FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Metaviz: Průvodce kreativním programováním
Student:	Bc. Radka Hošková
Vedoucí:	Ing. Radek Richtr, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem rešeršní části je mapování a popularizace kreativního programování, vizualizací a digitálního umění. Výstupem praktické části práce je webová aplikace Metaviz – digitální výstava dílčích implementovaných vizualizací.

1) Proveďte rešerši
   a) v oblasti digitálního umění a vizualizací;
   b) různých technologií pro kreativní programování.
2) Analyzujte vybraná řešení kladoucí si za cíl popularizaci, výuku, nebo sdružující komunitu kolem kreativního programování.
3) Navrhněte a implementujte
   a) alespoň dvě dílčí vizualizace vhodně demonstrující spektrum technických a uměleckých vizualizací;
   b) celkovou podobu webové aplikace, zobrazující tyto vizualizace.
4) Otestujte aplikaci, debatujte nad projektem a jeho možným pokračováním.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 3. února 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Metaviz: Průvodce kreativním programováním

Bc. Radka Hošková

Katedra softwarového inženýrství
Vedoucí práce: Ing. Radek Richtr, Ph.D.

3. května 2022

Poděkování

Karel Kryl - Děkuji

Stvořil Bůh, stvořil Bůh ratolest,
bych mohl věnce vázat,
děkuji, děkuji za bolest,
jež učí mne se tázat.

Děkuji, děkuji za nezdar,
jenž naučí mne píli,
bych mohl, bych mohl přinést dar,
byť nezbývalo síly.

Děkuji, děkuji, děkuji.
Děkuji, děkuji za slabost,
jež pokoře mne učí,
pokoře, pokoře pro radost,
pokoře bez područí.

Děkuji, za slzy děkuji,
ty naučí mne citu,
k živým, již, k živým, již žalují
a křičí po soucitu.
Děkuji, děkuji, děkuji.

Pro touhu, pro touhu po kráse
děkuji za ošklivost,
za to, že, za to, že utká se
láska a nevraživost.

Pro sladkost, pro sladkost usnutí
děkuji za únavu,
děkuji, za ohně vzplanutí
i za šumění splavu.

Děkuji, děkuji, děkuji,
Děkuji, děkuji za žízeň,
jež slabost prozradila,
děkuji, děkuji za trýzeň,
jež zdokonalí díla.

Za to, že, za to, že miluji,
byť strach mi srdce svíral,
beránku, děkuji, marně jsi neumíral.
Děkuji, děkuji, děkuji,
děkuji, děkuji.

Mé díky patří Ing. Radku Richtrovi, PhD. za vedení a pomoc při psaní diplomové práce. Díky patří všem, kteří se mnou práci diskutovali a podporovali mě. Závěrem děkuji Karlu Krylovi za složení zde citované písně.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 3. května 2022 ……

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Hošková, Radka. *Metaviz: Průvodce kreativním programováním.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Cílem této práce je strukturalizace dostupných informací o kreativním programování. Součástí práce je skupina interaktivních vizualizací, které je možné prohlížet včetně popisu ve webové aplikaci Metaviz.

Klíčová slova generativní umění, digitální umění, kreativní kódování, kreativní programování, umění nových médií, video textury, crown shyness

Abstract

The goal of this work is to structuralize available information about creative programming. A part of this work is a set of interactive visualizations, available with description in web application Metaviz.

Keywords generative art, digital art, creative coding, creative programming, new media art, video textures, crown shyness

Obsah

Seznam obrázků

Úvod

Teoretický fyzik Richard P. Feynman popisuje v knize *Radost z poznání* [1] rozepře se svým kamarádem umělcem o krásе květiny. Ohrazuje se proti umělcově tvrzení, že by vědci svou analýzou ubírali květině na krásе a naopak tvrdí, že studiem těchto vnitřních procesů ji na krásе pouze přidávají. Kromě vizuální krásy je totiž možné ocenit i důvtip přírody, jakým způsobem a efektivitou organismy fungují.

Tuto úvahu si dovoluji půjčit a rozvést jí dále v kontextu dnešních možností. Podobně jako se v kresbě využívají znalosti anatomie lidí či zvířat pro věrnější zachycení pózy, může počítačová grafika usilovat o co nejvěrnější zachycení přírodních jevů studiem jejich fungování. Dnes je možné nejen ocenit vizuální podobnu květiny a její procesy na různých úrovních, ale dokonce tyto procesy do jisté míry simulovat, alternovat, či vymýšlet a vytvářet vizualizace s vlastními zákonitostmi.

To, že se nejedná o revoluční myšlenky, dokazuje řada uměleckých prací simulující jevy přírody a algoritmů inspirovaných přírodou. Právě kombinace umění a technologie se potkává v umění nových médií a kreativním programování. Tato záliba se stává stále přístupnější veřejnosti, například díky grafické knihovně *Processing*, nebo nástroji pro vizuální programování jako *vvvv* a *Touchdesigner*.

V krátké eseji *I, Pencil* [2], popisuje Leonard E. Read, kolik věcí je potřebných pro výrobu obyčejné tužky a měřítko, ze kterého je patrná komplexita jednotlivých kroků, které ale nebyly nikým zadány a koordinovány. Podobně vznikla komunita digitálního umění a kreativního programování společně s dostupnými nástroji a materiály – spontánně snahou individuálních lidí věnujících se něčemu, co je baví a naplňuje. Z toho vyplývá, že neexistuje žádné nutné minimum pro možnost přispět komunitě a potenciálně každá snaha může mít vliv na definování této oblasti.

Cíle práce

Přestože je využití vizualizací často umělecké, není pravidlem, že by umělec a programátor musel být stejný člověk. Prioritním cílem práce je strukturalizovat dostupné informace o kreativním programování a zjednodušit tak programátorům i umělcům cestu pro ztvárnění jejich myšlenky. Dalším předmětem práce je i vytvořit vlastní přehledový set vizualizací využívající znalosti z rešeršní práce a vhodně je prezentovat.

Cílem práce není prezentovat kreativní programování jako něco převratného a přínosného pro každého. Odvětví jakými jsou informatika, počítačová grafika, umění a design by se pravděpodobně bez pojmu kreativní programování obešly. Kreativní programování inspiruje programátory k umění a umělce k programování, případně přispívá ke spojení programátorů a umělců k vzájemné kolaboraci.

Struktura

Diplomová práce je rozdělena na teoretickou a praktickou část. Úvodem teoretické části je Kapitola 1, kde se nachází vymezení pojmů, velmi stručný náhled do vývoje umění nových médií a příkladů algoritmů využitelných v kreativním programování. Následuje Kapitola 2 zaměřující se na instituce a komunitu, dále následuje Kapitola 3 obsahující rešerše technologií a Kapitola 4 rozebírá události festivaly a další vybrané projekty. Čistě výukové materiály pro tématiku kreativního kódování popisuje Kapitola 5 a závěrem teoretické části je Kapitola 6, obsahující příklady umělců a popularizátorů kreativního programování, kteří sdílejí části svého *know-how*.

Úvod praktické části patří vizualizacím. Kapitola 7 se věnuje vizualizaci *Video Textures* a Kapitola 8 vizualizaci *Crown Shyness*. Obě vizualizace jsou představeny, je definován jejich cíl a popsán jejich vývoj. V Kapitole 9 jsou stručněji popsány vizualizace menších rozměrů, vizualizace *Kensuke Koike* a *Painting*. Následuje část věnující se aplikaci *Metaviz*.[1] Kapitola 10 se věnuje analýze, Kapitola 11 návrhu a Kapitola 12 implementaci. Závěrem práce je Kapitola 13 věnující se testování, debatě nad využitím kreativního programování a možnému pokračování projektu. Součástí textu práce je příklad popisu vizualizace v aplikaci Metaviz dostupný v příloze C.

[1] Pro název *Metaviz* existuje trojice důvodů a žádný z nich nesouvisí se společností *Meta*. Přímočarými důvody pro název je podobnost s předchozí vizualizační iterací – bakalářskou prací *Fraviz* a faktem, že aplikace vizualizuje několik vizualizací. Hlubším důvodem pro název práce *Metaviz* je snaha o posuzování vizualizací za rámec prosté počítačové grafiky a analýza více vizualizací včetně obecných informací o nich.

Část I

Teoretická část

KAPITOLA 1

Kreativní programování

Kreativní programování je typ počítačového programování. Vyznačuje se tím, že jeho cílem je spíše než vytvoření něčeho praktického pro zjednodušení lidské činnosti, vytvoření něčeho expresivního či uměleckého. Nejedná se o vědní obor a nemá proto exaktní[2] definici. Kreativní kódování nezahrnuje pouze vizualizace, ale například produkování hudby, *VJing*, *videomapping*, *live coding*, či různé interaktivní projekty.

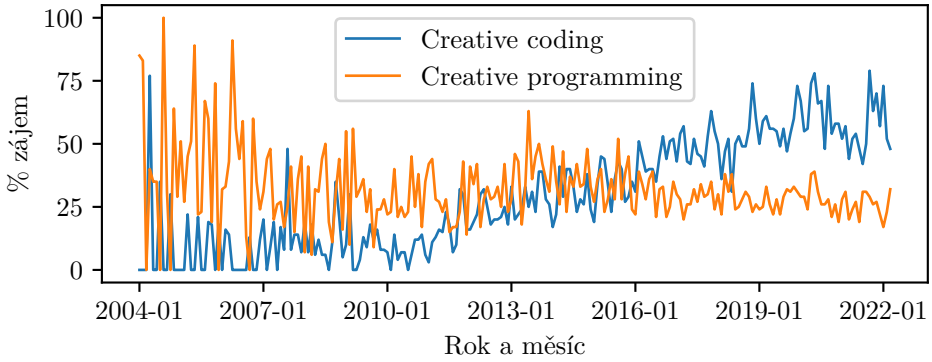
Kreativní programování si neklade žádné obecné podmínky pro výstup, užitečnost pojmu spočívá především v agregaci a vzniku technologií či materiálů za společným účelem – poskytnout jednoduchý způsob jak zkoumat určitou myšlenku pomocí počítače, ideálně s odstíněním co nejvíce technických záležitostí, které v daný moment nejsou předmětem zkoumání.

Kreativní programování vs. kódování

Vyskytují se termíny *kreativní kódování* i *kreativní programování*. V informatice je programování je proces od návrhu řešení problému pomocí výpočetní techniky ke spustitelnému počítačovému programu. Kódování je jednou z činností, který tento proces zahrnuje, tedy zápis zdrojového kódu v cílovém programovacím jazyce.

Celosvětové zastoupení těchto pojmů ve vyhledávání přes *Google.com* zobrazuje graf 1.1, kde osa y představuje počet vyhledávání vzhledem k nejvyššímu počtu vyhledávání, který je označen jako 100% (nejvyšší zájem byl v srpnu roku 2004 u kreativního programování). Přestože z grafu vyplývá *kreativní kódování* jako termín oproti *kreativnímu programování* na vzestupu, byl vybrán název programování pro jeho význam, tedy pro poskytnutí vyšší obecnosti. Data byla převzata z *Google Trends*.

[2]Nebo alespoň nebyla autorkou práce nalezena. Za vědní disciplínu se dá považovat *Creative Computing*.



Obrázek 1.1: Porovnání vyhledávání výrazů *creative coding* a *creative programming*

1.1 Pojmy

Paralelně se při spojení domén umění a technologií vyskytuje několik pojmů. Významově se tyto výrazy mohou částmi překrývat a existují mezi nimi nuance. Byly zkoumány často se objevující se pojmy i méně známé. Snahou této sekce bylo nalézt definice pro účely lepší orientace tématem.

Digital art

Digital art – digitální umění – je obecný termín pro umělecká díla, která využívají digitální technologie. Ty mohou být použity k jeho vytvoření nebo prezentaci. V dnešní době není nutné vytvářet vlastní nástroje, nýbrž umělec může používat mnoho programů pro vizuální umění i hudbu. Obecnějším pojmem do kterého patří digitální umění, je umění nových médií.

New media art

S vývojem technologií se mění kdysi tradičně používaná média. Například zpravodajství může místo papírových novin používat televize, pro ukládání dat je možné místo pevných disků používat *cloudová* uložiště a v umění je možné místo plátna používat digitální obrazovky. Lze se domnívat, že není snadné dosáhnout univerzální shody v otázce, co jsou to nová média.

Umění nových médií je velmi obecný pojem, popisující umění produkované elektronickými médii. Patří mezi ně například digitální umění, počítačová grafika, 3D tisk, robotika, videohry, interaktivní umění a další.

Computer art

Computer art je druh umění libovolně zahrnující počítače. Jak bylo zmíněno v knize [3], termín může konotovat éru průkopníků v této oblasti. Lze se domnívat, že na rozdíl od digitálního umění lze do computer artu zahrnout arte-

fakty vytvořené pomocí analogových počítačů a mechanických zařízení, jako například osciloskopu.

Generativní umění

Termín generativní či procedurální umění bývá používán například v kontextu algoritmického komponování hudby, počítačové grafiky a průmyslového designu či architektury. Obecnou definici poskytuje Philip Galanter z roku 2003: „generativní umění označuje jakoukoli uměleckou praxi, kde umělec používá systém, jako je soubor pravidel přirozeného jazyka, počítačový program, stroj nebo jiný procedurální vynález, který je uveden v pohyb a s určitou mírou autonomie přispívající nebo vedoucí k hotovému uměleckému dílu.“

Digitální morfogeneze

Digitální morfogeneze[3] je typ generativního umění, ve kterém algoritmus umožňuje získání komplexního tvaru či morfogeneze. Morfogeneze značí biologický proces, který způsobuje, že buňka, tkáň nebo organismus vyvinou svůj tvar, avšak později se pojem rozvinul do oblastí geologie, geomorfologie a architektury.

V kontextu digitálního umění by se pojem dal rozšířit na objevování tvarů, forem a vzorů pomocí výpočetního modelování a generativních systémů, založené na biologických, chemických a fyzikálních procesech. Je zde možné aplikovat výzkum za prakticky každého odvětví přírodních věd. Pro digitální morfogenezi existuje přehled[4] otevřený příspěvkům veřejnosti.

Creative Computing

Creative computing[5], volně přeloženo „kreativní výpočetní technika“ je mezioborová disciplína, kde se střetává kreativní umění a výpočetní technologie. Obor je možné (pod různými názvy) studovat na soukromých i veřejných univerzitách, kde je možné získat i magisterský titul.

Poznámka: Existují další umělecké směry související s počítači jako například *multimedia art*, *software art*, *net art*, *algorithmic art*, jejichž rešerše je nad rámec této práce. Pojem umění nových médií byl shledán jako nejobecnější.

[3] Z řeckého *morphê* (tvar) a *genesis* (vytvořáření) – vytváření formy.

[4] Na adrese `https://github.com/jasonwebb/morphogenesis-resources`.

[5] Pojem nebyl přidat do grafu1.1, kvůli shodě v názvu se stejnojmeným časopisem vydávaným v letech 1974 až 1985.

1.2 Vizualizace dat

Velkou a podstatnou aplikací počítačové grafiky jsou datové vizualizace. Tyto mají význam ve vědě i pro širokou veřejnost. Jejich složitost se může různě pohybovat na škále od jednoduchých bodových grafů, po (interaktivní) vizualizace zahrnující komplexní znalosti a zkušenosti v oblastech informatiky, matematiky, designu i Human – Computer Interaction (HCI) (interakce člověka s počítačem).

Fáze vizualizace dat

Ben Fry ve své dizertační práci[4] i knize[5] popisuje 7 kroků procesu tvorby vizualizace dat. Každá datová vizualizace má svoje specifika a tudíž může některé kroky absentovat.

1.	*Acquire* – získání dat	} informatika
2.	*Parse* – zpracování dat	
3.	*Filter* – filtrování dat	} matematika
4.	*Mine* – statistika	
5.	*Represent* – základní reprezentace	} graficky design
6.	*Refine* – čistší, hezčí reprezentace	
7.	*Interact* – interakce	} infografika a HCI

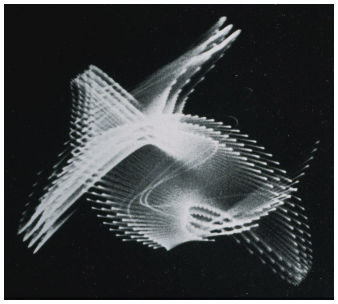
Jedná se o rozsáhlé téma. Těchto sedm kroků zahrnuje vědní disciplíny jakými jsou informatika, matematika a grafický design, případně infografika a HCI. Mezi přínosy vizualizací patří skutečnost, že lidský mozek je schopen přijímat vizuálním kanálem velký objem dat paralelně.

Historie

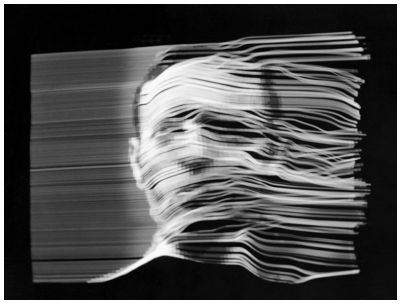
Jako disciplína je vizualizace od roku 1987. Jsou známé příklady vizualizací před používáním počítačům například mapa epidemie cholery v Londýně, kde byly do mapy zaznamenávány oběti a díky tomu bylo možné vyvodit spojitost závadu vodovodního potrubí. Dalším příkladem je litografie Charlese Minarda zobrazující graficky úbytek vojáků během tažení Napoleona na Moskvu (z knihy *Carte figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812–1813*, vydané r. 1869).

1.3 Vývoj umění nových médií

Podobně jako jiné oblasti lidské činnosti, i umění prochází vývojem. Po cestě tohoto vývoje se mnozí snažili hledat hranice možností nově se objevujících



(a) Ben Laponsky



(b) Vasulka

technologií. Databází pro umění, média a humanitní vědy je například stránka Monoskop [6]. Postupný vývoj byl popsán na mnoha místech, například na webu *Wikieducator* a stránce *Digital art timeline* [7], v příspěvku *Creative Coding: Perspectives and Case Studies* [8], článcích *The Pioneers (1950-1970)* [9] a *Computer Art History, Characteristics of Digital Imagery* [10], či v práci *Visual Intelligence: The First Decade of Computer Art (1965-1975)* [11]. V této sekci se nachází pouze velmi rámcový průřez osobností, objevů a technologií, kteří byli předcházeli tématu kreativnímu programování.

19. století

Podle výkladu definice nových médií, která považuje nová média jako cokoliv nového po tradiční architektuře, malování a sochařství, by se dalo považovat za první technologii fotografii, která vznikla technikou heliografie využívající snímků z *camery obscury* roku 1826, jejímž autorem je Nicéphore Niépce. Další technologické pokroky jsou mimo rámec této práce a nebudou zde[6] zmíněny.

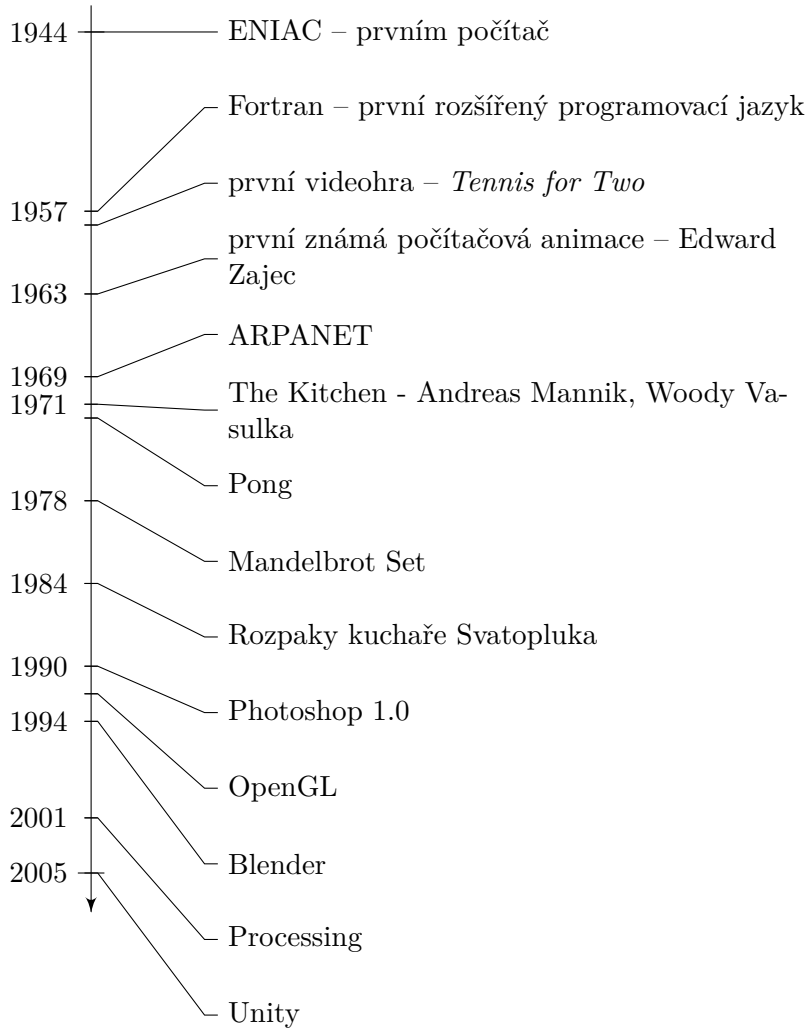
20. století

Za autora první počítačové grafiky se považuje Ben F. Laponsky. Jeho použití osciloskopu pro umění z roku 1950 pro umění je obrázeku 1.2a. Mezi další průkopníky patřila například Lillian Schwart a John Whitney, který v padesátých letech vytvářel počítačové animace díky mechanickému analogovému počítači, který vznikl úpravou protiletadlového sytému z druhé světové války.

První výstavy *computer artu* byly spíše než umělci pořádány vědci. Roku 1965 v New Yorkské Howard Wise galerii vystavovali Bela Julesz a A. Michael Noll, v Německu Georg Nees a Frieder Nake v *Galerie Niedlich*. První generace počítačových umělců pracující v *Bell Labs* a *Technische Universitat Stuttgart* přispěla množstvím informací k základům počítačové grafiky.

K tvorbě nového umění kromě počítačů byly používány i kamery a televize. Mezi významné umělce *video artu* patří Woody Vasulka pocházející z Brna se

[6]Některé z nich (jako je například mechanická televize) popisuje například kanál *Technology connections* na platformě *Youtube*.



Obrázek 1.3: Časová osa

ženou Steinou Vasulka, kteří založili v New Yorku interdisciplinární prostor „The Kitchen“, kde se pořádali výstavy, promítání, koncerty. Prostor je aktivní i v roce 2022. O umění Vasulkových byly natočeny dokumenty a příklad jejich díla je vidět například na obrázku 1.2b.

