VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BLENDER ADD-ON PRO VIZUALIZACI DAT

DATA VISUALIZATION BLENDER ADD-ON

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ZDENĚK DOLEŽAL**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ CHLUBNA**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Doležal Zdeněk**
Program: Informační technologie
Název: **Blender add-on pro vizualizaci dat**
Data Visualization Blender Add-On
Kategorie: Počítačová grafika

Zadání:

1. Naučte se základy práce s 3D modelovacím programem Blender (verze 2.8 a výše)
2. Seznamte se se skriptovacím Blender Python API
3. Nastudujte možné formy vizualizace dat, zejména běžně používané druhy grafů
4. Vyberte vhodné vizualizace a navrhněte uživatelské rozhraní pro výsledný add-on
5. Add-on implementujte a demonstrujte jeho použití na různých typech dat
6. Zdokumentujte a zveřejněte výsledný add-on pro použití dalšími uživateli
7. Vytvořte video reprezentující výsledky vaší práce

Literatura:

- Dle zadání vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4, experimenty vedoucí k bodu 5.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Chlubna Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 1. listopadu 2019

Abstrakt

S rostoucím zájmem o vizualizační techniky se nabízí možnost rozšíření již existujícího software pro 3D počítačovou grafiku o tvorbu vizualizací. Práce se zabývá vizualizací dat obecně, výběrem vhodných vizualizací, návrhem, vytvořením a zveřejněním add-onu pro vizualizaci dat do open-source programu Blender. Blender je díky této práci rozšířen pomocí Python API o rozhraní pro načtení dat a generování běžné používaných grafů. Publikovaná implementace byla zhodnocena uživatelským průzkumem a je dostupná open-source.

Abstract

With growing interest in visualisation techniques, possibility to extend existing 3D computer graphics software by visualisation tools arises. Thesis deals with theory of data visualisation in general, selection of suitable visualisations, design, creation and publication of an data visualisation add-on for the open-source software Blender. Thanks to this work Blender is extended using its Python API by an interface used for data loading and commonly used charts generation. The published implementation was evaluated by user survey and it is available open-source.

Klíčová slova

Vizualizace dat, 3D vizualizace, Grafy, Animované grafy, Počítačová grafika, Blender, Add-on

Keywords

Data visualisation, 3D visusalisations, Charts, Animated charts, Computer graphics, Blender, Add-on

Citace

DOLEŽAL, Zdeněk. *Blender add-on pro vizualizaci dat.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Chlubna

Blender add-on pro vizualizaci dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Chlubny. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Zdeněk Doležal
28. května 2020

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Tomáši Chlubnovi za ochotu, rychlou odezvu a pomoc při řešení neočekávaných problémů při tvorbě práce. Díky patří také všem přátelům a uživatelům, kteří add-on vyzkoušeli a poskytli drahocennou zpětnou vazbu.

Obsah

1 Úvod **2**

2 Teorie **3**
2.1 Data . . . 3
2.2 Vizualizace dat . . . 5
2.3 Vizualizační metody . . . 8
2.4 Typy grafů . . . 11
2.5 Blender . . . 15
2.6 Existující řešení . . . 17

3 Návrh **19**
3.1 Uživatelské rozhraní . . . 19
3.2 Import dat . . . 22
3.3 Vytvoření grafů . . . 22
3.4 Komponenty grafu . . . 27
3.5 Barvy . . . 28
3.6 Animace . . . 30

4 Realizace **31**
4.1 Využité části API . . . 31
4.2 Správa dat . . . 33
4.3 Vytvoření grafů . . . 34
4.4 Barvy . . . 37
4.5 Publikování add-onu . . . 38

5 Uživatelský průzkum **39**
5.1 Dotazník . . . 39
5.2 Výsledky . . . 40
5.3 Jiná zpětná vazba . . . 41

6 Závěr **42**

Literatura **43**

A Obsah paměťového média **45**

B Uživatelské rozhraní **46**

C Rendery vizualizací vytvořených pomocí add-onu **47**

Kapitola 1

Úvod

Zdrojů, ze kterých jsou získávána data neustále přibývá. Součástí novodobého lidského života je práce s technologiemi a s daty, ať už se jedná o zobrazování elektronické pošty, firemních zisků za kvartál nebo využívání nejmodernějších aplikací pro silniční navigaci. Pro lidské vnímání jsou data ve strojové podobě v mnoha odlišných formátech nesrozumitelná, proto důležitost vytvoření jejich abstraktní prezentace v moderní, interaktivní, čitelné, vizuálně a uživatelsky atraktivní formě čím dál více stoupá.

Cílem práce je poskytnout uživatelům nástroj pro vizualizaci dat v 3D prostoru v prostředí open-source programu Blender. Nástroj podporuje několik základních typů vizualizací a umožňuje přizpůsobit jejich vzhled obdobně jako již jiná existující řešení. Add-on vytváří vizualizace ve scéně z objektů, se kterými je následně možné manipulovat pomocí standardních operací nad 3D scénou, což ve výsledku poskytne uživateli možnost si vizualizaci přizpůsobit do posledního detailu. Natáčení pohledu ve 3D scéně umožní vytvořené vizualizace pozorovat z mnoha úhlů. Cílem je výsledný add-on poskytnout uživatelům, otestovat, upravit a vyladit programové řešení na základě zpětné vazby.

V kapitole 2 jsou rozebrány datové formáty, vizualizace dat a jednotlivé typy grafů. Následně v kapitole 3 je popsán postup při návrhu řešení. Zajímavé pasáže implementace, integrace vizualizace dat do Blenderu a publikování nástroje jsou probrány v kapitole 4. Získaná zpětná vazba a zkušenost uživatelů s add-onem je popsána v kapitole 5. Shrnutí a zhodnocení, zda a jakým způsobem bylo dosaženo stanovených cílů se nachází v kapitole 6.

Kapitola 2

Teorie

V této kapitole jsou uvedeny minimální nutné informace k porozumění problematice práce. Jedná se o teoretické výňatky týkající se dat, vizualizace dat, vizualizačních metod, Blenderu a jeho Python API.

2.1 Data

Data jsou definována jako hodnota schopná přenosu, uchování, interpretace či zpracování [3]. Data jsou základním stavebním kamenem pro vizualizaci. Jejich systematickým sběrem a konzistentním uložením s následným předzpracováním lze značně ovlivnit míru správné interpretace pozorovatelem po jejich vizualizaci.

2.1.1 Reprezentace dat

Data se běžně vyskytují ve dvou formách: data v tabulkách (datové sady) nebo jako izolované hodnoty [8]. Izolované hodnoty představují potencionálně využitelná fakta nebo data, která nejsou shromážděna na konkrétním místě. Pokud se jedná o data, jenž neprošla žádným zpracováním, hovoříme o datech nezpracovaných, či surových (raw data).

Tabelovaný způsob reprezentace představuje rozdělení každého naměřeného vzorku do kategorií (sloupců), kde každý sloupec znázorňuje jednu veličinu. První řádek obvykle popisuje jednotlivé kategorie, jedná se o záhlaví tabulky. Jednotlivé záznamy jsou reprezentovány na konkrétním řádku. Příkladem je tabulka 2.1 s daty o znečištění ovzduší v Soulu.

Measurement date	Station code	Item code	Average value	Instrument status
2017-01-01	101	1	0.004	0
2017-01-01	101	3	0.059	0
2017-01-01	101	5	1.2	0
2017-02-07	108	1	0.006	0
2017-02-07	108	3	0.046	0
2017-02-07	108	5	0.7	0

Tabulka 2.1: Příklad datové sady – Znečištění ovzduší v Soulu (výňatek)[1]
Na každém řádku se kromě informací o stanici a datu nachází specifický kód položky a stav přístroje.

[1] Převzato a upraveno z: https://www.kaggle.com/bappekim/air-pollution-in-seoul/data

2.1.2 Typy dat

Data jsou dělena do dvou fundamentálních skupin: kategorická a numerická. Kategorická též kvalitativní data jsou typem dat, která mohou být za pomoci jmen nebo štítků (labelů) zařazena do skupin nebo kategorií. Dále se dělí na nominální (nelze je uspořádat), ordinální (uspořadatelná) a binární. Numerická data neboli kvantitativní data jsou na druhou stranu typem dat, která lze reprezentovat číselnými symboly [12].

Numerická data lze dále rozlišit na spojitá a diskrétní data [8]. Spojitá data mohou nabývat všech reálných hodnot v rámci konečného nebo nekonečného intervalu. Diskrétní data mohou nabývat v daném intervalu pouze izolovaných číselných hodnot, zpravidla se jedná o přirozená čísla [13].

Numerická data mohou být rozptýlená (scattered) nebo uspořádaná do mřížky (gridded). Rozptýlená data se skládají z množiny bodů a jím odpovídajícím hodnot, tyto body nemají žádnou strukturu nebo uspořádání mezi jejich relativními lokacemi. Naopak data uspořádaná do mřížky se skládají z množiny bodů a jim odpovídajících hodnot, kde body mají strukturu a jejich pozice jsou vůči sobě uspořádané a zarovnané s konstantním krokem [10]. K uspořádání rozptýlených dat do mřížky, či odhadnutí hodnoty funkce mimo mřížku je využívána interpolace. Interpolace je soubor metod umožňující nalezení neznámé hodnoty mezi hodnotami známými.

2.1.3 Datové formáty

Jedním z nejvyužívanějších formátů pro uložení rozsáhlých datových sad je formát **comma separated values** neboli hodnoty oddělené čárkami (dále CSV). Řádky představují jednotlivé záznamy, obdobně jako v tabelované reprezentaci. Řádek obsahuje jednotlivé položky separované oddělovačem, který je specifický pro soubor. Oddělovač je obvykle čárka, někdy se využívá oddělovačů jiných, např. se jedná o tečku, či bílé znaky. Pokud je nutné v položce uvést stejný znak jako zvolený oddělovač pro soubor, tak se položka uzavírá do uvozovek. CSV formát je značně využíván hlavně pro jeho jednoduchost, strojovou i lidskou čitelnost a jeho načítání podporuje většina software. Příkladem je výpis 2.1, na kterém je uvedena tabulka z předchozí strany reprezentována ve formátu CSV.

```
Date,Station code,Item code,Average value,Instrument status
2017-01-01,101,1,0.004,0
2017-01-01,101,3,0.059,0
2017-01-01,101,5,1.2,0
2017-02-07,108,1,0.006,0
2017-02-07,108,3,0.046,0
2017-02-07,108,5,0.7,0
```

Výpis 2.1: Datová sada jako CSV

Datové sady pro vizualizaci mohou být také extrahovány z databáze a nebo získávány z webového aplikačního rozhraní. V tomto případě se využívá serializačních formátů, jenž přináší nutnost zavedení metadat (data, která vyjadřují informaci o jiných datech) a složitější analýzu. Pro představu je na výpisu 2.2 uveden jeden záznam z předchozí datové sady ve formátu **JSON**. Na výpisu 2.3 je uveden příklad formátu **XML**. V obou případech se oproti CSV v záznamu kromě samotných hodnot vyskytují hojně metadata, což může mít vliv na případnou redundanci položek nebo na rychlost vyhledávání.

```
[
  {
    "Date": 2017-01-01, "Station code": 101, "Item code": 1,
    "Average value": 0.004, "Instrument status": 0
  },
  ...
]
```

Výpis 2.2: Záznam ve formátu JSON

```
<Data>
    <Entry>
        <Date>2017-01-01</Date>
        <StationCode>101</StationCode>
        <ItemCode>1</ItemCode>
        <AverageValue>0.004</AverageValue>
        <InstrumentStatus>0</InstrumentStatus>
    </Entry>
    ...
</Data>
```

Výpis 2.3: Záznam ve formátu XML

2.2 Vizualizace dat

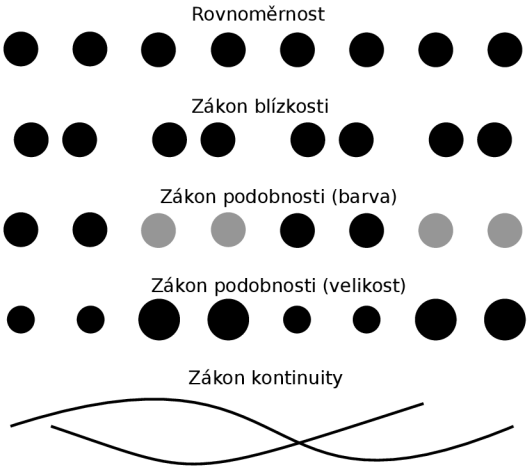
Vizualizace dat je proces zkoumání dat a informací po jejich převedení do grafické podoby. Jejím cílem je pochopení zkoumaných jevů a vniknutí do problému [15]. Jedná se o grafickou reprezentaci informací a dat. Pomocí elementů jako jsou tabulky, grafy a mapy poskytují uživateli vizualizační nástroje dostupný způsob, jak rozlišovat a pochopit trendy, odlehlé hodnoty nebo vzory v datech [16]. Jedná se o podobor disciplíny zvané analýza dat. Analýza dat obecně spočívá ve zpracování a transformaci dat do takové formy, která z dat odhaluje další informace a přináší nové znalosti pro podporu rozhodování.

Forem vizualizace existuje mnoho, přičemž každá má své klady a zápory, případně specifické uplatnění. Vizualizace mohou být přesnější a více odhalující alternativou statistickému zobrazování. Správně zvolená a aplikovaná vizualizace dělá složitá a objemná data srozumitelnými a čitelnými. Jedná se o abstraktní prostředek, který přibližuje informace uživateli a umožňuje mu zkoumat závislosti a podobnosti mezi vzorky dat.

Využití vizualizace je mnoho. Vědecké vizualizace zahrnují matematické grafy, interaktivní zobrazení lidského těla, signály nebo např. simulace sluneční soustavy. Hojné využití vizualizace dat se také nachází v marketingové sféře, kde primárně slouží pro prezentační účely, jelikož je vizuálně atraktivní, působí profesionálně a dokáže předat kvanta informací. Management je také obor, ve kterém je vizualizace dat podstatná. Nasbírané statistiky ohledně nejrůznějších parametrů, které mají vliv na chod podniku, bývají prezentovány ve formě grafů, tabulek aj., jenž následně slouží pro rozhodování a plánování dalších kroků. Vizualizovány jsou také data na mapách např. pro poskytnutí dynamických informací ze získaných dat jako je aktuální hustota provozu, koncentrace pevných částic v ovzduší nebo pro zobrazení statických informací ohledně populace, zaměstnanosti atd.

2.2.1 Psychologické efekty vizualizace

Při vizualizaci dat, hraje důležitou roli lidské vnímání, jelikož lidské vnímání je právě to, co je vizualizacemi vybuzováno k analýze problému. U tvoření vizualizace a při výběru vizualizační metody je vhodné myslet na to, jaká sémantická informace má být uživateli předána a jakým způsobem může zvolení vizualizační metody ovlivnit jeho vnímání, případně jeho postřeh. Příkladem mohou být Gestalt zákony (součást tvarové psychologie [9]), popisující tendence, ke kterým se lidské vnímání schyluje při pozorování tvarů, vidno na obrázku 2.1. Pochopení a aplikace Gestalt zákonů při tvorbě vizualizace může zvýšit čtivost a prezentovatelnost informace a zároveň zaručit správnost interpretace.



Obrázek 2.1: Gestalt zákony uplatnitelné při vizualizaci dat
Na obrázku se nachází vybrané gestalt zákony aplikovatelné na vzhled vizualizace. Velikost objektu znázorňující určitou hodnotu by měla být stejná jako velikost jiného objektu znázorňující hodnotu stejnou (zákon podobnosti velikosti). Stejné proměnné ve více kategoriích je vhodné znázornit v každé kategorii stejnou barvou (zákon podobnosti barvy) nebo je seskupit ke každé proměnné v určitém pořadí (zákon blízkosti). Objekty, které na sobě nezávisí – nejsou rozděleny do kategorií – by měly být zobrazeny rovnoměrně. Zákon kontinuity vyjadřuje lidskou tendenci organizovat objekty dle souvislých linií a kontur.

Nedílnou součásti vizualizace jsou barvy. Použitím barev lze zjednodušit komplexní grafy. Udržet a porovnávat velké množství stejně vyobrazených hodnot v mysli je značně náročné. Použitím kontrastních barev pro rozlišení jednotlivých souborů dat nebo gradientu pro reprezentaci měřítka je uživateli umožněno se zaměřit na důležité vlastnosti. Každá barva vyvolává v člověku jiné specifické pocity. Modrá vyvolává pocit chladu, klidu, červená pocity tepla, agrese [4]. Vhodné zvolení odstínu barvy může ovlivňovat dopad na celkové vnímání vizualizace a měnit její důležitost. Barvou lze přidávat hloubku do vizualizace a využít ji například při prezentaci. Lze zvýraznit konkrétní hodnotu, která je něčím zajímavá a poukázat na ni. Barvy by měly být používány konzistentně, v širokém a rozlišitelném rozsahu poskytujícím uživateli dostatečný kontrast (případně jas) pro jednoznačné rozlišení prezentovaných informací.

