychle velké množství návrhů, a co je důležitější, tento způsob návrhu často vede k nečekaným kombinacím a propojením, které by jinak návrhář mohl minout. Parametrický způsob designu se dnes uplatňuje hlavně v architektuře, jako parametry se použijí data o pozemku, o tom, jak by měl být dům velký, kolik by mělo být kde

1 Pozn. Není myšleno jako objektově orientovane programováni. Podle úhlu pohledu, můžeme říci, že modularita v programování je buď jednodušší formou objektově orientovaného programování a nebo naopak jeho vyšším stupněm, přístupem k samotnému designu programu.

2 REAS, Casey a Chandler MCWILLIAMS. *Form code in design, art and architecture: guide to computational aesthetics*. New York: Princeton Architectural Press, 2010, s. 95. ISBN 978-1-56898-937-2.

světla, jaký by měl být tvar budovy atd. Pokud shromáždíte dostatečné množství parametrů, počítač je schopen poskládat podle něj nejefektivnější rozložení budovy velice rychle. Tento návrh nebývá konečný a stává se jen mezikrokem v celém procesu. Postupně se parametrický způsob návrhu dostává i do dalších odvětví designu a stává se jedním z nových přístupů a hnacích motorů inovace.

3.4.3 Simulace

Hlavní schopností počítače jsou velmi přesné výpočty. Studováním přírody — opět za pomocí parametrů (bez těch se simulace neobejde) — jsme schopni simulovat to, jak se budou objekty chovat v reálném světě. Tradičním využitím simulace je meteorologie nebo zátěžové simulace materiálu. Zde jde čistě o přesnost a realitu naší simulace. Jiným využitím může být, když zapojíme biologii a zjistíme principy toho, jak rostou např. korály, jsme schopni tuto znalost vzít a s její pomocí doslova nechat vyrůst korálovou strukturu. Tímto způsobem jsme schopni vytvořit objekty, které mají všechny vlastnosti korálů, ale můžeme je využít novým způsobem. Tím však možnosti simulace nekončí, simulace postavená na přesnosti a zákonech reálného světa je ve výzkumu nutná, v umění a designu však tato realita tak důležitá není. Ohýbání a měnění pravidel přírody může přinést něco velmi nečekaného a nového. Simulace může být nástrojem přesnosti, ale také základem pro něco daleko za hranicí reality.

3.4.4 Co počítače nedokážou

Počítače za nás žádnou práci neudělají, jen nám můžou pomoci a ulehčit. Nedokáží samostatně přemýšlet, to znamená, že vše, co počítače dělají, musí být dopředu promyšleno tím, kdo je naprogramoval. Při tvorbě programu se tedy musí autor zabývat nejen tím, co má program vykonat když se stane A, ale také tím, co se bude dít, když A nenastane. Pokud chceme dosáhnout kvalitního programu, musíme se naučit počítat s nečekaným. Vetšinou to dopadá tak, že programátor musí trávit daleko delší dobu úpravou programu, aby fungoval i v neideálních podmínkách než mu zabral samostatný návrh a naprogramování funkcionality.

3.5 Využití kreativního programování

Kreativní programování je v zásadě natolik nový obor, že lze těžko říci, kde všude najde svoje využití. Soudě podle toho, jak moc důležitým se stalo programování v neuměleckých oborech, lze v následujících letech očekávat silný růst lidí, kteří začnou využívat programování ve své designérské práci. Do dnešního dne lze za nejdůležitější příklady

kreativního programováni považovat:

- Interaktivní instalace
- Vizualizace dat
- Projekce na velkých show a koncertech
- Video mapping
- Generovaná grafika a tiskoviny
- Projekty počítačové vize (rozpoznáváni obrazu)
- Generování 3D objektů
- Na míru vytvořené aplikace
- Interaktivní webové projekty
- Nová uživatelská rozhraní (analogové i digitální)
- Ovládání strojů
- Tvorba animace

Většina projektů spojuje více zde zmíněných prvků, například interaktivní instalace, které propojují video mapping s na míru vytvořeným ovladačem (uživatelským rozhraním) a počítačovou vizí.

4 PŘÍKLADY A VŮDČÍ OSOBNOSTI

Výběr nejlepších projektů a největších současných průkopníků na poli designu s využitím kreativního programování je velice subjektivní. Záměrem následujících řádků je shrnout ty nejvýraznější s ohledem na jejich obor.

4.1 Osobnosti a studia

4.1.1 Robert Hodgin (Flight404)

Tvůrce interaktivních animací, simulací a generované grafiky. Mezi jeho nejdůležitější projekty patří vizualizátor hudby v Apple iTunes nebo například práce pro CERN (Evropská organizace pro jaderný výzkum), kde měl za úkol popularizovat vědu pomocí nových technologií. Obzvlášť zajímavé je také to, že nemá žádné formální vzdělání v designu nebo programování. Samostudiem se z něj stala hvězda Processingové komunity. Využívá programovacích jazyků Processing a Cinder.

4.1.2 Nervous systems

Studio skládající se z Jessicy Rosenkrantzové a Jessieho Louis-Rosenberga založené v roce 2007. Jessica Rosenkrantz vystudovala architekturu a biologii na MIT a později také architekturu na Hardward Graduate School of Design. Jessie Rosenberg vystudoval na matematiku na MIT. Dohromady studují přírodní principy růstu v rostlinách a živočiších. Ze svých poznatků poté vytvářejí algoritmy, simulace, k tvorbě nových výrobních procesů užívaných při výrobě šperků, lamp, váz a nábytku. Jejich výtvory se pak vyrábí pomocí 3D tisku. Ke své práci používají programovacího jazyku Processing.

4.1.3 Marius Watz

Abstraktní umělec využívající ke své práci programovací jazyk Processing. Jeho práce se zabývá softwarovou syntézou tvarů a vztahy mezi barvou, tvarem a pohybem. Je znám používáním ostrých geometrických tvarů a velmi výrazných barev. Jeho práce jsou vystavovány v galeriích po celém světě a jeho výstupem jsou projekce, velkoformátový tisk nebo objekty vytvořené pomocí 3D tisku. Je také zakladatelem unikátní konference Generator.X, která se zabývá rolí softwaru a generativních strategií v designu a umění.

4.1.4 Casey Reas

Spoluautor Processing současně využívá svůj software k tvorbě instalací, projekcí a tiskové grafiky. Jeho práce je velmi minimalistická, někdy jde jen o ukázku určitého procesu pomocí programu. Dalo by se říci, že jeho práce částečně navazuje na práci Johna Madey. Vystavuje po galeriích na celém světe, nadále se podílí na vývoji Processing a učí na University of California v Los Angeles. Je také spoluautorem několika knih o jazyku Processing.

4.1.5 Onformative

Designérské duo Cedric Kiefer a Julia Laub. Zabývají se hlavně klasickými tiskovinami a data vizualizacemi, které tvoří pomocí Processing. Dalo by se tedy říci, že využívají kreativní programování jako tradiční software. Jejich práce je však kontroverzní, pomocí parametrizace a generativních postupů tvoří automatické systémy na tvorbu knih a plakátu, čímž částečně zpochybňují roli grafického designéra v naší době. Jsou autory velmi ceněné knihy Generative-Gestaltung, která vysvětluje mnoho generativních technik a jejich použití v grafickém designu. Žijí v Berlíně a oba učí na různých školách po Německu.