Jedny z prvních pokusů začlenit československé publikum učinil seriál *Rozpaky kuchaře Svatopluka,* kde občané pomocí rozsvěcování žárovek hlasovali o osudu hlavní postavy. Časová osa na obrázku 1.3 zobrazuje výběr z letopočtů vývoje technologií a událostí od roku 1944 do roku 2005.

21. století

Jako autor prvního jazyku pro kreativní programování se označuje John Maeda z Massachusetts Institute of Technology (MIT) Media Lab, roku 1999. Následně jeho studenti Casey Reas a Ben Fry oznámili vytvoření Processingu. V současném[7] umění nových médií existuje mnoho možností, jaké technologie použít z nichž některé jsou popsány v kapitole 3.

1.4 Příklady algoritmů a technik

Existují volně dostupné i komerční knihy a kurzy, týkající se tématice kreativního programování. Pod licencí *CC BY-NC 3.0*[8] je dostupná kniha[12] Daniela Shiffmana z roku 2012. Tato sekce jí bude inspirována.

V této sekci budou zmíněny příklady technik, využívaných převážně generativními umělci. Častým zdrojem inspirace jsou přírodní jevy a procesy, jako například chemické reakce a růst rostlin. Kromě inspirace v přírodě je možné inspirovat se tradičním uměním a umělci.

Náhodnost

Základním a užitečným nástrojem je použití náhody pomocí procedurálního šumu. Jak bylo psáno v práci [13]: „šum je náhodný generátor čísel počítačové grafiky.“ Má široké spektrum použití od procedurálního texturování, přes simulace mraků a vln, po obecnou snahu rozbití symetrie a přidání náhodnosti. Výhodou je i typicky rychlý výpočet a kompaktnost (například oproti alternativnímu přístupu – přidání šumu pomocí obrázku).

Jedná se o funkci, typicky s výstupními hodnotami mezi $0-1$. V závislosti na definičním oboru, čili vstupních parametrech, může být šum 1D, 2D, nebo 3D. Jedním ze známých algoritmů je Perlinův šum, původně vyvinutý pro potřeby filmu *Tron* začátkem osmdesátých let 19. století, jeho hodnoty lze promítnout do různých dimenzí, jak ukazují obrázky 1.4 a 1.5[9]

Místo přiřazování hodnot celému prostoru, lze zobrazovat například pohyb částic. Tomuto přírodnímu jevu se přezdívá Brownův pohyb (náznak pohybu je na obrázku 1.6[10]), matematicky ho popisuje Wienernův proces.

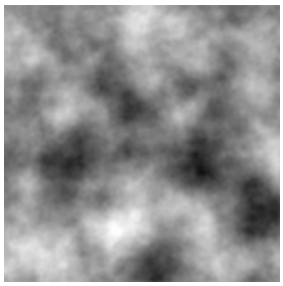
[7] O rozdílech práce v oblasti umění nových médií na přelomu 80. a 90. let a nyní hovoří například Lucie Svobodová – `https://artycok.tv/45434/vidis-tedy-jsem-lucie-svobodova`.

[8] Uvádění původu, nekomerční použití.

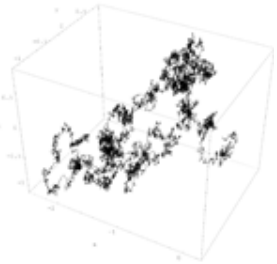
[9] Zdroj: `https://rtouti.github.io/graphics/perlin-noise-algorithm`.

[10] Zdroj: `https://en.wikipedia.org/wiki/Brownian_motion`.

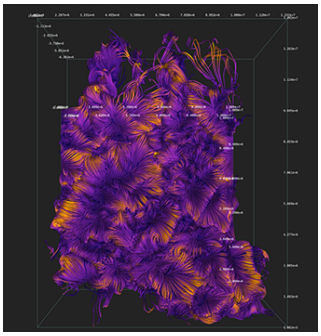
Obrázek 1.4: 3 oktávy Perlinova šumu 1D



Obrázek 1.5: Perlinův šum v 2D



Obrázek 1.6: Brownův pohyb, cesta jedné částice ve 3D



Obrázek 1.7: Vizualizace 3D dat

Obrázek 1.8: Vápenatka mnohohlavá

Obrázek 1.9: Conway's *Game of Life*

Reprezentace pohybu, vektorová pole

Jedním ze stavebních kamenů užitečným pro reprezentování reálných i imaginárních světů je euklidovský vektor s definovanou velikostí a směrem. Pomocí vektorů lze vyjádřit působení sil na objekty ve scéně v čase, simulovat částicové systémy a tak dále. Živé příklady minimalistického použití základních konceptů včetně potřebné matematiky lze prohlížet online[11] v již zmíněné knize. Možnostmi vizualizace 3D vektorových dat a vektorovými poli se věnovala práce [14], jejíž příklad je na obrázku 1.7.

[11] Na adrese `https://natureofcode.com/book/`.

Autonomní agenti, chování a systémy

Pro určité jevy jakými jsou například termitiště a pohyb hejna ptáků, je výhodné zavést pojem *autonomní agent*, kladoucí tři požadavky:

1. autonomní agent má limitovanou schopnost vnímat okolí,
2. autonomní agent zpracovává informace z okolního prostředí a vyhodnocuje svojí další akci,
3. autonomní agent nemá centrálního vůdce.

Tvoření komplexních systémů pomocí emergence z jednoduchých lokálních pravidel se zabýval například Craig Reynolds, který roku 1986 vyvinul program *Boids* simulující shlukování ptáků a podobných bytostí, které následně popsal v práci [15]. Další příklad organismu využívající pro rozhodování vjemy z blízkého okolí je houba anglickým názvem *Physarum polycephalum* (na obrázku 1.8[12]), jejížch chování popisuje Jeff Jone v článku [16].

[12]Zdroj: `https://www.sciencealert.com/this-creeping-slime-is-changing-how-we-think-about-intelligence`.

Celulární automat

Dalším způsobem modelování systémů je celulární automat (CA), které definovali Stanisław Ulam ve snaze zkoumat růst krystalů a John von Neumann, ve snaze o sebe–replikující roboty. CA se skládají z:

- pravidelné mřížky buněk,
- kde každá buňka má konečný počet stavů
- a pro každou buňku je definované okolí.

Příkladem CA je Hra života J. H. Conwaye (na obrázku[13] 1.9). Hra, či spíše simulace, má několik pravidel a její vývoj je závislý na výchozím stavu. Existuje mnoho implementací s komunitou hledající zajímavé[14] vzorce.

Pomocí CA lze simulovat různé jevy – například *Belousov-Zhabotinsky* reakci, či *Diffusion–Limited Aggregation*.[15] Reakce *Belousov–Zhabotinsky* reakce příkladem nerovnovážné termodynamiky, což vede ke vzniku nelineárního chemického oscilátoru. *Diffusion-Limited Aggregation* je proces difúze částic a agregace do fraktálních struktur. O CA a o mnohém dalším pojednává například kniha *A New Kind of Science* od Stephena Wolframa, zakladatele *Wolfram Research*.

Vzory v přírodě

Alan Turing se v roce 1952 zajímal v práci [17] vznikem vzorů v přírodě z homogenních stavů. Nyní nazývané Turingovy vzory lze generovat pomocí modelů reakční difúze a *Gray–Scott* modelem. Zde je výčet dalších příkladů typů vzorů.

- Symetrie
- Stromy, fraktály
- Spirály
- Chaos, tok, meandry
- Vlny, duny
- Bubliny, pěna
- Teselace
- Praskliny
- Skvrny, pruhy

[13] Zdroj: `https://www.wolfram.com/language/gallery/implement-conways-game-of-life/`.

[14] Jedním ze zajímavých počátečních stavů má rekurzivní vlastnost a tedy po oddálení (změně měřítka) se objevuje stejný vzor jako v počátečním stavu (zdroj: `https://www.youtube.com/watch?v=uRIHR55tyko`.

[15] Zdroj: `www.hermetic-systems.com/compsci/cellular_automata_algorithms.htm`.

KAPITOLA 2

Instituce, komunita a výzvy

Tato kapitola se věnuje grafickým komunitám a skupinám, které mohou být užitečné i pro oblast kreativního programování. Nejvýznamnější vědecké publikace a výzkum je možné sledovat například pomocí konferencí Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH). Příkladem dlouholetého fenoménu je komunita *Demoscene*. Další skupiny se sdružují například kolem konkrétních technologií, kde mohou sdílet své výtvory, nebo pořádat soutěže.

2.1 SIGGRAPH

SIGGRAPH[16] je konference o počítačové grafice, skládající se z akademických přednášek i průmyslového veletrhu. Součástí konferencí jsou vzdělávací kurzy, panelové diskuze a různé interaktivní techniky. Hlavní konference se pořádá od roku 1974 v Severní Americe a druhá v od roku 2008 v Asii. SIGGRAPH uděluje několik cen za příspěvky v počítačové grafice.

Konference jsou pořádané skupinou *ACM* SIGGRAPH, speciální zájmovou skupinou *ACM. Association for Computing Machinery* je mezinárodní učená společnost působící v oblasti výpočetní techniky. Byla založena v roce 1947 a jedná se o největší vědeckou a vzdělávací informatickou společnost.

2.2 The Node Institute

NODE Institute[17] umožňuje komunitě kreativního softwaru sdílet znalosti a růst pořádáním setkání, workshopů, webinářů a online kurzů. Institut funguje jako centrum pro kreativce na pomezí umění a technologie.

[16] `https://www.siggraph.org`
[17] `https://thenodeinstitute.org`

2.3 Datové vizualizace, infografika

Kromě čistě uměleckého záměru, lze počítačovou grafiku využít například pro zobrazení informací a dat. I přes rozsáhlost tohoto oboru, budou ve stručnosti některé webové stránky zabývající se vizualizacemi dat a informační grafikou v této sekci zmíněny. Byly nalezeny příklady webů, kde je možné podle zadání přispět datovými vizualizacemi.

storytellingwithdata.com Webová stránka, kterou provozuje Cole Nussbaumer Knaflic nese stejné jméno, jako její kniha *Storytelling with data*. Je zde možné nalézt články, podcasty a připojit se ke komunitě jejíž cílem je sdělit myšlenku pomocí dat. Přispět ke komunitě je možné například sdílením na platformě *Twitter* s #SWDchallenge.

makeovermonday.co.uk Eva Murray a Andy Kriebel, autor knihy *#MakeoverMonday*, provozují stejnojmennou webovou stránku, kde v letech 2016-2021 byly zveřejňovány každé pondělí datové sady s instrukcemi pro vytvoření datových vizualizací, kde byl zaznamenáván počet přispěvatelů. Přes skutečnost, že momentálně v roce 2022 není stránka aktivní, jsou jejich datové sady používány,

kaggle.com Kaggle je dceřinná společnost *Google* a online komunita pro *data science* a strojové učení. Nabízí prostor pro inspiraci i soutěže, zahrnující různá témata například od vylepšení gest pro *Microsoft Kinect* po vylepšení hledání Higgsonova bosonu v *CERNu*.

Inspirace

Další weby je možné sledovat pro inspiraci kreativní způsoby použití grafiky a témata, která popisují.

pudding.cool Webovou stránku *Pudding* obsahující vizuální eseje zamířené na interaktivitu, provozuje Matt Daniels.

visualcapitalist.com Zakladatel *Visual capitalist* Jeff Desjardins spolu s týmem publikují datové vizualizace zaměřené na témata jakými jsou technologie, energie a globální ekonomie.

informationisbeautiful.net Webovou stránku provozuje David McCandless, autor několika infografických knih. Spolu s týmem sdělují informace pomocí přehledných barevných vizuálů, vytvořených interním nástrojem *VizSweet*.

2.4 Demoscene

Demoscene[18, 19, 20] je fenomén na pomezí digitálního umění, kreativní tvorby a undergroundu. Jedná se o mezinárodní decentralizovanou a nekomerční subkulturu, která se zaměřuje na produkování digitální audiovizuální tvorby. Artefakty – *dema* – jsou typicky několikaminutové, počítačem generované sekvence animací, kombinace videí, textu a 3D grafiky.

Události kde soutěžící *demosceneři* živě produkují svá *dema* se nazývají *demoparty*. Výtvory musí být při prezentaci generovány v reálném čase. Soutěže mají kategorie, kde je nutné splnit různé limity. Například některá *dema* musí mít maximálně čtyři kilobyty, nebo musí být spuštěna na konkrétních počítačích, například C64.

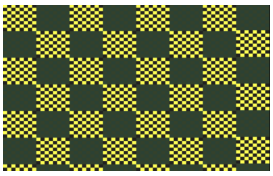
Fenomén *demoscene* byl přijat jako kulturní dědictví organizací *UNESCO* například ve Finsku (v roce 2020) a v Německu (v roce 2021). Anglickým portálem pro databázi *dem* je *pouet.net*. Počátky *demoscene* jsou v osmdesátých letech 19. století v *cracker* kultuře, kdy počítačový nadšenci přidávali své podpisy do her zbavených protekce proti kopírování, odkud se odloučili na nezávislé vyvíjení *dem*.

Demoscéna měla dopad na oblasti jako vývoj počítačových her a umění nových médií. S demoscénou souvisí například vznik *JavaScriptové* knihovny *Three.js demoscénerem* Ricardem Caballo[18], nebo platformy *Shadertoy* pro sdílení a upravování vizuálů generovaných v reálném čase pomocí jazykaGLSL. Díky hardwarovému pokroku nemusí již být malá velikost programu nutností, avšak zkušenosti s kompaktním a efektivním kódem jsou nadále ceněné.

Memories

Jedním příkladem takového dema je MS DOS program *Memories* od uživatele *HellMood*. Velikost programu je 256B a vyhrál soutěž "PC 256 byte"události *Revision* komunity *demoscene* v roce 2020 i cenu popularity. Analýzu tohoto programu popisuje článek[19] na stránce *Sizecoding wiki*. Demo obsahuje několik efektů, které se do sebe prolínají a hudbu. Příklad jednoho z efektů ukazuje 2.1.

```
xchg dx,ax ; get XY into AX
sub ax,bp ; subtract time from row
xor al,ah ; XOR pattern (x xor y)
or al,0xDB ; pattern for array of boards
add al,13h ; shift to good palette spot
```



Obrázek 2.1: Vizualizace *Array of chessboards* s velikostí 16 bajtů

[18] O vzniku hovoří v přednášce `https://www.youtube.com/watch?v=LXWYOF4VibE`.
[19] `http://www.sizecoding.org/wiki/Memories`

2.5 Psaní malého kódu

Programování co nejkompaktnějším způsobem bez ohledu na čitelnost rozhodně nepatří mezi nejlepší praktiky, avšak i tak má koncept svojí přidanou hodnotu. Kompaktní programování může motivovat programátory objevovat hranice jazyka i algoritmů, nebo už jen sloužit k pobavení a výzvě mezi programátory.

つぶやき processing, つぶやき GLSL

Na sociální síti *Twitter* se vyskytuje *hashtag* '# つぶやき Processing'.[20] Jedná se o výzvu[21] v tvoření pomocí *Processingu*, kdy je kód limitovaný délkou[22] znaků. Součástí příspěvku bývá i video náhled výsledku. Dále se výzva rozšířila i na jazyk GLSL.

Na příkladu[23] na obrázku 2.2 je vidět kód a (původně animovaný) vizuál, který kód generuje. Program lze spustit pomocí knihovny *p5.js*, což umožňuje zkomprimovat *JavaScript* do jinak nezkompilovatelné podoby. Je například možné vynechat klíčové slovo *let* u definice proměnných a *function* u definice funkcí (dokonce je použita *arrow function*). Pro vynechání *setup* funkce je použit operátor ||, díky kterému se vytvoří *canvas* pouze při prvním vykreslení funkce *draw*.

```
t=0;draw=$=>{t||createCanvas
    (540,600);background(0);circle
    (270,270,50);for(j=-4+(t+=s=PI
    /20)/10%s;j<4;j+=s)for(i=0;i<6;i
    +=s*2)quad(...[[i,j],[b=i+s*sin(t
    +i*j)*2,j],[b,j+s],[i,j+s]].
    flatMap(([u,v,e=v**2*40+80])=>[e*
    cos(u+j)+270-v*19,v*80+250+e*sin(
    u+j)]))}
```

Obrázek 2.2: Příklad つぶやき *processingu* od uživatele *yuruyurau*

Dwitter, Twigl

Podobně jako předchozí příklady, se stránky *Dwitter.net* a *Twigl.app* věnují výzvě psaní krátkého kódu s různou limitací počtu znaků. *Dwitter* umožňuje zobrazování výtvorů komunity, *Twigl.app* je spíše editor umožňující export videa. Stránky mají integrovaný editor pro vytváření příspěvků.

[20] つぶやき znamená japonsky *tweet*.
[21] `https://www.deconbatch.com/2020/01/what-is-processing.html`
[22] V roce 2022 to je 280 znaků.
[23] Zdroj: `https://twitter.com/yuruyurau/status/1485987035207041024`.

2.6 Genuary, Nodevember

Některé komunitní výzvy se konají opakovaně. Příkladem těch momentálně aktivních je *Nodevember* konající se v listopadu a *Genuary* v lednu[24]. Rámcová zadání pro daný den v měsíci – *prompts* – se odkrývají postupně, nebo všechna najednou. Vývojáři a umělci participující se těchto událostí poté sdílejí svoje výtvory na sociálních sítích s *hashtagy* události.

Nodevember

Při události *Nodevember* lze použít libovolný procedurální či *node-based* software, jakými jsou například *Blender*, *Substance*, *Houdini*, či *Unity*. Podle pravidel není možné například používat obrázkové textury, načítat externí data, či manuální modelování a scriptování. Na stránkách (`https://nodevember.io`) jsou dostupné statistiky pro roky 2019, 2020, 2021 ukazující počet účastníků, vstupů, používaných programů a další. Nejúspěšnější rokem je 2020 s 3840 vstupy od 496 účastníků.

Genuary

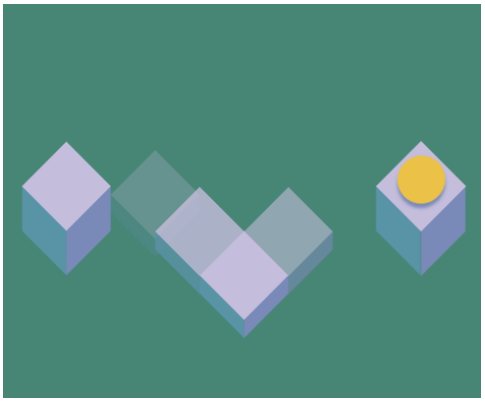
Genuary (`https://genuary.art`) nemá striktní pravidla a je možné vytvářet díla pomocí jakéhokoliv frameworku. Zadání na další roky je možné navrhnout prostřednictvím stránky *Github*. Zadání z roku 2022 zahrnují výzvy k inspirací se malířem *Sol LeWitt Wall*, použití *ditheringu*, letištní koberec, nebo pouze barevnou paletu.

7daysofcode

Kromě těch měsíčních existuje například i sedmi denní výzva *7daysofcode*. Podle oficiální stránky (`https://7daysofcode.art`) se jedná o velmi neformální soutěž, která by se měla konat každý měsíc.

[24] Poslední běh *Codevember.xyz* byl roku 2019.

(a) HTML a CSS olejomalba



(b) HTML a CSS minihra

2.7 Webové technologie

Zvýšení kreativity omezením možností využívají i nadšenci používající webové technologie. Jednou takovou umělkyní je *Diana Smith*, která vytváří „olejomalby“ pouze pomocí Hypertext Markup Language (HTML) a Cascading Style Sheets (CSS), jak ukazuje 2.3a[25].

Existují i příklady jednoduchých mini her, opět využívající pouze technologie HTML a CSS, tentokrát i pro implementaci jednoduché logiky. Příkladem je hra 2.3b[26], jejíž cílem je projít pomocí kurzoru myši postupně se odkrývajícím labyrintem.

Lze se domnívat, že ačkoliv se pravděpodobně jedná především o výzvu, může být tento přístup vhodný v situacích, kdy není možné použít *JavaScript* (například když prohlížeč zakazuje provádění *scriptů*).

2.8 Kreativní uskupení

Dalšími místy, kde se koncentrují lidé za účelem objevování spojení umění a technologií, jsou například školy. V této sekci budou představeny dvě taková uskupení na MIT a ČVUT.

The Future Sketches

Příkladem uskupení zajímající se o využití kódu kreativním způsobem je *The Future Sketches* na MIT. Jejich slovy se zabývají zkoumáním *softwaru* jako média pro umění a design, a zároveň otázce, jak mohou nástroje a pedagogické přístupy zasvětit novou generaci do „výpočetního řemesla“. Kromě používání existujících nástrojů se soustředí i na vytváření nástrojů nových, neb „nástroje dneška pomáhají utvářet umění zítřka“ [21]. Polem jejich současného výzkumu

[25] Zdroj: `https://diana-adrianne.com`.

[26] Zdroj: `https://codepen.io/nathantaylor/pen/KaLvXw`.

je generativní forma, strojové učení a rozšířená realita s důrazem na to, jak pochopit esenci těchto technologií a použít je nečekanými, poetickými způsoby.

NI-CCC

Skupina, která se snaží spojovat umění s technologiemi existuje i na FIT ČVUT v rámci předmětu Kreativní programování (Creative Coding and Computational Art), *NI-CCC*. Jedná se o kreativně pojmenovaný předmět věnující se kreativnímu programování. Projekty, které zde vznikly, jsou dostupné na stránkách[27] předmětu. Práce se věnují různorodým tématům, například sonifikaci obrazu či vymírajícím druhům. Další projekty využívají open data měst, videa z tramvají a další.

2.9 Generativní script

Současným trendem je tvoření generativních *scriptů*, do nichž je nutné pro získání výsledku vložit *seed*, unikátní náhodný řetězec, parametrizující výsledný artefakt. Jedná se o jiný přístup k tvorbě a umění, kdy autor místo jednoho díla parametrizuje program tak, aby bylo možné získat různé (ideálně použitelné) variace. Tímto způsobem se dají napsat programy pro získání generativních stromů, domů, či celých světů[28].

NFT

Generativní scripty mají využití například ve světě obchodu s Non-fungible token (NFT). Platformy, jakou je například *Art Blocks*, umožňují digitálním umělcům nahrát generativní *script* na *Etherum blockchain* a následně specifikovat kolik z tohoto *scriptu* lze vygenerovat iterací. Sběratel poté například iteraci zakoupí a tím se mu vygeneruje unikátní artefakt, tudíž nikdo předem neví jakou variaci dostane.

[27] `https://ni-ccc.github.io`
[28] Jako ve hře *Valheim* či *Minecraft*.