2.2.2 Vizualizace dat v 3D prostoru

Lidská mysl je zvyklá na vnímání trojrozměrného prostoru z reálného světa. Zavedením 3. rozměru do vizualizace lze docílit zobrazení objemů, zasadit data do prostorů, jenž jim dodávají hlubší kontext, a také dosáhnout zesílení kognitivního vnímání závislostí. Při správném provedení vizualizace ve 3D lze zvýšit čtivost informací poskytovaných vizualizací a ulehčit následnou analýzu dat.

Vizualizace ve 3D nejlépe s pohledným zbarvením je pro pozorovatele přitažlivá a příjemná na pohled, působí reprezentativně a přitom nemusí být ani výstižná. Této vlastnosti se využívá například pro marketingové účely[2], kdy pro laika vypadají více-dimenzionální grafy profesionálněji, přestože mnohdy nevyjádří více než obyčejný graf. Ve vědě přidání třetí dimenze umožní vizualizovat reálné procesy, zasadit grafy do prostředí, jenž jim rozšiřují kontext (např. sloupcový graf na mapě), komplexní simulace kapalin nebo větru, další závislosti naměřených veličin, plochy představující matematické funkce, objemy a jiné.

Ve specifických případech může zavedení třetího rozměru zvýšit chybovost získávání informací z vizualizace. Příkladem může být zmatení perspektivou u výsečového grafu, pozorovaného pod takovým úhlem, jenž zkresluje velikosti jednotlivých řezů (zadní řez se jeví menší než je jeho skutečná velikost). Graf se jeví pohlednější, což pro většinu běžných nebo marketingových účelů je výhodné, ale pro prezentaci technických dat nikoliv. Špatným exportováním, tedy vznikne matoucí vizualizace, degradující schopnost pozorovatele číst a interpretovat data správným způsobem. Vliv zkreslení se dá omezit například ortografickou projekcí (pro vnímání nepřirozenou) ulehčující odečítání hodnot a nebo poskytnout pozorovateli možnost se v prostoru pohybovat, díky čemuž lze na vizualizaci pohlížet z více úhlu a následně informace interpretovat na základě více pohledů.

2.2.3 Animace

Zavedením animace do grafu umožníme pozorovateli vysledovat v grafu trend vývoje za uplynulou časovou jednotku. Přidáním animace v čase do vizualizace, která popisuje rozdělení do kategorií, je možné pozorovat přeskupení, nárůst či pokles hodnot v jednotlivých kategoriích. Vizualizace se stane dynamičtějším a objem prezentovaných dat se zvýší, čímž se zvedne i počet informací, které pozorovatel zpracuje, což poskytne prostor pro nalezení dalších asociací mezi daty. Pohyb po časové ose přidává vizualizaci možnost interakce. Animace tvoří vizualizaci pohlednou, zapamatovatelnou a umožňuje klást důraz na jiné závislosti, než vizualizace statická. Příkladem velmi povedené animace vizualizace je služba windy[3]. Jedná se o zobrazení rozmanitých dat o počasí, na kterých je díky animaci možné pozorovat vývoj v čase.

2.2.4 Interaktivita

Běžní uživatelé vyhledávají vizualizaci za účelem nabytí nových znalostí, či ověření zjištěných skutečností. Přidáním interaktivity do zobrazení je více angažována uživatelova pozornost a nabízí se příležitost objevovat. Možnost prozkoumávání vizualizace do hloubky skýtá prostor pro zahrnutí detailů ohledně konkrétních témat, či dávkování informace po částech k zjednodušení jejího pochopení (zanořování neboli drill-down). Základním zástupcem interaktivity je přibližování nebo natáčení pohledu na vizualizaci, které umožňuje vizualizaci prohlédnout z více stran a pozorovat více-dimenzionální data z různých perspektiv.

[2] Apple 2008, https://paragraft.wordpress.com/2008/06/03/the-chart-junk-of-steve-jobs/
[3] Dostupné na: www.windy.com

Dalšími způsoby je proklikávání pro zobrazení detailu nebo přeskupení na základě změny parametrů.

2.3 Vizualizační metody

Vizualizační metoda je způsob vytvoření vizualizace, definuje prostor, nad kterým je vizualizace prováděna a také systém, jakým jsou prezentovány uživateli data. Správný grafický displej, čili přístroj zobrazující data ve formě srozumitelné člověkem, by měl splňovat dle Edwarda Tufteho [17] určité principy a vlastnosti aplikovatelné i na vizualizační metody:

- Zobrazovat data - základní účel jakékoliv vizualizace
- Přimět pozorovatele, aby přemýšlel o zobrazované látce, nikoliv o metodologii, grafickému designu, způsobu grafické produkce, či čemkoliv jiném.
- Vyhnout se zkreslení toho, co data mají vyjadřovat
- Prezentovat velký počet hodnot na malém prostoru
- Učinit velké soubory dat jednotnými
- Nabádat lidské oko k porovnání různých hodnot dat
- Odhalovat data na různých úrovních detailu, od širokého spektra po jemnou strukturu
- Sloužit rozumnému a jednoznačnému účelu: popis, průzkum, tabelace nebo dekorace
- Být úzce propojen se statistickým a verbálním popisem souboru dat

Principy poukazují na zmiňované psychologické efekty vizualizací a jsou s nimi úzce spjaty. Vizualizační metoda by měla poskytovat prostředek pro vytvoření jednoduchého, přehledného a efektivního způsobu pro interpretaci velkého množství hodnot na malém prostoru bez zbytečných rozptylujících vlivů.

Zvolení vizualizační metody závisí na jejím účelu a oblasti použití. Jelikož je sféra užití pro vizualizaci dat široká, zmíněny jsou pouze hlavní kategorie vizualizačních metod. Graf je popsán obecně a v nadcházející kapitole jsou popsány konkrétní typy grafů vhodné pro 3D.

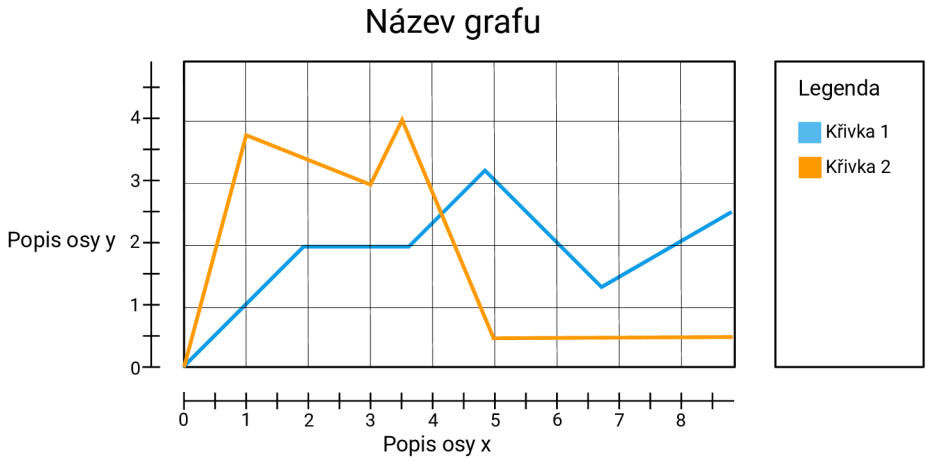
2.3.1 Grafy

Grafy neboli diagramy jsou obvykle zobrazeny v kartézské soustavě souřadnic a jejich hlavním účelem je zobrazení dat pomocí tvarů, přímek či bodů, v takové formě, kdy z grafu lze vyčíst hodnotu určitého vzorku a porovnat i s hodnotou vzorku jiného.

Pro běžné potřeby obvykle stačí dvě dimenze tvořeny osou x a osou y popisující souřadnice na ploše. Pomocí objektu, jenž znázorňuje hodnoty v jednotlivých dimenzích, je možno číst jaká hodnota na ose y náleží hodnotě na ose x nebo vyčíst do jaké kategorie jednotlivé objekty patří a tyto vlastnosti porovnávat s hodnotami jiného vzorku. Pro zobrazení více asociací a většího počtu dat lze graf rozšířit do třetí dimenze přidáním osy z (hloubky) nebo rozlišit jednotlivé hodnoty do kategorií pomocí barev, popisů či výměnou bodů za primitivní objekty. Kromě samotného objektu reprezentující hodnoty dat jsou nedílnou součástí také komponenty dodávající grafu kontext a při správném použití zlepšují čitelnost a intuitivnost.

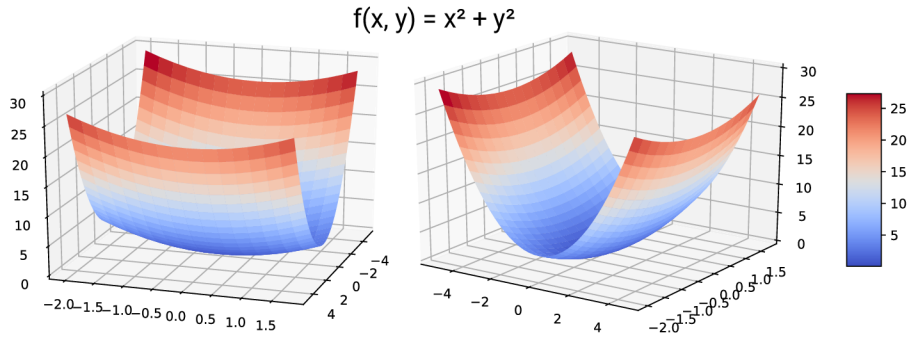
2.3.2 Komponenty grafu

Typický graf lze rozložit na několik důležitých komponent, kde každá má svoji určitou roli pro předání informace uživateli.



Obrázek 2.2: Dekompozice základního grafu

Na obrázku 2.2 lze vidět dekompozici plochy grafu na jednotlivé komponenty [5]. K uvedení uživatele do problému slouží **název grafu**, který by měl být výstižný a popisovat co na grafu uživatel nalezne, čeho se graf týká. Středem celé plochy grafu je **posloupnost dat** zobrazena způsobem, který vyjadřuje vztah veličin ve dvou nebo více dimenzích. Graf může obsahovat jednu či více posloupností. Data mohou být reprezentována křivkami, sloupci, plochou a jinými tvary. Měřítko datům dodávají **osy**. Osa znázorňuje rozsah a hodnoty numerické nebo kategorické veličiny. Osa obsahuje stupnici reprezentující vymezené hodnoty z rozsahu spojité či diskrétní veličiny nebo kategorie. Stupnice může být lineární nebo v elektrotechnice často využívaná logaritmická. Na stupnici se nachází značky reprezentující určité hodnoty z rozsahu osy. V případě tří dimenzí graf obsahuje osu z popisující další dimenzi (hloubku), vertikální osa y bývá následně umísťována dle pohledu na graf tak, aby nezastiňovala data viz. obrázek 2.3 (na druhém grafu by osa bez přizpůsobení překrývala vyobrazený povrch). Každá osa by měla obsahovat **popis osy** specifikující jakou veličinu osa představuje a u numerických hodnot jednotku veličiny. K zjednodušení nalezení hodnoty na ose y náležící hodnotě na ose x slouží volitelně **mřížka**. Mřížka je tvořena vertikálními a horizontálními přímkami na stejných pozicích jako kroky stupnice, což poskytuje uživateli vodítko pro nalezení žádané hodnoty. V trojrozměrných grafech je mřížka znázorněna na zadních stěnách kvádru ohraničujícího objem grafu. Pokud je zobrazeno více posloupností dat nebo je využito jejich rozlišení dle stylů (barva, styl čáry aj.), tak je pro zvýšení čitelnosti využívána **legenda**. Legenda přiřazuje danému stylu posloupnosti název. Pokud barva vyjadřuje v grafu také hodnotu, obvykle se uvádí gradient barev v legendě (někdy nazýváno barevná mapa) také a je uvedeno jaká barva náleží jaké hodnotě, představeno vpravo na obrázku 2.3.

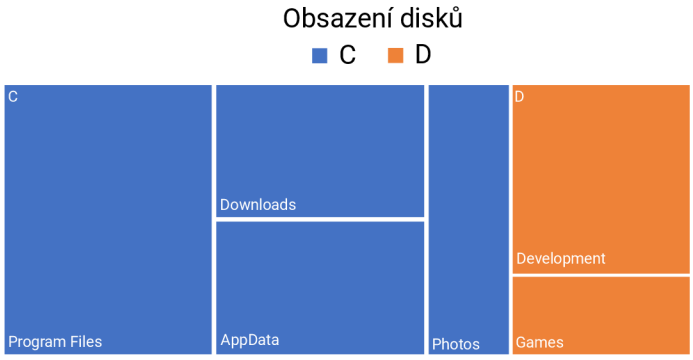


Obrázek 2.3: Posunutí osy dle natočení pohledu na graf a barevná mapa
Na obrázku je znázorněn posun osy a natočení popisů dle úhlu pohledu na graf. Vertikální osa je přemístěna aby nezakrývala vyobrazenou plochu. V pravé části obrázku se nachází barevná mapa, jenž znázorňuje barvy odpovídající hodnotám.

2.3.3 Informační vizualizace

Snahou informační vizualizace je uživateli zprostředkovat interní strukturu dat, kauzální vztahy a závislosti v nich [7]. Do této kategorie spadají vizualizace pracovních postupů, časových os a nebo informační grafiky popisující například historické události.

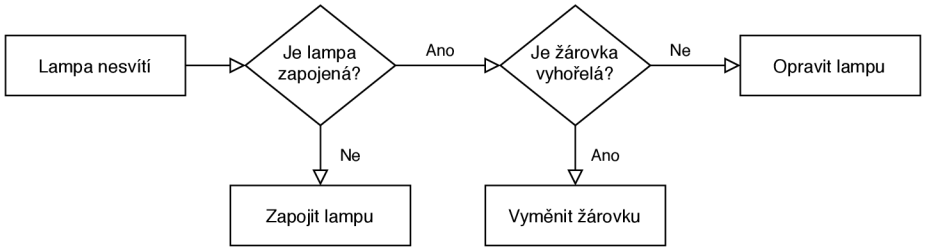
Představitelem informační vizualizace je **stromová mapa**. Stromová mapa poskytuje hierarchický pohled na data, což umožňuje jednodušší rozpoznání vzorů a výjimečných kategorií. Příklad stromové mapy se nachází na obrázku 2.4.



Obrázek 2.4: Příklad stromové mapy
Stromová mapa reprezentující hierarchický pohled na rozložení obsahů disků. Jednotlivé větve stromu (zde složky) jsou reprezentovány čtverci a jednotlivé pod-větve čtverci menšími. Jednotlivé kategorie (disky) jsou znázorněny odlišnou barvou.

Speciálním typem informační vizualizace je **geografická mapa**. Jedná se o intuitivní a efektivní zobrazovací prostředek pro vizualizaci geografických dat (data souvisejících se státy, regiony či městy a nebo lokalitami). Typickým příkladem geografické vizualizace je předpověď počasí. Teploty, tlak nebo pohyb srážkových oblak jsou zobrazeny na mapě, která informaci o počasí dodává polohu, pojem o rozpoložení, prezentuje pohyb aj. Toto propojení umožňuje pozorovat a interpretovat souvislosti, které by nebylo možné jednoduše pozorovat bez pokročilých vizualizací. Dalším příkladem je mapa městské hromadné dopravy.

Mezi informační vizualizace patří také **vývojový diagram** (flowchart). Jednoduchý příklad vývojového diagramu je znázorněn na obrázku 2.5. V praxi jsou využívány pro popis pracovních postupů, algoritmů nebo pro vizualizaci řešení problémů.



Obrázek 2.5: Vývojový diagram řešení triviálního problému
Vývojový diagram je symbolická reprezentace kroků zahrnutých v nějakém procesu. Různé druhy kroků (rozhodovací, sekvenční) jsou znázorněny různými symboly, vývoj mezi kroky je znázorněn šipkami.

Dalšími známými a používanými vizualizacemi, které patří mezi informační vizualizace jsou **časové osy**, **Vennovy diagramy** nebo **ER diagramy**.