4.1.6 Field

Londýnský digitálně designérský powerhouse zaměřující se na interaktivní instalace a tvorbu nových zážitků poháněných technologií. Experimentují s barvou, pohybem a zvukem. Jejich práce využívá ty nejnovější technologie a jsou známi svým vědeckým přístupem k práci. Jejich škála práce sahá od webových prezentací a filmů, přes interaktivní instalace pro galerie až po aplikace pro mobilní zařízení.

4.1.7 Universal Everything

Všestranné designérské studio z Velké Británie zaměřující se především na tvorbu klipů, reklam a interaktivních instalací. Využívají mnoho různých nástrojů od komerčního softwaru po kreativní programování. Jejich reklamní instalace pro automobilky zaznamenaly obrovský úspěch.

4.1.8 V Squared labs

Jedno z předních studií na tvorbu koncertních show a velkých reklamních projekcí. Jejich jádro tvoří VJové, kteří postupem času přijali do svých kruhů specializované programátory, animátory a architekty. Ke své tvorbě využívají platformy Touchdesigner nebo D3, díky

nimž mohou najednou ovládat nejen projektory, ale i světla, lasery a další koncertní techniku. Studio má sídlo v Los Angeles a dělá show po celém světě.

4.1.9 Matt Pearson

Autor soustředící se hlavně na generování obrazů pomocí Processing. Jeho kniha GENERATIVE ART: A PRACTICAL GUIDE byla jednou z prvních knih vysvětlujících, jak náhodně generovat komplexní obrázky, a přitom se sama nezabývala základy Processing. Důležitá je jeho druhá kniha Novelty Waves. Provokativní esej a kritika současného digitálního umění zabývající se např. Opensource uměním a náměty jako "Proč nikdo nikdy nebrečel nad webovou stránkou".

4.1.10 Daniel Shiffman

Společně s tvůrci Processing je to nevýraznější postava v učení kreativního programování pomocí Processing. Shiffmanova první kniha jménem Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction je detailním vstupem do světa kreativního programování. Jeho druhá kniha The Nature of Code sklidila velký úspěch u Processing komunity. Je zaměřená na pokročilejší techniky a využití matematiky při kreativním programování. Knihu dokonce doprovází videopřednášky, které je možné streamovat z portálu vimeo.com. Nature of Code byla financována na Kickstarteru a dnes je volně přístupná všem ke stažení.

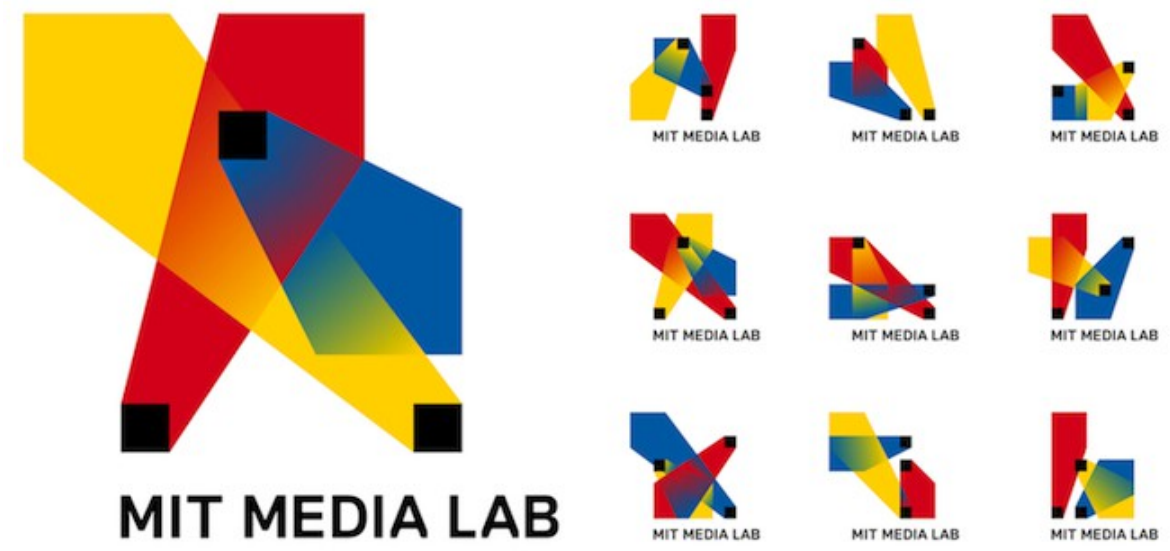
4.1.11 Jer Thorp

V současnosti jeden z nejznámějších tvůrců data vizualizací. Jeho práce shlédnou každý měsíc tisíce lidí, protože pracuje pro The New York Times Company Research & Development Lab, což je odnož New York Times, která hledá nové perspektivy na velké objemy dat, a později ze svých poznatků tvoří data vizualizace do těchto světoznámých novin. Thorp umí odhalit mnohdy jinak naprosto skrytá fakta o světě kolem nás. Jeho vizualizace jsou součástí mnoha knih a sborníků, například Data Flow nebo Beautiful Visualisation, což samo o sobě mluví o vysoké kvalitě jeho práce.

4.2 Typové příklady kreativního programování

4.2.1 MIT Media Lab identity

V posledních pěti letech vzniklo hned několik generativních identit, které si získaly oblibu: české logo Domu Umění v Brně, identita koncertní haly Casa da música od Stefana Sagmaistera nebo nečekaně progresivní logo pro město Burlney ve Velké Británii. Žádná z nich však nedosahuje kvalit vizuální identity MIT Media Labu z roku 2012. Není překvapením, že nejpokrokovější designérská vzdělávací instituce na světě chápe podstatu a smysl generované identity. Je to taky přesně sedící identita, komu jinému by také seděla generovaná identita více, než místu, kde má generovaný grafický design kořeny. Co ale staví identitu Media Labu před ostatní generované indentity, není silnější koncept, nýbrž čistě vizuální kvalita. Identita Media Labu působí svěže a velmi nadčasově. Má nádech retra a aplikace identity jsou chytré, účelné a krásné. Projekt byl vytvořen samozřejmě v Proces