KAPITOLA 3

Technologie pro kreativní programování

V této kapitole budou představeny typické technologie pro kreativní programování jakou je například grafická knihovna *Processing*. Další programy jsou známé pro použití v *new media art*, jako například *Touchdesigner*. Dalším zohledněným hlediskem bude dostupnost webového editoru. Bude zde snaha prezentovat i kontext, ve kterém by mohly být použité nástroje jako *Blender* a *Desmos*.[29]

3.1 Processing

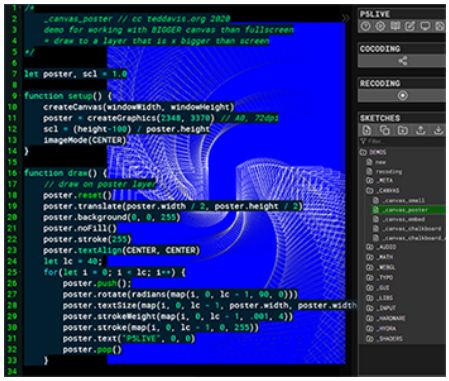
Processing je jedním z prvních nástrojů pro kreativní programování a pochází z roku 2001. Nyní je dostupný nejen v původní formě programu s vlastním vývojovým prostředím, ale i jako knihovna, kterou lze použít ve vlastním projektu:

Processing Program je možné stáhnout pro operační systémy Linux, Windows a Mac. Obsahuje editor pro psaní kódu, display pro zobrazení výsledku a manažer, pomocí kterého lze stáhnout přídavné knihovny, příklady, módy a nástroje. Nativním programovacím jazykem je Java, ale je zde možnost stáhnout módy pro Python, R, *p5.js* (JavaScript), GLSL a další. Módy mohou a nemusí být udržovány pro aktuální verzi Processingu.

Processing.py Dalším způsobem jak vyvíjet v jazyce Python je instalace knihovny p5 pro Python. Prerekvizitou je instalace knihovny *GLFW*.

[29] Více nástrojů lze nalézt například na adrese `https://github.com/terkelg/awesome-creative-coding`.

(a) Alfa verze *Processingu* na CD ve velikosti vizitky



(b) Editor Theodora Davise pro živé performance

p5.js Pro webový vývoj existuje knihovna *p5.js*, kterou lze přidat pomocí HTML tagu script, či nainstalovat do projektu pomocí balíčkovacího systému (*npm*, *yarn*)

Díky *Processingu* je možné s psaním relativně málo řádků kódu (do souborů zvaných *sketche*) vytvořit zajímavý vizuál. V Každém *sketchi* je funkce *setup* určená pro instrukce které se vykonávají před začátkem animace a funkce *draw* určená pro instrukce, které se provádí v nekonečné kreslící smyčce.

Mimo kreslení primitiv je knihovna nativně vybavena například funkcemi šumu a různými matematickými funkcemi. Lze načítat soubory, pracovat s pixely, detekovat vstupy z klávesnice a myši, pracovat s křivkami a maticemi. Je možné pracovat ve 2D i ve 3D.

Lze se domnívat, že vzhledem k množství materiálů a referencí je *Processing* (potažmo *p5.js*) momentálně (v roce 2022) nejpopulárnější knihovnou pro začátky v kreativním programování. Projekt *Processing* sponzoruje nezisková organizace *Processing Foundation*.

Online editory

Pravděpodobně nejpopulárnější volbou pro rychlý začátek bez nutnosti instalace žádného programu je online editor. Místo názvu je uvedena rovnou url adresa editoru.

editor.p5js.org Jednoduchý způsob, jakým zkusit vyvíjení pomocí *p5.js*. Nachází se zde intuitivní editor a po přihlášení je možné své *sketche* ukládat a sdílet ostatním. Z hlediska výkonu může být lepší lokální vývoj.

openprocessing.org Openprocessing je komunita pro sdílení *sketchů* vytvořených pomocí knihovny *p5.js*. Na stránce je možné vytvářet nové *sketche* i prohlížet výtvory uživatelů.

teddavis.org/p5live/ Stránka Theodora Davise pro živé kódování, například pro *VJing* (živé vizualizace) i výuku. Výchozí nastavení zobrazuje průsvitný editor, za kterým je vidět výsledek renderu. Mezi funkce patří například i zobrazení tabule. Stránka umožňuje kolaborativní kódování, různé možnosti nastavení editoru a zobrazení, ukázkové kódy i vestavěnou dokumentaci *p5.js*.

3.2 Kreativní kódování v C++

Tato a následující sekce budou věnovány specifickým jazykům a operačním systémům. Existují knihovny (fameworky), umožňující vývoj v C++. Budou představeny projekty *openFrameworks* a *Cinder*.

openFrameworks

Open source C++ sada nástrojů pro kreativní kódování *openFrameworks* je navržena tak, aby asistovala v kreativním procesu poskytováním jednoduchého a intuitivního *frameworku* pro experimenty. Kód je napsán tak, aby byl kompatibilní mezi platformami. Podle oficiální stránky[30] projekt podporuje vývoj na pěti systémech (*Windows*, *OSX*, *Linux*, *iOS*, *Android*) a čtyřech vývojových prostředích (*XCode*, *Code::Blocks*, *Visual Studio* a *Eclipse*). Program je distribuovaný pod MIT licencí. Jedním z cílů *openFrameworks* je možnost kolaborace.

Cinder

Další C++ knihovnou pro kreativní programování je *Cinder*. Jeho užití předpokládá programování s estetickým záměrem, což zahrnuje domény jakými je grafika, audio, video výpočetní geometrie. Nástroj je multiplatformní a podle oficiální stránky[31] dostupný pro *macOS*, *Windows*, *Linux*, iOS a *Windows UWP*. Stránka rovněž popisuje, že *Cinder* je praxí prověřený software, vhodný jako primární nástroj pro profesionály, ale zároveň vhodný pro učení a experimentování.

Možnou nevýhodou *Cinderu* je oproti *openFrameworks* menší transparentnost a debuggovatelnost, jelikož *Cinderu* závisí na vestavěných knihovnách operačních systému. Místo toho *openFrameworks* je závislý na větším množství *open source* projektů.

[30] `https://openframeworks.cc`
[31] `https://libcinder.org`

3.3 Kreativní kódování na iOS - framework C4

Pro operační systém *iOS* existuje například *framework C4*. Podle oficiální stránky[32] *C4* využívá sílu nativního *iOS* programování v kombinaci se zjednodušeným Application Programming Interface (API). Díky *frameworku* je možné objevovat možnosti práce s médii a interakcí, vytvářet umělecká díla a navrhovat rozhraní.

3.4 Technologie pro webový vývoj

Prezentované příklady knihoven pro webový vývoj používají programovací jazyk *JavaScript*, případně je u některých knihoven možné programovat v *TypeScriptu*. Zmíněné technologie je možné v projektu použít například lokálním uložením souboru, instalací pomocí balíčkovacích systémů, či přidáním odkazu, pokud je soubor uložen na *CDN*. Výhodou webových knihoven je například jistá nezávislost na operačním systému a možnost existence webového editoru pro vývoj přímo v prohlížeči, například *codesandbox.io*, *JSFiddle*, či editory pro konkrétní knihovny.

Technologie s 2D *enginem*

Tyto webové technologie se dají rozdělit například podle vykreslovacího *enginu*. Příkladem 2D *enginu* jsou uvedeny *p5.js*, *PixiJS* a *Paper.js*.

p5.js O *p5.js* již bylo psáno v sekci 3.1.

Paper.js Podle oficiálních stránek[33] je *Paper.js* Švýcarským nožem v psaní *scriptů* pro vektorovou grafiku.

PixiJS *PixiJS* je *lightweight* multiplatformní 2D knihovna používající *WebGL* pro hardwarovou akceleraci. Knihovnu je mimo jiné možné použít pro vývoj her.

plotly.js *Plotly.js*[34] je *Javascriptová* knihovna pro datové vizualizace, umožňující produkovat 2D i 3D grafy.

Technologie s 3D *enginem*

Další možnosti přináší využití 3D *enginu*. Budou zde stručně představeny knihovny *Three.js* a *Babylon.js*.

[32] `https://www.c4ios.com/examples/`
[33] `http://paperjs.org`
[34] `https://plotly.com/javascript/`

Three.js Knihovna, jejíž vznik byl stručně naznačen v sekci 2.4, využívá pro hardwarovou akceleraci *WebGL*. Mezi funkce této knihovny patří například různé nastavení kamer, různé druhy světel, materiálů, možnost definovat vlastní materiály přes *shadery*, animace, sadu pomocných funkcí a lze pomocí *Three.js* tvořit projekty pro virtuální a rozšířenou realitou. Oficiální stránka[35] obsahuje řadu ukázek reálných projektů.

Babylon.js Vývoj *Babylon.js*[36] začal o 3 roky později, než *Three.js*, tedy v roce 2013 a to dvěma zaměstnanci firmy *Microsoft*. Kód je rovněž dostupný na platformě *Github*. Ve srovnání s *Three.js* je *Babylon.js* ucelenějším *frameworkem*, obsahujícím v základu více funkcí, které nemusí být dodatečně importovány (jako různé ovládání kamery v *Three.js*), díky čemuž je velikost balíčku *Babylon.js* větší. *Babylon.js* byl napsán v programovacím jazyce *Typescript* a vývojáři cílí o zpětnou kompatibilitu mezi verzemi. *Babylon.js* rovněž nabízí *Node Material Editor*[37], tedy webový editor materiálů, používající vizuální programování. S vývojem *WebGPU* se vývojáři *Babylon.js* snaží o podporu této rychlejší alternativy *WebGL* ve svém projektu. Vývojáři rovněž představili nástroj *Spector.js* pro debugování *WebGL* scén.

Pomocné knihovny

V závislosti na projektu mohou být nepřekvapivě užitečné různé knihovny. Bude zde zmíněna například knihovna vhodná například pro datové vizualizace a jednoduchá knihovna *quicksettings* vhodná pro mnoho různých projektů.

D3 *Data-Driven Documents (D3)* je knihovna pro jazyk *JavaScript* umožňující vytváření dynamických, interaktivních vizualizací dat ve webových prohlížečích. Podle oficiální stránky[38] je díky knihovně možné z dat generovat tabulky či interaktivní grafy.

Quicksettings Pro urychlení vývoje webových vizualizací je možné použít knihovnu *quicksettings*, umožňující uživatelsky ovládat nastavené parametry. Jedná se o velmi jednoduchou knihovnu, poskytující řadu vstupů jako výběr barvy, nahrání obrázku, zadání čísel a podobně.

Z hlediska výkonu

Většina prezentovaných webových technologií využívá pro akceleraci výkonu přes grafickou kartu standart *WebGL* (který staví na základech *OpenGL* a *DirectX*), některé používají modernější a rychlejší *WebGPU* (využívající *Vulcan*)

[35] `https://threejs.org`
[36] `https://www.babylonjs.com/`
[37] `https://nme.babylonjs.com/`
[38] `https://d3js.org/`

– například *Babylon.js*. Další alternativou je využití *WebAssembly*, což je cílem například projektu *GLAS*[39].

3.5 Vizuální programování

Pro multimediální projekty, kdy je potřeba propojit několik vizuálních a zvukových vstupů i výstupů, může být užitečné použití programů využívajících vizuálního programování, které tyto úkoly jednoduše a intuitivně abstrahuje. V tradičním programování se instrukce zadávají psaním klíčových slov a znaků. Vizuální programování místo toho používá manipulaci grafický elementů, do kterých se případně dají zadávat hodnoty a je tedy nutné použití myši i klávesnice.

V této sekci budou ve stručnosti představeny programy využívající vizuálního programování *TouchDesigner* a program *vvvv*, ale existují i další programy, například *Noitch VFX*. Na rozdíl od doposud zmíněných technologií nejsou tyto projekty typicky zadarmo pro komerční užití.

TouchDesigner

Program *TouchDesigner*[40] dostupný pro operační systémy *Windows* a *macOS* vyvíjí společnost *Derivative*. Program je zadarmo pro nekomerční užití v maximálním rozlišení 1280x1280 a umožňuje tvorbu multimediálních interaktivních vizualizací v reálném čase pomocí vizuálního programování. *TouchDesigner* je možné použít pro 2D projekty a je vybaven i 3D *enginem*. Je možné ho použít pro video mapování, živé performance, výstavy, libovolné obohacení výstupu webové kamery a další.

vvvv

vvvv[41] je sada nástrojů pro živé vizuální programování, jednoduché prototypování a vývoj. Program je navrhnut pro práci s různými médii s fyzickým rozhraním, umožňující generování grafiky v reálném čase s interakcí mnoha uživatelů zároveň. Program je zadarmo pro nekomerční užití a skupina za *vvvv* věří férovému úsudku uživatelů na to, kdy je vhodné zakoupit licenci pro komerční užití.

[39] Odkaz na projekt: `https://github.com/lume/glas`.
[40] `https://derivative.ca/`
[41] `https://vvvv.org/`

3.6 Modelovací programy a herní *enginy*

V závislosti na povaze projektu může být výhodné použití modelovacích programů a herních *enginů*. Modelovací programy umožňují například generování procedurálních objektů a v herních *enginech* je například možné experimentovat s fyzikou.

Herní enginy

Herní *engine* je softwarový *framework*, který soustřeďuje obecné funkce používané v počítačových hrách. Budou zde ve stručnosti zmíněny programy *Godot*, *Unity* a *Unreal*. Programy umožňují produkovat 2D i 3D hry.

Godot *Godot* je multiplatformní *open source* herní *engine* pod licencí MIT vyvinutý komunitou. Je dostupný pro řadu operačních systémů a umožňuje vývoj i pomocí webového editoru. Pro vývoj *Godot* nabízí 5 programovacích jazyků: *GDScript*, C#, *VisualScript*, and C++ a C.

Unity *Unity* je herní engine vyvinutý společností *Unity Technologies*. Kromě grafického prostředí pro tvorbu podporuje také tvorbu skriptů v jazyce C#.

Unreal *Unreal* je herní *engine*, který byl vytvořen firmou *Epic Games*. Pro vývoj her v *Unreal enginu* lze použít jazyk C++ a vizuální skriptování pomocí *Blueprints*.

Modelovací programy

Existuje řada 3D modelovacích programů, neboli programů používající matematické reprezentace pro objekty a jejich povrchy. Jako zástupci širokého spektra možnosti zde budou jmenovány programy *Bledner* a *Houdini*.

Blender *Blender* je *open source* program pro 3D grafiku, který může být použit mimo jiné například pro animované filmy, vizuální efekty, umění, modelování či virtuální realitu. V kontextu kreativního programování lze *Blender* použít zejména například pro procedurální grafiku pomocí uzlových editorů pro materiály či pro geometrii, nebo psaním scriptů.

Houdini Apprentice *Houdini* je program od firmy *SideFX*, který je využíván předními společnostmi v oblasti vizuálních efektů. *Houdini Apprentice* je limitovaná verze, která je dostupná zadarmo pro nekomerční užití.

3.7 GLSL

GLSL je programovací jazyk pro psaní shaderů. GLSL připomíná syntaxi jazyka C a je od roku 2004 součástí *OpenGL*. Tvorbu shaderů pomocí GLSL podporují ve svých projektech například *Three.js*, *Touchdesigner*, či *Unity*.

Shadertoy

`Shadertoy.com` je online komunita a platforma pro profesionály v počítačové grafice, akademiky i nadšence pro sdílení výtvorů shaderů v GLSL. Každý sdílený výtvor se vykresluje v reálném čase a je u něj možné zobrazit kód, který ho vygeneroval. Součástí *Shadertoy* je i editor pro vytváření shaderů.

Shadershop

Pro začátečníky může být programování v tomto jazyce nepřímočaré a proto se nabízí například projekt *Shadershop*[42]. Jedná se o program, který umožňuje graficky měnit parametry funkcí a provádět nad nimi operace podobně, jako v programu *Photoshop*. *Shadershop* zároveň vypíše předpis takové funkce. *Shadershop* lze možné stáhnout, či použít webový editor.

Desmos

Dalším nástrojem kde je možné vizualizovat matematické funkce pro potřeby nejen vývoje v GLSL je například *Desmos*[43]. *Desmos* má bohatou komunitu (například na platformě *Reddit*).

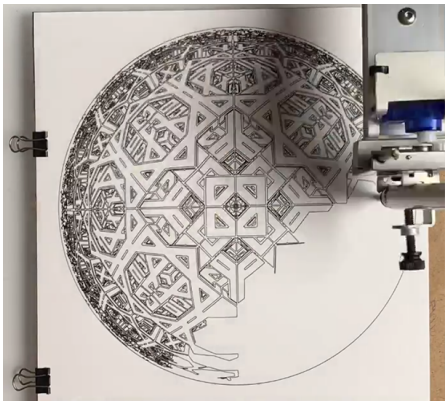
3.8 Kreativní programování mimo digitální grafiku

Kromě generování počítačové grafiky lze podobné projekty sledovat v jiných oblastech. Budou zde zmíněny projekty například z oblasti hudby a plotter artu.

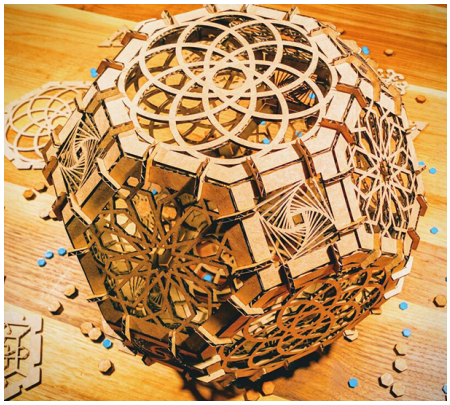
Hudba

Kromě nástrojů jakými jsou *FL Studio* či *Ableton*, lze hudbu produkovat i pomocí psaní kódu. Hudbu lze produkovat kódem i živě, například na událostech jakými jsou *Algorave*. Budou zde uvedeny příklady programů klasického i vizuálního programu.

[42] `http://tobyschachman.com/Shadershop/`
[43] `https://www.desmos.com/calculator`



(a) Plotter art



(b) Počítačem ovládané vyřezávání

SuperCollider *SuperCollider* je platforma pro audio syntézu a algoritmickou kompozici, používaná hudebníky, umělci i vědci pracujícími se zvukem. Původně ho vyvinul James McCartney roku 1996, který ho po 8 letech zpřístupnil pod GNU licencí. Nyní je program aktivně udržován a vyvíjen komunitou. *SuperCollider* je naprogramování v C++17 a používá několik knihoven třetích stran, jako Qt a Boost. Program je dostupný pro *Windows*, *macOS*, řadu *Linux* a *BSD* distribucí, *Raspberry Pi*, a *BeagleBone Black*.

Sonic Pi *Sonic Pi*[44] je prostředí pro živé kódování hudby, napsané v *Ruby*. Používá *engine* pro syntézu ze *SuperCollideru*.

Max, Pure Data Mezi vizuální programovací jazyky patří *Max* (známý také pod *Max/MSP/Jitter*), nebo program *Pure Data*.

Mimo digitální svět

Mimo digitálního zobrazování a přehrávání existují nástroje pro produkování výsledků ve fyzickém světě. Mezi tyto manifestace patří například *plotter art*, 3D tisk, nebo vyšívání.

Plotter art *Plotter art* (na obrázku 3.2a[45] je druh umění, které je uzpůsobeno pro tisk na plotterech, grafickému výstupnímu zařízení počítače. Zařízení může být zakoupeno, či zkonstruováno. Příkladem takového zařízení je *AxiDraw*.

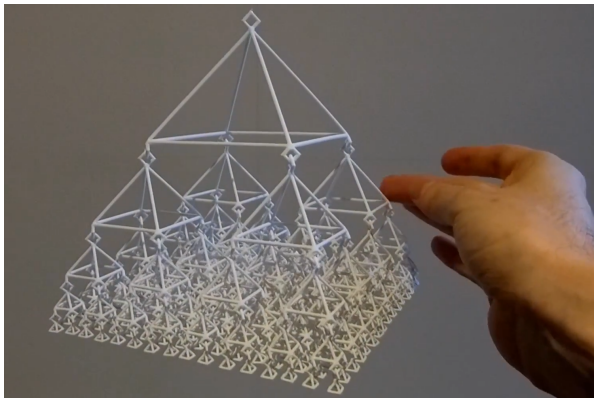
Vyšívání Dalším způsobem jak exportovat vytvořený design je použití vyšívacího stroje, který dovoluje automatizované vyšívání pomocí instrukcí.

[44] Stránka: `https://sonic-pi.net`.
[45] Zdroj: `https://www.instagram.com/p/CbtxhTqLMCs/`.

Příklad jak docílit tohoto zadání docílit pomocí programu *Houdini* a jazyku *Python* ukazuje například tento návod[46].

Laserové digitální vyřezávání Pro digitální vyřezávání z různých materiálů lze použít například nástroj *Cuttle*[47]. Příklad ukazuje obrázek 3.2b[48].

3D tisk Analogicky může být umění produkováno pomocí 3D tiskáren. Příkladem tvůrce zajímavých designů pro 3D tisk je *Henry Segerman*[49]. Příklad jeho práce je na obrázku 3.3.



Obrázek 3.3: Příklad výstupu z 3D tiskárny od Henryho Segermana

[46] https://entagma.com/python-in-houdini-controlling-an-embroidery-machine-dst-export/

[47] https://cuttle.xyz

[48] Zdroj: https://fabacademy.org/2021/labs/waag/students/nadieh-bremer/blog/week-3/

[49] https://www.youtube.com/c/HenrySegerman

KAPITOLA 4

Vybrané projekty a události

Cílem této kapitoly je poskytnout příklady známých i méně známých projektů, přibližující rozsah a možnosti umění nových médií, kreativního programování a vizualizací.

4.1 Události a festivaly

Tato sekce se věnuje festivalům umění nových médií a kreativního programování. Pro Českou republiku je známý festival *Signal*.

Obrázek 4.1: Signal festival 2021

Signal Festival

Signal je pražský festival probíhající pravidelně čtyři dny v říjnu a to každoročně již od roku 2013. Program je tvořen renomovanými zahraničními i českými umělci z oblasti *light designu*, vizuálního a digitálního umění, umělé inteligence, ale i konceptuálního umění. Slovy oficiální stránky projektu [22] – „Festival rád vzdělává sebe i návštěvníky“. Příklady z ročníku 2021 jsou na obrázku 4.1[50].

[50]Zdroj: `https://www.signalfestival.com/rocnik/2021/`.

Další události

Seznam dalších festivalů ze které si vzala inspiraci tato část lze nalézt stránkách *Awesome Creative Coding*[51]. Další události a festivaly lze vyhledat například na internetové stránce `events.thesupply.com`.