2.4 Typy grafů

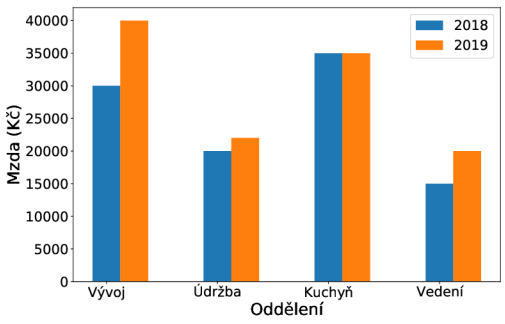
Následující sekce popisuje jednotlivé typy grafů, které lze vizualizovat ve 3D prostoru. V případě čistě dvou-dimenzionálních grafů lze v 3D prostoru natočit kameru, aby její zachycená plocha (viewport) byla rovnoběžná s plochou grafu, dodatečně lze zmenšit zkreslení využitím ortografické projekce. Informace v následujících podsekcích byly čerpány z dokumentací k programům Excel [11], Adobe Flex [1], z dokumentace knihovny matplotlib [6] a z článku Data and Information Visualization Methods, and Interactive Mechanisms: A Survey [7].

2.4.1 Sloupcový graf

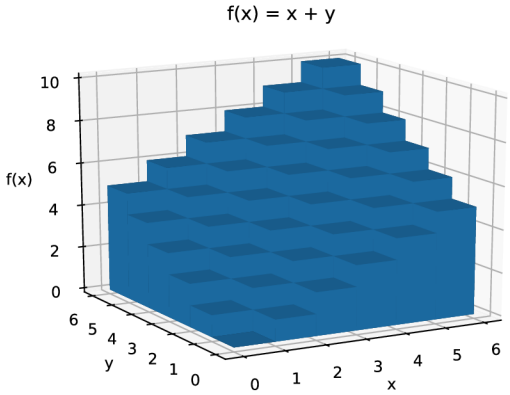
Jednou z nejběžněji používaných forem vizualizace dat je sloupcový graf anglicky nazývaný bar chart nebo column chart. Typicky vyobrazuje diskrétní a kategorická data. Data jsou reprezentována horizontálními či vertikálními sloupci, jejichž výška (délka) odpovídá hodnotám dat. Sloupce mohou popisovat i více posloupností, v takovém případě jsou sloupce spadající do jedné kategorie shluknuty k sobě – aplikace gestalt zákonu blízkosti. Na obrázku 2.6 jsou posloupnostmi mzdy ve dvou odlišných letech (znázorněny barvou a do-definovány legendou) a kategoriemi jednotlivé oddělení společnosti. Pro uživatele je následně jednodušší si utvořit asociace o celkovém vývoji.

Méně používanou variantou sloupcového grafu je plovoucí (floating column chart), jenž nemá fixní spodní hodnotu a sloupec nezačíná zarovnaný s osou, ale na konkrétní hodnotě. Využívá se pro zobrazení rozsahu hodnot v jednotlivých kategoriích a někdy bývá speciální značkou uvedena průměrná nebo střední hodnota.

Sloupcový graf popisující hodnoty, které je možno vizualizovat i ve 2D, je často vytvářen ve 3D prostoru. V takovém případě pouze dochází ke zkreslení a ztížení čtivosti informace. Sloupcového grafu ve 3D lze efektivně využít za podmínky, pokud jsou vizualizovány troj-dimenzionální diskrétní hodnoty v mřížce (např. naměřený vzorek závislý na dvou proměnných), diskrétní funkce a nebo více-dimenzionální kategorická data. Příklad 3D sloupcového grafu se nachází na obrázku 2.7.



Obrázek 2.6: Sloupcový graf popisující změnu mzdy v odděleních společnosti (ilustrační hodnoty)

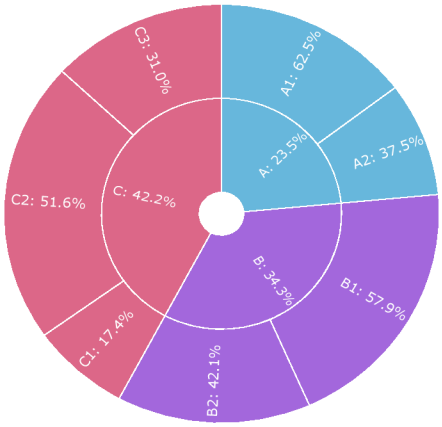


Obrázek 2.7: Sloupcový graf ve 3D zobrazující součet diskrétních hodnot na osách

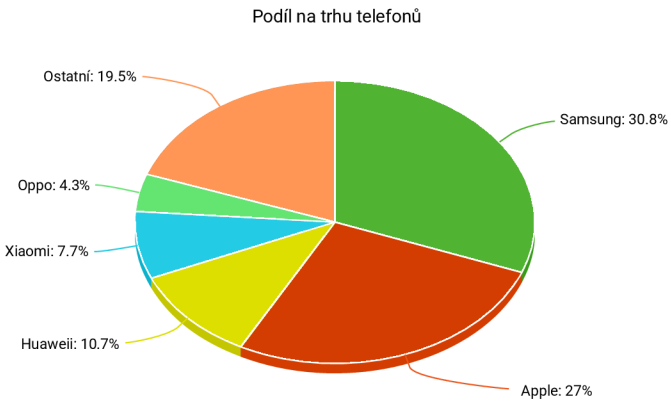
2.4.2 Výsečový graf

Výsečový, či jindy přezdívaný jako koláčový nebo kruhový graf (anglicky pie chart) je tvořen z kruhu rozděleného do výsečí, kde každá výseč reprezentuje část celkové kvantity. Výseče jsou v literatuře nazývány jindy také klíny, řezy či sektory. Výseče popisují rozdílné kategorie nebo části dat s podobnou charakteristikou. Každá výseč je obvykle znázorněna dostatečně kontrastní barvou od okolních, aby bylo možné přehledně rozlišit hranice. Výsečový graf je vhodný pro data, kde se nevyskytují hodnoty záporné nebo blízké nule a data neobsahují více než sedm kategorií. Ukázka výsečového grafu je na obrázku 2.9.

Speciálním typem výsečového grafu je prstencový graf, kde je z kruhu zachován pouze prstenec na obvodu. Rozšířením je vícevrstvý prstencový graf též nazývaný jako paprskový diagram, jenž umožní oproti klasickému výsečovému grafu zobrazení hierarchie v jednotlivých kategoriích, což je reprezentováno širším vnějším prstencem s výsečemi rozdělenými do podkategorií, příkladem obrázek 2.8.



Obrázek 2.8: Demonstrace rozdělení kategorií u vícevrstvého prstencového grafu



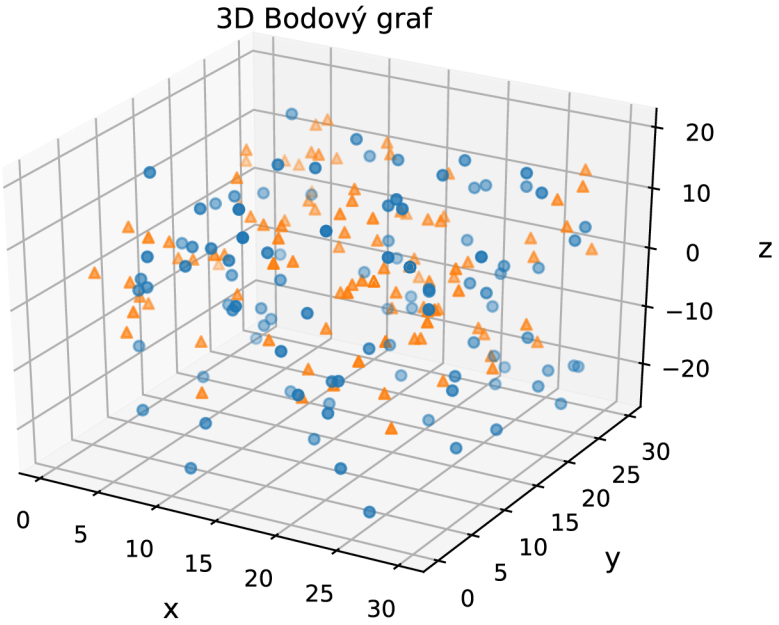
Obrázek 2.9: Podíl na trhu chytrých telefonů, výsečový graf ve 3D

K popisu jednotlivých výsečí existují dva hlavní způsoby. Prvním způsobem je vyjádření hodnoty u každé z výsečí a následného odkázání v legendě na kategorii dle barvy (běžně využíváno při dlouhých názvech kategorií). Při zaznačení kategorie i hodnoty přímo u výseče viz. obrázek 2.9 se jedná o druhý způsob. Ke zvýšení čitelnosti se využívá křivky, která míří od kategorie k popisu, aby nedošlo k přiřazení výseče k popisu, který k ní nepatří. Hodnoty u výsečového grafu se obvykle uvádí v procentech.

Výsečových grafů ve 3D prostoru se stále využívá, přestože mohou vést k zkreslení hodnot. Tento problém lze eliminovat vytvořením grafu ve 3D prostoru, kdy kamera má ortogonální projekci a je umístěna přímo nad grafem, čímž je graf zobrazován jako 2D.

2.4.3 Bodový graf

Bodový graf neboli korelační diagram, XY graf, anglicky scatterplot zobrazuje v kartézských souřadnicích data jako množinu bodů. Každá proměnná v datech je reprezentována na jedné ose. Bodový graf umožňuje pozorovat závislosti v datech, vzájemné vlivy jednotlivých proměnných a jednoduše odhalit odlehlé hodnoty. Numerická data zobrazována bodovým grafem ve 2D jsou ve formátu dvojice (x, y) a ve 3D se jedná o trojici (x, y, z), kde jednotlivé prvky reprezentují pozici bodu v prostoru grafu. Bodový graf je vhodné využít pokud data obsahují mnoho vzorků (bodů) a nebo cílem vizualizace je porovnání obsáhlých datových sad. Bodový graf je také výhodné využít, pokud je při vizualizaci manipulováno se stupnicí způsobem, který může odhalit v datech další korelace.



Obrázek 2.10: 3D bodový graf vyjadřující závislost více proměnných s rozlišením kategorií pomocí značek

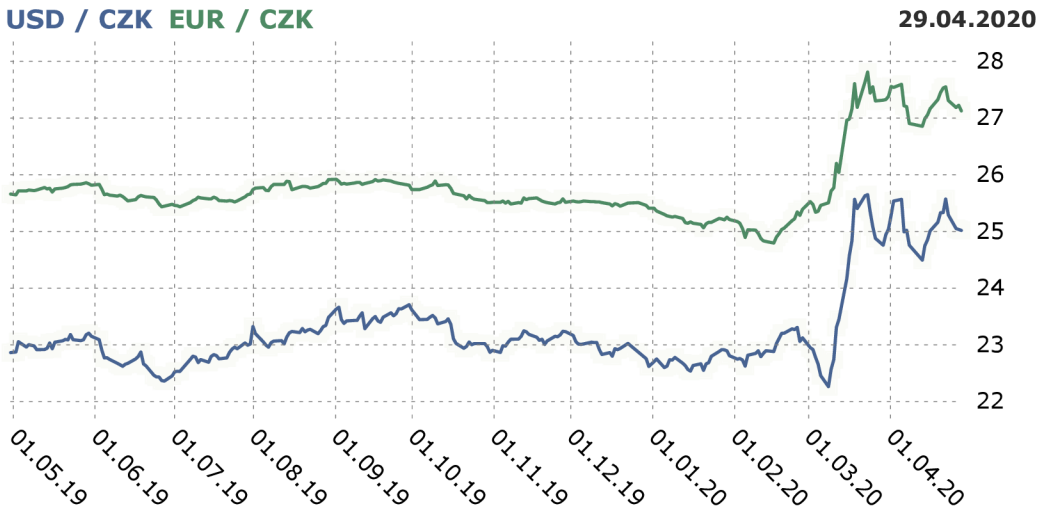
Další dimenze mohou být přidány do grafu změnou barvy bodů majících společnou vlastnost nebo změnou jejich značky (např. využití trojúhelníku místo kolečka pro danou kategorii), jedná se o aplikaci gestalt zákonu podobnosti. Přidáním další osy a rozšíření prostoru grafu do 3D lze vizualizovat závislost hodnot na další proměnné, viz. obrázek 2.10.

Variací bodového diagramu je bublinový graf. Bublinový graf je také složen z bodů reprezentovaných kruhy (koulemi ve 3D), ovšem velikost těchto bodů reprezentuje další hodnotu

z dat. Ve 3D jsou data pro bublinový graf čtveřicí (x, y, z, h), kde h je zmiňovaná velikost koule ležící na pozici definované prvními třemi hodnotami. K lepší analýze a předpovědi budoucího vývoje mohou být body proloženy křivkou – interpolovány.

2.4.4 Spojnicový graf

Spojnicový graf, anglicky line chart, znázorňuje data jako posloupnost bodů propojených spojitou křivkou nebo po částech přímkami. Body mohou být reprezentovány symboly nebo se někdy nezobrazují vůbec a je ponechána pouze samotná spojnice. Vhodný pro vizualizaci kategorických dat s časovými jednotkami. Použití spojnicového grafu odhaluje tendence vývoje a ilustruje vývoj dat v čase. Obvykle se v jednom grafu vizualizuje více posloupností bodů (např. stejné měsíce měření, ale odlišné roky nebo odlišné produkty). Zobrazení více posloupností představuje jednoduchou metodu pro porovnání vývoje trendu mezi kategoriemi. Tradičně se spojnicový graf využívá v ekonomii pro zobrazení kurzů měn. Příkladem je uveden spojnicový graf na obrázku 2.11, znázorňující vývoj kurzů eura a dolaru v průběhu posledního roku.



Obrázek 2.11: Vývoj kurzu eura a dolaru v závislosti na koruně[4]

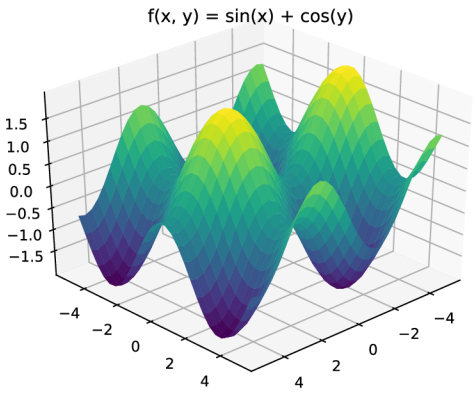
2.4.5 Povrchový graf

Povrchový graf, anglicky surface chart, je tvořen plochou ve 3D prostoru složenou z primitiv (čtyřstěnů nebo trojúhelníků). Obvyklým využitím je vizualizace funkcí dvou proměnných. Data reprezentující body, ze kterých je povrchový graf tvořen, jsou uspořádány do mřížky, kvůli správnému vytvoření povrchu z bodů v datech.

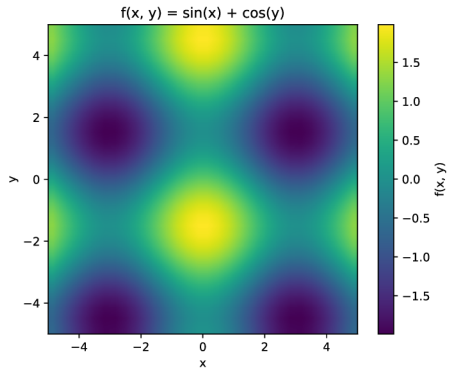
Vizualizace rozptýlených dat povrchovým grafem poskytuje o datech další informace, zejména v bodech, kde nejsou dostupné žádné naměřené hodnoty. Aby bylo možné vizualizovat rozptýlená data povrchovým grafem, je nutné data převést do mřížky. K převedení dat do mřížky se používá matematická technika nazývaná **interpolace**. Geometricky je interpolace prokládání naměřených bodů křivkou, která prochází všemi naměřenými body. Matematicky popsána jako nalezení přibližné hodnoty funkce v intervalu, jsou-li známy hodnoty funkce jinde v tomto intervalu. Z rozptýlených dat je interpolována mřížka, kterou

[4]Převzato z webu www.kurzy.cz, upraveno

lze následně propojit primitivy a vytvořit povrch. Povrchový graf funkce dvou proměnných z mřížky o 30ti bodech na délku a šířku pro ilustraci na obrázku 2.12.



Obrázek 2.12: Funkce dvou proměnných zobrazena povrchovým grafem na intervalu (-4, 4)



Obrázek 2.13: Graf funkce dvou proměnných jako rastr

Stejnou informaci jako vyjadřuje povrchový graf, lze také vyjádřit pomocí 2D rastru, kde šířka a výška představují osy x a y a hodnota dat je reprezentována barvou na dané pozici, což je vhodné pro tisk, ale ztrácí se možnost prohlédnutí dat z více úhlů. Předchozí graf vizualizován jako 2D rastr na obrázku 2.13.

2.5 Blender

Blender je zdarma dostupný open-source software poskytující sestavu prostředků pro práci s 3D grafikou[5]. Konkrétně se jedná o modelování, rigging, animování, simulování, renderování a editací videí. Pro pokročilé uživatele program Blender poskytuje rozhraní pro programování (Python API).