Obr. 3. Identita MIT Media Labu

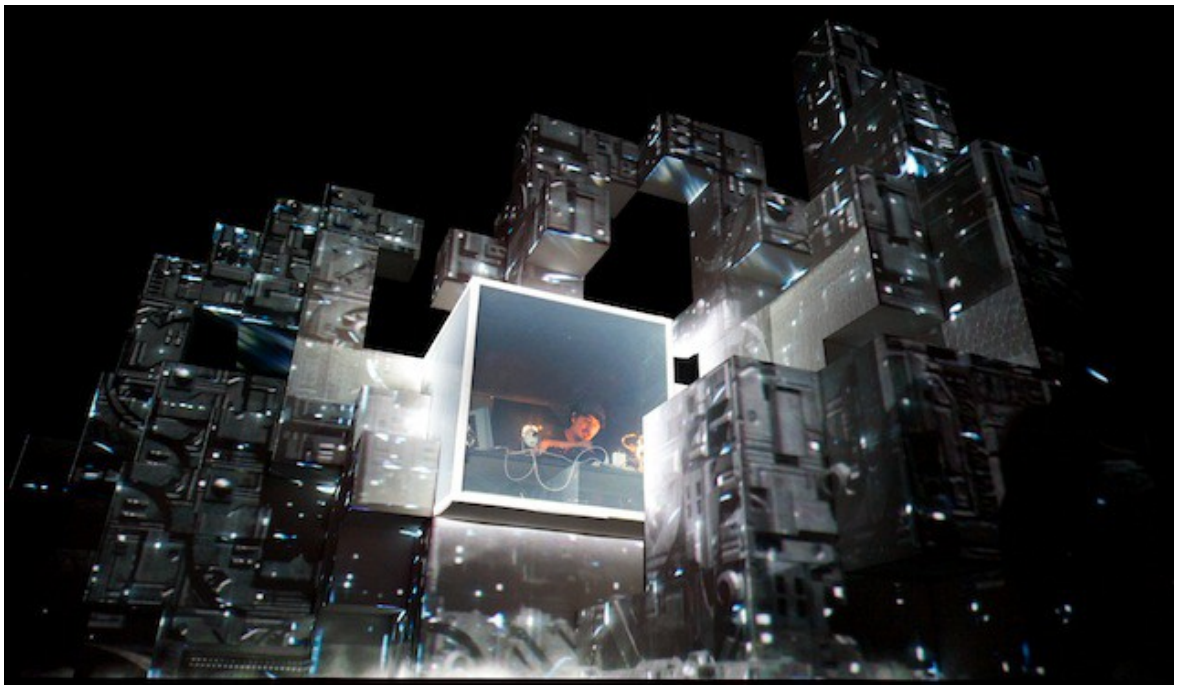
4.2.2 House of cards

V roce 2009 kapela Radiohead opět předběhla dobu a ukázala nám budoucnost. Rok před vznikem Microsoft Kinect vytvořil tým kolem Aarona Koblina hudební klip čistě pomocí skenování 3D prostoru (a míru provedenými laserovými čidly). Klip ukázal vizuální styl, který se o pár let později stal typickým pro práci s Microsoft Kinect. Klip je pokrokový v tom, že místo čistého videozáznamu nabídl divákovi interaktivní rozhraní a ten si tak mohl režii klipu řídit sám.

Obr. 4. Hudební klip House of Cards

4.2.3 Amon Tobin ISAM

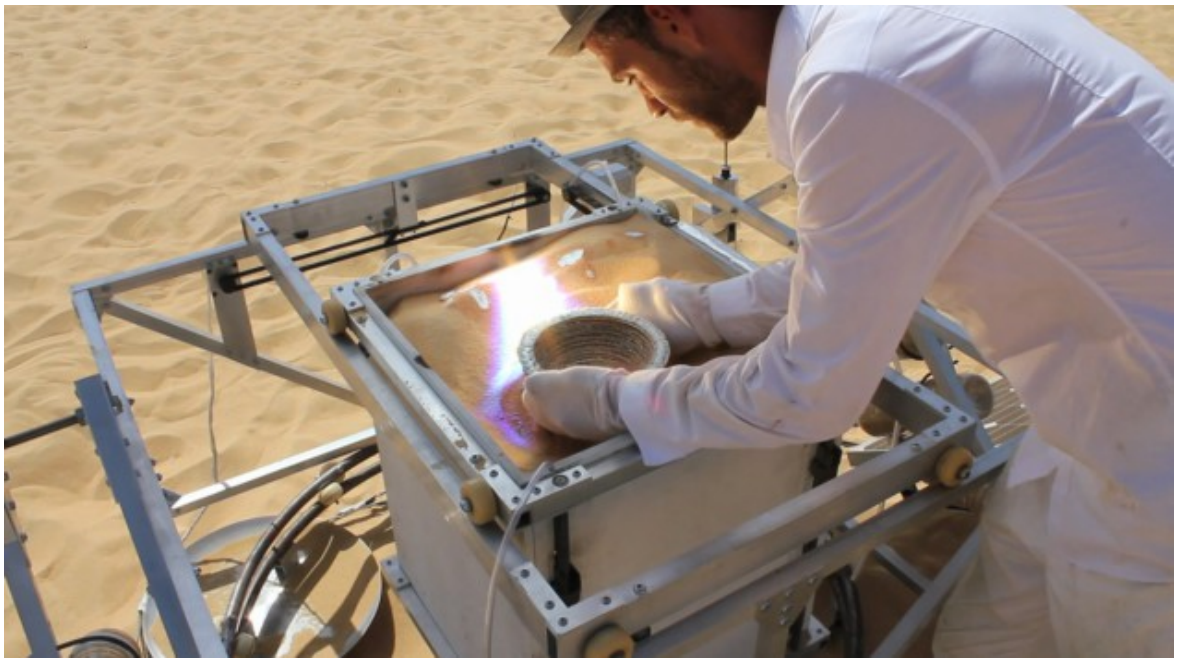
Audiovizuální show, která doprovázela hudbu Amona Tobina po celém světě. Jedná se o velkou krychlovitou fragmentovanou skulpturu, která má uvnitř kabinu, ze které Amon Tobin hraje. Stzdui V Squared Labs celou projekci vytvořili v TouchDesigneru. Projekt je speciální nejen komplikovaným využitím několika projektorů, ale také svým vznikem. Skulptura byla nejprve navrhnuta designovým studiem Vita Motus [www.vitamotus.com] a ještě před fyzickou výrobou se na ní připravovaly projekce. Designové studio zaslalo přesný 3D model studiu V Squared Labs a ti mapovali projekci na virtuální model, aniž by viděli výsledný objekt.



Obr. 5. Projekce Amon Tobin ISAM

4.2.4 Solar Sinter

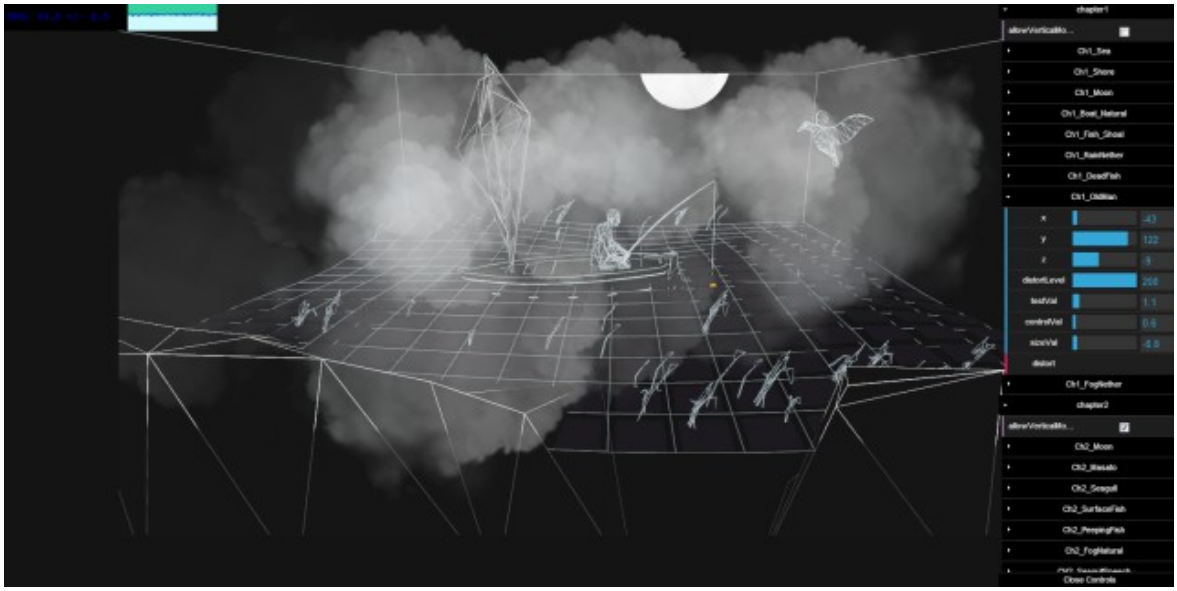
Studentský projekt Markuse Kaysera [www.markuskayser.com] je pouštní 3D tiskárna využívající jen sluneční energii a písek z pouště. Stroj je schopný nekonečné autonomní výroby skleněných předmětů, které vznikají tavením písku. Postavený je na open-source technologiích a velmi levné platformě Arduino. Projekt vzbudil velký ohlas a poukazuje na alternativní využití jinak neobyvatelných částí planety



Obr. 6. Solar Sinter

4.2.5 The Carp and The Seagull

Krátký umělecký interaktivní webový film. Web byl vyroben na zakázku pro The Creators Project a vyhrál mnoho cen. Mnoho lidí (například Matt Pearson ve své eseji No-one Ever Cried At A Website[1]) považuje tento Web za první, který je schopen v lidech vyvolat opravdové vřelé emoce. Jak Pearson zmiňuje, je to dáno především tím, že projekt nikdy nebyl podřízen technologii, jako jsou dnes všechny webové projekty. Autoři se jen snažili najít co nejlepší formu pro svoje vyjádření a náhodou se tím stala webová stránka.