- **Festival OFFF**[52] je mezinárodním festivalem kreativity, digitálního designu a umění. Jedná se o přehlídku trendů v oblastech kreativity, designu, hudby, umění, zábavy a digitální kultury. Koná se v Barceloně.
- **Eyeo Festival** [53] je setkání komunity kreativní technologie. Spojuje zájemce z oblastí umění, kreativního kódování, virtuální a rozšířené reality a vývoje softwaru. Festival se odehrává v Minneapolis.
- **Gray Area Festival**[54] je pořádán neziskovou organizací *Gray Area*. Tématem je kreativní kódování, umění a technologie.
- **Mutek**[55] je mezinárodní festival digitální kreativity a elektronické hudby.
- **NODE Forum for Digital Arts** je festival pořádaný institucí *The Node Institute* zmíněné v sekci 2.2. Jedná se o otevřenou platformu pro kulturu, umění a technologii.
- **ART+TECH**[56] je festival pořádaný skupinou CODAME oslavující kreativitu workshopy, uměleckými galeriemi, výstupy a instalacemi. Festival je pořádán každoročně od roku 2010.

Obrázek 4.2: *Expo 2020*

[51] `https://github.com/terkelg/awesome-creative-coding#events`
[52] `https://www.offf.barcelona`
[53] `https://eyeofestival.com`
[54] `https://grayareafestival.io`
[55] `https://mutek.org`
[56] `https://codame.com`

4.2 Expo

Expo, neboli světová výstava je velká mezinárodní výstava průmyslu a kultury jednotlivých zemí. Organizátorem je Mezinárodní úřad pro výstavnictví a rozlišuje dvě kategorie výstav *Expo*: Světová výstava konající se každých 5 let a trvající až 6 měsíců a Mezinárodní specializovaná výstava konající se v termínu mezi světovými výstavami, trvající do 3 měsíců a mající specifické téma.

Mezinárodní úřad pro výstavnictví

Roku 1928 byla pro účel dohlížení na světové a mezinárodní výstavy založena mezivládní organizace *Mezinárodní úřad pro výstavnictví*[57]. Mezi jejich 170 členů patří například Česko a Slovensko. Hymna *Mezinárodního úřadu pro výstavnictví* je čtvrtá část 9. Symfonie Antonína Dvořáka.

Expo 2020

Expo 2020 se konalo v Dubaji od listopadu 2021 do března 2022 na téma *Connecting Minds, Creating the Future*. Výstava českého pavilonu byla nazvána *Czech spring* a kromě sytému *S.A.W.E.R* který dostal cenu za nejlepší inovaci zde bylo několik uměleckých výstav. Ukázka je na obrázku 4.2[58]

Obrázek 4.3: Stěna v SAGElabu na FIT ČVUT

Obrázek 4.4: Průchod budovami Národního muzea

Obrázek 4.5: Stěna v Centru architektury a městského plánování

4.3 Příklady atypických vizualizačních ploch v Praze

V této sekci budou zmíněny příklady některých ploch použitelných pro vizualizace. Jmenované plochy mají atypické rozměry či specifické rozhraní pro projekci.

[57] Anglicky *Bureau of International Expositions* (`BIE`).
[58] Zdroj: `https://www.czexpo.com/en/national-day/`.

Linky Světelná instalace s rozlišením 5px x 240px na budově FEL ČVUT nazývaná *Linky* oživuje dejvický kampus proměnlivými vizualizacemi. Linky tvoří 5 pásků a má vlastní API.

Tunel Další projekční plocha se nachází ve spojovacím tunelu mezi Historickou a Novou budovou Národního muzea. Obraz je promítán po celé délce obvodu chodby po obou stranách s plochou bezmála 270 m^2 na ultra-krátkou projekční vzdálenost. Příklad ukazuje obrázek 4.4.

CAMP Sál s unikátní velkoplošnou projekcí jehož součástí je 25 metrů široká stěna (na obrázku 4.5) se nachází v *Centru architektury a městského plánování* (CAMP). O spojení a prolnutí obrazu ze tří projektorů se stará media server *d3 4×2pro*.

SAGElab Zobrazovací stěnu v *SAGElabu* v budově ČVUT tvoří 20 monitorů o celkovém rozlišení 9600 x 4320 pixelů. Obraz je generován na jednom místě a hardwarově synchronizován. Stěna je vidět na obrázku 6.2.

Glab Interaktivní stěna v grafické laboratoři v budově ČVUT o rozměrech 6m x 3m je kombinací projekčních technik a lidarových dat umožnujících unikátní způsob interakce uživatele s projekcí.

4.4 Nervous system

Jako příklad designového studia bylo vybráno studio *Nervous system*. Specializuje na generativní design, tedy v tomto kontextu specifický způsob tvorby, kde místo modelování finálního produktu se soustředí na pochopení, objevování a simulaci procesů, kterými mohou teoreticky vygenerovat nekonečné množství designů. Založili ho v roce 2007 Jessica Rosenkrantz a Jesse Louis-Rosenberg. Studio se nachází v *New Yorku*. V rozhovoru[59] prozradili, že se specializují na produkty menších rozměrů, aby byla cena pořízení a výroby malá a produkty byly tak dostupné široké veřejnosti.

Některé[60] své designy pro 3D tisk zpřístupňují veřejnosti. Mezi jejich produkty patří například různé šperky, puzzle, či lampy. Ukázka je na obrázku 4.6.

4.5 Electric Sheep

Jak je uvedeno na stránce projektu[61], Electric Sheep je kolaborativní abstraktní umělecké dílo. Aplikaci je možné nainstalovat na mnoho různých zařízení a použít například jako spořič obrazovky.

[59] `https://www.youtube.com/watch?v=4Yl09NiI1lU`
[60] `https://www.thingiverse.com/nervoussystem/designs`
[61] `https://electricsheep.org`

Obrázek 4.6: Produkty studia *Nervous system*

Autorem projektu Electric Sheep[62] je Scott Draves. Využívá druh fraktálů *Fractal flame*, který Draves napsal roku 1992 a následně zpřístupnil pod GNU General Public License (GPL) licencí. Oblíbenější animace – ty, pro které více lidí hlasovalo v aplikaci – žijí v systému déle a reprodukují se genetickým algoritmem pomocí mutace a křížení.

4.6 Vizualizace znalostí a informací

Jak již bylo zmíněno v 1.2, lidský mozek dokáže vizuálním kanálem přijímat velké množství informací. Jednou z aplikací vizualizací a grafiky je proto znázornění informací a spojitostí vizuálně. Například projekt *See, also*[63] agreguje různé vizualizace dat z *Wikipedie*.

Seealsology

Jako zástupce ze oblasti informačních vizualizací byl vybrán projekt Seealsology[64], od *DensityDesign Research Lab* z Milána, dostupný i se zdrojovým kódem na stránce *Github*[65].

Vstupem vizualizace jsou příspěvky na stránce *wikipedia.org* a nastavení, jaké odkazy a do jaké hloubky se mají procházet. Výstupem je graf propojení různých stránek na *Wikipedii*. Tento nástroj byl použit i pro hledání pojmů související s tématem této diplomové práce.

Aplikaci je vhodné používat interaktivně, kromě odkazů na vyhledané stránky je možné například pomocí klávesové zkratky přidat další vyhledávací vrchol v reálném čase. Příklad výstupu[66] pro stránku z *Wikipedie* na téma *creative coding* ukazuje obrázek 4.7.

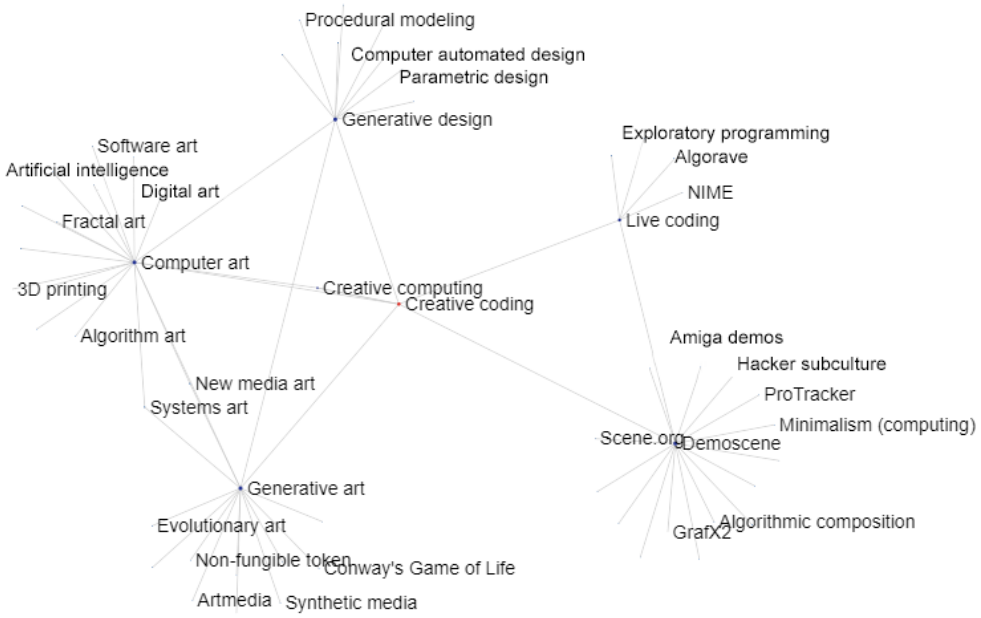
[62] Název je narážkou na román Philipa K. Dicka – *Do Androids Dream od Electric Sheep*? z roku 1968.

[63] `https://seealso.org`

[64] Zdroj: `https://densitydesign.github.io/strumentalia-seealsology/`

[65] `https://github.com/densitydesign/strumentalia-seealsology`

[66] Mírně editovaný s lepší viditelností popisků.



Obrázek 4.7: Příklad výstupu z projektu *Seealsology*

Six degrees of Wikipedia

Stránka *Six degrees of Wikipedia*[67] vyhledá nejkratší cesty mezi dvěma zadanými stránkami na *Wikipedii* a následně zobrazí orientovaný graf vybraných nejkratších cest. Jméno referuje hypotézu „Šest stupňů odloučení“, tedy myšlenku, že lidé jsou v průměru šest nebo méně sociálních kontaktů od sebe.

Obsidian

Obsidian[68] je aplikace pro psaní poznámek, obsahující jednoduchou grafovou vizualizaci. Vizualizace ukazuje všechny poznámky jako vrcholy grafu a hrany označují reference mezi poznámkami. Je dostupná i alternativa ve formě rozšíření pro textový editor *Visual Studio Code*[69].

[67] `sixdegreesofwikipedia.com`
[68] `https://obsidian.md`
[69] `https://foambubble.github.io/foam/`

KAPITOLA 5

Výukové kanály a materiály

Pod pojmem *creative coding* lze na internetu nalézt[70] mnoho knih, učebních materiálů a návodů pro různé technologie. Úkolem této kapitoly je vytvořit základní přehled výčtem příkladů volně dostupných učebních materiálů.

5.1 Processing a webové technologie

Processing je jedním z prvních z nástrojů kreativního kódování. Knihovna *Processing* je spolu s jeho interpretací v jazyce *JavaScript*, knihovnou *p5.js*, stále populární a používaný jazyk pro výuku kreativního kódování. Proto v této sekci budou pro ně společně s dalšími webovými technologiemi uvedeny příklady dostupných materiálů.

Dan Shiffman

Dan Shiffman je člem představenstva *Processing Foundation* a profesorem na univerzitě v New Yorku. Napsal knihy *The Nature of Code* a *Learning Processing*. Provozuje populární kanál na platformě *YouTube* jménem *The Coding Train*. Začátkem roku 2022 je počet jeho odběratelů 1,37 milionů.

The Coding Train *The Coding Train* je kanál na platformě *YouTube*. Zaměřuje se především na kreativní kódování v *Processingu* a *p5.js*. Kromě základů používání těchto technologií se zde nacházejí i ukázky implementace a využití různých algoritmů, například CA a fraktály. Ve videích jsou prezentovány i možnosti interakce pomocí pohybového senzoru *Kinect* a webkamery, využití strojového učení, či datových API.

[70] Rozcestníkem pro mnohé materiály je například tento komunitou vytvořený seznam: `https://github.com/terkelg/awesome-creative-coding`.

The Nature of Code Text i ukázky kódu knihy *The Nature of Code* jsou volně dostupné. Kniha se zaměřuje na představení a konkrétní strategie a techniky pro simulaci přírodních systémů za použití knihovny *Processing*.

Theodor Davis

Ted Davis vyučuje ve Švýcarsku kreativní kódování a *glitch*. Jeho webový editor pro živé kódování v knihovně *p5.js* byl již zmíněn v kapitole 3.1. Na platformě *YouTube* demonstruje jeho použití prostřednictvím kanálu *newandnewermedia*[71], kde se začátkem roku 2022 nachází kolem 22 hodin živého kódování vizualizací. Rovněž se zde nachází seznam video návodů představujících funkce tohoto editoru.

Keith Peters

Keith Peters se věnuje kódování v oblastech jakými je například grafika, animace, dynamika, interakce, fyzikální a matematických simulace, hry a generativní umění. Vytvořil a přispíval do řady projektů a napsal knihu *Playing With Chaos*.

Coding Math *Coding Math*[72] je nyní již neaktivní projekt výukových videí vyučující například kreativní používání matematiky a fyziky pro grafiku a animace. K videím a článkům jsou dostupné i zdrojové kódy.

BIT-101 Mezi současné veřejné materiály K. Peterse patří například jeho blog.[73]

Franks laboratory

Franks laboratory[74] je příkladem výukového kanálu na platformě *YouTube*, obsahující návody s tématikou kreativního programování čistě pomocí jazyka *JavaScript* a technologií HTML a CSS.

Three.js, Babylon.js

Dalším způsobem, jak se naučit konkrétní technologie, je oficiální dokumentace, například ty pro *Three.js*[75] či *Babylon.js*[76].

[71] `https://www.youtube.com/channel/UCGgCYuwGOm4LFslMsrTRhXA`
[72] `http://www.codingmath.com`
[73] `https://www.bit-101.com/blog/category/tutorial/`
[74] `https://www.youtube.com/c/Frankslaboratory`
[75] `https://threejs.org/docs/`
[76] `https://doc.babylonjs.com/`

5.2 Modelovací programy, vizuální programování, herní *enginy*

Tato sekce se věnuje softwarovým programům jakými jsou například *Touchdesigner*, *Unity*, *Unreal* či *Blender*. Prezentované příklady výukových materiálů v této sekci jsou na platformě *YouTube*.

Freya Holmér

Freya Holmér[77], známá také jako Acegikmo, je autorkou například například plug-in ve formě uzlového editoru do programu *Unity* (jež se stal průmyslovým standartem), nebo grafické knihovny *Shapes*. Na platformách *YouTube*, *Twitter* a *Twitch* vysvětluje počítačovou grafiku mimo jiné i prostřednictvím svých specifických animací. Acegikmo má i početnou komunitu na platformě *Discord*.

Erindale

Erindale[78] se zaměřuje na procedurální tvorbu v programu *Bledner*. Jeho návody se týkají například geometrického uzlového editoru, či výzvě *Nodevember* zmíněné v kapitole 2.6. Erindale má rovněž komunitu na platformě *Discord*.

Kohui – Noto The Talking Ball

Umělec Kohui (`https://kohui.xyz/`) vytváří audiovizuální díla inspirovaná přírodními fenomény, i společností. Zároveň produkuje tutoriály v programu Touchdesigner pod přezdívkou *Noto The Talking Ball*[79].

5.3 GLSL

V této sekci budou příklady dostupných materiálů pro grafiku vytvořenou pomocí psaní matematických funkcí, tedy například kódováním GLSL shaderů.

The Book of Shaders

The Book of Shaders[80] je webová stránka, která postupně vysvětluje tvorbu shaderů v jazyce GLSL. Autory jsou Patricio Gonzalez Vivo a Jen Lowe. *The Book of Shaders* je dostupná v několika jazycích.

[77] `https://www.youtube.com/c/Acegikmo`
[78] `https://www.youtube.com/c/Erindale`
[79] `https://www.youtube.com/c/NotoTheTalkingBall`
[80] `https://thebookofshaders.com`

Obrázek 5.1: 3D grafika vytvořena pouze pomocí matematiky

Inigo Quilez

Inigo Quilez je softwarový vývojář, technický umělec, produktový manager a učitel s dlouholetou praxí, jež pracoval například pro *Pixar*, či *Oculus*. O svém hobby, počítačové grafice, natáčí video návody[81], píše články a podílel se vývoji projektu *Shadertoy*.

Je autorem mnohých dem[82]. Na svých stránkách[83] má pod MIT licencí užitečné funkce a části kódu. Příklad jeho vizualizací v reálném čase ukazuje obrázek 5.1.

Martijn Steinrucken

Dalším příkladem video návodů na GLSL shadery je *The Art of Code*[84], jejímž autorem je Martijn Steinrucken. Součástí je i představení funkcí užitečných pro herní design.

[81] `https://www.youtube.com/c/InigoQuilez`
[82] Zmiňovaných zde 2.4
[83] `https://iquilezles.org/www/index.htm`
[84] `https://www.youtube.com/c/TheArtofCodeIsCool`

KAPITOLA 6

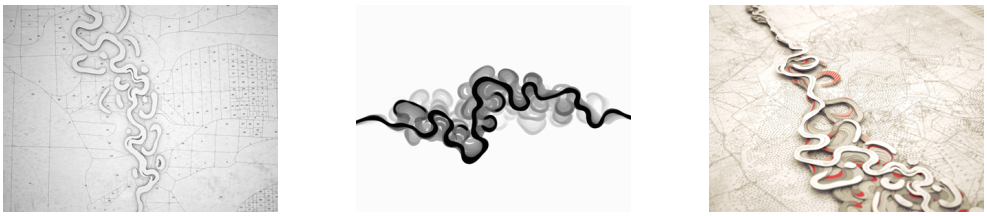
Osobní portfolia

Je mnoho artistů, kteří zasluhují obdiv. Pro účely této práce zde budou jmenováni někteří z těch, kteří algoritmy a postupy tvorby popisují a sdílejí prostřednictvím internetu, například formou blogových článků, či videí. Na rozdíl od předchozí kapitoly, jsou popsané algoritmy vysvětleny v kontextu vlastních uměleckých projektů.

6.1 Robert Hodgin

Robert Hodgin je mimo jiné například zakladatel designového a technologického studia *Rare Volume* a spolutvůrce C++ *frameworku Cidner*. Jeho práce se pohybuje od 2D datových vizualizací po imerzivní 3D simulace a zahrnuje teoretickou fyziku, astronomii, simulace hejn ptáků a audiovizualizace. Primárně pracuje v programu *Houdini*. Přednášel na řadě konferencí a festivalů.

Příklad z jeho portfolia[85] je uveden na obrázku 6.1. Jedná se o projekt *Meander*, procedurální systém pro generování historických map řek, které nikdy neexistovaly. Projekt a jeho vývoj je popsán pomocí textu, obrázků, videí i interaktivně.



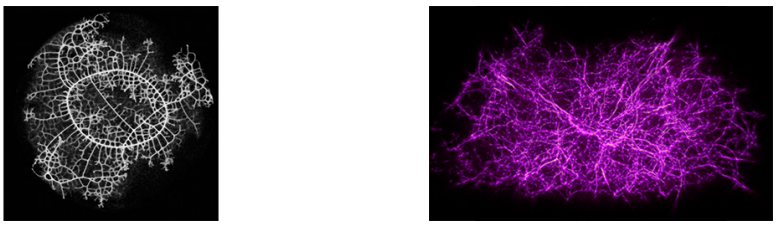
Obrázek 6.1: Projekt *Meandr*

[85] `https://roberthodgin.com/project/meander`

6.2 Sage Jenson

Sage Jenson[86] je umělec, jeho výzkum zahrnuje *biocomputation* a interakce pomocí hmatu. Vytváří interaktivní audiovizuální systémy a programuje chování organismů a bytostí. V jedné práci kterou popisuje na blogu[87] simuloval Vápenatku mnohohlavou (Physarum polycephalum, zmínění již v sekci 1.4 na obrázku 1.8). Implementace využívá popis z článku [16] a je napsána v jazyce C++ za použití nástroje *openFrameworks*.

Agentura *NASA* se při tvorbě simulace temné hmoty ve vesmíru inspirovala[88] právě touto prací, kde byla nalezena podobnost mezi chováním Vápenatky a sítěmi, které vznikly působením gravitační síly. Příklad výstupu Sage Jensona (vlevo) a 3D adaptace pro NASA od Oskara Eleka (vpravo) je na obrázku 6.2.



Obrázek 6.2: Sage Jenson, Oskar Elek

6.3 Webové a plotter projekty

Následující umělci svoje díla prezentují převážně pomocí webových stránek, či fyzicky vytištěné pomocí plotterů. Přínosem této sekce je inspirace a zdroj odkazů úspěšných, populárních, či technicky popsaných a propracovaných projektů.

Tyler Hobbs

Tyler Hobbs ve svých esejích[89] píše o generativním umění, kterému se věnuje profesionálně. Jeho slovy „používá způsob přemýšlení naučený z programování pro analýzu vizuálního světa kolem nás“. Některé jeho inspirativní přednášky (například z události *Strange Loop Conference*) a rozhovory jsou na platformě *YouTube*.

[86] Sage Jenson používá zájmeno „jejich“. Autorka v textu používá zájmeno „on“ kvůli nedostatku znalostí správného skloňování.

[87] `https://cargocollective.com/sagejenson/physarum`

[88] `https://www.nasa.gov/feature/goddard/2020/slime-mold-simulations-used-to-map-dark-matter-holding-universe-together`

[89] `https://tylerxhobbs.com/essays`

Tyler Hobbs ve svých projektech již od roku 2016 věnoval vektorovým polím (*flow fields*) a tuto techniku použil například ve svém díle *Fidenza*. *Fidenza* je příkladem digitální umění, které vzniká generativním skriptem (vytvořeným pomocí knihovny *p5.js*) a je prodáváno přes platformu *Art Blocks*. Výdělky z prodeje následně přispěl organizacím *Processing Foundation*, *Girls Who Code* či *AGE of Central Texas*. Příklad výstupu díla *Fidenza* zobrazuje obrázek 6.3.

Baptiste Crespy

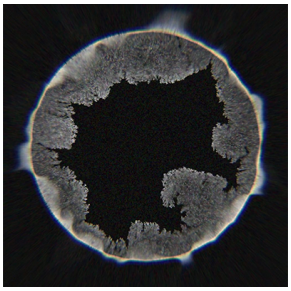
Baptiste Crespy (známý pod přezdívkou *ciphrd*) je generativní umělec a *WebGl* vývojář. Zajímá se například o autonomní systémy, tvořící život připomínající chování. Na svém blogu [90] dokumentuje jakým způsobem simuloval známé reakce a procesy, jakými je například *Diffusion-Limited Aggregation* či *Reaction–diffusion*. Obrázek 6.4 je příkladem jeho práce.