2.5.1 Základní popis

Manipulace s objekty a jejich daty probíhá pomocí editorů v uživatelském rozhraní. Blender obsahuje mnoho editorů pro specifické účely (např. *Image Editor* pro obrázky nebo *Graph Editor* pro nastavování animačních křivek). Nejčastěji používaným je *View3D*, ve kterém je zobrazována scéna. Geometrie scény je tvořena jedním čí více objekty. Pracovní pozice ve scéně je znázorněna 3D kurzorem, jenž definuje polohu sloužící k přidávání nových objektů. Každý objekt v Blenderu obsahuje:

- Pozici ve 3D prostoru, jeho natočení (rotaci) a velikost (scale).
- Data popisující zbytek informací o objektu. Např. tvar objektu (vrcholy, hrany, stěny), materiály popisující vzhled (barva, textura, interakce se světlem), popis animací a mnoho dalších.

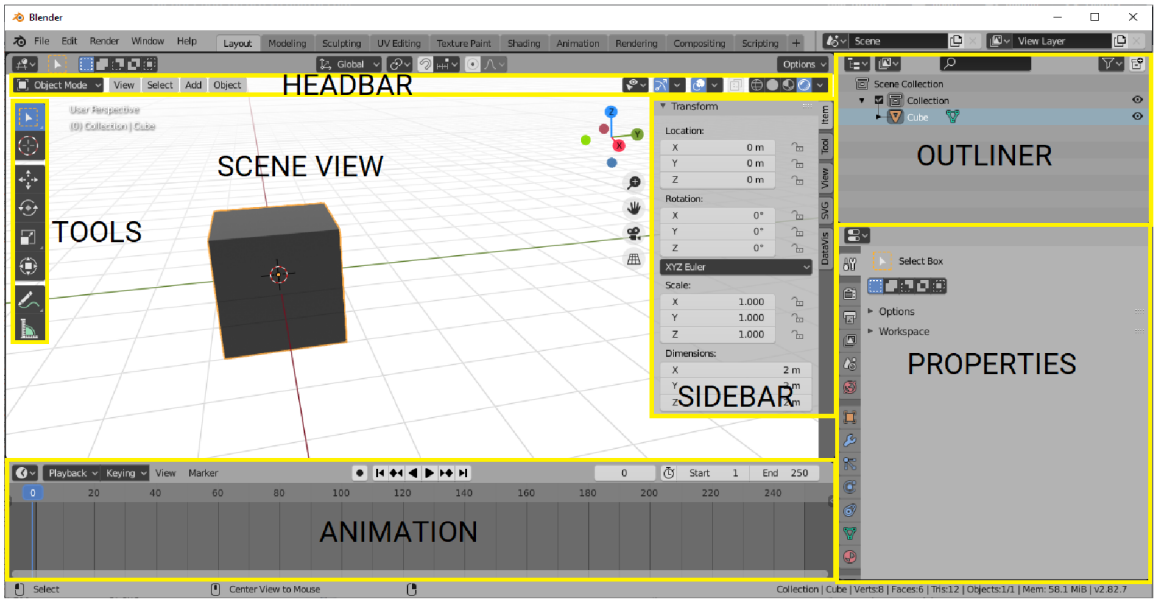
Manipulace se scénou (objekty a jejich daty) je umožněna panely specifickými pro vlastnosti objektu (transformace, animace, data objektu, modifikátory aj.). Objekty ve scéně mohou být aktivní, vybrané a nebo neoznačené. U aktivního objektu je možné přepínat mezi dvěma režimy: objektový režim (object mode), který je implicitní a slouží k práci s objektem jako

[5]Blender https://www.blender.org

celkem (změna posunu, natočení) a editační režim (edit mode) pro manipulaci s geometrií objektu. Určité operace jsou dostupné pouze v specifických módech – např. geometrii nelze upravovat v režimu objektu. Vytvořené scény je možné uložit do souboru formátu blend a nebo exportovat do jiných používaných formátů pro uložení 3D objektů jako jsou fbx nebo obj. Kompletní popis programu a jeho funkcí lze nalézt v referenčním manuálu [2].

2.5.2 Uživatelské rozhraní

Program Blender je primárně ovládán pomocí značně přizpůsobitelného grafického uživatelského rozhraní. Výchozí rozložení uživatelského rozložení je na obrázku 2.14.



Obrázek 2.14: Uživatelské rozhraní Blenderu, lokace důležitých částí rozložení
Obrázek popisuje vybrané části uživatelského rozhraní Blenderu (ohraničené žlutě). K manipulaci se scénou slouží tlačítka v **headbar**. Nástroje aktuálního editačního režimu se nachází v **tools**. Scéna je zobrazena ve **scene view** a hierarchie jejího obsahu v **outliner**. V části **properties** je možné upravit parametry scény a nebo aktivního objektu. V sekci **sidebar** se nachází dodatečné informace o objektu, nástroji a nebo funkcionality add-onů. K základní editaci animací slouží část **animation**.

2.5.3 Python API

Rozhraní Blenderu pro programování (také Blender Python API) umožňuje programátorovi vytvořit skript v jazyce Python pomocí importovaného balíčku *bpy*, jenž zahrnuje třídy, proměnné a funkce k manipulaci s Blenderem pomocí kódu. Skripty lze vytvářet a upravovat přímo v editoru skriptů v Blenderu. Tato možnost skýtá prostředí vhodné pro prototypování, kdy je nutné často upravovat kód a ladit ho. Druhou možností je vytvořit add-on, který si uživatel jednorázově importuje, nainstaluje a bude ho mít dostupný v Blenderu dlouhodobě pro všední použití.

API umožňuje rozšiřování Blenderu do různých směrů, různými způsoby. Zmíněny budou pouze klíčové vlastnosti a následně v kapitole 4 budou popsány konkrétní využité části. API je rozděleno do Python modulů dle poskytované funkcionality. Výčet hlavních modulů:

- `bpy.data` — Vytváření, změny dat, přístup k atributům, přidávání objektů do kolekcí atd.
- `bpy.context` — Přístup ke kontextu aplikace, který specifikuje uživatelský výběr, aktivní scénu, nastavení nástrojů a další.
- `bpy.ops` — Operátory (nástroje) obecně používané uživatelem – vyvolány například stisknutím tlačítka. Obsahují vlastnosti (properties), které dále specifikují chování operátoru.
- `bpy.types` — Definice typů (tříd). Zahrnuje definice využívané pro rozšiřování uživatelského rozhraní nebo také třídy pro definici vlastností operátorů aj.
- `bmesh` — Odkrývá interní API pro práci s geometrií objektu pomocí kódu.
- `mathutils` — Matematické operace.

Programové rozšíření spočívá ve využití definovaných tříd a metod s přidáním vlastní funkcionality jako jsou výpočty nebo speciální kombinace uživatelského vstupu v určitém kontextu. Např. zjednodušeně: vytvoření vlastního operátoru zděděním `bpy.types.ops`, který vykoná požadovanou akci nad zvolenými objekty z `bpy.context.selected_set`, přidaným do uživatelského rozhraní rozšířením třídy `bpy.types.Panel`. Detailní popis API se nachází v dokumentaci [2].

2.5.4 Komunita

Cílem práce je add-on publikovat mezi uživatele Blenderu, aby mohli add-on vyzkoušet a případně používat pro vizualizaci jejich dat. Možnými místy pro akvizici uživatelů se nabízí komunitní fóra[6], webové stránky prezentující novinky[7], obchody s výtvory[8] a nebo skupiny na sociálních sítích reddit[9] a facebook[10].

2.6 Existující řešení

Software, zabývajících se vizualizací dat, existuje mnoho. Hlavním faktorem při jeho výběru hrají roli data, která chce uživatel vizualizovat, a také za jakým účelem vizualizaci vytváří. Pro daný účel je vhodné využít specifický software. Jedná-li se např. o data korespondující k lokaci na mapě, je vhodné použít software, který se specializuje na geografické vizualizace a je pro konkrétní účel vyladěný.

Existující řešení běžně obsahují mnoho možností pro přizpůsobení vizualizací. Nastavení lze rozčlenit od kategorií podle části grafu, které se týká. Typické nastavení vyskytující se v již existujícím software:

[6] Blender artists https://blenderartists.org/
[7] Blender Nation https://www.blendernation.com/
[8] Blender Market https://blendermarket.com/
[9] Reddit https://www.reddit.com/r/blender/
[10] Facebook skupina https://www.facebook.com/groups/blenderartists/

- Rozsah osy, rozestup stupnice, tloušťka čar pro osy
- Formát textu u jednotlivých značek např. počet desetinných míst u čísel nebo zarovnání textu u kategorie
- Formát a obsah popisů os, jejich umístění
- Barevné schéma grafu, zda barva závisí na hodnotě
- Umístění legendy, co má zobrazovat

Specifické grafy mají své vlastní nastavení a neobsahují všechny parametry z výše zmíněných (výsečový graf nemá osy ani stupnici). Obvykle nástroje obsahují možnost nevytvářet osy, stupnice nebo legendu vůbec.

2.6.1 Řešení pro běžné uživatele

Jedná se o řešení převážně ovládaná pomocí grafického uživatelského rozhraní (GUI), uživatelsky přívětivá a přístupná. Nejznámějším vizualizačním nástrojem na platformě Windows je Microsoft Excel a na unixových platformách LibreOffice Calc. Tyto nástroje umožňují pracovat s tabulkovými daty, provádět na nich matematické operace a následně je vizualizovat ve formě grafů. Nástroj, který díky svému propracovanému uživatelskému rozhraní a intuitivnosti ovládání, lze zařadit do této skupiny je Tableau, jenž poskytuje řadu přizpůsobitelných vizualizačních metod.

2.6.2 Pokročilá řešení

Obvykle se jedná o matematické knihovny nebo specifické programovací jazyky poskytující sadu metod a parametrů pro vytvoření vizualizací. Ve vědecké sféře se používá pro výpočty a následné vizualizace grafů placený software Matlab nebo jeho open-source protějšek Octave. V poslední době roste na popularitě Python knihovna Matplotlib [6], jenž je využívána pro vizualizace grafů ve spojitosti s matematickými knihovnami SciPy, NumPy a strojovým učením. Vizualizacím multi-platformně pro android, iOS a Windows se věnuje knihovna SciChart. Nevšedním přístupem k vizualizaci dat je nástroj VTK [14], kde je vizualizace tvořena unikátním procesem pomocí filtrů a odvozováním vyfiltrovaných výsledků. Vizualizacemi dat na webu se zabývá knhovna D3.js, primárně určená pro tvorbu data-driven dokumentů a webových prezentačních vizualizací.

2.6.3 Řešení pro Blender

Specifický je add-on VTK for Blender, který využívá node editoru pro tvorbu vizualizací propojením s VTK[11]. Add-on je pořád v beta verzi a jeví se, že poslední dobou na něm neprobíhá vývoj.

Pro Blender zatím neexistuje unifikované řešení poskytující uživateli možnost vytvářet vizualizace. Uživatelé grafy buď modelují a nebo využívají Python API k vytvoření konkrétního typu grafu. Na internetu se nachází návody a příklady, jak vytvořit např. sloupcový[12] nebo koláčový graf[13].

[11] Dostupný z GitHub https://github.com/simboden/BVtkNodes
[12] Sloupcový graf v Blenderu https://www.youtube.com/watch?v=vhBziRSDclI
[13] Koláčový graf v Blenderu https://www.youtube.com/watch?v=L44q456du-w

Kapitola 3

Návrh

Kapitola zabývající se návrhem add-onu — od interakce uživatele a zakomponování do Blenderu až po problémy, které mohou nastat při tvorbě jednotlivých grafů a jejich komponent, a způsoby jak je řešit.

3.1 Uživatelské rozhraní

Uživatelské rozhraní add-onu by mělo být do Blenderu zakomponováno tak, aby uživatel mohl očekávat jednotlivé operace na takovém místě, kde se nachází operace podobné. Použití add-onu by mělo být pro uživatele jednoduché a přitom by mělo poskytovat rozsáhlé možnosti přizpůsobení. Pokud uživatel nebude chtít vizualizaci přizpůsobovat, měla by být vytvořena s výchozími parametry v co nejlepší formě podle známých informací o datech. V uživatelském rozhraní by měla být vizuálně oddělena část pro načítání dat a pro vytváření grafů. Před zveřejněním add-onu by každá důležitá sekce měla obsahovat ikonku, aby uživatel intuitivněji poznal, čeho se daná sekce týká. Pro uživatele by měl být zobrazen název souboru s jakým aktuálně pracuje a také získané informace o datech.

Vlastní prvky lze do uživatelského rozhraní vložit do různých editorů (editor animace, videa, scény). Uživatel chce vizualizaci vytvořit a následně si data prohlédnout, případně si vizualizaci přizpůsobit dle svých představ. Nejvhodnějším se jeví prvky zařadit do editoru scény, kde se běžně pracuje s objekty a jejich parametry.

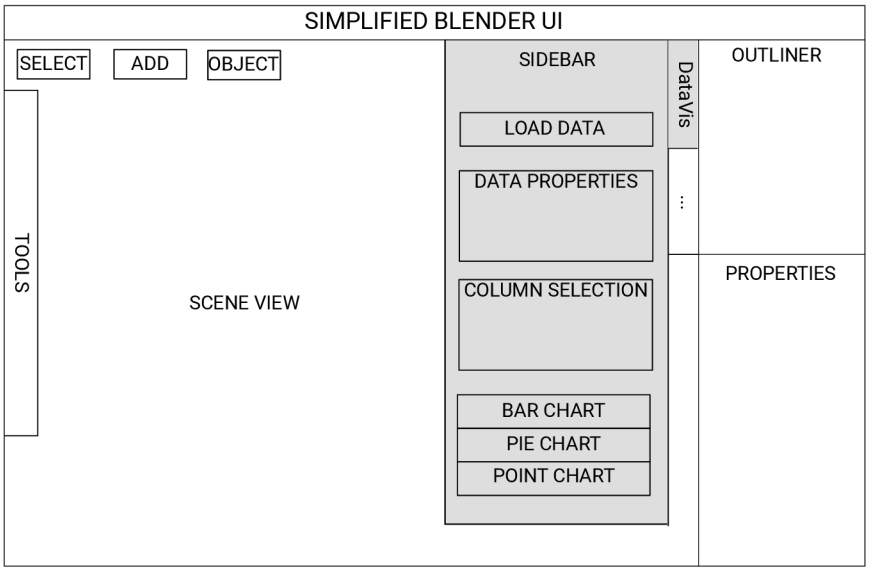
Interakce s vytvořenou vizualizací je realizována využitím zobrazení ve 3D scéně Blenderu, kde uživatel může pohybovat, rotovat a přibližovat pohled na graf. Pro přesnější zobrazení a eliminování chyby při odečítání hodnot z grafu lze přepnout pohled do módu ortografického zobrazení.

3.1.1 Rozložení

Při zohlednění implementačních možností pro rozšíření uživatelského rozhraní Blenderu je důležité definovat, kde se jednotlivé prvky budou nacházet. Vzhled jednotlivých částí uživatelského rozhraní (tlačítek, boxů pro výběr hodnot, modálního okna aj.) je definován globálně a nelze ho jednoduše programově přizpůsobit. Jednotlivé koncepty návrhu jsou znázorněny na wireframe, jenž reprezentuje zjednodušené uživatelské rozhraní Blenderu.

První návrh

První navržené rozložení je reprezentováno obrázkem 3.1. V Blenderu je zvyklostí umísťovat funkcionality nebo příp. celou funkčnost add-onů, které nesouvisí s konkrétní částí pracovní plochy, právě do postranního panelu. Umístěním rozložení na jedno místo je uživateli zprostředkována veškerá funkcionalita blízko sebe a uživatel nemusí hledat, kde se jednotlivé části add-onu nachází. První návrh obsahoval možnost výběru sloupců dat pro jednotlivé osy, od čehož bylo následně upuštěno, kvůli komplexnosti analýzy dat a animování.



Obrázek 3.1: První návrh uživatelského rozhraní
Na obrázku je ilustrován koncept zařazení uživatelského rozhraní do postranního panelu. Obsah postranního panelu je rozdělen na načítání dat a vytvoření grafů.