Obr. 7. The Carp and The Seagull

4.2.6 Project Cascade[2]

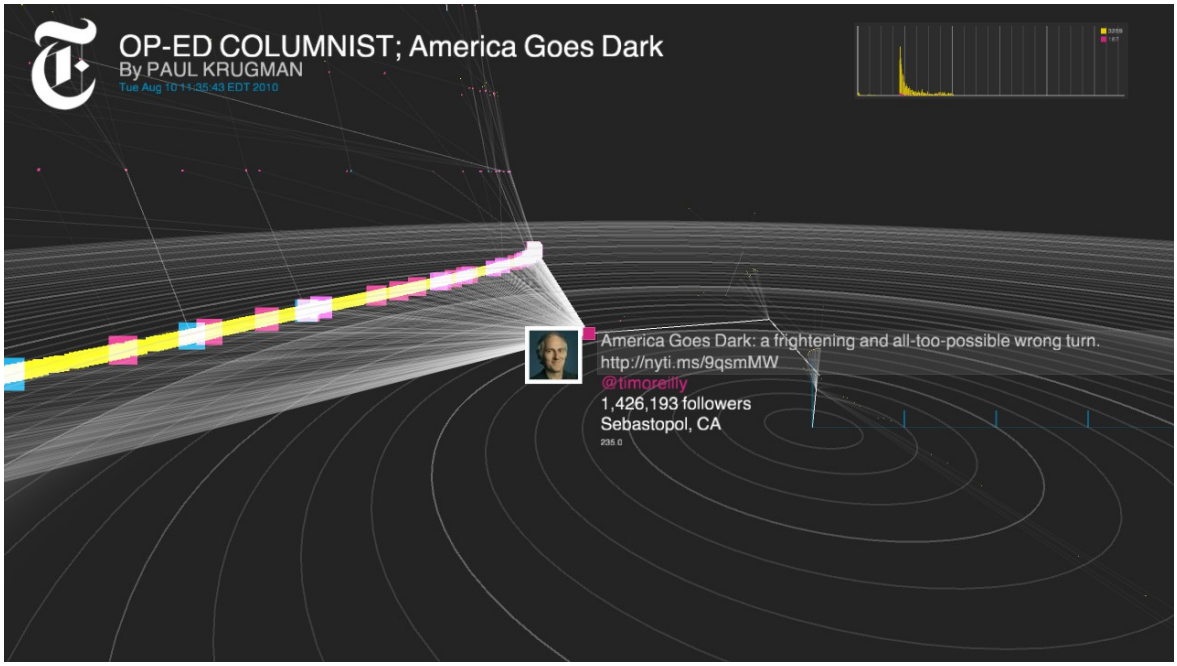
Interaktivní data vizualizace do galerie, kterou vytvořil Jer Thorp. Jedná se o ambiciozní vizualizační nástroj, který se snaží ukázat principy, jakými mezi sebou sdílíme informace. Vysvětluje jak putují tweety[3] zaměstnanců New York Times[4].

1 PEARSON, Matt. No-one Ever Cried At A Website. *Creativeapplications.net* [online]. [cit. 2013-05-12]. Dostupné z: http://www.creativeapplications.net/theory/no-one-ever-cried-at-a-website-matt-pearson/

2 THORP, Jer. Project Cascade. *THE NEW YORK TIMES COMPANY RESEARCH & DEVELOPMENT LAB* [online]. 2012 [cit. 2013-05-12]. Dostupné z: http://nytlabs.com/projects/cascade.html

3 *Twitter* [online]. 2006 [cit. 2013-05-12]. Dostupné z: https://twitter.com/

4 NEW YORK TIMES. *New York Times* [online]. 1996 [cit. 2013-05-12]. Dostupné z: http://www.nytimes.com/



Obr. 8. Project Cascade

4.3 Důležité portály a komunity

4.3.1 Creative Applications [creativeapplications.net]

Nejznámější portál zabývající se čistě projekty, které využívají kreativní programování jako svoji základní složku. Web se dnes vyvinul také do tištěného magazínu Holo [ww.holo-magazine.com], který vychází jednou za půl roku.

4.3.2 Visualizing [visualizing.org]

Novější portál zaměřený čistě na data vizualizace. Hlavním cílem autorů tohoto webu je pokusit se najít cesty, které lidem pomohou vyznat se v dnešním nadbytku informací.

4.3.3 Create digital motion [createdigitalmotion.com]

Sesterský web portálu createdigitalmusic.com. Zabývá se především VJingem a technologickými projekty spojenými s tvorbou hudby.

4.3.4 Creative coding podcast [creativecodingpodcast.com]

Web s pravidelnými podcasty zaměřenými na kreativní programování.

4.3.5 Resonate festival [resonate.io]

Stránky konference o kreativním programovaní a spojení umění, designu a technologie. Každý rok se zde objevují ty nejprogresivnější osobnosti.

4.3.6 Open processing [openprocessing.org]

Web, kde uživatelé Processing mohou sdílet zdrojový kód svých programů s ostatními. Openprocessing.org je jeden z důvodů, proč je Processing tak oblíbený, umožňuje uživatelům učit se od ostatních. Mnoho škol a workshopů sdílí práci studentů na tomto webu, stejně tak zde lze najít programy od profesionálů a známých jmen Processing scény.

4.3.7 The Creators Project [thecreatorsproject.vice.com]

Projekt financovaný společností Intel a magazínem Vice. The Creators Project sponzoruje různé umělce a designéry, aby mohli vytvořit projekty, které chtějí, a nemuseli se dívat na komerční úspěch. Velká část projektů je založena na kreativním využití technologie.

II. PRAKTICKÁ ČÁST

5 PROGRAMOVACÍ JAZYKY A NÁSTROJE PRO KREATIVNÍ PROGRAMOVÁNÍ

Existují stovky různých programovacích jazyků a mnoho z nich užívají umělci a designéři ke své práci. Každý jazyk je v něčem lepší a v něčem horší. Některé jsou si velmi podobné a spojuje je stejná filozofie myšlení. Vše se to odvíjí od jejich původu, vývoje a k jakému účelu byly stvořeny. Například jazyk C ze sedmdesátých let byl (a pořád je) jedním z nejrozšířenějších programovacích jazyků a mnoho jiných na něm staví nebo se inspiruje jeho filozofií. C++ je vylepšenou verzí C. Firma Apple využívá výhradně svůj programovací jazyk Objective-C, což je C s přidanými funkcemi od Apple (takže klasické C bude v Objective-C fungovat). Programovací jazyk Java je vylepšením C od firmy Sun Microsystems, ne mnoho se lišící od C++. Microsoft má také svoji mutaci C pojmenovanou C#. Všechny tyto jazyky jsou rozdílné, ale sdílí stejné kořeny a podobnou filozofii. Tak je to se všemi jazyky a obecně platí, že pokud člověk chápe filozofii jednoho jazyka, pochopit ostatní jazyky jemu příbuzné mu nebude dělat problémy. Programovací jazyky se dlouhodobě vyvíjí a nové jazyky si berou zaběhnuté konvence z těch předešlých a zároveň se je snaží vylepšit.