Je autorem projektu *Seed Simul*, interaktivní webové stránky, kde uživatelé mohou zasadit virtuální semínka ve 3D prostředí. Tyto vstupy od uživatelů poté pomalu časem rostou a tvoří se tak unikátní kolaborativní struktura. Momentálně *ciphrd* zakládá a vyvíjí platformu *FxHash*, sloužící pro tvorbu a sběr generativních NFT na *tezos blockchainu*.

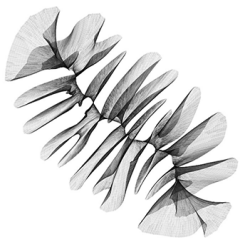
Anders Hoff

Další inspirativní generativní artista Anders Hoff sdílí svoje nadšení pro simulaci vzorů pomocí svého blogu *Inconvergent*[91]. Příklad jeho práce ukazuje obrázek 6.5. Kolem let 2015 a 2016 napsal generativní eseje inspirující se růstem vláken hub a stromů, či tvorby různých umělců. K těmto esejím je dostupný kód. Pozdější články se inspirují písečnými dunami (*Grains of sand*), či využívají grafovou strukturou (*Tangle of webs*). Je autorem frameworku pro generativní systémy jménem *Weir*[92], využívající programovací jazyk *Lisp*.

Obrázek 6.3: *Fidenza*



Obrázek 6.4: DLA



Obrázek 6.5: Incovergent

[90] `https://ciphrd.com`
[91] `https://inconvergent.net/generative/`
[92] `https://github.com/inconvergent/weir`

6.4 Virtuální světy a hry

Následující sekce se opět zaměřuje na vývojáře sdílející svůj kreativní postup s veřejností, ovšem zaměřuje se na ty, kteří tvoří převážně komplexnější interaktivní světy, či přímo hry. Zmínění autoři často používají pro sdílení obsahu platformu *YouTube*.

Sebastian Lague

Mezi tématy kanálu *Sebastian Lague*[93] je například tvorba modelů v programu *Blender*, seznam videí o herním vývoji a především seznam videí *Coding Adventures*. V rámci *Coding Adventures* je například implementován pohyb pomocí již umiňovaného článku [15], jsou simulovány ekosystémy, procedurální světy, termitiště a dále.

CodeParade

CodeParade[94] je známý například pro *Marble Marcher*, hru ve které je vytvořen fyzikálně responzivní svět tvořený fraktály, nebo hru *Hyperbolica* odehrávající se v neeuklidovském prostoru. Určité aspekty obou projektů jsou na kanále *CodeParade* popsány ve videích.

Martin Donald

Martin Donald[95] popisuje pomocí herního *enginu Godot*, jak ve vývoji her a virtuálních světů použít například algoritmus kolapsu vlnové funkce[96]. Pomocí krátkých videí využívající bezplatné či *open-source* programy jako *Blender* a *FlowMapPainter*, popisuje jak například vytvořit *vertex animation textures*, *flowmaps* a další. Občasné součásti videí jsou i doprovodné kresby či vizualizace matematických konceptů.

APH Games

Portál *APH Games*[97] poskytuje ucelené materiály týkající se vývoje her se zaměřením na uměleckou a *low–code* technickou stránku. Použitou technologií je především *PIXI.js*. V návodech je ukázáno například použití architektonického vzoru *Entity component system*. Stránky poskytují i řadu zdrojů, výběr článků a projekty z předmětu NI–APH. Součástí projektu jsou i videa[98] týkající se mimo jiné například tutoriálů či přednášek.

[93] `https://www.youtube.com/c/SebastianLague/`
[94] `https://www.youtube.com/c/CodeParade`
[95] `https://www.youtube.com/channel/UC8bYucAICXmYet8pZ5Ja9Dw`
[96] `https://github.com/mxgmn/WaveFunctionCollapse`
[97] `https://aphgames.cz` či `https://aphgames.io`
[98] `https://www.youtube.com/channel/UC1PTa9NCygXV_0IgOBRMw-Q`

Část II

Praktická část

KAPITOLA 7

Vizualizace 1, Video textures

Úvodem praktické části práce jsou představeny implementované vizualizace. V této kapitole je představena vizualizace detekce periodicity ve videích a dynamických texturách. U každé vizualizace bude představen její záměr, postup vývoje a implementace.

7.1 Cíl vizualizace

Cílem této vizualizace je zkoumat možnosti získávání periodicity z videa a kreativním způsobem vizualizovat data použitá pro tuto techniku. Je potřeba implementovat článek [23] popisující tuto problematiku a následně vytvořit vizualizaci, zobrazující syntézu pomocí dat z analýzy.

Motivace za vizualizací

Motivací za touto vizualizací je její následovné použití pro synchronizaci videa a hudby. Vstupními daty navazujícího projektu jsou videa ve standardizovaném stylu *rubber hose* animací, které byly populární kolem roku 1920. Z toho důvodu i pro tento projekt budou použita tato videa.

7.2 Videotextury

Video textury je typ média na pomezí fotografií a videem. Plynulý a nekonečně se opakující proud obrázků. Analýzou videa se obdrží jeho struktura, jíž syntézou vznikne nové video, podobné tomu původnímu, o *libovolné délce*. Použití video textur zahrnuje zobrazení dynamických scén na webových stránkách, tvorbu dynamických kulis pro scény například v počítačových hrách nebo interaktivní ovládání videem - poháněných animací. Způsob, jakým by se daly využít v tomto projektu je ukázán na stránce projektu [24].

Popis algoritmu

Prvním, kdo definoval pojem videotextura byl Schödl ve svém článku [23] z roku 2020, od té doby byly provedeny optimalizace a rozšíření, například [25] či [26]. Algoritmus je popsán například i v jedné z implementací [27].

Analýza videa

První částí je analýza videa a uložení dat do struktur, z nichž bude možné vycházet v následující části sestavení videotextury.

Černobílé hodnoty V matici $\mathcal{I}_i$ o rozměrech $M \times N$ (šířka x výška videa) jsou pro každý snímek i uloženy hodnoty od 0 do 255 (černobílý jas pixelu získaný z červené, zelené a modré složky).

Podobnost snímků Pro každou dvojici snímků se zjistí jejich vzdálenost (Euklidovská norma) a uloží do matice $D_{ij} = ||\mathcal{I}_i - \mathcal{I}_j||_2$.

Způsoby zachování dynamiky Pokud se objekt hýbe například zleva doprava, algoritmus by mohl mylně měnit směr. Zachování dynamiky se může zajišťovat například optickým tokem, či váženým oknem, kdy se v úvahu berou i příspěvky sousedících snímků (podle binomického rozdělení).

Vážené okénko Dalším krokem je přičtení příspěvků okolních $m+m$ snímků přenásobených o váhy $[w_{-m}, \ldots, w_{m-1}]$. Přepis matice je tedy

$$D'_{ij} = \sum_{k=-m}^{m-1} w_k D_{i+k,j+k}, \tag{7.1}$$

kde hodnota pohyblivého okénka m je typicky 1 či 2. Výsledkem je matice s rozměry $(N - 2m) \times (N - 2m)$, jelikož na každém kraji bude chybět m snímků pro porovnání.

Očekávaná budoucí cena Pro generování potenciálně nekonečného videa by bylo velmi nevýhodné vybrat přechod, ze kterého je málo přechodů, případně po kterém už video končí. Toto se optimalizuje výpočtem ceny přechodů. Definuje se matice D''_{ij} očekávané ceny přechodu ze snímku $i-1$ na snímek j součtem přes všechny očekávané ceny:

$$D''_{ij} = \left(D'_{ij}\right)^p + \alpha \sum_k P''_{jk} D''_{jk}, \tag{7.2}$$

kde

$$P''_{ij} \propto exp\left(-D''_{i+1,j}/\sigma\right). \tag{7.3}$$

Konstanta p parametrizuje jestli má být výsledkem spíše málo (ale kvalitních), nebo více (méně kvalitních) přechodů. Konstanta $0 < a < 1$ reprezentuje relativní váhy budoucích přechodů. Systém rychle konverguje pokud $0.99 < a < 0.999$.

Jedním ze způsobů řešení je simultánní iterativní vyhodnocování D''_{ij} a P''_{ij} přivádějící systém blíže do konvergence a přerušení, další výpočty neovlivňují matici. Bohužel tento způsob konverguje pomalu.

Jelikož při $\sigma \to 0$ půjde hodnota P''_{ij} k 1 pro nejlepší přechod a k 0 pro nejhorší, lze přepsat rovnici 7.2 na

$$D''_{ij} = \left(D'_{ij}\right)^p + \alpha \min_k D''_{jk}. \tag{7.4}$$

Tato rovnice je známá jako *Q-Learning* (metoda model-free reinforcement learningu). Díky tomu se matice bude procházet jako graf, kde vrcholy jsou vážené přechody. Je zde možnost selektivního přepočítávání hodnot řádků v D''_{ij} pro každý krok. Nejnižší cena často zahrnuje přechod ze snímku, který je blízko konci, což lze dále propagovat dopředu. Dále se inicializuje $D''_{ij} = \left(D''_{ij}\right)^p$ a definuje

$$m_j = \min D''_{jk} \tag{7.5}$$

Od posledního řádku k prvnímu a naopak se počítá

$$D''_{ij} = \left(D''_{ij}\right)^p + \alpha m_j \tag{7.6}$$

a aktualizuje se hodnota m_j použitím rovnice 7.5.

Pravděpodobnosti z ceny přechodů Dalším krokem je získanou cenu přechodů vyjádřit jako pravděpodobnost. Toho je docíleno namapováním na exponenciální funkci následujícím způsobem:

$$P''_{ij} = exp(-D_{i+1,j}/\sigma). \tag{7.7}$$

Tímto budou vyšší ceny přechodu namapovány na menší pravděpodobnost. Nižší hodnoty konstanty σ upřednostňují méně lepších přechodů, vyšší hodnoty více horších přechodů.

Výběr nejlepších přechodů Nabízejí se dva způsoby, jak zredukovat celkový počet stavů (uzlů/přechodů) v matici:

1. braní v potaz pouze lokální maxima, nebo
2. nastavení pravděpodobnost přechodů pod určitou hranici na 0.

Syntéza textur

V práci [27] jsou jmenovány 4 způsoby syntézy textur ze získané matice přechodů.

Náhodné přehrávání Náhodné přehrávání znamená, že není použito předchozí analýzy a v každém snímku i se vybere snímek j tak, že $j > i$. Původní článek uvádí náhodné přehrávání pro kontrast s informovaným náhodným přehráváním.

Nejdelší cyklus Nejdelší cyklus jednoduše identifikuje nejdelší možný cyklus z matice přechodů. Ve výsledku tohoto přístupu lehce pozorovatel pozná opakování.

Informované náhodné přehrávání Po přehrání snímku i je vybrán následující snímek j podle pravděpodobnostní matice P_{ij}.

Video loop play Kombinací primitivních smyček, které se překrývají, se vytvoří složená smyčka.

Mezi dostupné implementace videotextur patří například ještě [28], [29], [30], [31], [32] a [33]. Repozitář [33] funguje, má komentáře a testy. Pro použití by možná šel optimalizovat. Repozitář [27] má výborné video i text k videotexturám a kód se povedl po přeložení do jazyka *Python 3* a troše snahy autorce této práce spustit, avšak výpočet se po několika hodinách nedokončil na systému *Windows* ani *Linux*.

7.3 Implementace

Vizualizace byla rozdělena na část analýzy a interaktivní přehrávač syntézy snímků. Pro implementaci analýzy vstupních snímků videa byl použit programovací jazyk *Python 3*. Vizualizace přehrávání syntézy textur je uzpůsobena pro využití na webu. Pro vizualizaci byla použita knihovna *p5.js* a klasické webové technologie (*Javascript*, HTML a CSS).

Analýza videa

Implementace analýzy videa byla inspirována pracemi [27] a [33]. Analýzu je možné parametrizovat v konfiguračním souboru. Příklad vstupních parametrů ukazuje kód 7.1. Pro generování snímků ze vstupního videa je k dispozici *script* `generate_frames.sh` s parametrem cesty k videu.

Listing 7.1: Příklad parametrů

```
clock_parameters = {
  "input_folder"    : "clock",
  "qualityExponent" : 2,
```

```
  "futureCostAlpha" : 0.999,
  "sigmaMult"       : 2,
  "thresholdValue"  : 0.5,
}
```

input_folder Název složky obsahující snímky videa. Umístění složky v kmeni adresáře je nutnou podmínkou pro přehrávač textur.

qualityExponent Tento parametr ovlivňuje jestli má být výsledkem spíše málo kvalitních, nebo více méně kvalitních přechodů.

futureCostAlpha Parametr α ovlivňuje relativní váhu budoucích přechodů. Doporučením pro jeho hodnoty je interval $(0.99, 0.999)$.

sigmaMult Parametr *sigmaMult* je použit při stanovení hodnoty σ, která je použita při převádění ceny přechodu na pravděpodobnost přechodu. Hodnota σ je definována průměrem matice *očekávaných budoucích cen* přenásobeným o *sigmaMult*. Nižší hodnoty σ upřednostňují méně lepších přechodů, vyšší více horších přechodů.

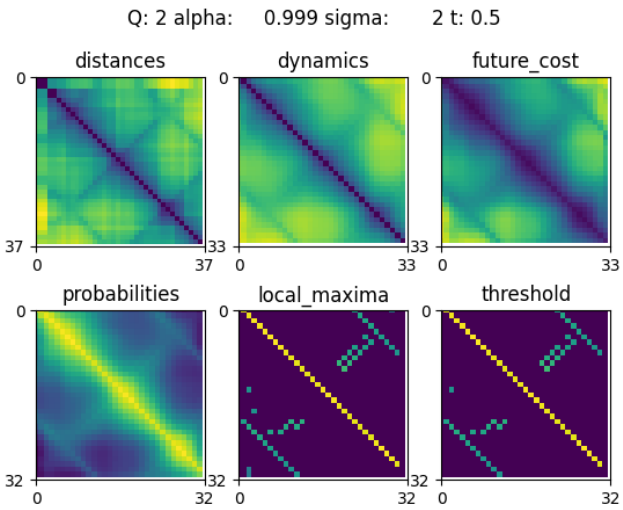
thresholdValue Tento parametr udává od jaké pravděpodobnosti se má pravděpodobnost zahodit, tedy nastavit jako 0%.

Výsledná matice pravděpodobnosti přechodů i postupné kroky algoritmem byly vizualizovány do grafů pomocí knihovny *Matplotlib*. Barevné spektrum se aplikuje pouze v rámci absolutních hodnot konkrétního grafu. Nejmenší hodnoty mají fialovou barvu, největší žlutou. Škála je zobrazena na obrázku 7.1 a je známá pod názvem *viridis*. Rozhodnutí při volbě mapování barev byla inspirována prací [14]. Vizualizace je důležitá pro ladění parametrů.

Obrázek 7.1: Volba barevného schématu

Na obrázku 7.2 je zobrazen příklad výstupu matic podobnosti analytické části. Na ose x a y každého grafu jsou čísla snímků. První matice *distances* znázorňuje vzdálenosti snímků. Další matice *dynamics* je výsledkem aplikace binomického filtru a matice *future_cost* zohledňuje cenu budoucích přechodů. Matice pravděpodobností *probabilites* je inverzní, díky rovnici 7.7. Další grafy ukazují redukci hodnot, *local_maxima* ponechání lokálních maxim a *threshold* odstranění hodnot pod zadanou pravděpodobnost.

Nastává otázka, kolik snímků je mezi sebou vhodné porovnávat a testovat tak jejich podobnost. Je zřejmé, že porovnávané jsou takové snímky, mezi kterými očekáváme nalezení podobnosti. Bylo implementováno „posuvné okénko“

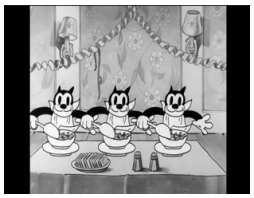


Obrázek 7.2: Příklad výstupu matic podobnosti z analytické části ukázky *clock*

o velikosti m. Nejprve jsou generováno pole intervalů $(k * m, (k + 1) * m)$ a $(k * (m/2), (k + 1) * (m/2)$ tak, aby byly pokryty všechny snímky m. Tyto intervaly jsou vstupy předané do funkce `videotexture_analysis`, kterou *pool* procesů. Jednotlivé výsledky jsou následně sjednoceny do matice přechodů. Příklad složené matice přechodů je na obrázku 7.6. Z toho příkladu je patrné, že by větší posuvné okénko zaznamenalo více přechodů. Výhodou menšího posuvného okénka je lepší paralelizovatelost a tedy kratší dobu výpočtu.

Obrázek 7.3: Ukázka *schody*

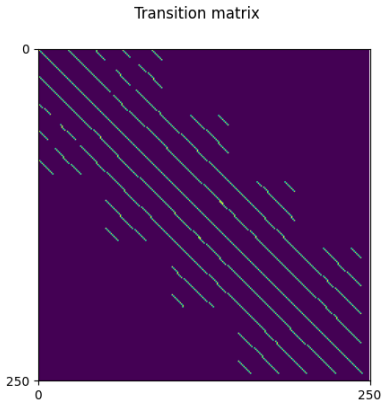
Obrázek 7.4: Ukázka *tráva*



Obrázek 7.5: Ukázka *cartoon*

Pro testování výsledků byla mimo jiné použita videa z profesionální databáze dynamických textur *DynTex* [34]. Bylo provedeno porovnání získání smyček videí s metodou založenou na trojitě toroidních 4D dlaždicích dynamických textur [35, 36, 37]. Příklad ukázky *schodů* (`54pe210`) je na obrázku 7.6.

Metoda video textur zachovává velmi dobře dynamiku videa například v ukázce *schody* 7.3. V případech videí, které jsou buď více náhodné a méně strukturální (*tráva* 7.4), nebo je v nich vysoká shoda snímků napříč videem (ukázka *cartoon* 7.5) pak poskytuje výsledky srovnatelné, nebo lepší. V porovnaní s matematickým modelem *CAR*, publikovaným Haindlem v [38] produ-



Obrázek 7.6: Složená matice přechodů ukázky *schody*

kuje metoda video textur ostřejší výsledek. Na rozdíl od matematických modelů zde prezentovaná metoda totiž upravuje pouze obsah některých snímků (*crossfading*), a ostatní ponechává v kvalitě originálu, její kompresní poměr je však o řády horší.

Možnosti vylepšení implementace analýzy videa

Závěrečná matice přechodů (obrázek 7.6) je zbavena informace o pravděpodobnosti přechodů a závěrečná syntéza pracuje pouze s binární hodnotou zda existuje, či neexistuje přechod. Hledání odpovídajících vygenerovaných obrázků může být uživatelsky nepohodlné a do budoucna by i část analýzy mohla být vybavena grafickým rozhraním pro zadávání parametrů a zobrazení matic.

Interaktivní syntéza videa

Pro syntézu videa byla kromě triviálního lineárního přehrávání přidána možnost informovaného náhodného přehrávání. Toto informované náhodné přehrávání může běžet automaticky, nebo může být odkrokováno vypnutím automatického přehrávání a stisknutím tlačítka *jump*. Díky inspiraci z práce [26] bylo do části syntézy přidáno prolínání mezi snímky.[99] V prohlížeči lze vybrat žádné prolínání, nebo malé či velké.

[99] V přehrávači nazvané *crossfading*.

KAPITOLA 8

Vizualizace 2, Crown Shyness

8.1 Cíl vizualizace

Cílem této vizualizace je hledání způsobů jak simulovat jev *Crown Shyness* a výsledek použít například jako pozadí webové stránky. V kapitole bude představen vývoj vizualizace s pomocí různých, v počítačové grafice známých a používaných, matematických konstruktů a metod.

Motivace za vizualizací

Tento jev byl vybrán pro simulaci pro jeho možnost implementace, jako generativní *script*, zmíněný v sekci 2.9. Není tedy cílem vytvořit jednu statickou verzi, ale vizualizaci, která bude parametrizovatelná a při každém generování volaná s náhodnými parametry. Kromě estetických parametrů (jako například barva stromů) lze u vizualizace parametrizovat například:

- počet stromů,
- způsob rozmístění stromů,
- velikost korun stromů,
- míra členitosti větví a
- mezera mezi korunami stromů.

8.2 Crown Shyness

Crown shyness je přírodní fenomén, který se vyskytuje u některých druhů stromů, kdy se koruny stromů navzájem nedotýkají, ale tvoří se mezi nimi

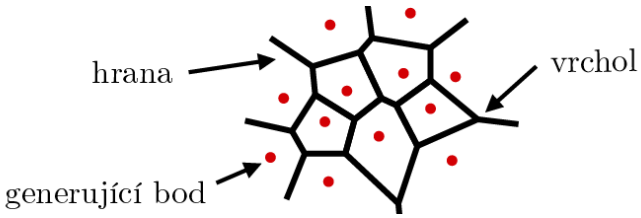
prázdná místa. Hypotézy pro tento jev debatuje například práce [39]. Vizuální inspirací jsou fotky na obrázku 8.1[100].

Obrázek 8.1: Fotografie jevu *crown shyness*

Voroného diagram

Z pozorování fotografií a videí jevu *crown shyness* byla učiněna úvaha, že koruny mají strukturu podobnou Voroného diagramu. Byly implementovány různé použití této myšlenky.

Voroného diagram prvního řádu[101] dané množiny bodů je kolekce oblastí, které rozdělují rovinu. Každá oblast koresponduje s jedním ze zadaných bodů a všechny ostatní body náležící jedné oblasti jsou v dané metrice blíž svému odpovídajícímu bodu, než všem ostatním. Typicky používanou metrikou je Euklidovská metrika. Na obrázku 8.2 je popsána terminologie.



Obrázek 8.2: Popis Voroného diagramu

8.3 Postupný vývoj

Na následující stránkách budou popsány různé verze implementace vizualizace včetně technologií a náhledu. Pro všechny verze byla použita knihovna *d3-delauay*[102] pro optimalizované generování Voroného diagrmau.

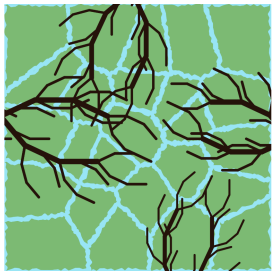
[100] Zdroj: `https://en.wikipedia.org/wiki/Crown_shyness#/media/File:Dryobalanops_Aromatica_canopy.jpg`, `https://art-sheep.com/25-truly-beautiful-photos-depicting-crown-shyness/`

[101] Voroného diagramy jsou samy o sobě v grafice poměrně využívané a známé. Ne tolik však Voroného diagramy vyššího řádu pro skupiny bodů.