Druhý návrh

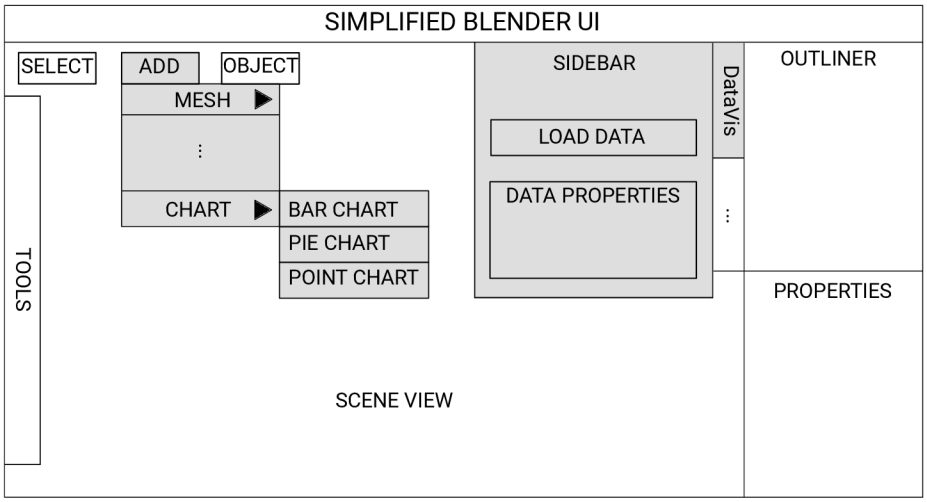
Rozložení bylo předěláno na základě brzké zpětné vazby k prototypu s prvním rozložením. Od prvního se liší rozdělením načítání dat a vytváření grafů do částí Blenderu, kde by je uživatel při práci s 3D scénou očekával více. Druhý návrh je ilustrován obrázkem 3.2. Parametry týkající se celého add-onu, které nejsou přímo spjaty s vytvářením grafů budou umístěny, jak je u add-onů v Blenderu zvykem, v nastavení.

3.1.2 Workflow

Po načtení dat si uživatel vybere graf, volitelně nastaví parametry a potvrzením graf vytvoří. Změna parametrů je dostupná i po vytvoření grafu. Pokud uživatel není s výsledkem spokojen, může pozměnit parametry a vizualizaci upravit, dokud nevykoná další akci.

Načtení dat

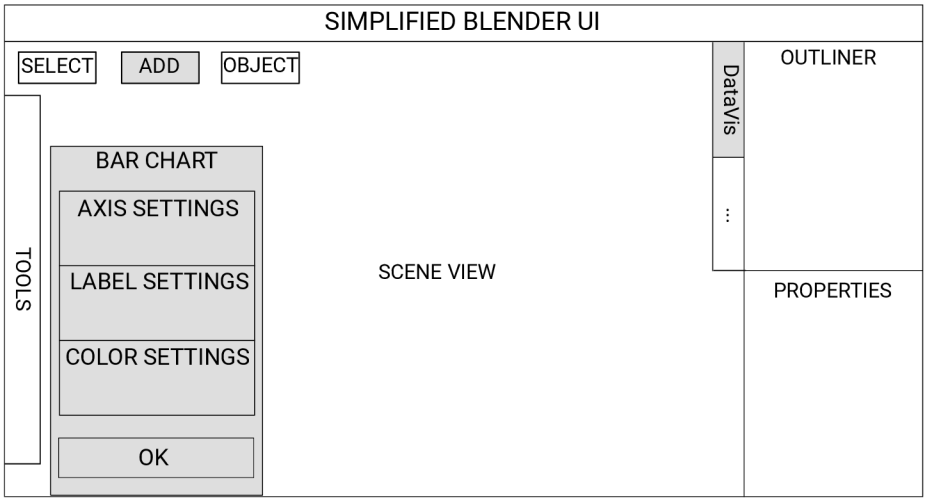
Stiskem tlačítka pro načtení dat bude vyvoláno modální okno souborového systému, které umožní uživateli vybrat soubor obsahující data pro vizualizaci. Pokud je soubor úspěšně načten, zobrazí se informace o datech. V případě nepodporovaného formátu dat dat bude uživateli zobrazena chyba.



Obrázek 3.2: Druhý návrh uživatelského rozhraní
Načítání dat se nachází v postranním panelu. Vytváření grafů bylo přesunuto do menu pro přidávání objektů do 3D scény. Menu je rozšířeno o skupinu charts, ve které se nachází tlačítka k vytvoření jednotlivých typů grafů.

Tvorba a přizpůsobení grafu

Po stisknutí tlačítka pro vytvoření konkrétního grafu bude vyvoláno okno s parametry *properties dialogue*, ilustrováno obrázkem 3.3. Po potvrzení bude graf vytvořen a díky implementaci akcí v Blenderu pomocí operátorů může uživatel následně parametry upravit a vizualizace se bude měnit (tzv. redo akce u operátoru). Pokud nebudou načtena žádná data nebo z aktuálně načtených dat nelze daný typ grafu vytvořit bude tlačítko pro vytvoření daného grafu zašedlé.



Obrázek 3.3: Uživatelské rozhraní pro přizpůsobení grafu
Na obrázku je znázorněno uživatelské rozhraní pro přizpůsobení jednotlivých parametrů grafů, které jsou shromážděny do skupin.

3.2 Import dat

Většina existujících řešení zobrazuje data ve formě tabulek a poskytuje možnost pro export do formátu CSV a na internetu existuje řada služeb poskytující CSV datové sady zdarma. Cílem je uživateli umožnit načtení dat ze souboru ve formátu CSV do paměti Blenderu s použitím již existující knihovny pro zpracování CSV souborů.

Po načtení dat do paměti proběhne analýza, při které bude zjištěno o jaká data se jedná. Výsledkem analýzy by měly být následující informace o datech:

- Počet dimenzí — počet sloupců
- Typ dat — kategorická nebo numerická
- Zda data obsahují dostatek hodnot pro animace
- Obsah prvního řádku — nachází-li se na prvním řádku řetězce, pravděpodobně se jedná o popisy jednotlivých dimenzí
- Rozsah dat v jednotlivých dimenzích
- Počet řádků

Na základě analýzy bude vytvořena predikce o jaká data se jedná a jaké grafy je z načtených dat možné vytvořit. Pokud jsou v datech numerické hodnoty bude implicitně předpokládáno, že se jedná o numerická data, přičemž při vytvoření grafu bude uživateli poskytnuta možnost taková data zobrazit i jako kategorická.

Možným rozšířením je průběžné načítání více datových sad z více souborů a do uživatelského rozhraní zakomponovat seznam datových sad s možností přidání, odebrání a vybrání aktuální datové sady, ze které jsou vizualizace tvořeny.

3.3 Vytvoření grafů

Grafy by měly být vytvářeny způsobem, jenž by uživatel očekával od software určeného pro vizualizaci dat při zachování zvyklostí v Blenderu — bez zbytečných složitostí, ale s širokou možností přizpůsobitelnosti. Cílem je umožnit tvorbu základních grafů pro numerická i kategorická data — řešení pro rendery grafů pro prezentační účely, ale také grafy zkoumatelné pro vědecké účely.

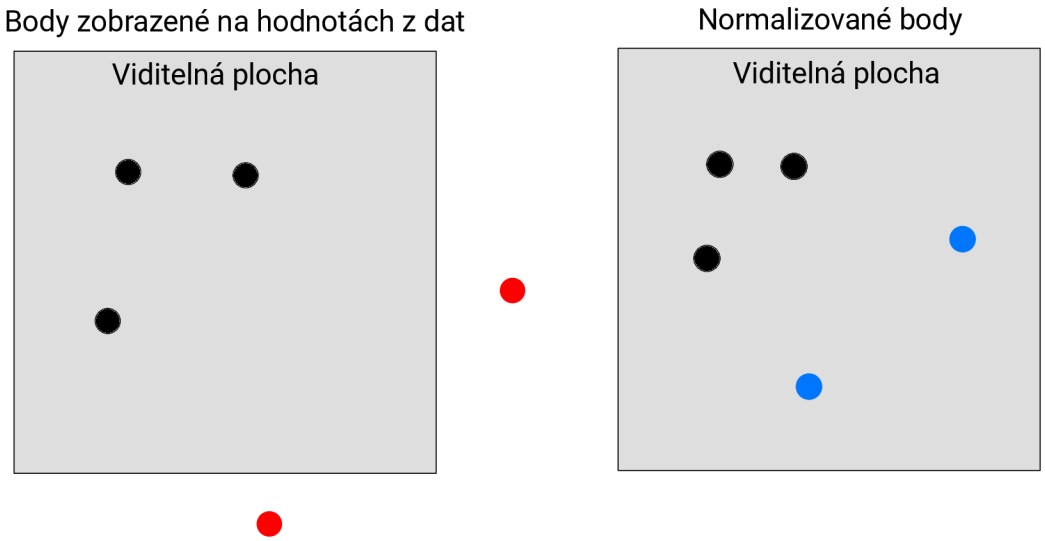
3.3.1 Sdružení komponent grafu

Jednotlivé objekty, ze kterých se bude graf skládat, budou v hierarchii objektů zabaleny do kontejneru grafu. Kontejner pro graf umožní graf i s komponenty přesouvat, škálovat a otáčet vybráním jednoho objektu a usnadní uživateli práci. Lokace kontejneru bude také sloužit jako výchozí bod pro tvorbu všech objektů, ze kterých se graf skládá. V Blenderu jsou objekty vytvářeny na pozici 3D kurzoru, kterou si uživatel postupně mění při své práci. Kontejner bude při vytvoření grafu umístěn na pozici 3D kurzoru.

3.3.2 Normalizace velikosti grafu

Uživatelem nahraná data mohou mít v každé dimenzi různě velký rozsah. V existujícím software jsou grafy vytvářeny nezávisle na rozsahu os do normalizovaného prostoru určité

velikosti, kde jednotlivé vizualizace vůči sobě nemusí mít zachováno stejné měřítko. Ve 2D je prostor grafu definován obdélníkem a ve 3D kvádrem. Stejný problém by měl být řešen i v Blenderu, předpokladem je, že uživatel chce svá data mít zobrazena v prostoru, který může ihned prozkoumat a v tomto prostoru může očekávat všechna data — znázorněno na obrázku 3.4. Například při zvolení řešení bez normalizace — vizualizace jednotlivých hodnot bodového diagramu přímo na pozicích v prostoru podle dat, by mohla znemožnit okamžité nalezení odlehlé hodnoty. V Blenderu je také limitující view distance, kdy při velkém oddálení (velké hodnoty v datech) již nejdou vidět všechny objekty.



Obrázek 3.4: Normalizace bodů do viditelného prostoru

Rozsahy (minima a maxima) jednotlivých dimenzí budou známy po analýze dat. Následně je možné normalizovat jednotlivé hodnoty v datech do intervalu $[0, 1]$ pomocí vzorce 3.1, kde x je pole dat v dané dimenzi, x_i je konkrétní hodnota z x, a z_i výsledná normalizovaná hodnota.

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{3.1}$$

Hodnoty ve všech dimenzích budou normalizovány. Vynásobením výsledné normalizované hodnoty z_i uživatelem zadaným parametrem pro danou dimenzi může být umožněno přizpůsobení velikosti grafu v jednotlivých dimenzích.

Díky využití normalizace dat je možné více vizualizací ze stejných dat kombinovat. Například vytvořit spojnicový graf s osami a popisky a následně konkrétní body zvýraznit bodovým grafem, u kterého budou osy a popisky vynechány.

3.3.3 Podporované typy grafů

Zvoleny byly základní, běžně používané grafy. Informační vizualizace byly vynechány, jelikož jsou příliš specifické podle účelu použití, je náročné zachytit všechny možnosti a 3D prostor jim nedodává přidanou hodnotu. V následujících podsekcích je popsán návrh vytvoření jednotlivých typů grafů s ohledem na implementační možnosti Blender Python API.

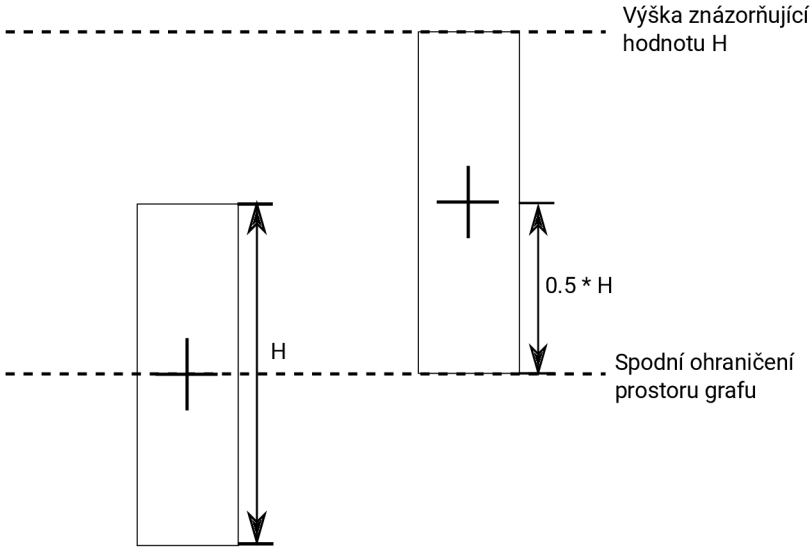
Sloupcový graf

Sloupcový graf bude možné vytvořit ve 3D pro numerická data a ve 2D pro kategorická i numerická data. Vhodná numerická data pro vizualizaci sloupcovým grafem jsou diskrétní, často pouze celá čísla. Nutné je zohlednit způsob, jakým zobrazit data z množiny reálných čísel. První přístup nabízí nástroj Excel, který k takovým datům přistupuje jako ke kategorickým a vizualizuje je postupně nezávisle na numerické hodnotě na horizontální ose (zobrazí je s konstantním krokem). Druhý méně omezující způsob představuje knihovna matplotlib, kde je řešení ponecháno na uživateli a sloupce jsou při numerických hodnotách v datech zobrazeny na pozicích odpovídajících datům, přestože se vizuálně mohou překrývat. Pro add-on do Blenderu byl zvolen druhý způsob, protože uživateli umožní širší možnosti přizpůsobení grafů, ale přitom ho neomezí.

Každý sloupec sloupcového grafu bude vytvářen z primitivního kvádru přidaného do scény. Podporován bude základní typ sloupcového grafu, kdy všechny sloupce začínají na minimu ve vertikální ose a velikost sloupce představuje jeho hodnotu. Šířku sloupce si uživatel může zvolit. Pokud budou vizualizovány kategorická data, tak lze vypočítat šířku jednotlivých sloupců, aby se nepřekrývaly, dle vzorce 3.2. Vypočítaná šířka bude nastavena jako výchozí, přičemž uživatel bude moci hodnotu přepsat.

$$sirka\_sloupce = \frac{velikost\_grafu}{pocet\_kategorii} \tag{3.2}$$

Pozice x a y sloupců budou v případě numerických dat určené hodnotami dat. U kategorických dat budou sloupce rozmístěny od počátku s konstantním krokem odpovídajícím velikosti sloupce. Kvádr je zobrazen ve scéně na poloze s počátkem ve středu jeho geometrie. Ponecháním kvádru s velikostí a pozicí přímo z dat by bylo dosaženo špatného (poloviční) znázornění velikosti odpovídající hodnotě. Pro dosažení velikosti reprezentující hodnotu z dat je nutné kvádr posunout vertikálně o polovinu jeho velikosti, ilustrováno obrázkem 3.5. Uživateli bude poskytnuta možnost místo kvádru použít válec a nebo vlastní objekt ze scény.



Obrázek 3.5: Posun kvádru s počátkem ve středu geometrie

Výsečový graf

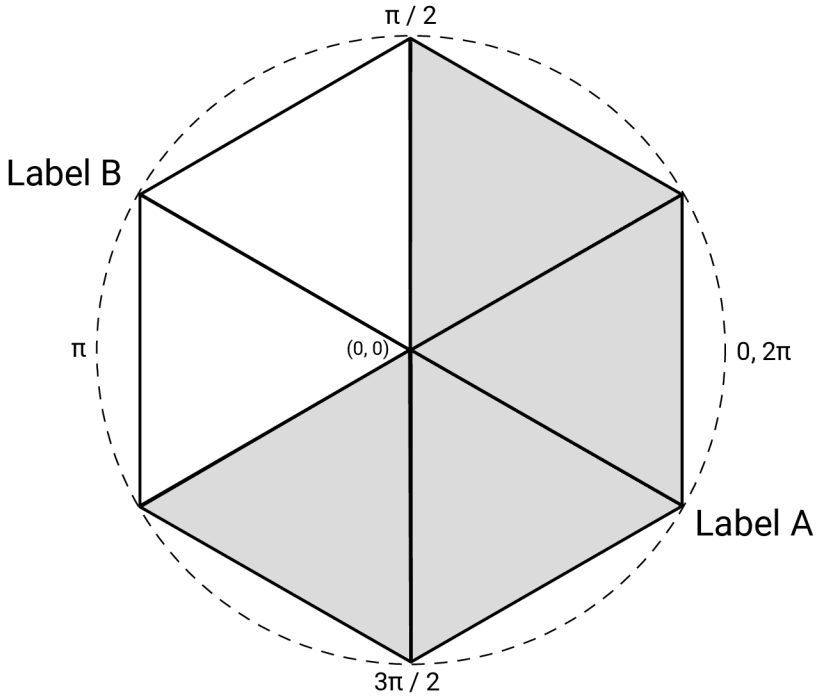
Výsečový graf bude možné vytvořit pouze z kategorických dat s nezápornými hodnotami, které neobsahují na prvním řádku popisy jednotlivých os.