5.0.1 Vysokoúrovňový programovcí jazyk

Vysokoúrovňový programovací jazyk (neboli také vyšší programovací jazyk) je jazyk, který využívá větší míru abstrakce. To znamená, že čím "vyšší" programovací jazyk je, tím více se přibližuje myšlení člověka. Naopak nízkoúrovňové programovací jazyky se svým zápisem přibližují tomu, jak pracuje a přemýšlí počítač. Oba přístupy mají svoje výhody a nevýhody. Obecně platí, že čím vyšší programovací jazyk je, tím je pomalejší. Počítač musí kód vyššího programovacího jazyku přeložit do sobě pochopitelného nižšího jazyka. Vyšší programovací jazyky jsou zase lépe čitelné a jejich zápis je kratší. Tím může programátor pracovat rychleji a snižuje se počet míst, kde lze udělat chybu.

5.0.2 Důležité otázky při volbě programovacího jazyka

Pro kterou platformu náš program je? Je pro nás důležitá rychlost programu? Pokud tvoříme aplikaci na více platforem, je nutné, aby byla rychlá? To by znamenalo programovat aplikaci pro každou platformu znova. Nebo nám na rychlosti nezáleží a využijeme jazyk, který můžeme spustit na všech platformách? Hodí se náš jazyk k učení programování? Dají se v něm rychle zkoušet nápady? Otázek je mnoho. Díky tomu, jak jsou programovací jazyky rozdílné, je nezbytně nutné dobře vybrat jazyk, který použijeme.

5.1 Základní rozdělení a přístupy

Existují dva odlišné typy programovacích jazyků: textové jazyky a takzvané vizuální jazyky. Naprostá většina dnes používaných jazyků je textových, ale vizuální jazyky mají svoji váhu právě u umělců a designérů. Vizuální jazyky mají totiž viditelný výstup a programátor si může svůj program organizovat v prostoru. Mají vlastní GUI a přidávají další vrstvu zjednodušení pro programátora, který se nemusí zabývat technickými detaily. Na druhou stranu jsou tyto jazyky zpravidla pomalejší (jejich základem jsou textové jazyky), omezené a při větších projektech se GUI muže plést do cesty a dělat práci daleko komplikovanější. Vizuální programovací jazyky si našly oblibu především v tvorbě zvuku a také úpravě videa. Takzvané nodové editační programy jsou standardem ve světě editování videa a jde o typ vizuálního programovacího jazyka. Zde je nutné klást si otázku, do jaké míry zjednodušení vizuálního jazyka se pořád jedná o programovací jazyk a ne o vývojové prostředí. Odpověď se nedá přesně určit (je subjektivní) a je důležitá jen pro klasifikaci jednotlivých jazyků. Následující výčet programovacích jazyků rozhodně není kompletní, ale snaží se zkráceně mapovat dnes nejvyspělejší nástroje vhodné k použití v umění a designu.

5.2 Textové jazyky

5.2.1 ActionScript

Jazyk, který vznikl speciálně pro Adobe Flash. Flash je animační nástroj vyrobený především pro tvorbu interaktivního obsahu na webu. Umí velmi dobře pracovat s vektory a lze ho tedy použít ke tvorbě generované grafiky. Jazyk se pořád vyvíjí, ale odklon webu od Flashe (příchod HTML5/JavaScript) mu zkomplikoval budoucnost. Firma Adobe postupně mění účel ActionScriptu a dnes je díky platformě Adobe Flex možné ActionScriptem

programovat multiplatformní aplikace. Pro ActionScript existuje mnoho učebních zdrojů a knih, sama firma Adobe dělá učební materiály pro uživatele.

5.2.2 JavaScript

JavaScript byl vyvinut firmou Netscape pro jejich webový prohlížeč. I přes slovo Java ve svém jméně nemá JavaScript s Javou mnoho společného. Z jazyku vhodného jen na web se postupem času stal překvapivě univerzální programovací jazyk. Scriptographer [scriptographer.org], nástavba na Adobe Illustrator, umožňuje JavaScriptem programovat Adobe Illustrator a díky tomu zpracovávat vektorovou grafiku. Basil.js zase umožňuje JavaScriptem programovat Adobe Indesign. JavaScript lze také použít při programování Adobe Photoshop a Adobe After Effects. Programování grafiky na web ulehčuje mnoho knihoven. Paper.js [paperjs.org] na programování rastrové grafiky, Raphaël.js [raphaeljs.com] zpracování grafiky vektorové, Three.js [threejs.org] používá knihovnu k práci s 3D grafikou, nebo d3.js [d3js.org] (celým jménem Data-Driven Documents) knihovnu na tvorbu data vizualizací. Pomocí JavaScriptu lze dnes dokonce programovat webové servery díky Node.js. Díky tomu všemu je JavaScript jazykem s obrovským potenciálem a budoucností.

5.2.3 Python

Velmi všestranný vysokoúrovňový programovací jazyk, který je výborný jako první programovací jazyk. Obecně byl vždy skloňován jako jazyk vhodný k učení, postupem času se však vyvinul v jeden z nejvšestrannějších a nejsilnějších jazyků vůbec. I přesto, že programy v něm mohou být pomalejší, Python to vynahrazuje tím, jak rychle v něm lze vyvíjet. Velká část internetových aplikací od Googlu je napsána pomocí Python. Python v sobě nemá obsaženou grafickou knihovnu, ale grafiku s ním lze generovat pomocí externích knihoven. Lze s ním programovat webové servery (framework Django [djangoproject.com]), hry (PyGame [pygame.org]), mobilní aplikace (Kivy [kivy.org]) a mnoho dalšího. Externích knihoven pro Python je obrovské množství, dokonce existuje i pythonovská mutace jazyku Processing, jménem processing.py [github.com/jdf/processing.py]. Python si také osvojilo mnoho programů a vývojových prostředí jako svůj scriptovací jazyk (lze s ním ovládat program). Takovým softwarem je například opensource 3D grafický editor Blender [blender.org]. Dále s ním lze rozšiřovat schopnosti vizuálních programovacích jazyků Nodebox a Touchdesigner (viz. sekce Vizuální jazyky). Poslední a možná největší výhodou je množství kvalitních zdrojů k učení Pythonu.

5.2.4 Java a Processing

Jazyk Processing byl do této chvíle zmíněn několikrát, je to také jazyk, kterým byla stvořena projektová část této bakalářské práce. Přesto Processing není tak úplně samostatným programovacím jazykem. Processing je projekt, který se snaží zjednodušit programování v Javě, aby byl využitelný i umělci a designéry. Jedná se o vlastně o vrstvu nad Javou, která využívá mnoha Java knihoven a všechny je spojuje do propojeného celku. Uživatelům poskytuje také editor kódu (PDE - Processing Development Environment), se kterým lze rychle začít kód a později ho jednoduše sdílet a exportovat na web, nebo z kódu vytvořit aplikace. Síla Processingu spočívá právě v tom, jak rychle s ním lze začít, jak dobře se mu daří ulehčovat práci začínajícím programátorům a v neposlední řadě nápomocnou komunitou, s velkým množstvím učebních zdrojů.