[102] `https://github.com/d3/d3-delaunay`

První verze, Voroného diagram

Verze:	1.0
Technologie:	p5.js, P2D renderer
Funkce:	nepravidelné hrany, členité větve



Statická první verze zkoumá možnosti knihovny *p5.js*. Z bodů vygenerovaných do téměř pravidelné mřížky byly získány polygony pomocí Voroného teselace. Tyto polygony reprezentují koruny stromů.

Hrany polygonů byly vykresleny pomocí nepravidelých čar pro dojem listů. Jedna taková úsečka byla zkonstruována z několik menších čar s malou náhodností směru. Je to jeden z důvodů, proč by následná animace této verze mohla být problém.

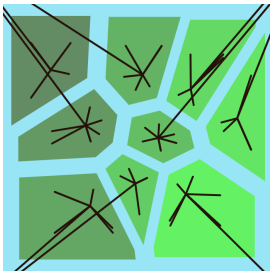
V této verzi byly pro generování členitých větví zkoumány možnosti *L-Systémů*, tedy jedné ze skupin fraktálů typicky používané pro generování rostlin. Větve jsou generovány náhodně a nesouvisí se skutečnou pozicí korun stromů. Toto by bylo možné napravit například generováním korun stromů podle pozice konců větví. Následně se ale další verze rozvíjely jiným směrem.

Zafixované větve, animace

Verze: 1.1

Technologie: p5.js, P2D renderer

Funkce: animace



Další verze měla za cíl najít optimální body pro umístění konců větví. Byly zvoleny body polygonu posunuté směrem ke středu o násobek vzdálenosti ke středu, tedy příslušnému generujícímu bodu. Tímto způsobem byly zkráceny i všechny vrcholy polygonu, pro vytvoření mezery mezi polygony. Kmeny stromů mají počátek v nejbližším rohu. K této souřadnici je přičtena mírná náhoda, aby stromy neměly počátek ve stejném místě.

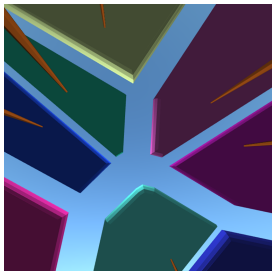
Všechny generující body byly animovány. K souřadnici ve směru osy x byl přičten násobek funkce *cosinus* s parametrem času přenásobený o šum rovněž s parametrem času. Případ souřadnice ve směru osy y byl analogický s výjimkou použití funkce *sinus* místo *cosinu*. Díky tomu se vytvořil mírně proměnlivý krouživý pohyb. Pro zachování vlastností Voroného diagramu je v každém kroku diagram přepočítáván. Tímto způsobem je vizualizace připravena pro implementaci ve 3D.

Three.js verze

Verze: 2.0

Technologie: Three.js

Funkce: 3D, osvětlení



Další iterace zkoumala možnosti 3D a byla pro ní použita knihovna *Three.js*. Motivací posunutí do 3D byla možnost použití perspektivy, místo její simulace ve 2D. Další motivací byla možnost využití GLSL*shaderů* a světla.

Základem pro vytvoření geometrie objektů byly polygony generovány analogickým způsobem jako v předchozích verzích. Z těchto polygonů byl vytvořen `THREE.Shape`, který byl obohacen o další dimenzi pomocí `THREE.ExtrudeGeometry`. Díky tomu je možné objektům přidat materiál. Pro přidání vlastního materiálu pomocí *shaderů* by avšak bylo vhodné geometrii rozdělit (subdivide) na více trojúhelníků. Kmeny stromů jsou v tomto případě jednoduché kužely.

Díky použití třídy `OrbitControls` je možné scénu ovládat pomocí myši, tedy přibližovat a otáčet pohled. Nevýhodou této verze byl jev, který nastával během animace. Koruny stromů měnily svůj tvar a svou velikost nepřiměřeným způsobem. I v této verzi se bohužel pro animaci v každém kroku vyhodnocuje Voroného diagram a následně se upravuje geometrie korun stromů, což by v případě většího počtu stromů bylo prakticky nepoužitelné. Z těchto důvodů se následně od tohoto 3D provedení upustilo a bylo snahou nejprve optimalizovat původní 2D řešení.

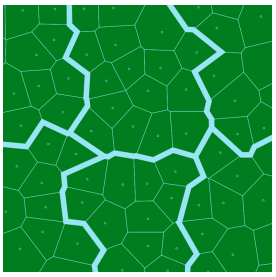
Důležitou vlastností této a následujících vizualizací je generování velikosti plátna v závislosti na velikosti okna prohlížeče. Toto je ovšem pro potřeby pozadí webu ještě dále upravit tak, aby se velikost vizualizace přizpůsobovala i změně v rozlišení.

Verze 3

Verze: 3.0

Technologie: p5.js

Funkce: složená koruna stromů



Verze 3 se vrací ke knihovně *p5.js* a zkoumá nové možnosti generování korun stromů. Nejprve byl upraven počáteční algoritmus pro generování počátečních bodů. Před přidáním každé souřadnice bodu bylo otestováno, zda v definovaném okolí už bod není. Tímto způsobem bylo docíleno náhodnější distribuce bodů.

Další myšlenkou této verze je implementace korun stromů jako skupinu polygon. První přímočarou myšlenkou bylo rozřazení polygonů do skupin podle pozice souřadnice generujícího bodu v mřížce. Takto byly polygony rozřazeny do skupin. Pro získání souřadnic (často nekonvexního) polygonu obalující polygony náležící jedné skupině, byla použita knihovna `Polybool.js`. Hranice této obálky byly na zvýrazněny silněji, než hranice jednotlivých polygonů, které tvoří korunu stromu.

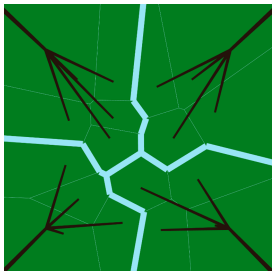
Řídící body všech polygonů byly animovány jako v předchozích verzích. Kontrolovali ovšem pozice ostatních řídících bodů a posunuly se pouze pokud v definovaném okolí žádný další nebyl. Tato verze neodstranila neefektivitu. Účelem této verze bylo odstranit nepřiměřeně se zvětšující koruny stromů a snažila se tím docílit rozdělením koruny stromu na menší polygony, které se nemohou k sobě přiblížit na definovanou vzdálenost. Popsaný způsob problém částečně vyřešil, nastal avšak problém oddělených dílčích polygonů od koruny.

Použití `WEBGL` rendereru

Verze: 3.1

Technologie: p5.js, `WEBGL` renderer

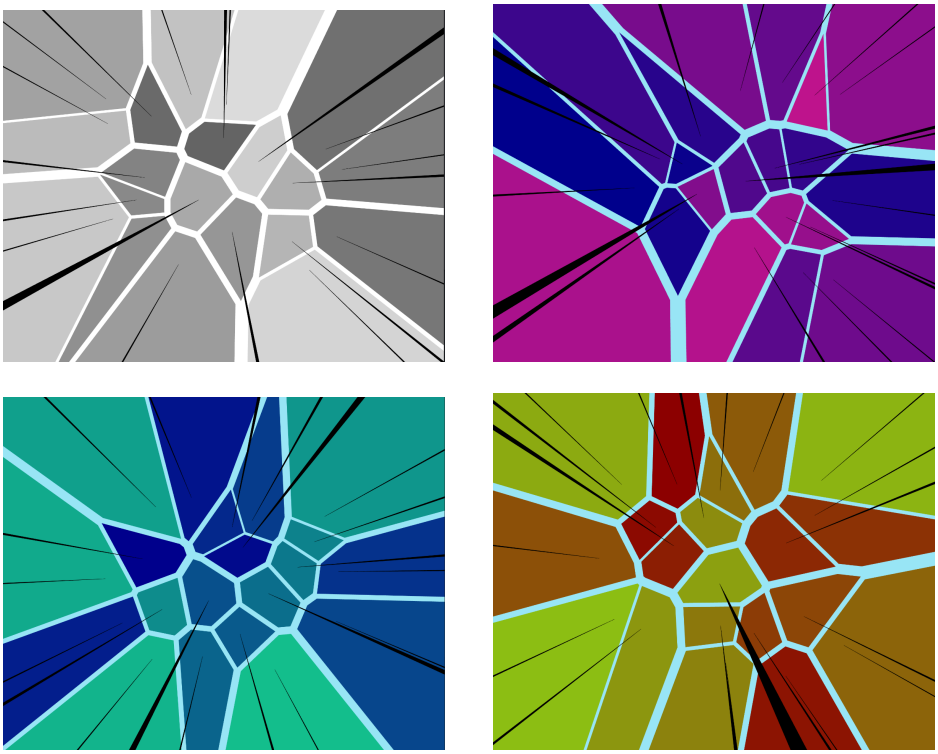
Funkce: perspektiva pomocí `WEBGL`



Cílem další verze bylo vyzkoušet možnosti změny rendereru `P2D` na `WEBGL`. Díky `WEBGL` rendereru je možné pracovat v pomocí knihovny *p5.js* ve 3D. Pro úpravu stávající aplikace na tento renderer bylo potřeba pouze posunout souřadnice, neb je střed `WEBGL` rendereru uprostřed, oproti levému hornímu rohu jak to bylo u `P2D` rendereru.

Celé dosavadní plátno z vizualizace *Verze 3* bylo vykresleno na vzdálenější souřadnice z. Následně byly přidány jednoduché stromy podobné *Verzi 1.1*. I přes jednoduchost scény má vizualizace díky 3D dojem perspektivy a hloubky.

Nabízí se zde vylepšit tuto vizualizaci například sofistikovanějším generování stromů, nebo zaměnit pravidelné regiony stromů za méně pravidelné. Výsledná vizualizace je avšak stále nevhodná jako pozadí webu pro výpočetní složitost a proto tato verze nebyla dále rozvíjena.



Obrázek 8.3: Barevná témata verze 4.0

Popis finální implementace

Finální implementace používá technologie z verze 3.1, tedy knihovnu *p5.js* s `WEBGL` rendererem. Byl dodržen požadavek použitelnosti pro webové pozadí a vizualizace je proto velmi jednoduchá. Body se generují pouze při načtení stránky, či změně velikosti okna prohlížeče. Animaci v tomto případě poskytuje autonomní otáčení kamery. Vizualizace reaguje na pohyb myši a lze ji pomocí kolečka myši přibližovat a oddalovat. Díky použití perspektivní kamery nastávaly nepřiměřené změny vizualizace při atypických poměrech stran prohlížeče. Bylo snahou tyto problémy vyřešit pomocí změny úhlu perspektivní kamery v závislosti na poměrech stran vizualizace. Při spuštění vizualizace se náhodně vybere schéma z předdefinovaných barevných schémat, z nichž některá jsou na obrázku 8.3.

KAPITOLA 9

Další vizualizace

Dalším vizualizacím menšího rozsahu vybraným pro tuto práci bude věnována společná kapitola. Vizualizace byly vybrány pro doplnění pestrosti spektra uměleckých a technických vizualizací. K vizualizacím inspirovaným umělcem Kensuke Koike byl v aplikaci *Metaviz* zpracován popis.

9.1 Vizualizace 3, Kensuke Koike

Kensuke Koike[103] je umělec původem z Japonska, který produkuje své umění pomocí fyzické manipulace s fotografiemi. Pro své umění často používá staré černobílé fotografie. Následující vizualizace byla přímo inspirována jeho prací. Jedná se o digitalizaci jeho tvorby.

Implementace

Byly implementovány dvě adaptace díla umělce Kensuke Koike. Konkrétní dílo, kterým jsou vizualizace inspirovány ukazuje obrázek 9.1. První, zobrazena na obrázku 9.2, pomocí knihovny *p5.js*. Vizualizace byla provedena naivním způsobem načtení obrázkových dat do paměti, otočením *canvasu* a vykreslení pouze těch pixelů, vzdálených o specifický poloměr od středu plátna. Další implementace (na obrázku 9.3) používá knihovnu *Three.js*. Začátkem je plátno tvořené texturou s původním obrazem. Obrazec nazývající se Penrosův trojúhelník je tvořen trojicí stejných objektů rovněž s původní texturou s tím rozdílem, že objekty jsou posunuté a orotované pro vytvoření dojmu Penrosova trojúhelníka. Výsledný komplexní popis implementace včetně ukázek kódu, ilustrací a rovnic vysvětlující fungování vizualizace je v příloze C.

[103] `https://www.kensukekoike.com`

Rešerše prací inspirovaných tímto umělcem

Jedna z prací[104] inspirovaných tvorbou umělce Kensuke Koike využívá nápad vyřezání z obrázku pravidelné obdélníky a pomocí nich složit nový obraz. Další nalezené práce[105] umožňují vyřezávaný obdélník zvětšit mimo možnosti fyzické fotky.

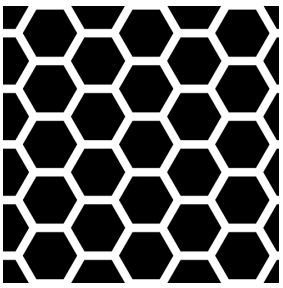
Obrázek 9.1: Kensuke Koike

Obrázek 9.2: Vizualizace *Circles*

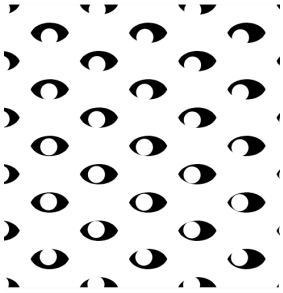
Obrázek 9.3: Vizualizace *Penrose*

9.2 Vizualizace 4, Painting

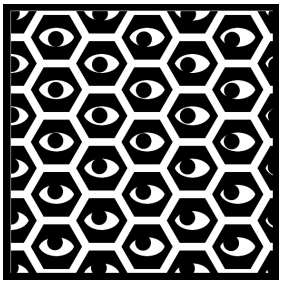
Další příklad implementované vizualizace ukazuje obrázek 9.6 vpravo. Jedná se o velmi jednoduchou vizualizaci, která reaguje na pohyb myši v prohlížeči. Vizualizace (obrázek 9.6) je provedena za použití knihovny *p5.js* a je tudíž možné ji spustit v prohlížeči. Obraz tvořený okem tvořeným dvěma polokruhy (obrázek 9.4) je replikován do šestiúhelníkové mřížky (obrázek 9.5).



Obrázek 9.4: První krok vizualizace – šestiúhelníky



Obrázek 9.5: Druhý krok vizualizace – oči reagující na myš



Obrázek 9.6: Výsledná vizualizace *Painting*

[104] `https://stephenpardy.github.io/projects/kensukekoike.html`
[105] `https://positlabs.github.io/selectif/`, `https://github.com/zij212/style_image`

KAPITOLA 10

Analýza

Před návrhem a implementací finální podoby aplikace *Metaviz* byla provedena analýza, která je obsahem této kapitoly. Byla provedena rešerše podobných aplikací a definovány požadavky na aplikaci.

10.1 Rešerše podobných webových aplikací

Před návrhem podoby samotné aplikace *Metaviz* byla provedena rešerše aplikací, mající podobný záměr, tedy umožňující zobrazit vizualizace. Cílem této sekce je definovat parametry, které budou aplikovány při návrhu aplikace.

Cineshader, ISF, Shadertoy, Threejs

První skupinou analyzovaných aplikací jsou ty, zastřešující jednu konkrétní technologii. Obsah tvořený příspěvky od komunity je častým společným rysem těchto aplikací. Budou zde analýzy aplikací zaměřující se na GLSL.

Shadertoy[106] umožňuje vytvářet, upravovat a prohlížet GLSL *shadery* ostatních uživatelů a byla spuštěna v roce 2013. Umožňuje komentáře a pro vývoj je zde možnost přidat vstupy například z webové kamery a mikrofonu.

Interactive Shader Format (ISF) je souborový formát popisující GLSL *shadery* pro použití v interaktivních aplikacích. Webová stránka[107] tohoto projektu je příklad použití ISF. Stránka umožňuje vytvořit nový shader pomocí této technologie a je možné procházet existující shadery. ISF poskytuje shaderům Graphical User Interface (GUI).

[106] `https://www.shadertoy.com`
[107] `https://editor.isf.video`

Cineshader[108] je vizualizér GLSL shaderů z API aplikace *Shadertoy*. Kromě estetické vizuální stránky je zde přítomný i hudební podkres.

Devart

Devart[109] je příkladem aplikace, kde je více přispěvatelů, ale projekty jsou tvořeny v jiných technologiích. Jedná se o kolekci kreativních projektů různých výstav, kde vývojáři – umělci popisují své projekty po technické a technologické stránce. Mezi funkce webové aplikace patří mimo vyhledávání filtrace programovacích jazyků a technologií. Projekty ovšem nejsou v aplikaci vizualizovány, je dodán multimediální obsah.

GenerativeArtistry, Jurasic, RedblobGames

Další skupinou aplikací jsou interaktivní návody, které například postupným *scrollováním* po stránce rozkrývají části kódu, který produkují dané vizualizace. Příkladem takových interaktivních tutoriálů je *GenerativeArtistry*[110]. Stránky jsou vygenerovány pomocí knihovny pro statické generování stránek *Hugo*. Dalšími interaktivními stránkami s vysvětlením algoritmů jsou například *RedblobGames*[111] a *jurasic*[112].

Výsledky rešerše podobných aplikací

Mnohá řešení jsou různým způsobem interaktivní. V případě některých aplikací lze vytvářet uživatelské účty a přidávat vlastní artefakty. Některé aplikace umožňují přidávat komentáře. Dalším aplikace umožňují interagovat s vizualizací, například upravováním přednastavených parametrů, nebo pohybem myši, vstupem z mikrofonu, webové kamery a podobně. Pro potřeby některých stránek je statická webová aplikace dostatečná.

1. Mají mít vizualizace GUI?
   a) Uživatelské rozhraní je vhodné minimálně v kontextu komplexnějších vizualizací, viz Kapitola 7.
   b) V případě uměleckých vizualizací může být případný text a GUI rušivým elementem.
2. Do jaké míry má doprovodný materiál objasňovat funkci vizualizace?
   a) V případě estetické vizualizace (viz Kapitola 7) se lze domnívat, že doprovodný materiál není stěžejní.

[108] `https://cineshader.com`
[109] `https://devart.withgoogle.com`
[110] `https://generativeartistry.com/tutorials/`
[111] `https://www.redblobgames.com`
[112] `https://www.jurasic.dev`

   b) V případě technické vizualizace (viz Kapitola 8) je objasnění vhodné.

3. Má být prohlížeč vizualizací součástí aplikace? Pokud ano, měl by být estetický?

   a) V případě, že by aplikace měla zobrazovat všechny vizualizace, mohlo by být požadavkem, aby byla aplikace snadno rozšiřitelná o další technologie. Náročnost rozšiřitelnosti závisí na množství použitých technologií.

10.2 Požadavky na aplikaci

Aplikace *Metaviz* má za cíl představit konkrétní vizualizace, čili požadavek na přidávání dalších artefaktů komunitou není ambicí. Aplikaci je však nutné přizpůsobit možnosti použití různých technologií.

Funkční požadavky

V této části budou vymezeny funkcionality, kterými bude aplikace disponovat. Každému funkčnímu požadavku bude přiřazena priorita a složitost na škále od 1 do 5, kde 1 znamená nejmenší prioritu, či složitost.

- **FP01 Zobrazení vizualizací**

  Aplikace by měla zobrazovat vizualizace, či by měla na vizualizace poskytnout odkaz.

  – **Priorita:** 5

  – **Složitost:** 3

- **FP02 Stručný popisu vizualizací**

  Součástí vizualizací bude stručný popis jejich významu a částečný popis jejich fungování.

  – **Priorita:** 4

  – **Složitost:** 2

- **FP03 Zobrazení kódu**

  Aplikace umožní prohlížet zdrojový kód vizualizací, například pomocí odkazu na repozitář.

  – **Priorita:** 3

  – **Složitost:** 1

- **FP04 Tutoriál**

  Součástí informací o vizualizaci bude návod jak byla vizualizace vytvořena.

  – **Priorita:** 2
  – **Složitost:** 4

- **FP05 Online dostupnost vizualizací**

  Implementované vizualizace budou dostupné přes web.

  – **Priorita:** 2
  – **Složitost:** 5

Nefunkční požadavky

Následující text představuje technické požadavky na aplikaci.

- **NFP01 Webová aplikace**

  Aplikace bude dostupná výhradně přes web.

- **NFP02 HTTPS**

  Aplikace bude komunikovat za pomoci protokolu *HTTPS*[113].

- **NFP03 Responzivní design**

  Aplikace bude responzivní pro mobily.

Případy užití

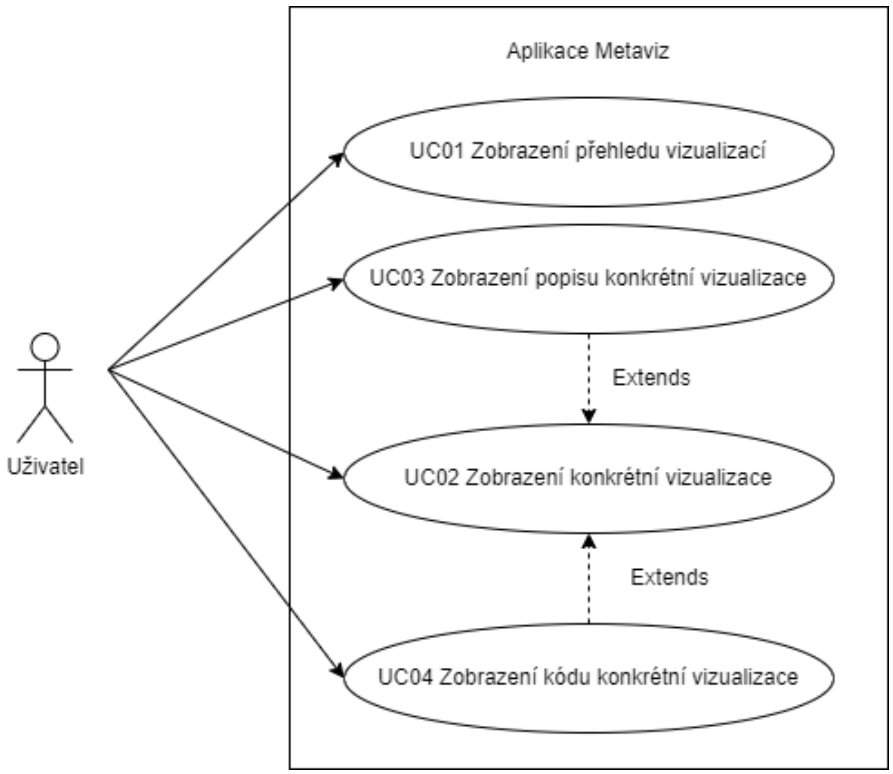
Případy užití rozvíjí funkční požadavky a dále je specifikují a člení na jednotlivé současné případy užití. Pokrytí všech funkčních požadavků případy užití je znázorněno v tabulce 10.1. Diagram aktivit je zobrazen na obrázku 10.1.