Výsečový graf bude tvořen z několika trojbokých hranolů, jejichž počet si uživatel bude moci zvolit. Hranoly budou postupně vytvářeny a rotovány kolem počátku (osa Z) o hodnotu inkrementu vycházející ze vztahu 3.3. Základ koláčového grafu rozděleného do šesti hranolů lze vidět na obrázku 3.6. Díky zařazení hranolů do hierarchie pod kontejner grafu jsou hranoly vytvořeny lokálně vždy na souřadnicích $(0, 0)$ a není nutné využívat dalších transformací.

$$inkrement = \frac{2 \cdot \pi}{celkem\_hranolu} \tag{3.3}$$

$$pocet\_hranolu = \frac{x}{\sum_i \cdot x_i} \cdot celkem\_hranolu \tag{3.4}$$

Po vytvoření hranolů bude zjištěno, jakou část grafu zabírají jednotlivé kategorie dle vztahu 3.4, kde x je aktuálně zpracovávaná kategorie a jmenovatel zlomku je suma všech hodnot. Vynásobením celkovým počtem hranolů získáme desetinné číslo vyjadřující kolik hranolů je nutných pro zobrazení dané kategorie. Po zaokrouhlení na celé číslo bude vybrán počet hranolů reprezentující danou kategorii a tyto hranoly budou spojeny do jedné geometrie (výsledné výseče), aby na ně bylo možné aplikovat jeden materiál (obr. 3.6 — šedá a bílá část). Zvýšením počtu hranolů se zvýší přesnost zobrazení hodnot a zachování poměrů mezi nimi. Nízký počet hranolů také omezuje maximální počet kategorií, které lze zobrazit, proto bude zvolena dostatečně velká výchozí hodnota pro běžné účely.



Obrázek 3.6: Koncept vytvoření výsečového grafu z hranolů
Výseče se skládají z několika hranolů majících společný materiál (šedá a bílá barva). Následně je k nim vytvořen popisek.

Popisy ke každé z kategorií budou umístěny vždy v polovině výseče (obr. 3.6 — Label A pro šedou výseč a Label B pro bílou výseč). Lokace v prostoru je vypočítána pomocí goniometrických funkcí v jednotkové kružnici. Úhel popisu na kruhu vypočítáme dle velikosti výseče a jejího počátku. Souřadnice x je následně vypočítána funkcí *cos* a souřadnice y funkcí *sin*. Vzdálenost popisu lze měnit vynásobením x a y komponent konstantou (uživatelské nastavení).

Bodový graf

Bodový graf bude dostupný pro numerická data ve 2D i 3D. K znázornění hodnot bude jako výchozí využívána koule, přičemž uživateli bude poskytnuta možnost vybrat jiný objekt ze scény. Nastavitelným parametrem bude velikost tohoto objektu. Vybraný objekt bude umístěn na normalizovanou pozici (x, y, z) (kde y je ve 2D 0) odpovídající hodnotám z dat.

Možným rozšířením je využít dalších hodnot v datech, které by mohly být reprezentovány změnou objektu, barvy a nebo změnou velikostí bodu za cenu zvýšené komplexnosti analýzy dat a složitosti animace takového grafu.

Spojnicový graf

Vytvoření spojnicového grafu bude umožněno pro kategorická i numerická 2D data. Spojnice bude tvořena křivkou definovanou vrcholy z dat. Křivka bude obalena objektem (bevel object) plochy, což umožní na tvar křivky aplikovat materiál. V případě kategorických dat budou horizontální kroky konstantní a vrcholy křivky budou odpovídat hodnotám z dat. Bude-li se jednat o numerická data, budou vrcholy umístěny rovnou na normalizovanou pozici vypočtenou z hodnot dat. Před obalením křivky objektem je nutné vrcholy propojit. K zachování správné topologie bude nutné vrcholy vytvořené z numerických dat seřadit dle x-ových hodnot. Následně mezi sousedními vrcholy bude vytvořena hrana. K vyhlazení rohů křivky bude možné využít operaci bevel, která zjemní rohy a přidá k ostrým místům další vrcholy.

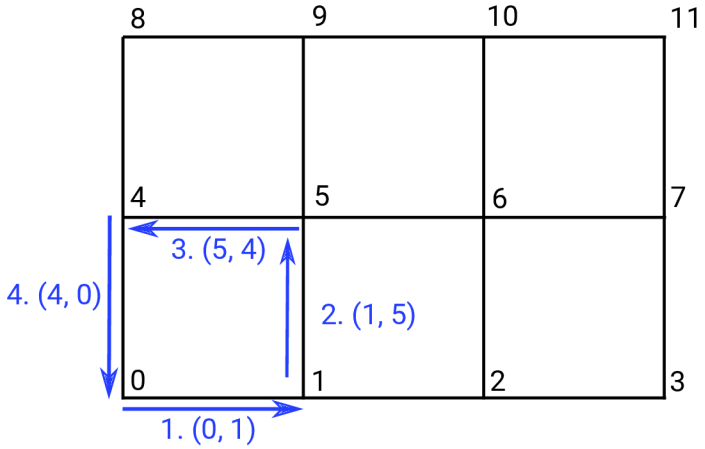
Spojnicové grafy často znázorňují více posloupností, v add-onu bude tato možnost poskytnuta vytvořením dalšího spojnicového grafu se stejným počátkem z jiných dat, nastavením stejných rozsahů a vynecháním vytvoření objektů os.

Povrchový graf

Povrchový graf bude možné vytvořit z 3D numerických dat. Jak je zmíněno v kapitole 2, tak data mohou být roztroušená takovým způsobem, že by z nich nebylo možné jednoznačně utvořit povrch. K převedení do mřížky bude využita interpolace.

Interpolace je rozsáhlé téma a existuje mnoho možností a metod, jak k ní přistupovat. Výhodné se jeví použít již existující knihovnu, ve které je interpolace implementována. Na základě zkušenosti byla zvolena knihovna SciPy[1] poskytující řadu implementací interpolace. Načtená data budou převedena do formátu, se kterým knihovna pracuje, a budou interpolována do mřížky, jejíž velikost si určí uživatel. Hodnoty z mřížky budou následně vloženy do 3D scény jako vrcholy a mezi nimi budou vytvořeny stěny způsobem ilustrovaným na obrázku 3.7. — pro vytvoření správné geometrie pomocí Python API jsou vrcholy uloženy v listu a je nutné specifikovat ve správném pořadí vrcholy tvořící stěnu. Vytvořením stěn vznikne plocha (povrch) reprezentující načtené hodnoty dat.

[1] Dokumentace k modulu pro interpolaci ve SciPy: https://docs.scipy.org/doc/scipy/reference/interpolate.html



Obrázek 3.7: Postup vytvoření stěn u povrchového grafu
Na obrázku je znázorněna mřížka, kde jednotlivé vrcholy mají svůj index. Modré šipky představují pořadí, v jakém je nutné specifikovat indexy vrcholů (tvořících hrany) k vytvoření stěny se zachováním správné topologie.

Nevýhodou využití již existující knihovny je exponování Pythonu a instalace závislostí běžnému uživateli. Nabízí se řešení vytvořit návod, dle kterého může uživatel knihovny doinstalovat a nebo zakomponovat instalaci knihovny do uživatelského rozhraní Blenderu.

3.4 Komponenty grafu

Sekce návrhu zabývající se možnostmi vytvoření jednotlivých komponent grafu. Komponenty by měly být znovupoužitelné u různých typů grafů a nezávislé na jejich implementaci. Uživatel si bude moci zvolit, zda chce jednotlivé komponenty vytvořit. Pro všechny komponenty budou vytvořeny výchozí materiály, aby bylo možné barvu komponent upravovat nastavením barev materiálu.

Každá komponenta bude v hierarchii opět potomkem kontejneru komponenty, aby s ní šlo jednoduše manipulovat. Kontejner komponenty bude potomkem kontejneru grafu, ke kterému náleží.

3.4.1 Osy

K vytvoření os a značek bude využito primitivních objektů. Do scény budou přidány kvádry, kterým následně bude upravena pozice a škála, aby dohromady představovaly osu.

Jelikož jsou osy důležitou součástí grafu, je nutné poskytnout uživateli dostatek přizpůsobitelných elementů. Výčet podporovaných parametrů pro osy:

- Rozsah — Minimum a maximum zobrazovaných hodnot
- Krok stupnice — Po jakém kroku mají být zobrazeny značky na ose
- Tloušťka čar — Tloušťka osy, tloušťka značek
- Pozice — Pozice vertikální osy
- Vzdálenost od grafu — Mezera mezi grafem a osami

Rozsah os bude implicitně nastaven, aby zahrnoval všechna načtená data. Uživatel si dále bude moci přizpůsobit rozsah os dle svých požadavků. Na základě rozsahu os se budou měnit data zobrazovaná grafem v normalizovaném prostoru grafu. Graf bude zobrazovat pouze data, která jsou ve zvoleném rozsahu.

Na ose bude na každé hodnotě kroku vygenerována značka. Krok bude také přizpůsobitelný, výchozí hodnota bude vypočítána tak, aby byl zobrazen určitý čitelný počet značek a nedošlo k vytváření mnoha se překrývajících značek a textových objektů.

3.4.2 Popisy

Popisy budou vytvářeny z dostupných dat. Pokud data nebudou obsahovat dostatek informací pro vytvoření popisů, uživatel bude moct zadat hodnotu popisů ručně. Velikost textu popisů bude přizpůsobitelná.

Titulek bude zobrazen vycentrovaný na střed nad grafem. Kvůli zachování jednoduchosti a jednoznačnosti dat bude text titulku zadávat vždy uživatel.

Popis značek stupnice budou vytvořeny automaticky na základě rozsahu osy a délky kroku. U popisů os numerických dat bude možné zvolit číselný formát a počet desetinných míst. Popisy kategorických dat budou zkoseny, aby byly čitelnější.

Popisy os se budou nacházet vedle každé z os a jejich obsah bude tvořen hodnotami z prvního řádku z dat a nebo uživatelem zadaným vstupem.

3.4.3 Legenda

Legendu bude možné vytvořit pro grafy znázorňující kategorie nebo pro grafy využívající barvu k rozlišení hodnot. Legenda bude tvořena plochou, na které budou zobrazeny jednotlivé kategorie — barva a název kategorie v případě kategorických dat. Poloha legendy vůči grafu bude přizpůsobitelná.

3.4.4 Mřížka

Pro jednodušší odečítání hodnot pomocí mřížky lze využít mřížku ve 3D scéně, kterou implicitně Blender zobrazuje. Tato mřížka je škálovatelná a přizpůsobitelná. Uživatel si může graf nebo mřížku transformovat do pozice, kde čáry mřížky odpovídají hodnotám, které chce analyzovat. Vytvoření mřížky z objektů by v Blenderu přeplňovalo scénu a snižovalo čitelnost grafů.

3.5 Barvy

Pro obarvení vizualizací bude uživateli poskytnuto více přizpůsobitelných možností. Existující řešení poskytují z hlediska obarvení pestrou škálu možností. Běžně vizualizační software poskytuje uživateli možnost všechna data zobrazit jednou barvou nebo využít předdefinovanou paletu barev nebo aplikovat na graf přechod. V existujícím software je také možnost zvýraznit určitou hodnotu v datech, na kterou chce uživatel upozornit.

Je nutné zohledenit navrhované složení grafů ve 3D scéně a možnosti aplikace materiálu na objekty nebo geometrie v Blenderu. Výčet vhodných variant obarvení grafu s využitím materiálů v Blenderu:

- Konstantní — Jedna barva pro geometrii nebo všechny objekty grafu, nepřenáší další informaci, pouze vizuální efekt.

- Gradient — Určení barvy na základě pozice v geometrii nebo polohy objektu v prostoru, zvýrazňuje hodnotu v určité dimenzi. Také může být využito pro zlepšení vizuální prezentace grafu.
- Náhodné — Náhodná barva dle polohy / pozice v geometrii, nepřenáší další informaci. Pouze vizuální efekt.

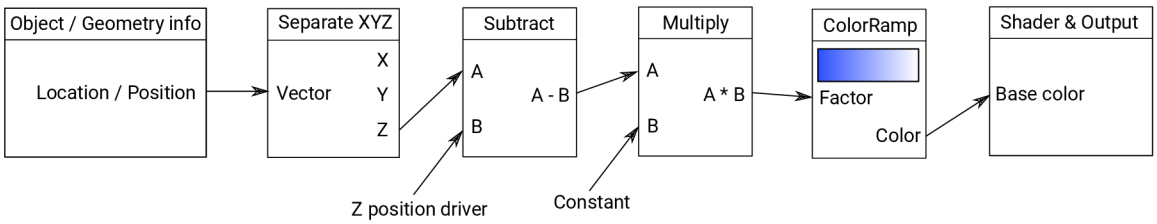
3.5.1 Materiály

Použití materiálu bez definice shaderu lze využít pouze pro grafy, kde jsou data zobrazena pomocí objektů, kterým lze přiřadit materiál s barvou odpovídající hodnotě. Tento přístup je časově i paměťově náročný, protože se pro každý bod v datech vytváří nový materiál. Výhodou je, že uživatel při analýze vizualizace může vybrat jakýkoliv bod grafu (např. odlehlou hodnotu) a změnit jeho barvu (zvýraznit ho), bez toho aniž by se změnily barvy ostatních bodů reprezentujících stejnou hodnotu. Uživatel může definovat barvu, od které se bude daný způsob obarvení odvíjet.

3.5.2 Shader Nodes

Pokud je graf složený pouze z jedné geometrie (povrchový, spojnicový graf), tak na ni lze aplikovat pouze jeden materiál. V takovém případě lze využít obyčejného materiálu k obarvení grafu jednou barvou. K obarvení geometrie nebo objektů grafu lze také definovat shader. Shadery jsou v Blenderu tvořeny pomocí Shader Nodes. Shader se skládá z různých bloků, které mohou provádět matematické operace, poskytovat informace o objektu, generovat texturu, počítat vlastnosti materiálu aj.

Pro obarvení grafů byl navržen shader, který nabízí dostatečné možnosti přizpůsobení a je ním možno obarvit body v grafu závisle na jejich poloze. Bloky shaderu jsou zobrazeny na obrázku 3.8. **Object / Geometry info** poskytuje výstupní hodnotu informující o pozici objektu nebo pozici v geometrii. **Separate XYZ** rozdělí jednotlivé komponenty vektoru pozice na hodnoty reprezentující jednotlivé dimenze. K zachování invariace vůči posunu grafu ve scéně bude využíván *Z position driver*. Driver automaticky získává polohu kontejneru grafu ze scény, která je pomocí bloku **Subtract** odečtena od Z souřadnice geometrie nebo objektu, čímž je převedena do lokálního prostoru grafu. Blok **Multiply** slouží k násobení konstantou *Constant*, kvůli posunutí polohy u sloupcového grafu. Nejdůležitější součástí je blok **ColorRamp**, kterému je na vstup *Factor* přivedena vertikální poloha objektu nebo geometrie normalizovaná předchozími kroky do intervalu $[0, 1]$. Vstupem je vybírána pozice v gradientu, který si může uživatel nadefinovat a přizpůsobit a výstupem je barva na této pozici, která je následně aplikována jako *Base color* ve výstupním bloku shaderu **Shader & Output**. Tento shader bude nutné implementovat pomocí API a aplikovat ho na geometrii nebo všechny objekty vyobrazených dat.



Obrázek 3.8: Blokové schéma Node Shaderu pro obarvení grafů

Uživatel si bude moci díky navrženému shaderu jakkoliv přizpůsobit barvy vizualizace pomocí nastavení barev a jejich pozic v bloku **ColorRamp**. Znalý uživatel bude moci využít shader jako základ pro vytvoření vlastního shaderu, který dokáže poskytnout ještě větší možnosti pro analýzu dat díky barvě.

3.6 Animace

Animace jsou v Blenderu tvořeny pomocí snímků, které jsou vázány na vlastnosti objektů a nebo na vlastnosti geometrie. Hodnota vlastnosti je následně interpolována v čase mezi dvěma odlišnými snímky.

Animované grafy bude možné tvořit z dat, kde se bude nacházet více hodnot k zobrazení na vertikální ose. Pokud uživatel bude mít taková data načtená, automaticky se mu nabídne možnost data animovat a specifikovat vzdálenost jednotlivých snímků. Cílem je pro uživatele vytvořit animovaný základ, který si následně bude moci upravit.