5.2.5 C++

Programovací jazyk C vznikl jako nízkoúrovňový jazyk používaný pro programování operačních systému a ovladačů. Programy v něm napsané jsou velice rychlé, ale programování v něm je někdy zbytečně zdlouhavé. Proto vznikl jazyk C++, který se snaží zachovat rychlost C a zároveň urychlit vývoj. Na C++ jsou postaveny knihovny, které dramaticky ulehčují zpracování obrazu a zvuku, v porovnání s přímým C++. Je nutno však říci, že i přesto je kreativní programování pomocí C++ náročnější než ostatní zde zmíněné jazyky. Jedná se o knihovny zaměřené na profesionály a vznikají s nimi ty nejnáročnější projekty na světě. Všechny jsou také zadarmo a open-source.

5.2.6 OpenFrameworks [www.openframeworks.cc]

Nejpoužívanější knihovna na kreativní programování pomocí C++. Jedná se o obrovskou databázi uživateli tvořenými nástroji. To je její největší přednost a zároveň největší nevýhoda. Znamená to neustálý přísun nových nástrojů, ve kterých může být komplikované se vyznat.

5.2.7 Cinder [libcinder.org]

Tato knihovna byla vytvořena firmou The BarbarianGroup [barbariangroup.com], která ji poskytuje a vyvíjí jako open-source. Vychází z vývoje iTunes visualiseru. Filozofie Cinderu je jiná hlavně v tom, že se soustředí na maximální využívání systémových knihoven podporovaných platforem. Tím je kompaktnější, rychlejší a celkově čistší.

5.2.8 Polycode [polycode.org]

Novější C++ knihovna, která má jiný přístup než dvě předešlé. Kromě C++ ji lze programovat také jednodušším programovacím jazykem Lua a zároveň přitom využít rychlosti knihoven napsaných v C++. I přes relativní novotu, je to možná dobrá knihovna ke vstupu do kreativního programování pomocí C++.

5.3 Vizuální programovací jazyky a rozhraní

5.3.1 Nodebox [nodebox.net]

Rodina nástrojů, které vznikly v Belgii. Jsou postaveny speciálně pro grafické designéry. Nodebox 3 je vizuální programovací prostředí, které se zaměřuje na práci s vektory a fonty. Nodebox je velice přístupný, je výborným nástrojem na vstup do světa kreativního programování a jeho funkcionalita se dá rozšířit pomocí programování v programovacím jazyku Python.

5.3.2 Quartz Composer

Vizualní programovací jazyk od firmy Apple. Jeho vývoj se prakticky zastavil, ale pořád je možné ho nainstalovat na počítačích s OSX pomocí vývojářského balíčku Xcode.

5.3.3 Max/MSP/Jitter [cycling74.com/products/max/]

Vizuální programovací prostředí od firmy Cycling '74. Není zadarmo. Je vhodné hlavně pro tvorbu hudby a virtuálních hudebních nástrojů. Max je součástí programu na tvorbu hudby Ableton Live [ableton.com] a proto je oblíbenou volbou muzikantů.

5.3.4 PureData [puredata.info]

Open-source varianta Max/MSP od stejného tvůrce. Od Max/MSP se liší především neopracovaností grafického rozhraní a chybějící podporou.

5.3.5 vvvv [vvvv.org]

Tradiční vizuální programovací jazyk z roku 1998. Zadarmo pro nekomerční použití. Jeden z nejpoužívanějších balíčků na video mapping a VJing. Funguje jen pod platformou Windows a díky stáří svého jádra začíná být pomalejší než jiná řešení.

5.3.6 TouchDesigner [derivative.ca]

Novější vizuální programovací jazyk z roku 2008. Základem mu byl software na tvorbu vizuálních efektů Houdini [sidefx.com]. Dnes se oba programy už značně liší. Všechny komponenty v TouchDesigneru jsou akcelerovány grafickou kartou, tím pádem je schopen zpracovat opravdu veliké množství dat a je určitě nejrychlejším vizuálním programovacím jazykem k zpracování videa. Projekty, kde jeden počítač s TouchDesignerem ovládá 10 a více projektorů najednou, nejsou ničím neobvyklým. TouchDesigner je pro nekomerční účely zdarma.

6 SEZNAMOVÁNÍ SE S KREATIVNÍM PROGRAMOVÁNÍM

Je důležité začít s jazykem, který bude nováčky inspirovat, a ke kterému bude existovat velké množství učebních zdrojů. Dobrou volbou jsou určitě Processing, Python a Nodebox.

6.1 Užitečné zdroje

Všechny programovací jazyky mají svoji dokumentaci a fóra. Toto je seznam několika dalších ověřených kvalitních zdrojů.

6.1.1 The Nature Of Code [natureofcode.com, vimeo.com/shiffman]

Kniha Daniela Shiffmana je volně ke stažení a na jeho Vimeo profilu názorně vysvětluje příklady z jeho knihy. Je zde už přes 130 kvalitních lekcí o Processing.

6.1.2 MOOC [edx.org, udacity.com, coursera.org]

MOOC - massive open online course jsou otevřené internetové přednášky velkých zahraničních univerzit. Lze zde najít mimo jiné přednášky na téma různých programovacích jazyků zaměřených na nováčky i pokročilé.

6.1.3 MIT open-courseware [ocw.mit.edu]

Portál s přednáškami z MIT. Jejich kurzy programování jsou ty nejlepší na světě a jejich základní kurzy programování lze zvládnout bez předchozí programovací schopnosti. Hlavním účebním jazykem na MIT je všestranný Python.

6.1.4 Khan Academy [khanacademy.org]

Khan Academy, nezávislý e-learningový portál, má vlastní speciální ást zam enou na programování. Interaktivní u ení doprovázené videi. Vhodné i pro úplné za áte níky.

6.1.5 Code Academy [codecademy.com]

Nejkvalitnější portál na interaktivní učení programování. Obsahuje kurzy na jazyky jako Python nebo JavaScript a využívá herních prvků k motivaci studentů.

6.2 Doporučené knihy k jazyku Processing

6.2.1 Processing: A Programming Handbook

Rozsahem a hloubkou je tato kniha od samotných autorů Processing vysoce kvalitním zdrojem. Není strukturována jako učebnice s postupně komplexnějšími příklady, ale jednotlivé kapitoly se nezávisle zabývají důležitými praktikami, při programování s Processing.[1]

6.2.2 Learning Processing

Učebnice s klasickou linearní strukturou, postupně vysvětlující programování v processing. Jde o první knihu Daniela Shiffmana a autor sám předpokládá znalost materiálu této knihy, před postoupením k jeho další knize Nature of Code.[2]

6.2.3 Generative Design

Kniha zaměřená jen na praxi. Jde o soubor ukázek s příslušným kódem, kterým byly stvořeny. Vyrobeno studiem Onformative.[3]

1 REAS, Casey a Ben FRY. *Processing: a programming handbook for visual designers and artists*. Cambridge, Mass.: MIT Press, c2007, xxvi, 710 p. ISBN 02-621-8262-9.

2 SHIFFMAN, Daniel. *Learning Processing: a beginner's guide to programming images, animation, and interaction*. Boston: Morgan Kaufmann/Elsevier, c2008, xvii, 453 p. ISBN 01-237-3602-1.

3 HARTMUT BOHNACKER, Benedikt Gross, Claudius Lazzeroni EDITOR a Translated by Marie FROHLING. *Generative design: visualize, program, and create with processing*. New York: Princeton Architectural Press. ISBN 16-168-9077-0.