1. **Zobrazení vizualizací**
   - UC01 Zobrazení seznamu vizualizací
   - UC02 Zobrazení konkrétní vizualizace
2. **Zobrazení popisu vizualizací**
   - UC03 Zobrazení popisu konkrétní vizualizace
3. **Zobrazení kódu**
   - UC04 Zobrazení kódu konkrétní vizualizace

[113]Hypertext Transfer Protocol Secure

	FP01	FP02	FP03	FP04	FP05
UC01	x	x	-	-	x
UC02	-	-	x	-	x
UC03	-	x	-	x	-
UC04	-	-	x	-	-

Tabulka 10.1: Tabulka pokrytí funkčních požadavků případy užití



Obrázek 10.1: Diagram případů užití

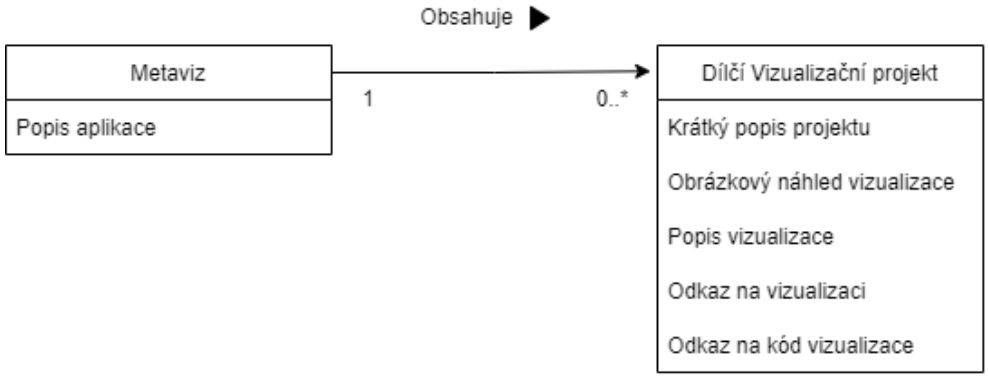
KAPITOLA 11

Návrh

Před implementací aplikace *Metaviz* byl proveden návrh. V této kapitole bude popsán doménový model, struktura a architektura aplikace. Budou zde popsány používané technologie a návrh uživatelského rozhraní.

11.1 Doménový model

Samotná aplikace *Metaviz* slouží ke sdružování vizualizací. Aplikace umožňuje uživateli prohlížet vizualizace, ale neumožňuje je nahrávat ani upravovat. UML diagram doménového modelu ukazuje obrázek 11.1.



Obrázek 11.1: Doménový model

Jekyll	Hugo
prerekvizity pro instalaci	jednoduchá instalace
jazyk Ruby	jazyk Go
více témat (∼1200)	méně témat (∼370)
podpora od organizace GitHub	rychlejší kompilace
funkce dostupné pomocí plug-inů	vestavěné funkce
nový projekt má výchozí téma	novému projektu je nutné přiřadit téma

Tabulka 11.1: Porovnání některých funkcí technologií *Jekyll* a *Hugo*

11.2 Technologie

Popis vizualizací vyžaduje rozsáhlé možnosti formátování. Pro tyto účely byl vybrán značkovací jazyk *Markdown*, který lze převést do HTML a splnit tak požadavek dostupnosti na webu. Z analýzy vyplývá, že výsledná aplikace popisující vizualizace je podobná webovému blogu. Z toho důvodu se nabízí použít technologii pro statické generování stránek. Při výběru konkrétní technologie se ze všech možností volba zúžila na rozhodnutí mezi technologiemi *Jekyll* a *Hugo*. Porovnání vybraných funkcí je v tabulce 11.1.

Díky vlastnostem rychlé kompilace a jednoduché instalace a předchozí zkušenosti s *frameworkem Hugo*[114] byla vybrána konkrétně tato technologie. Motivací za použitím těchto technologií je i vyřešení nefunkčního požadavku rezponzibility, jelikož existují témata řešící tuto problematiku.

Hugo

Hugo je *framework* pro statické generování stránek. To znamená, že se stránka generuje v pouze v případě tvorby či úpravy obsahu, nikoliv při požadavcích návštěvníků stránky (jak je to v případě dynamicky generovaných stránek). Staticky generované stránky jsou vhodné v případě, že se obsah málo mění.

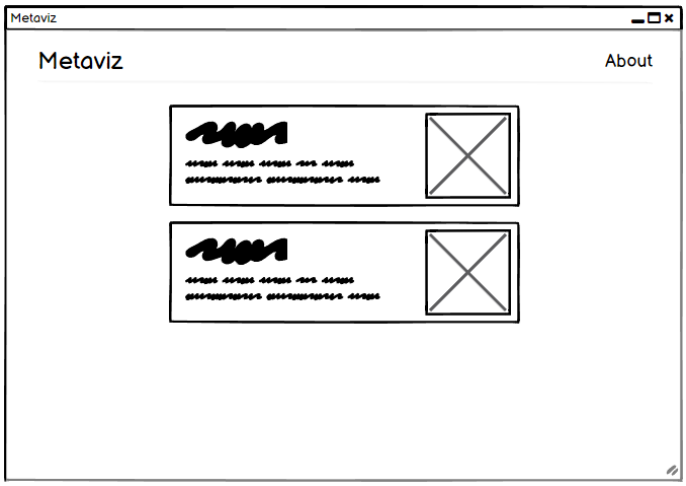
11.3 Struktura a architektura

Aplikace nevyžaduje přihlašování, ani rozhraní pro úpravu obsahu. Příspěvky budou uloženy v souborech *Markdown* a z toho důvodu není potřeba databáze. Výsledné statické stránky vygenerované technologií *Hugo* má pouze klientskou část a z toho důvodu je architektura označená jako monolitická.

[114] `https://gohugo.io`

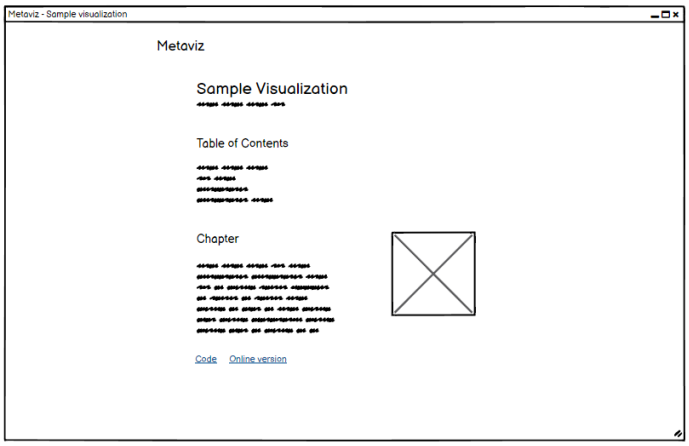
11.4 Uživatelské rozhraní

Uživatelské rozhraní by mělo být jednoduché, elegantní a přímočaré. Obrázek 11.2 zobrazuje *low-fidelity* prototyp domovské stránky aplikace. V aplikaci by bylo být možné zobrazit seznam vizualizací, který by měl ukázat její název, náhled, popis a měla by umožňovat přesměrování na více informací o vizualizaci.



Obrázek 11.2: Lo–fi prototyp domovské stránky

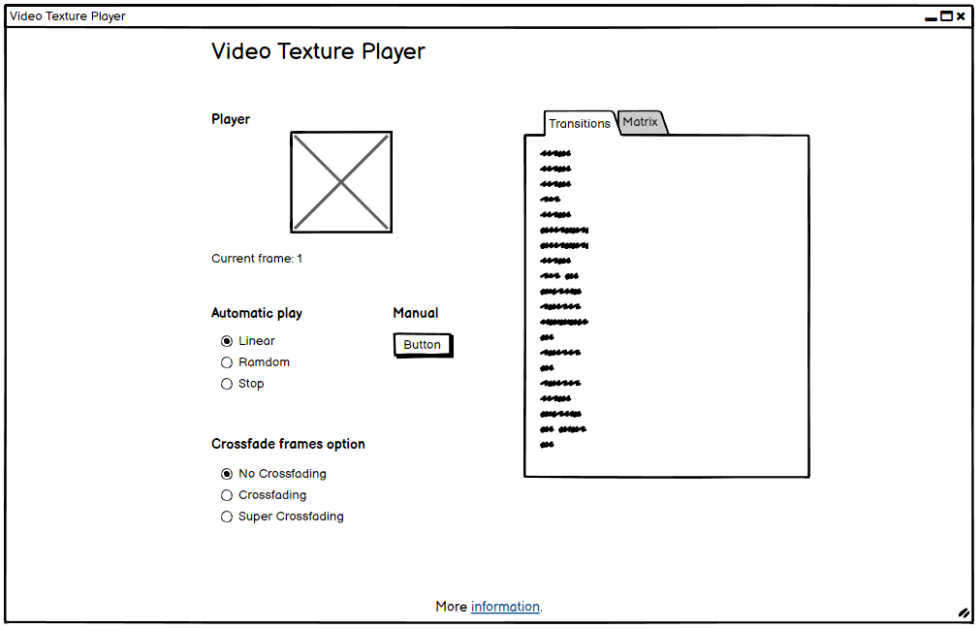
Rozhraní jednotlivých vizualizací by mělo být rovněž minimalistické. Stránky by měly obsahovat odkaz na domovskou stránku. V případě vizualizací obsahující delší popis by měl být přítomen rejstřík. Příklad takové vizualizace je na obrázku 11.3.



Obrázek 11.3: Lo–fi prototyp dílčí vizualizace

Uživatelské rozhraní vizualizace video textur

Uživatelské rozhraní vyžaduje i jedna z dílčích vizualizací. Pro přehrávání video textur je možné vybrat typ přehrávání a nastavit možnosti prolínání mezi snímky při nelineárním skoku. Vizualizace kromě přehrávání zobrazuje matici přechodů a stejnou informaci zobrazuje v podobě seznamu se zvýrazněním aktuálního snímku. Pro tento přehrávač byl rovněž navržen prototyp. Zobrazuje ho obrázek 11.4. Pro návrh obou prototypů byl použit nástroj *Balsamiq*[115].



Obrázek 11.4: Lo–fi prototyp přehrávače video textur

[115] `https://balsamiq.cloud`

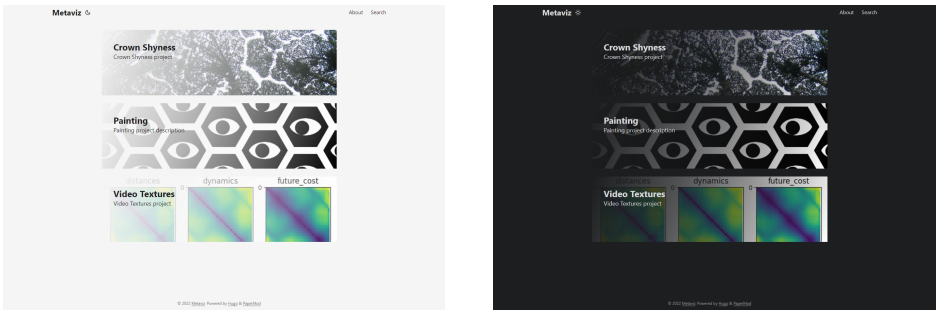
KAPITOLA 12

Implementace aplikace Metaviz

Tato kapitola stručně popisuje finální podobu samotné aplikace *Metaviz*, která vznikla na základě předchozí rešerše a analýzy. Cílem této kapitoly je kromě dokumentování postupu i poskytnutí informací pro případné rozšíření aplikace a vizualizací.

12.1 Téma PaperMod

Bylo potřeba vybrat téma pro technologii *Hugo* splňující požadavky návrhu. Pro jednoduchost, responzibilitu a podporu komunity bylo použito téma *PaperMod*[116]. Je dostupno ve třech variantách: *Home-Info Mode*, *Profile Mode* a *Regular Mode*, které bylo vybráno pro aplikaci. Téma bylo vybráno i pro jeho možnost přepnutí světlého i tmavého režimu. Tyto varianty ukazuje obrázek 12.1.



Obrázek 12.1: Barevné varianty výsledné podoby aplikace *Metaviz*

[116] `https://github.com/adityatelange/hugo-PaperMod/`

Téma bylo dále upraveno tak, aby odpovídalo požadavkům aplikace. Bylo upraveno stylizování přehledu článků tak, aby ukázkový obrázek zakrýval celý náhled. Tyto varianty byly upraveny tak, aby byly použitelné pro obě barevná schémata tématu.

KaTeX

Pro *renderování* matematických výrazů a rovnic byl použit *KaTeX*[117]. Jedná se o knihovnu napsanou v programovacím jazyce JavaScript. Renderování je založeno na programu TeX od Donalda Knutha.

Do tématu *PaperMod* je *KaTeX* přidán v souboru `math.html`, kde se nachází scripty, odkazující na knihovnu uloženou pomocí *CDN*. Tento soubor je do hlavičky HTML stránky přidán, pokud daná vizualizace obsahuje parametr `math`, jak bylo definováno v souboru `extend_head.html`.

Popis repozitáře aplikace Metaviz

Následující struktura je v mnohém identická pro stránky využívající technologii *Hugo*. Pro případy budoucího rozšíření bude tato sekce popisovat lokaci typicky využívaných souborů.

- `archetypes/` — definování atributů článku
- `assets/` — přizpůsobení CSS a JS tématu
  - `js/`
  - `css/`
    - `extended/` — složka pro vlastní styly
- `content/` — obsah webové aplikace
  - `post/` — složka pro příspěvky
  - `about.md` — příklad stránky přístupné z menu
- `layouts/` — přizpůsobení HTML šablon tématu
- `static/` — obrázky a média
- `themes/` — soubory tématu
- `config.yml` — konfigurační soubor

12.2 Netlify

Pro hosting aplikace byla použita služba *Netlify*[118], která poskytuje uživatelsky přívětivou správu webových stránek a je proto vhodná pro statické webové stránky, jakou je i aplikace *Metaviz*. Stránka se zkompiluje pomocí příkazu `hugo` a výstup je umístěn ve složce `public/`. Aplikace je dostupná na adrese `https://metaviz.netlify.app`.

[117] `https://katex.org/docs/autorender.html`
[118] `https://www.netlify.com`

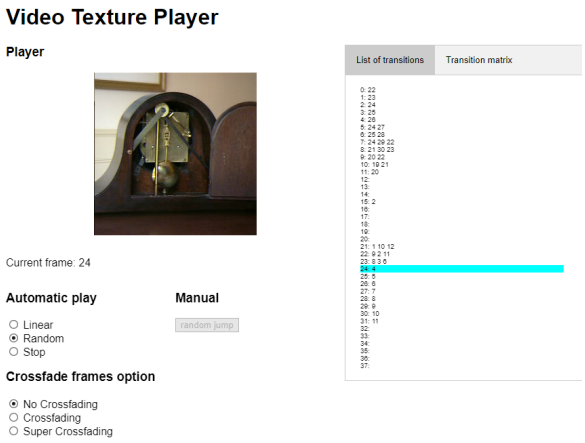
KAPITOLA 13

Testování a reflexe nad projektem

Bylo provedeno uživatelské testování, které by mělo poskytnout další názory na aplikaci a odhalit tak možné nedostatky, či nápady na zlepšení. Závěrem této kapitoly i práce je reflexe nad projektem.

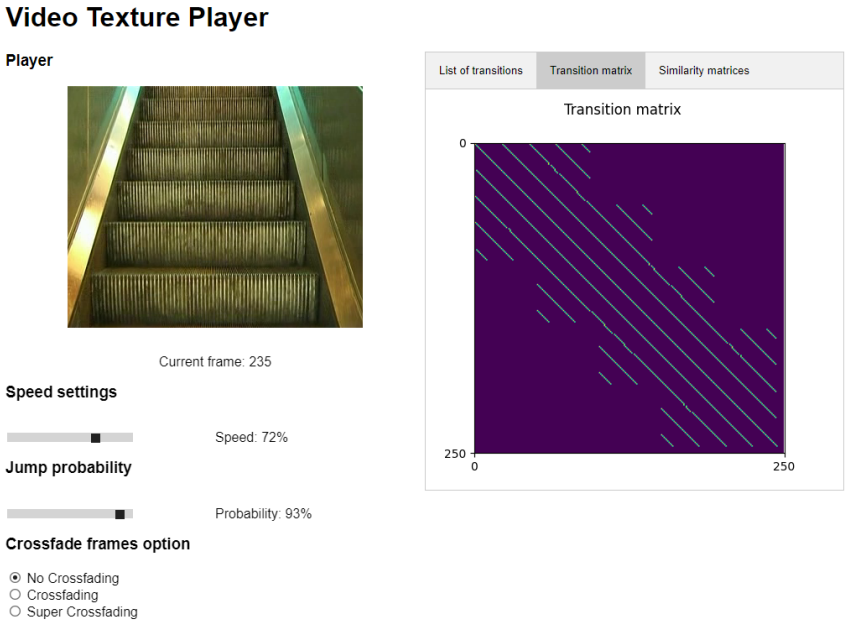
13.1 První testování

Předmětem prvního testování bylo uživatelské rozhraní přehrávače video textur. Obrázek 13.1 ukazuje původní podobu rozhraní. Původní rozhraní mělo tři možnosti automatického přehrávání z nichž jedna automatické přehrávání zastavila v umožnila manuální skok.



Obrázek 13.1: Stav přehrávače video textur před uživatelským testováním

Uživatelské testování na úzkém okruhu testerů odhalilo, že rozhraní je nepřehledné. Polovina testerů neodhalila význam nastavení automatického přehrávače. Proto bylo navrženo další rozhraní a to bylo použito při dalších běžích testování. Toto přetvořené rozhraní ukazuje 13.2. Nyní je přehrávání ovládáno pomocí dvou *sliderů*, ovládající rychlost a pravděpodobnost skoku.



Obrázek 13.2: Upravená verze přehrávače video textur po testování

13.2 Scénáře testování

Bylo provedeno testování formou dotazníku. Úvodem dostali respondenti instrukci, aby se soustředili na problémy při plnění úkolů. Poslední osmá otázka byla nepovinná. Účastníci testování pracovali samostatně a nebyli pozorováni. Vyplňovaný dotazník obsahoval následující úkoly a otázky:

1. Otevřete prosím stránku `https://metaviz.netlify.app` na libovolném zařízení a napište, jaké používáte (mobil, notebook, ...).
2. Rozklikněte vizualizaci *Crown Shyness*.
   - Zkuste na stránce nalézt Finální verzi a otevřít živou vizualizaci.
   - Jmenujte jednu barvu z barevného schématu otevřené vizualizace.
3. Vraťte se na stránku `https://metaviz.netlify.app` a rozklikněte vizualizaci *Kensuke Koike*.

- Vyzkoušejte si navigaci na stránce pomocí *Table of Contents*.

4. Vraťte se na stránku `https://metaviz.netlify.app` a rozklikněte vizualizaci *Painting*.
   - Zobrazují se Vám na stránce matematické rovnice?
5. Vraťte se na stránku `https://metaviz.netlify.app` a rozklikněte vizualizaci *Video Textures*.
   - V odstavci *Transition matrix* nalezněte informaci, co je zobrazeno v grafu na osách x a y.
6. Otevřete prosím stránku `https://vt-player.netlify.app/clock.html`.
   - Nastavte parametry „Speed“ a „Probability“ přibližně na hodnotu 70%.
   - Zobrazil se Vám přehrávač s hodinami a reagoval na nastavování parametrů?
7. Znáte podobnou aplikaci, či webovou stránku jako *Metaviz*?
8. Jaké jsou vaše dojmy z aplikace?

Výsledky testování

Respondenti byli většinou studenti, především informatiky. Respondentů bylo celkem 10 a jejich závěry pomohly odhalit některé další nedostatky aplikace a přispěly nápady na zlepšení. Ke každé otázce budou ve stručnosti shrnuty vstupy získané od respondentů.

1. Většina respondentů měla notebook či stolní počítač, dva lidé testovali aplikaci na mobilu.
2. Z obdržených barev se lze domnívat, že všichni nalezli živou verzi vizualizace, což bylo účelem otázky.
3. Bylo odhaleno, že téma používá nezvyklý kurzor (lupu) při „najetí“ myši na navigaci. Další respondent by uvítal například připnutí názvu aktuální sekce, či celé navigace z důvodu nutnosti „scrollování“ na začátek stránky.
4. Se zobrazováním matematických rovnic nebyl problém.
5. Otázka měla za cíl nalézt informaci z textu. Odpovědi značí, že rozhodnutí vybrat odstavec týkající se grafu vyprodukovala bonusové informace. Například bylo zjištěno, že začátek grafu v levém horním rohu je matoucí.

6. S úkolem nebyl problém ani na mobilních zařízeních.

7. Většina respondentů se s podobnou stránkou nesetkala. Dvěma respondentům stránka připomínala *jupyter notebook*, či webové blogy.

8. Jeden z prvních respondentů se svěřil, že by v dotazníku uvítal vysvětlení kontextu celé aplikace, jelikož si nedovedl představit, k čemu by jí v budoucnu využil. Na základě toho bylo do úvodu dotazníku přidáno „Stránka Metaviz sdružuje různou škálu vizualizací a popisuje jejich fungování.“

    - Dalším nápadem na zlepšení bylo přidání obsahu článku na levé straně pro lepší orientaci v případných větších dokumentech.
    - Bylo zjištěno, že umístění tlačítka na změnu barevného tématu je matoucí – respondent uvedl, že ho nejprve považoval za logo.
    - Jeden z respondentů si umí představit využití aplikace pro zábavu i běžné využití v případě rozumně popsaného postupu vytvoření u všech vizualizací.
    - Kontrast na hlavní stránce u popisu vizualizací nebyl dostatečný.
    - Obdržené dojmy z aplikace obsahovali pozitivní hodnocení vzhledu aplikace a obsah přínosným.
    - Jeden respondent se svěřil, že se mu tento styl kombinující vysvětlení, kód a živý výstup líbí a jako smysl stránky vnímá snahu o vysvětlení několika vizualizačních technik.