3.6.1 Animace objektů

Pokud je graf tvořen z objektů, je možné animovat jejich polohu, velikost či rotaci. Příkladem animace objektů může být změna velikosti a pozice sloupců na základě hodnot ve dvou měřeních — vytvoření snímku pro škálu a pozici pro první i druhé hodnoty dat.

3.6.2 Animace geometrie

Je-li graf tvořen jednou geometrií, tak v Blenderu nelze (bez instalace dalšího add-onu) animovat polohy nebo velikosti jednotlivých vrcholů, hran nebo stěn křivky přímo. Pro animace geometrie se využívá tzv. shape keys, které definují vzhled celé geometrie pro daný snímek. Vytvořen bude základní shape key, od kterého se budou následně odvíjet další shape keys definující geometrii pro jiné hodnoty dat.

Kapitola 4

Realizace

V této kapitole je popsána realizační část práce. Popsáno je využití Blender Python API (dále bpy) k vytvoření add-onu pro vizualizaci dat. Dále jsou popsány zajímavé způsoby, jakými byly překonány některé implementační problémy při zachování rozšiřitelnosti do budoucna. Poslední sekce se zabývá publikováním add-onu pro veřejnost.

4.1 Využité části API

Popis klíčových částí a konceptů API, které byly při implementaci skutečně využity. Popsány jsou obecné praktiky pro implementaci add-onu a detailně funkčnost operátoru, jelikož je to základní blok implementace a je na něj odkazováno v následujících sekcích.

4.1.1 Inicializace

Zdrojové soubory musí tvořit balíček, který je následně možné nainstalovat do Blenderu. Vstupním bodem add-onu je soubor `__init.py__` inicializující celý balíček, který se nachází v kořenovém adresáři `data_vis`. Ve vstupním souboru se nachází proměnná `bl_info`, kde jsou specifikovány informace o add-onu. Všechny třídy, které pracují s vnitřními daty Blenderu jsou zaregistrovány funkcí `bpy.utils.register_class`. Třídy, které není nutné pojmenovávat dle konvencí bpy, obsahují prefix `DV_` (Data Visualisation), aby nedošlo ke konfliktům s ostatními add-ony.

Pro správnou funkci API obsahuje každá zaregistrovaná třída třídní proměnné specifikující další informace. V každé zaregistrované třídě se musí nacházet proměnná `bl_idname` — identifikační jméno, kterým je na danou třídu odkazováno při definici uživatelského rozhraní a nebo mimo prostor skriptu add-onu. Správné je také volitelně definovat proměnnou `bl_label` — krátký popis dané třídy, zobrazován v uživatelském rozhraní.

Před registrací samotných tříd jsou načteny vlastní ikony ze složky `icons/` a lze na ně odkazovat a použít je v uživatelském rozhraní skrze proměnnou `preview_collections`.

4.1.2 Operátory

Operátor v bpy zapouzdřuje akci manipulující s vnitřními daty Blenderu. Akce může být vyvolána uživatelem (kliknutí na tlačítko) a nebo kódem. Vlastní operátor je tvořen implementací třídy, která je podtřídou `bpy.types.Operator`. Jména vlastních definic tříd

operátorů obsahují prefix dle konvence[1]. Při popisu jednotlivých tříd v následujících sekcích bude prefix pro zachování jednoduchosti vynechán.

Chování operátorů je specifikováno proměnnou `bl_options`, v rámci implementace je využito nastavení `REGISTER` a `UNDO`, jenž je nutné použít pro následné zobrazení redo panelu.

Vlastnosti

Přizpůsobení řízení toku kódu a parametrů operátorů se realizuje pomocí vlastností (angl. properties). Vlastnosti je možné zobrazovat v uživatelském rozhraní nebo animovat. U všech vlastností lze definovat jejich jméno (zobrazováno v uživatelském rozhraní) a popis (zobrazen po najetí myši). Typy vlastností využívané v add-onu:

- Float — Desetinné číslo, je možné definovat velikost kroku
- Int — Celočíselné hodnoty
- String — Řetězec, lze definovat jeho maximální délku, jeho podtypem je cesta pro modální okno souborového systému
- Boolean — Logická hodnota True nebo False
- Enum — Výčtový typ, umožňuje definovat jednotlivé možnosti, jejich hodnoty a popisy.
- Vector (Float, Int) — Vektor o velikosti 2 nebo 3 typu Int nebo Float.

Vlastnosti jsou definovány anotovanou třídní proměnnou operátoru. Vlastnost je anotována třídou odpovídající typu z modulu `bpy.types` a můžou být uvedeny parametry, které vlastnost do-specifikují. Na výpisu 4.1 je uveden příklad popisující celočíselný selektor počtu vrcholů se jménem, popisem, omezujícím rozsahem a výchozí hodnotou.

```
import bpy

vertices: bpy.types.IntProperty(
    name='Number of vertices',
    description='Specify desired number of vertices',
    min=1,
    max=20,
    default=5
)
```

Výpis 4.1: Ukázka definice a přizpůsobení vlastnosti

Vlastnosti je možné shlukovat do skupin a definovat skupinu vlastností (angl. property group). Na skupinu vlastností se dá následně v operátoru odkazovat speciální vlastností anotovanou typem `bpy.types.PointerProperty`.

[1]Konvence pojmenování tříd: https://wiki.blender.org/wiki/Reference/Release_Notes/2.80/Python_API/Addons

Metody

Popsány jsou základní metody operátorů, které jsou volány vnitřním systémem Blenderu při určitých akcích. Rozšíření Blenderu spočívá primárně v definici těchto funkcí.

- `poll` — Specifikace prerekvizit k správnému provedení metody `execute`.
- `draw` — Override této metody dovoluje definovat vlastní uživatelské rozhraní.
- `invoke` — Metoda volaná při uživatelské interakci.
- `execute` — Metoda volaná po `invoke` a nebo při redo akci operátoru. Definice chování opertároru je v této funkci. Funkce musí vracet specifickou návratovou hodnotu, nejčastěji `CANCELLED` nebo `FINISHED`. Parametrem metody je předávaný kontext aplikace.

4.1.3 Rozšíření uživatelského rozhraní

Uživatelské rozhraní je možné rozšířit buď implementací metody `draw` v operátoru a nebo implementací podtřídy třídy pro definici části uživatelského rozhraní.

Panel pro načtení dat je implementován třídou `DV_AddonPanel` a definicí rozložení v metodě pro vykreslení UI. Místo panelu v uživatelském rozhraní definují třídní proměnné.

Hodnoty proměnných definujících pozici panelu je možné upravit v nastavení, které je implementováno třídou `DV_Preferences`. V nastavení je umístěno tlačítko vyvolávající operátor `InstallModules`, který se pokusí najít interpret Pythonu využívaný Blenderem a nainstalovat knihovnu SciPy pro povrchový graf.

Zmiňované třídy a funkcionality jsou implementovány v inicializačním souboru add-onu, kvůli zjednodušení závislostí mezi třídami.

Výsledné uživatelské rozhraní pro tvorbu vizualizací se nachází v příloze B.

4.2 Správa dat

Práce s daty je zapouzdřena do třídy `DataManager` implementované v souboru `data_manager.py`. Pro dostupnost stejné instance ve všech částech implementace je třída implementována jako singleton[2].

4.2.1 Načtení dat

K vyvolání modálního okna souborového systému slouží implementace operátoru `DVLoadFile` nacházející se v `operators/data_load.py`. K uložení cesty uživatelem zvoleného souboru v modálním okně je definována třídní vlastnost `filepath` speciálního podtypu `FILE_PATH`.

Invokací operátoru (kliknutím na tlačítko `Load Data`) je vyvoláno modální okno souborového systému pomocí `window_manager.fileselect_add(self)`. Návratovou hodnotu metody `invoke` je v tomto případě `RUNNING_MODAL`, aby se operátor nepovažoval za ukončený dokud neproběhne metoda `execute`.

Jakmile uživatel vybere soubor je vykonána metoda `execute`, kde je zavolána metoda pro načtení dat z třídy `DataManager` a parametrem je předána cesta k souboru. Po zjištění cesty již načtení dat proběhne v třídě `DataManager` s využitím Python modulu `csv`.

[2] Implementace návrhového vzoru převzata z: https://python-3-patterns-idioms-test.readthedocs.io/en/latest/Singleton.html

4.2.2 Analýza dat

Po načtení je automaticky provedena analýza dat při které je iterováno přes všechny řádky z načteného souboru. Analýza dat spočívá v typové kontrole hodnot jednotlivých sloupců a co nejlepšího určení jejich reálného smyslu. Při analýze je také zkontrolována kompletnost dat, zda na určitém řádku se nenachází méně sloupců (což není podporováno). Výsledky analýzy určují následující podmínky:

- Pokud se ve všech sloupcích na prvním řádku nachází řetězce, tak je to indikováno proměnnou a tyto řetězce jsou uloženy pro následující využití pro popisy os.
- Nachází-li se na první pozici řetězec a na dalších čísla jsou data vyhodnocena jako kategorická.
- Nachází-li se na všech pozicích pouze čísla jsou data vyhodnocena jako numerická.

Celkový počet sloupců určuje zda jsou data vhodná pro animace. Výsledek analýzy může být nevalidní, pokud uživatel načte data, která se jeví jako numerická, ale jejich význam je kategorický (např. roky). Nevalidní výsledek analýzy je možné přepsat voláním metody `override`, čehož je využito v operátorech pro tvorbu grafu.

Analýzou dat je také zjištěno minimum a maximum hodnot numerických sloupců (jejich rozsah) a je uloženo do proměnné `self.ranges`. Ke zjištění výsledků analýzy a přístupu k instančním proměnným slouží metody `get_range`, `get_step_size`, `get_labels` a jiné.

4.3 Vytvoření grafů

Snahou při implementaci vytváření grafů bylo zachovat rozšiřitelnost do budoucna. Vytvoření grafu probíhá pomocí operátoru, jenž je podtřídou generického operátoru, definuje skupiny vlastností, které využívá a implementuje logiku vytvoření objektů popisujících data a jejich přidání do scény.

4.3.1 Skupiny vlastností

Původní myšlenkou bylo využít dědičnosti a implementovat společné vlastnosti v generickém operátoru. Po testování vyšlo najevo, že vlastnosti v podtřídě nezdědí anotovanou funkčnost a nelze je následně využít. Řešením bylo definovat skupiny vlastností, na které je možné v kódu jednotlivých grafů odkazovat pomocí `bpy.types.PointerProperty`, pokud dávají u daného grafu smysl. Tyto skupiny vlastností jsou definovány v souboru `properties.py`.

Vlastnosti byly rozděleny do skupin, podle toho zda se týkají stejné části grafu a nebo spolu logicky souvisí. Skupiny vlastností vytvářející objekty obsahují vždy vlastnost `create` typu boolean, díky které je rozhodováno zda vytvořit objekty reprezentující danou komponentu ve scéně. Výčet implementovaných skupin:

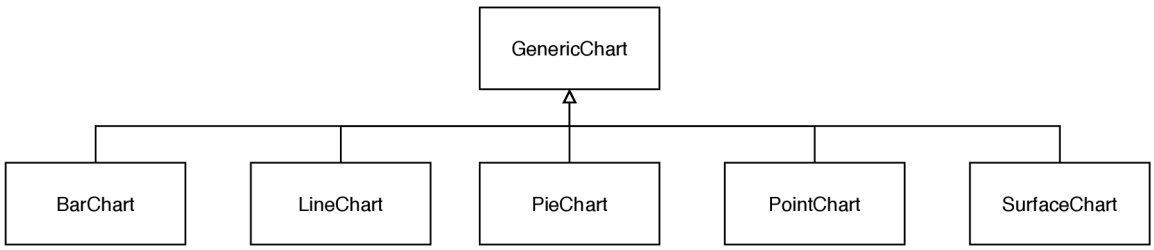
- `AxisPropertyGroup` — Nastavení rozsahu a kroku pro jednotlivé osy. Velikost objektů, ze kterých je osa tvořená, jejich pozice. Nastavení velikosti a formátu popisů značek os. Rozsahy jsou ve výchozím stavu nastaveny ze známých rozsahů z analýzy dat.
- `HeaderPropertyGroup` — Vlastnosti související s titulkem grafu. Obsahuje vlastnosti pro přizpůsobení velikosti textu a samotný text titulku.

- `LabelPropertyGroup` — Zda-li jsou popisy os tvořeny z dat. Pokud uživatel nechce tvořit popisky z dat, jsou zde definovány vlastnosti pro popisy jednotlivých os.
- `ColorPropertyGroup` — Vlastnosti pro nastavení barevného schématu grafu. Konkrétně barevný odstín, typ zbarvení a zda bude vytvořen shader nebo využit materiál bez shaderu.
- `AnimationPropertyGroup` — Vlastnosti týkající se animací. Specifikace vzdálenosti mezi keyframy.
- `LegendPropertyGroup` — Pozice legendy a velikost jednotlivých prvků legendy.

Rozdělení do skupin poskytuje prostor pro doplnění dalších vlastností specifikujících nové rozšiřující parametry a nebo pro přidání dalších skupin pro jiné typy vizualizací.

4.3.2 Generický operátor

Implementace generického operátoru umožňuje zapouzdřit společné operace a chování pro všechny grafy do jedné třídy. Generický operátor pro tvorbu grafu je implementován třídou `GenericChart` nacházející se v souboru `general.py`. Operátory pro vytvoření jednotlivých typů grafů se nachází ve složce `operators/`. Diagram tříd znázorňující hierarchii operátorů je na obr. 4.1.



Obrázek 4.1: Třídní diagram operátorů vytvářejících grafy

Aby bylo možné objekty grafů rozlišit, tak každý objekt má proměnnou obsahující unikátní číslo (id). Pro přístup k datům je při inicializaci instance operátoru je vytvořen odkaz na singleton objekt třídy `DataManager`. Také jsou inicializovány data do instančních proměnných pomocí metody `init_data`. K dispozici jsou také metody pro vytvoření titulku, vytvoření kontejneru, vybrání kontejneru a další.

Zobrazení uživatelského rozhraní

Generický operátor implementuje logiku zobrazování uživatelského rozhraní pro přizpůsobení grafů. Pro zobrazení skupiny vlastností stačí v podtřídě tuto skupinu definovat třídní proměnnou se správným jménem a anotací. Uživatelské rozhraní představuje okno s vlastnostmi (props dialog), které je v generickém operátoru vyvoláno v metodě `invoke` funkcí `window_manager.invoke_props_dialog`.

Zobrazené vlastnosti a jejich rozložení je definováno v metodě `draw`. V této metodě je definováno uživatelské rozhraní společné pro všechny grafy a následně jsou volány funkce vykreslující uživatelské rozhraní reprezentující vlastnosti jednotlivých skupin vlastností. Pokud je definována a správně pojmenovaná třídní proměnná anotovaná jako specifická

skupina vlastností, tak je pro ni vytvořena část uživatelského rozhraní. Na definice třídních proměnných je dotazováno funkcí `hasattr`.

Pokud konkrétní graf také rozšiřuje uživatelské rozhraní a definuje metodu `draw` je nutné zavolat rodičovskou metodu pomocí `super().draw` a následně specifikovat rozšiřující vlastnosti pro zobrazení.

4.3.3 Osy

Vytvoření objektů os a práce s nimi je zapouzdřena v souboru `axis.py`.

K vytvoření os slouží třída `AxisFactory` implementována podle návrhového vzoru továrny. Ve třídě je implementována statická metoda `create`, která na základě různých kombinací parametrů (počet dimenzí, popisky, rozsahy) rozhodne jakým způsobem vytvořit objekty osy z třídy `Axis`.

Třída `Axis` představuje abstrakci jedné osy. Voláním metody `create` je vytvořen kontejner, do kterého jsou vytvořeny kvádry představující čáru osy, kvádry představující značky na ose a textové objekty popisů.

4.3.4 Legenda

Legendu je možné vytvořit instancí třídy `Legend` nacházející se v souboru `legend.py`. Implementace legendy je vhodná pouze pro kategorická data. Legenda je vytvořena pomocí ploch obarvených barvami kategorií z grafu a k nim umístěným textům s popisem dané kategorie.

Jednotlivé položky legendy mají přizpůsobitelnou velikost a aby si legenda zachovala stejnou výšku jako graf, tak jsou položky automaticky zarovnávány dle jejich šířky a výšky.

Je přichystána i možnost vytvoření barevné mapy popisující hodnoty barev pro numerická data, ale není zatím používaná.