III. PROJEKTOVÁ ČÁST

7 PROJEKT: NÁHODNÝ_SEŠIT

Název tématu „autorská kniha“ může být v tomto případě značně zavádějící. Věru jedná se o malou knihu, která obsahuje pár pohledů a názorů autora. Na druhou stranu autor nemá v rukou to, jaký bude konečný výsledek. Pohledy, názory, pořebné znalosti i použité technologie jsou v mnoha směrech společnou snahou. Kniha by bez nepřímé podpory komunitou nikdy nemohla vzniknout, podpory, kterou může čerpat každý. Autor je v roli pozorovatele a vykonavatele. A je to právě tento proces, forma a jistá závislost na komunitě, která nejlépe vystihuje pohledy v knize obsažené. Způsob vzniku knihy tedy není jen formálním řešením, ale je důležitou součástí celkové zprávy, kterou se kniha snaží předat.

Dnešní mladá generace, příjmula internet jinak než ty ostatní. Tím, že s interentem vyrůstali, se internet stal jejich integrální součástí. Věci jako sdílení, pirátství a open source jsou pro ně neoddělitelnou součástí a tyto pojmy silně ovlivňují jejich myšlení. Dalo by se říci, že Náhodný_sešit, je zpovědí jednoho z takto silně ovlivněných lidí.

7.1 Pozice knihy v naší době

V době, kdy častěji než book slyšíme ebook, je význam papírové knihy velmi diskutabilní. Díky internetu lidé čtou víc než kdy dřív, paradoxně možnost volby je mnohokrát dovede k textům pochybné kvality. Překvapivě papírová kniha nikam nemizí. Naopak prodeje knih jsou zdravé a lidé knihy nepřestávají milovat. Knihy jsou totiž objektem s vlastním příběhem a vlastnostmi. Dokud nebudou zařízení na čtení digitální knih schopny měnit svoji formu podle obsahu, budou knihy pořád o krok napřed. Projekt se snaží zanechat tuto výhodu, kterou má papírová kniha oproti té digitální, a dokonce se tyto klasické hodnoty pokouší podpořit. Nejenom, že konzumentovi zůstane objekt, ale on s ním spojí svůj život. Tím, že bude originální, personalizovaná přímo pro něj a také tím, že se bude muset podílet na jejím vzniku, se z pěkného sériového objektu stává osobní artefakt. Konzument do ruky nedostane knihu, která jen čeká na to, až s ní spojí svůj život a přilne k ní, on je s ní svázán od první chvíle, co jí dostane do ruky. Pokud se tedy odhodlá ji vůbec zhotovit.

7.2 Open source

Celá kniha vznikla pouze pomocí open-source softwaru, její větší část dokonce vznikala na open-source operačním systému Linux. Fascinující je fakt, že prakticky všechna práce, která umožnila vznik nástrojů použitých k tvorbě knihy, je naprosto doborvolná a jen na popud svobodné vůle. Vůle, která věřila, že společně dokáže více. Open source software je

7.3 Pirátství

Open-source má svoje zákonitosti a k jeho využití je nutno mít určitou vyspělost. Je paradoxem, že právě díky pirátství jsou lidé tuto vyspělost schopni zdarma získat. Pirátství je pro mnoho lidí cestou k open-source. Co je ještě paradoxnější, je to, že náš školní systém místo toho aby podporoval, vychovával a posouval studenty k této vyspělosti, se je naopak snaží uzavřít do klece závislosti na komerčním softwaru.

7.4 Do it yourself

Pojem Do it yourself tedy doslova „Udělej si sám“, je dnes v naší společnosti velmi oblíbeným, jen díky internetu. Lidé sdíleji geniální jednoduchá řešení k problémům, které by jinak vyžadovalo drahá řešení. S příchodem 3D tisku do domácností, se tento trend bude jen rozrůstat. Hobby výroba doma na koleně, přináší kromě úspor také velkou míru satisfakce, kterou nelze dostat pouhým zakoupením produktu. Náhodný_sešit jde v myšlenkách Do it yourself a snaží se podpořit domácí hobby výrobu. Tomu také odpovída formát Náhodného_sešitu, který si může doma vytisknout každý a bez velké námahy tak vytvořit vlastní knihu.

7.5 Technologie za projektem

Projekt je celý vytvořen pomocí programovacího jazyka Processing. Dále využívá několik knihoven. Jde například o knihovny na práci s vektory, nebo na export do formátu PDF. Velikou výhodou jazyku Processing, je fakt, že se dá jednoduše vyvíjet na všech platformách. Náhodný_sešit byl vyvíjen na operačních systémech Linux a OSX, s použitím služby Dropbox[1].

7.5.1 Problematické části

Nejproblematičtější částí projektu, není vznik doprovázejících generativních grafik. Problematická jsou řešení kompatibility a vůbec celková stabilita výsledného programu. Kromě grafik museli vzniknout části programu, které budou dělat stránky v PDF, další nutná část je funkční uživatelské rozhraní a nebo rozumný systém sazby textu. V programováni je to nakonec celkem bežné, programátor stráví 10 krát více času nad úpravou a zajištěním stability, než nad tvorbou toho co je v projektu důležité.

1 Cloudové úložiště více na https://www.dropbox.com/

7.6 Software jako kniha

Software se postupně vyvíjí a stejně tak je tomu u Náhodného_sešitu. Novější verze, budou řešit nejen techické obtíže a případné chyby. Stejně jak se budou měnit autorovi názory, bude možné, že se pozmění i sdělení Náhodného_sešitu. Toto verzování, může také znamenat, že starší počáteční verze budou možná lepší, svoji menší sofistikovaností. Každý vygenerovaný Náhodný_sešit s sebou bude nést i svoje číslo verze, ve které byl vygenerován.

Kvůli verzování je nešťastné přidávat k práci statický soubor (protože nejde zajistit aktuálnost verze) a proto je k ní přiložen návod, jak si Náhodný_sešit vyrobit s odkazem na stránku projektu.

Projekt Náhodného_sešitu bude vždy aktivní na stránkách autora tedy **www.krisa.cz**

ZÁVĚR

Svět designu je ve fázi dramatického rozkvětu. Díky informačním technologiím a 3D tisku, se rychle mění pozice designéra a jeho vliv na svoje okolí. Vznikají stále nová odvětví a nové způsoby práce v designu. Tato práce začala jako průzkum možnosti využití informačních technologií v grafickém designu a snaží se zachytit význam programování pro designéry v budoucnosti. Projekt doprovázející teoretickou část, je riskantní výzvou, především kvůli celkové technologické náročnosti, nicméně obohatil autora o nové neocenitelné zkušenosti a pohled na programování v umělecké praxi.

Designéři budou do budoucna muset přijmout nové role, které jim doba informačních technologií přináší. Na druhou stranu, informační technologie přináší naprosto nové, nekonvenční rešení a tím nový prostor pro designéry k realizaci. Je důležité položit otázku, jestli designéři, kteří začnou plně využívat racionální programování, tím také nepřijdou o část svých emocionálně kreativních schopností, protože budou nejprve řešit technologické problémy (a tím se limitovat). Čas tedy ukáže, jestli může být programování kreativním rozšířením mozku a nebo je jen formální technologickou překážkou nutnou k vyřešení.

BIBLIOGRAFIE

[1] FRY, Ben. *Visualizing data*. 1st ed. Sebastopol: O´Reilly, 2007, xiii, 366 s. ISBN 05-965-1455-7.

[2] GREENBERG, Ira. *Processing: creative coding and computational art*. New York: Distributed to the book trade worldwide by Springer-Verlag, c2007, xxx, 810 p. ISBN 15-905-9617-X.