Ze získaných odpovědí je patrné, že by stránka mohla obsahovat popis vysvětlující význam stránky. Navigace po stránce je pravděpodobně pro delší texty nedostatečná. Z dotazníku vyplývá, že některé vizualizace (například video textury) by mohly být více v aplikaci popsány, aby přinášely přínos. Na základě výstupů z dotazníku byl upraven například kontrast u popisu vizualizací na úvodní stránce a lupa v navigaci. Další testování bude z důvodu obsazenosti *UX* laboratoře provedeno po dohodě s vedoucím až po odevzdání práce.

13.3 Aplikace kreativního programování

Závěrem práce mající za cíl zkoumat digitální umění, vizualizace a kreativní programování je krátká reflexe vycházející ze získaného náhledu do tématu. Cílem je představit nalezené příklady využití studovaného tématu.

Výuka programování

Nabízí se použít kreativní kódování pro výuku a samostudium programování. Vizuální výstup může začátečníkům zjednodušit pronikání do tématiky programování. To například menší náročností na abstrakci, nebo pro ně může být tvorba „umění“ zábavnější. Lze se domnívat, že nutnost instalace programů a vývojového prostředí může být dalším nepříjemným faktorem při začátcích v programování. V Kapitole 3 byly jmenovány internetové stránky, které umožňují vyzkoušet si knihovny pro kreativní programování (například *Processing*) přímo v prohlížeči. Alternativu k psaní kódu nabízí například jmenované příklady programů pro vizuální programování.

Interaktivní materiály

Nástroje pro kreativní programování mohou být bezesporu použity pro tvorbu materiálů, jak bylo demonstrováno v aplikaci *Metaviz*. Nástroje lze využít pro matematické, fyzikální, grafické a jiné animace či aplety. Interaktivnost může materiály obohatit a udělat názornější a zábavnější.[119] Kromě obohacení výukových materiálů se nabízí použití například i pro datové vizualizace.

Řemeslo, hobby

Kreativní programování poskytuje způsob, jak svojí myšlenku vyjádřit pomocí technologií. Na rozdíl od tradičních uměleckých technik (kresba, malba, hra na kytaru) digitalizace nepotřebuje aby dotyčná osoba měla potřebné vybavení (kytara, plátno, barvy, ...), ani manuální dovednost.

13.4 Pokračování projektu

Teoretická část měla za cíl provést strukturalizaci tématu. Vybrané segmenty by šly dále rozvinout a upřesnit. Možné pokračování se nabízí například způsobem zaměření se na konkrétní technologii či problém.

Vizualizace *Video textury* a *Crown Shyness* by mohly být dále rozšířeny po stránce popisné i praktické. Výsledná stránka *Metaviz* se díky otevřenému zdrojovému kódu a dostupnosti na platformě *Github* dá rozšířit například pomocí *pull requestu*.

Výslednou aplikaci by bylo možné rozšířit například rozšířením možností interakce. Například přidáváním komentářů a přímočařejší kolaborací na tvorbě obsahu, než zmíněné *pull requesty*. Další možností rozšíření je například konverze na progresivní webovou aplikaci[120], což by umožnilo například prohlížení materiálů v offline režimu. Aplikaci by mělo být díky technologii *Hugo* jednoduše lokalizovat do češtiny.

[119] Interaktivní výklad používá například služba `https://brilliant.org/`.
[120] Anglicky *Progressive Web Apps*, `PWA`.

Závěr

Podtitulem této práce je „Průvodce kreativního programování“. Z toho důvodu byla provedena snaha o rozumné strukturalizování dostupných informací a materiálů o tomto tématu. Popis těchto logických celků byl již popsán úvodem práce. Výsledkem je celý Part 1 této práce, který by mohl existovat samostatně. Tento fakt má samozřejmě svá negativa – autorka se například obává případných nepřesností a chyb, které mohly vzniknout ze snahy jmenovat i informace, které dále nejsou v práci dále využívány a nebyly proto více validovány. Nicméně práce může být pro množství a povahu nalezených referencí použita jako rozcestník.

Part 2 se věnuje vlastním vizualizacím, které jsou dostupné[121] online. Tvorba těchto vizualizací byla ovlivněna předchozí rešerší. Čerpalo se z ní například při volbě technologií pro vizualizace i výsledného webu. Podle vzoru některých rešeršovaných prací byly vizualizace vybaveny kratšími i delšími popisy jejich fungování.

Při tvorbě vizualizace *Video textur* bylo čerpáno především z disertační práce [27]. Výsledná implementace v kombinaci s dostupností[122] online by mohla posloužit jako začátek pro další rozšiřování tohoto tématu. Jelikož projekt obsahuje i zobrazení dat výstupu z algoritmu, reprezentuje tato vizualizace technické spektrum vizualizací.

Vizualizace *Crown shyness* prošla několika iteracemi. Autorka s výslednou podobou není spokojena a nevylučuje pokračování mimo rámec práce. Výsledek lze přisuzovat limitováním se na webové technologie, které sice lehce vizualizaci zpřístupní veřejnosti, ale pro tuto vizualizaci nemusí být ideální. Tato vizualizace reprezentuje spektrum uměleckých vizualizací, jelikož jejím cílem bylo především vytvořit něco estetického.

K dalším vizualizacím byl vytvořen popis jejich fungování především z matematického hlediska. Za tímto účelem vznikly i doprovodné ilustrace a animace, které jsou samy formo vizualizace. Materiál je dostupný v aplikaci v anglickém jazyce. Přínos dalších vizualizací je především popularizační. Jelikož tento typ vizualizací nebyl cílem této práce, nebyla srozumitelnost vysvětlení vizualizací dále testována.

[121] `https://metaviz.netlify.app`
[122] `https://vt-player.netlify.app`

Výsledná „aplikace“ je jednoduchá a podobá se osobním blogům, což vyplývá i z testování. Z hlediska definovaných funkčních požadavků byly splněny všechny až na požadavek tutoriálu, který je splněn pro vysokou náročnost a malou prioritu pouze částečně – tedy pro některé vizualizace. Analogicky požadavek FP05 o dostupnosti vizualizací online nebyl splněn kompletně. Testování odhalilo několik nápadů na zlepšení, týkající se především prohlížení stránky.

Literatura

[1] Feynman, R. P.: *The pleasure of finding things out: The best short works of Richard P. Feynman*. Basic Books, 2005.

[2] Read, L. E.: *I, pencil*, ročník 8. Freeman, 1958.

[3] Taylor, G. D.: *When the machine made art: the troubled history of computer art*. Bloomsbury Publishing USA, 2014.

[4] Fry, B. J.: *Computational information design*. Dizertační práce, Massachusetts Institute of Technology, 2004.

[5] Fry, B.: *Visualizing data: Exploring and explaining data with the processing environment*. "O'Reilly Media, Inc.", 2008.

[6] Monoskop. [online]. Dostupné z: `https://monoskop.org/Monoskop`

[7] Digital art timeline. [online]. Dostupné z: `https://wikieducator.org/Digital_art_timeline`

[8] Creative Coding: Perspectives & Case Studies. [online]. Dostupné z: `https://javascript.plainenglish.io/all-about-creative-coding-e79268d944e8`

[9] The Pioneers (1950-1970). [online]. Dostupné z: `http://www.vam.ac.uk/content/articles/a/computer-art-history/`

[10] Computer Art History, Characteristics of Digital Imagery. [online]. Dostupné z: `http://www.visual-arts-cork.com/computer-art.htm`

[11] Dietrich, F.: Visual Intelligence: The First Decade of Computer Art (1965-1975). *IEEE Computer Graphics and Applications*, ročník 5, 1985: s. 33–45.

[12] Shiffman, D.; Fry, S.; Marsh, Z.: *The nature of code*. D. Shiffman, 2012.

[13] Lagae, A.; Lefebvre, S.; Cook, R.; aj.: A survey of procedural noise functions. In *Computer Graphics Forum*, ročník 29, Wiley Online Library, 2010, s. 2579–2600.

[14] Vojtěch, T.: *Vizualizace 3-rozměrnỳch dat slunečního povrchu*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

[15] Reynolds, C. W.: Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, s. 25–34.

[16] Jones, J.: Characteristics of pattern formation and evolution in approximations of physarum transport networks. *Artificial Life*, ročník 16, 2010: s. 127–153, ISSN 1064-5462, doi:10.1162/artl.2010.16.2.16202. Dostupné z: https://uwe-repository.worktribe.com/output/980579

[17] Turing, A. M.: The chemical basis of morphogenesis. *Bulletin of mathematical biology*, ročník 52, č. 1, 1952: s. 153–197.

[18] Nationwide Inventory of Intangible Cultural Heritage. [online]. Dostupné z: https://www.unesco.de/en/culture-and-nature/intangible-cultural-heritage/demoscene-culture-digital-real-time-animations

[19] Demoscene. [online]. Dostupné z: https://en.wikipedia.org/wiki/Demoscene

[20] Kudra, A.: AoC | Art of Coding –The Demoscene as Intangible World Cultural Heritage. 2020.

[21] Group Overview' Future Sketches. [online]. Dostupné z: https://www.media.mit.edu/groups/future-sketches/overview/

[22] Signal Festival. [online]. Dostupné z: https://www.signalfestival.com/o-festivalu/

[23] Schödl, A.; Szeliski, R.; Salesin, D.; aj.: Video textures. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.

[24] Xiaofeng Tao : Video Textures. Dostupné z: http://cs.brown.edu/courses/cs129/results/final/taox/

[25] Sevilla-Lara, L.; Wulff, J.; Sunkavalli, K.; aj.: Smooth loops from unconstrained video. In *Computer Graphics Forum*, ročník 34, Wiley Online Library, 2015, s. 99–107.

[26] Panchal, K.: Improved Algorithm for Seamlessly Creating Infinite Loops from a Video Clip, while Preserving Variety in Textures. *arXiv preprint arXiv:2011.02579*, 2020.

[27] Struan McDonough: VideoTextures. Dostupné z: `https://github.com/VirtualVirtuoso/VideoTextures`

[28] Maxim Kazakov: CP_video_textures. Dostupné z: `https://github.com/max-kazak/CP_video_textures`

[29] sudeepgbits: video textures using python. Dostupné z: `https://github.com/sudeepgbits/video-textures-using-python`

[30] lucaslefaucheur: Video-Textures. Dostupné z: `https://github.com/lucaslefaucheur/Video-Textures`

[31] J. Mecom, A. Wang: video-textures. Dostupné z: `https://github.com/microaeris/video-texture`

[32] omnimeta: Video Texture Generator. Dostupné z: `https://github.com/omnimeta/video-texture-generator`

[33] Patrick Violette: video-texture. Dostupné z: `https://github.com/pviolette3/video-texture`

[34] Péteri, R.; Fazekas, S.; Huiskes, M. J.: DynTex : a Comprehensive Database of Dynamic Textures. *Pattern Recognition Letters*, ročník doi: 10.1016/j.patrec.2010.05.009, 2012, http://projects.cwi.nl/dyntex/.

[35] Richtr, R.: *Dynamic Texture Modeling*. PhD. thesis, České vysoké učení technické v Praze, Fakulta informačních technologií, 2018, disertační práce.

[36] Richtr, R.; Haindl, M.: Dynamic texture similarity criterion. In *2018 24th International Conference on Pattern Recognition (ICPR)*, Beijing: IEEE, 2018, s. 904–909.

[37] Richtr, R.; Haindl, M.: Dynamic texture editing. In *SCCG*, 2015, s. 133–140.

[38] Filip, J.; Haindl, M.; Chetverikov, D.: Fast Synthesis of Dynamic Colour Textures. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, ročník 4, 0-0 2006, ISSN 1051-4651, s. 25 –28, doi:10.1109/ICPR.2006.550.

[39] Hattimare, R.: Crown shyness in various tree species. *Int. J. Sci. Dev. Res*, ročník 3, 2018: s. 322–324.

PŘÍLOHA A

Seznam použitých zkratek

1D 1-dimenziální.

2D 2-dimenziální.

3D 3-dimenziální.

API Application Programming Interface.

CA celulární automat.

CSS Cascading Style Sheets.

D3 Data-Driven Documents.

GLSL OpenGL Shading Language.

GPL GNU General Public License.

GUI Graphical User Interface.

HCI Human – Computer Interaction.

HTML Hypertext Markup Language.

ISF Interactive Shader Format.

MIT Massachusetts Institute of Technology.

NFT Non-fungible token.

SIGGRAPH Special Interest Group on Computer Graphics and Interactive Techniques.

PŘÍLOHA B

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `text/` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `source/` ..... zdrojové soubory textu práce
- `metaviz` ..... repozitář aplikace Metaviz
- `visualizations` ..... repozitář vizualizací
  - `video-textures` ..... repozitář vizualizace Video textur
  - `crown-shyness` ..... repozitář vizualizace Crown Shyness
  - `others` ..... repozitář dalších vizualizací

PŘÍLOHA C

Příklad zpracovaného materiálu k vizualizaci

Následující stránky přílohy tvoří příklad zpracovaných materiálů k vizualizaci Kensuke Koike. Jedná se o verzi pro tisk. Online verzi lze nalézt na adrese `https://metaviz.netlify.app/post/kensukekoike/`.

Metaviz ☼ About Search

Kensuke Koike

Kensuke Koike inspired visualizations

▶ Table of Contents

Visualizations are inspired by brilliant japanese artist Kensuke Koike. Two visualizations were made using web technoliges. This text is containing an explanation how was those visualizations made. An image used is from *Johann (Hans) Beckmann*, the name of the painting is *Im Inntal*.

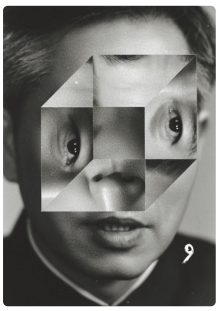
The visualizations

Circles	Penrose
Visualization *Circles* was made with p5.js.	Visualization *Penrose* was made with three.js.
	
Live version	Live version

Circles	Penrose
Online editor	Online editor

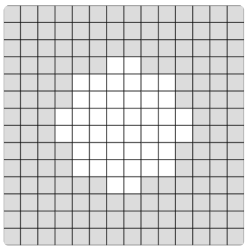
The inspiration

Kensuke Koike is famous for creative manipulations with real life printed photographs (especially black and white). His work can be found for example on Instagram. On the following pictures, the artworks that inspired the visualizations are shown.

Circles

In this example, the color of each square is defined by its distance from the center. If the $[x, y]$ coordinate is greater than given radius r, square is colored grey. This condition in math terms looks like this: $x^2 + y^2 > r^2$.



This principle is used for *cutting off* the alpha channel of an image, resulting in image drawn with given radius.

The code

The image is preloaded using built-in `preload()` function. This function is called before `setup()` function. In `setup()` function, `noLoop()` is called. This makes, the `draw()` function called just once.[1]

```
let WIDTH = 400;

function preload() {  img = loadImage('img/2.jpg'); }
function setup() {
  createCanvas(WIDTH, WIDTH);
  noLoop();
}
```

Each pixel has four values. Red, green, blue and alpha (this channel we want to manipulate). Values of pixels are stored in 1D array. Function `drawImage()` is checking each pixel and if it is greater than given radius, it sets the alpha channel to zero. Then it draws the image.

```
function drawCircle(radius) {
  for (let y = -WIDTH/2; y < WIDTH/2; y++) {
    for (let x = -WIDTH/2; x < WIDTH/2; x++) {

      // index of pixel is defined by x and y coordinates
      let index = x+WIDTH/2 + (y+WIDTH/2) * WIDTH

      // for every pixel, 4 values are stored so index needs to multiply index by 4
      index *= 4;

      // if the distance from center is greater than radius set alpha to 0
      if(pow(x,2)+pow(y,2) > pow(radius, 2)) {
```

```
        img.pixels[index+3] = 0;
      }
    }
  }

  // if we manipulate image pixels, updatePixels() has to be called
  img.updatePixels();
  image(img, -WIDTH/2, -WIDTH/2);
}
```

Then `draw()` function is called. In it, the canvas is shifted so that the origin is located in the middle. In default *p5.js* coordinate system, the $[0, 0]$ is located in the top left corner. Then the background image is drawn.

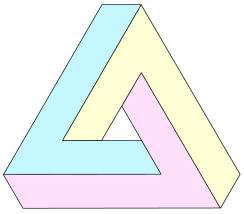
Image circles are then drawn within a for loop. For each iteration, the radius is increased by 30. In each iteration, the canvas is rotated by 12 degrees using the `rotate()` function. Degrees are converted to radians using `radians()` function.

```
function draw() {
  // loadPixels() has to be called before drawing an image
  img.loadPixels();

  translate(W/2, W/2);
  image(img, -W/2, -W/2);
  img.updatePixels();

  for(i=W-330; i>0; i-=30) {
    rotate(radians(12));
    drawCircle(i);
  }
}
```

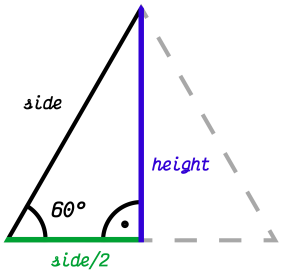
The result and code is available at p5 editor. Another visualization is trying to take a diffrent approach.

Penrose triangle

There is not only one way to determine the coordinates of the Penrose triangle points. In this section will be explained the method used in the visualization.



The idea here is to divide the penrose triangle into three shapes (yellow, cyan, pink) and get their coordinates.



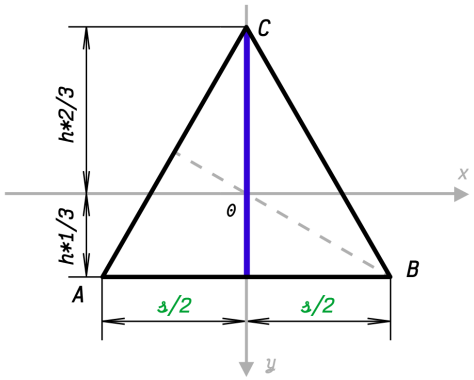
First, we define the height of the triangle.[2] The sum of all angles in any thiangle is 180 degrees. This triangle is equilateral and therefore all the angles has $180/3 = 60$ degrees. Using goniometric functions we express the height using a side of the triangle:

$$tan(60^\circ) = \frac{h}{s/2}$$

$$h = \frac{s}{2} * tan(60^\circ)$$

$$h = s * \frac{\sqrt{3}}{2}.$$

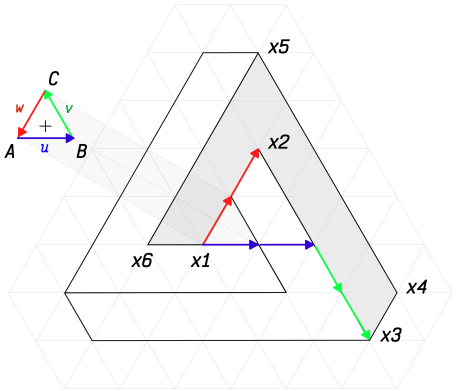
For the inner triangle of the Penrose triangle, points A, B, C and vectors u, v, w were defined[3] as:



$$(A, B, C) = ([-\frac{s}{2}, h * \frac{1}{3}], [\frac{s}{2}, h * \frac{1}{3}], [0, -h * \frac{2}{3}])$$

$$(u, v, w) = ([B - A], [C - B], [A - C]).$$

Then, the coordinates for grayed shape (x_1, x_2, x_3, x_4, x_5, x_6) were expressed as a combination of vectors u, v, w.

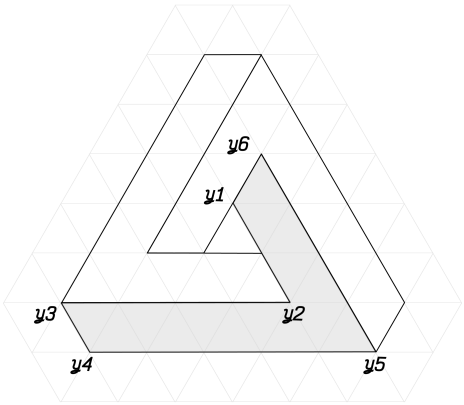


For every x_i coordinate, the origin was shifted to the point A.

```
x1 = A
x2 = A +   0  * u +   0  * v +  (-2) * w
x3 = A +   2  * u + (-2) * v +    0  * w
x4 = A +   3  * u + (-1) * v +    0  * w
```

```
x5 = A +  0 * u +  1 * v + (-3) * w
x6 = A + (-1) * u +  0 * v +  0 * w
```

This is just an example of linear combination, that is generating this shape. There are more ways this points could be obtained.



The next step is to generate the remaining two shapes. Those can be done using the same linear combination as before, but using the vectors u, v, w in different order and thus performing a rotation. An example of another shape:

```
y1 = C
y2 = C +  0 * w +  0 * u + (-2) * v
y3 = C +  2 * w + (-2) * u +  0 * v
y4 = C +  3 * w + (-1) * u +  0 * v
y5 = C +  0 * w +  1 * u + (-3) * v
y6 = C + (-1) * w +  0 * u +  0 * v
```

The coordinates for the third shape can be generated analogouslly. Thanks to this process, the coordinates of those three shapes forming Penrose triangle can be generated and parametrized by the side of the inner triangle.

Three.js

This code explains the *Three.js* implementation of the visualization. Coordinates are generated as was described above. Following code snippets are showing just specific part of the code.

```
// function creates one shape from given coordinates and adds a texture to it
function penrosePart(coordinates, image_path, scene) {
  // load texture
  let texture = new THREE.TextureLoader().load( image_path );

  // enable mirror wrapping
  texture.wrapS = THREE.MirroredRepeatWrapping;
  texture.wrapT = THREE.MirroredRepeatWrapping;

  // create texture material
  let textureMaterial = new THREE.MeshBasicMaterial( { map: texture } );

  // create geometry from coordinates
  let geomShape = new THREE.ShapeBufferGeometry( new THREE.Shape( coordinates ) );

  // create mesh and add to scene
  let mesh = new THREE.Mesh(geometry, material);
  scene.add(mesh);
}
```

The visualization is reacting to mouse movement (and touch event). Mouse (or touch) coordinates are used to offset the shape texture.

```
// edit texture offset on mouse move
function onDocumentMouseMove(event) {
  event.preventDefault();
  mouse.x = (event.clientX / window.innerWidth ) * 2;
  mouse.y = (event.clientY / window.innerHeight) * 2;
  shapeA.material.map.offset.x = positionX + Math.cos(mouse.x) * 0.3;
    shapeB.material.map.offset.y = positionY + Math.cos(mouse.y) * 0.3;
    shapeC.material.map.offset.x = positionX + Math.cos(mouse.y) * 0.3;
}
```

The resulting visualization and online code editor can be found here.

1. In case of this visualization, it is important to call `noLoop()`, because it is computationally expensive. ↩
2. Font used in visual materials is Victor Mono. ↩
3. This definition is using p5.js coordinate system, which has positive values in the right down corner. ↩