[3] HARTMUT BOHNACKER, Benedikt Gross, Claudius Lazzeroni EDITOR a Translated by Marie FROHLING. *Generative design: visualize, program, and create with processing*. New York: Princeton Architectural Press. ISBN 16-168-9077-0.

[4] HOFMANN, Armin. *Methodik der Form- und Bildgestaltung: Aufbau, Synthese, Anwendung. Manuel de création graphique. Forme, synthèse, application. Graphic design manual. Principles and practice*. 3d ed.). Arthur Niggli, ([Für] England: Academy Editions, London, 1973, 200 p. ISBN 37-212-0006-3.

[5] KLANTEN, Robert. *Data flow 2: visualizing information in graphic design*. Berlin: Gestalten, 2010, 271 p. ISBN 978-389-9552-171.

[6] LIMA, Manuel. *Visual complexity: mapping patterns of information*. 1st ed. New York: Princeton Architectural Press. ISBN 15-689-8936-9.

[7] LUTZ, Mark. *Python pocket reference*. 4th ed. Sebastopol, CA: O'Reilly, 2009. ISBN 05-961-5808-4.

[8] MAEDA, John. *The laws of simplicity*. Cambridge, Mass.: MIT Press, c2006, ix, 100, [6] p. ISBN 978-026-2134-729.

[9] MÜLLER-BROCKMANN, Josef. *Grid systems in graphic design: a visual communication manual for graphic designers, typographers and theree dimensional designers*. Niederteufen: Verlag Arthur Niggli, 1981, 175 s. ISBN 37-212-0145-0.

[10] PEARSON, Matt. *Generative art: a practical guide using processing*. London: Pearson Education [distributor], c2011, xli, 197 p. ISBN 19-351-8262-5.

[11] REAS, Casey a Chandler MCWILLIAMS. *Form code in design, art and architecture: guide to computational aesthetics*. New York: Princeton Architectural Press, 2010, 176 s. ISBN 978-1-56898-937-2.

[12] REAS, Casey a Ben FRY. *Processing: a programming handbook for visual designers and artists*. Cambridge, Mass.: MIT Press, c2007, xxvi, 710 p. ISBN 02-621-8262-9.

[13] RUDER, Emil. *Typographie: Typography*. [4th rev. ed.]. Niederteufen: A. Niggli, 1982c1967, 220 p. ISBN 37-212-0043-8.

[14] SHIFFMAN, Daniel. *Learning Processing: a beginner's guide to programming images, animation, and interaction*. Boston: Morgan Kaufmann/Elsevier, c2008, xvii, 453 p. ISBN 01-237-3602-1.

[15] STEELE, Julie a Noah P ILIINSKY. *Beautiful visualization: [looking at data through the eyes of experts]*. 1st ed. Sebastopol, CA: O'Reilly, c2010, xvi, 397 p. ISBN 14-493-7986-9.

[16] YAU, Nathan. *Visualize this: the FlowingData guide to design, visualization, and statistics*. Indianapolis, Ind.: Wiley Pub., c2011, xxvi, 358 p. ISBN 11-181-4025-7.

[17] ZELLE, John. Python Programming: An Introduction to Computer Science. 2. vyd. Franklin, Beedle, 2010. ISBN 978-1590282410. 3

SEZNAM OBRÁZKŮ

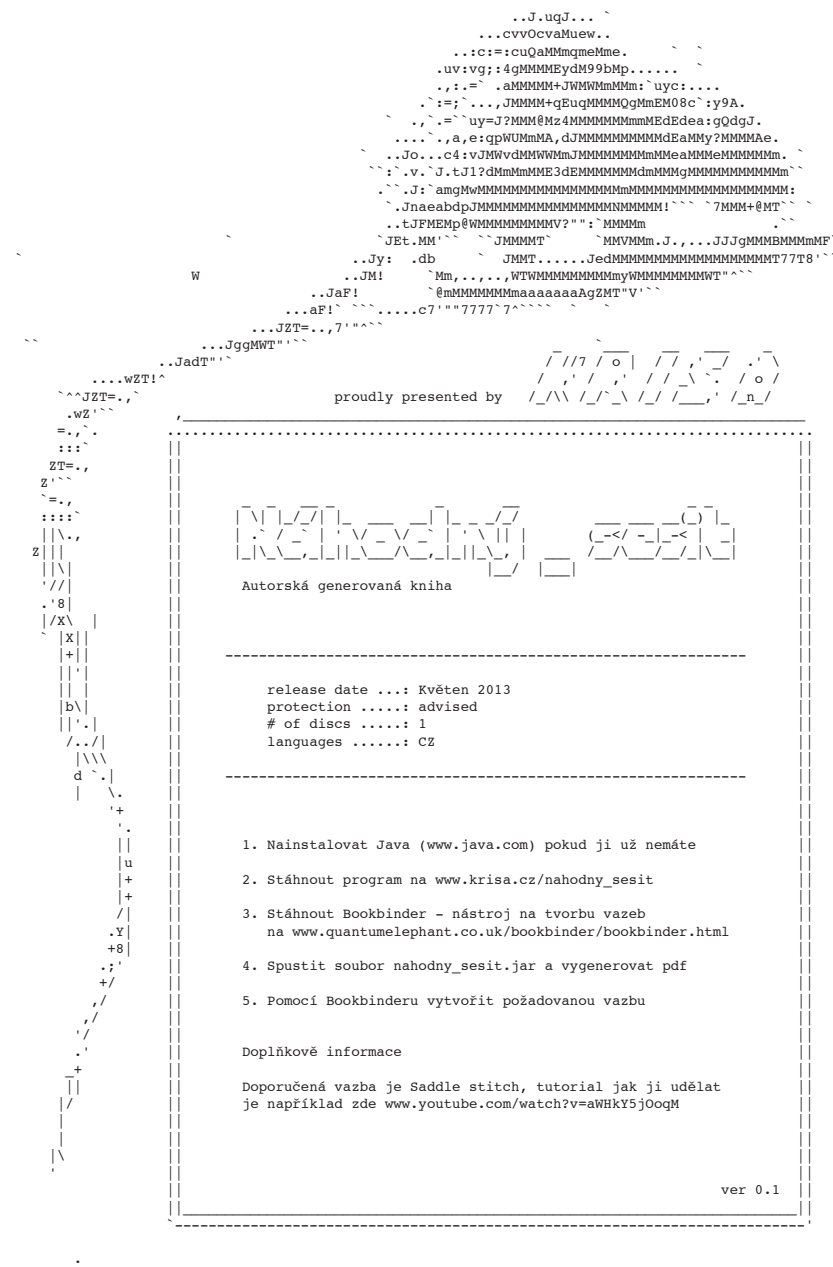
Obr. 1. Obrázek oscilogramu od B. Laposky..... 11
Obr. 2. Prvni rafický obrázek na počítači SAGE..... 12
Obr. 3. Identita MIT Media Labu..... 23
Obr. 4. Hudební klip House of Cards..... 24
Obr. 5. Projekce Amon Tobin ISAM..... 25
Obr. 6. Solar Sinter..... 25
Obr. 7. The Carp and The Seagull..... 26
Obr. 8. Project Cascade..... 27

SEZNAM PŘÍLOH

Příloha P 1: Návod projektu Náhodný_sešit.

Příloha P 2: Ukázky knihy Náhodný sešit

PŘÍLOHA P 1: NÁVOD PROJEKTU NÁHODNÝ_SEŠIT.



PŘÍLOHA P 2: UKÁZKY KNIHY NÁHODNÝ SEŠIT