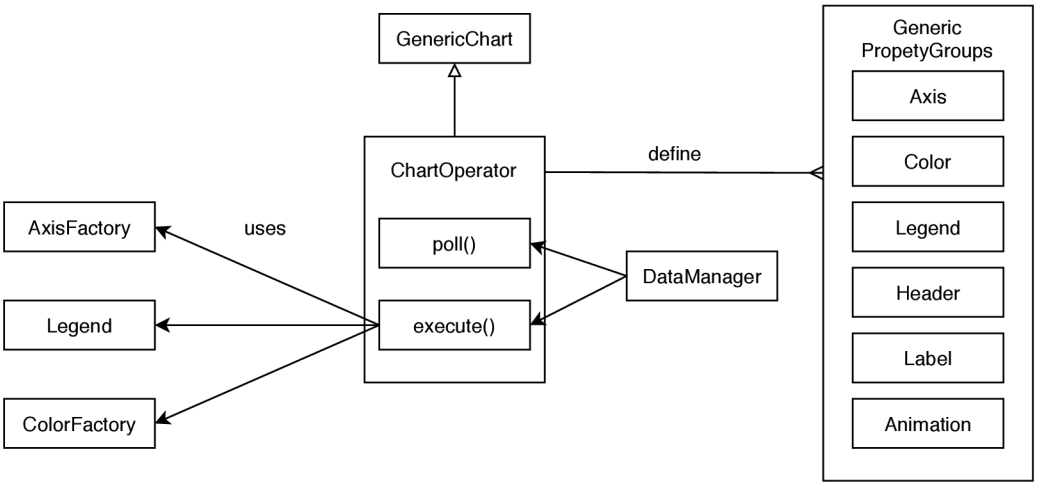
4.3.5 Animace

Pokud uživatel nahraje data, která lze animovat, je mu u grafů podporujících animace zobrazeno uživatelské rozhraní skupiny vlastností týkající se animací. Animace jsou podporovány u povrchového, sloupcového a bodového grafu. Pokud uživatel chce graf animovat, je získán aktuální snímek ve scéně a se zadaným rozestupem jsou vytvořeny snímky nebo shape keys reprezentující další hodnoty v datech.

4.3.6 Implementace operátoru grafu

Blokové schéma popisující obecnou implementaci operátoru je na obrázku 4.2. Operátor pro tvorbu grafu může definovat a využívat generické skupiny vlastností. V metodě `poll` je dotázano, zda jsou k dispozici správná data a je možné graf povolit. V metodě `execute` jsou inicializovány využívané komponenty a pro každý řádek v datech vytvořen objekt reprezentující data, na nějž je aplikována barva, případně jsou tyto objekty animovány. Jako poslední jsou vytvořeny osy nebo legenda a je vybrán kontejner grafu.

V rámci implementace generického operátoru je využíváno pomocných funkcí z modulů `data_utils` a `color_utils`.



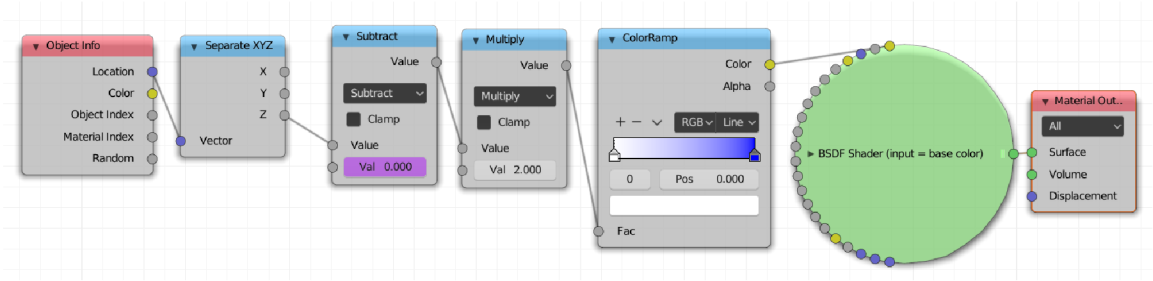
Obrázek 4.2: Blokové schéma reprezentující implementaci
`ChartOperator` je libovolnou podtřídou generického operátoru. Informace pro povolení grafu (v metodě `poll`) a data pro tvorbu vizualizace jsou získávány ze třídy `DataManager`. V operátoru mohou být definovány skupiny vlastností, dle kterých se následně řídí v metodě `execute` tvorba vizualizace. V případě definice skupiny vlastností pro osy, legendu nebo barvu je vytvoření objektů nebo materiálů abstrahováno do tříd nacházejících se na schématu vlevo.

4.4 Barvy

Obarvení grafu je implementováno v souboru `colors.py`. Práce s barvami je zapouzdřena třídou `ColorFactory`, která podobně jako u os dle zadaných parametrů vytvoří objekt pro generování materiálů pro každou hodnotu a nebo node shader. Materiály jsou také vytvořeny pro texty, osy a legendu, aby si uživatel mohl přizpůsobit jejich barvu.

4.4.1 Node Shader

Node shader pro obarvení objektů nebo geometrie grafu je implementován třídou `NodeShader`. Vlastnosti materiálu jsou určeny při volání konstruktoru, kdy je vytvořen samotný strom bloků popisujících shader. Pomocí bpy rozhraní pro práci se stromem bloků jsou vytvořeny a propojeny jednotlivé bloky, které jsou rozmístěny s dostatečným rozestupem, aby byly přehledně zobrazeny v uživatelském rozhraní. Výsledný strom vytvořený pomocí kódu lze vidět na obrázku 4.3. Voláním metody `get_material` je následně získán materiál používající shader nodes pro obarvení grafu.



Obrázek 4.3: Node Shader pro obarvení grafů vytvořený pomocí kódu

4.4.2 Materiály

Tvorba materiálů bez využití shaderu je ve třídě `ColorGen`. Zadáním parametrů při volání konstruktoru této třídy je vytvořen základní materiál. Pokud se jedná o obarvení gradientem, tak je vždy pro každý další objekt při volání metody `get_material` vytvořen nový materiál s barvou proporčně odpovídající hodnotě. Pro náhodné obarvení je vygenerovaný náhodně odstín barvy využitím modulu `random` a každé volání funkce `get_material` také vytvoří materiál. Při konstantním zbarvení je využíván pouze jeden materiál.

4.5 Publikování add-onu

Pro získání zpětné vazby a vylepšení add-onu na základě uživatelské zkušenosti bylo nutné add-on zveřejnit mezi komunitu. K zaujmutí uživatelů byly vytvořeny rendery vizualizací a ilustrační video. Akvizice uživatelů probíhala dvěma způsoby popsanými níže.

4.5.1 Open-source

Zdrojový kód projektu byl po celou dobu vývoje udržován na platformě GitHub[3]. S postupnou implementací nových vlastností byla vytvářena a zveřejňována nová vydání. Na GitHubu se nachází soubor *readme* sloužící jako uživatelská příručka obsahující užitečné odkazy, postup instalace addonu, příklady použití, akceptované datové formáty a aktuální stav add-onu. Detailnější dokumentace uživatelského rozhraní a parametrů je popsána na samostatné wiki stránce. Společně s odkazem na zdrojový kód byl add-on prezentován také na komunitních fórech a stránkách na sociálních sítích.

4.5.2 Blender Market

Add-on bylo původně zamýšleno publikovat pouze zdarma. Obchod BlenderMarket poskytoval rozsáhlou možnost zviditelnění projektu. Add-ony lze publikovat pouze jako placené s možností příspěvku organizaci vyvíjející Blender. Zvolena byla nízká cena a k zachování open-source přístupu je na titulní stránce uveden odkaz na GitHub. K zveřejnění add-onu bylo nutné si zažádat o účet tvůrce a následně vytvořit prezentační stránku s popisem a krátkou dokumentací[4].

[3] GitHub add-onu: https://github.com/Griperis/BlenderDataVis/
[4] Prezentační stránka dostupná na: https://blendermarket.com/products/data-visualisation-addon

Kapitola 5

Uživatelský průzkum

Kapitola zabývající se metrikami vyhodnocenými při testování uživateli po publikování add-onu pro veřejnost. Pro získání konzistentní a měřitelné zpětné vazby od více uživatelů byl vytvořen dotazník s průzkumem, testovacím scénářem a hodnocením add-onu.

5.1 Dotazník

Platformou pro dotazník byly zvoleny Google Forms[1]. Dotazník byl rozdělen do tří hlavních částí popsaných v podsekcích.

5.1.1 Informace o uživateli

První část slouží k získání informací o uživateli. Jelikož je add-on tvořen pro Blender, tak je důležité zjistit, jak znalý je uživatel s jeho používáním. Následují obecné otázky ohledně vizualizace dat, jak často uživatel vizualizuje data, jestli má zkušenosti s 3D vizualizacemi a jaký software již využíval k vizualizaci dat.

5.1.2 Testovací scénář

Testovací scénář spočívá v instalaci add-onu, stažení připravené datové sady a splnění úkolů stupňující se obtížnosti. Před plněním úkolů je testovanému uživateli doporučeno prostudovat soubor readme. Schéma testovacího scénáře:

1. Načtení dat a vytvoření vizualizace.
2. Kombinace dvou spojnicových grafů. Načtení dat, vytvoření spojnicového grafu s upraveným rozsahem a následné vytvoření druhého spojnicového grafu, z jiných dat, se stejným rozsahem, bez objektů os a popisů.
3. Přizpůsobení grafu. Načtení dat, vizualizace bodovým grafem s vlastním nastavením (kromě barev). Přizpůsobení barev v editoru shaderu, změna barev os.
4. Pokročilý úkol. Instalace závislostí v nastavení. Načtení obsáhlého souboru a její vizualizace jako povrchový graf.
5. Vizualizace vlastních dat.

[1]Dotazník dostupný na: https://forms.gle/HJ25HzZBvN4cmzmy9

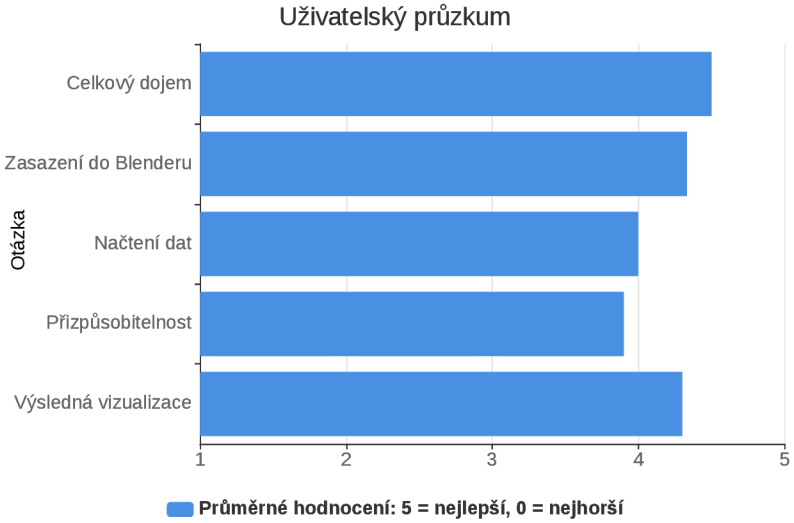
První úkol slouží primárně k seznámení uživatele s add-onem a poskytnutí ukázky, jak se add-on používá. Druhý úkol ukáže uživateli možnosti kombinace grafů a parametry pro nastavení os. Třetí úkol představí uživateli možnosti obarvení. Pokročilý úkol testuje kompatibilitu implementace a zároveň uživatelskou přívětivost instalace knihoven. Posledním úkolem je add-on zasazen do netestované situace, kdy uživatel sám otestuje intuitivnost používání na vlastních datech, bez předem připraveného postupu. V každém úkolu je uživateli zároveň představen jeden z podporovaných typů grafu.

5.1.3 Evaluace

V evaluační části dotazníku byly položeny obecné otázky ohledně práce s add-onem a konkrétní otázky ovlivňující následnou implementaci. Dotazováno bylo na proces načtení dat, vytvoření vizualizace, vzhled výsledné vizualizace, možnosti přizpůsobení, složitost používání a celkový dojem. Otázky ovlivňující následnou implementaci se týkaly rychlosti add-onu a zda by uživatel uvítal materiály pro osy a text unikátní pro každý graf. Poslední odpověď byla volitelná textová, aby uživatel mohl vyplnit své nápady a nebo připomínky.

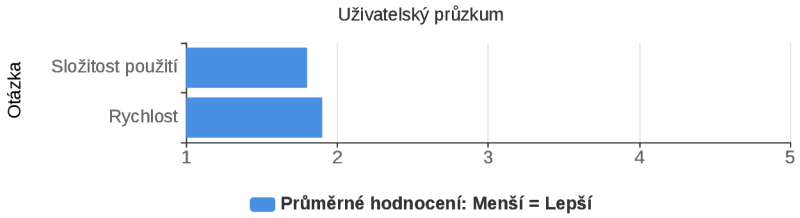
5.2 Výsledky

Výsledky uživatelského průzkumu jsou zobrazeny grafy na obrázcích 5.1 a 5.2. Většina uživatelů (77.8%) by uvítala, kdyby byly vytvořeny materiály pro osy a text unikátně pro každý graf, což bylo následně implementováno.



Obrázek 5.1: Uživatelské hodnocení add-onu
Zprůměrované hodnoty výsledků dotazníku dle jednotlivých sekcí.

Z výsledků byl vynechán uživatel, který z důvodu absence dokumentace v beta-verzi vyplnil všechny hodnoty záměrně jako nejhorší a jeho odpovědi se razantně lišily od průměrných odpovědí ostatních uživatelů. Toto bylo korigováno vytvořením uživatelské příručky pro další verzi. V uživatelské příručce jsou popsány jednotlivé části uživatelského rozhraní a práce s nimi. Detailně popsány jsou také jednotlivé skupiny vlastností a význam jednotlivých vlastností každé skupiny. Příručka byla zveřejněna na wiki stránce add-onu.



Obrázek 5.2: Uživatelské hodnocení složitosti a rychlosti add-onu
Průměrná hodnota odpovědí na otázky ohledně složitosti použití add-onu a zda-li jeho rychlost ovlivnila uživatelovu práci.

5.3 Jiná zpětná vazba

Zdrojem zpětné vazby byla mimo zveřejněný dotazník také fóra, BlenderMarket nebo GitHub. Zpětná vazba mimo publikovaný dotazník objevila několik chyb v implementaci a díky názorům uživatelů na fórech bylo zjištěno, co by se dalo zlepšit. Přidané vylepšení jsou popsány v následující podsekci.

5.3.1 Vylepšení

Jeden uživatel přidal návrh na vylepšení na GitHub stránce s add-onem. Problém byl rozumně popsán a s odkazem na článek. Konkrétně se jednalo o problém zaplnění bočního panelu u uživatelů, kteří používají velké množství add-onů. Uživatelské rozhraní se stává poté nepřehledným. Problém byl adresován umístěním možností pro nastavení pozice panelu add-onu, jenž byl předtím fixně v bočním panelu. Přizpůsobení pozice je umožněno zadáním vlastních hodnot do třídních proměnných určujících pozici panelu skrze nastavení add-onu v preferencích Blenderu.

Díky uživateli, který zkoušel add-on na unixové platformě, byl odhalen problém při instalaci závislostí pro povrchový graf. Problém byl vyřešen vylepšením implementace instalace závislostí, která již následně byla otestována v virtuálním unixovém prostředí.

Uživatelé poskytli návrhy i pro další možná rozšíření, jako například podporu jiných formátů dat, načítání dat z webového API a nebo podporu jiných typů grafů. Všechny zajímavé a realizovatelné nápady jsou uchovány na projektové stránce add-onu pro případné rozšíření.

Kapitola 6

Závěr

Cílem práce bylo vytvořit add-on pro vizualizaci dat do programu pro 3D počítačovou grafiku Blender. Při tvorbě práce neexistovalo pro Blender žádné jiné řešení, jenž by poskytovalo uživateli prostředek pro načtení dat a tvorbu grafů.

K řešení práce byla nejprve nastudována teorie vizualizace dat, z čeho se grafy skládají, a že existuje celá řada různých typů grafů vhodných pro různé účely. Následně byl proveden návrh uživatelského rozhraní a práce s add-onem. Poté byla postupně navrhována a implementována řešení pro vytvoření jednotlivých grafů ve 3D scéně. Pro splnění cílů práce bylo průběžně nutné nabýt co nejvíce zkušeností s Blenderem a jeho Python API. Základních zkušeností bylo nabyto experimentováním s Blenderem pomocí uživatelského rozhraní a následnou reprodukcí funkcionality pomocí Pythonu. Rozsáhlost a komplexnost API zapříčinila několikanásobné předělání celé struktury programu, protože vždy byla objevena nějaká vlastnost, která i skrze důkladné prostudování dokumentace byla přehlédnuta nebo využita špatným způsobem.

Výsledkem práce je na několika internetových platformách zveřejněný open-source nástroj pro vizualizaci grafů. Add-on si koupilo nebo stáhlo přes 40 uživatelů. Kvalita a použitelnost add-onu byla zhodnocena uživatelským průzkumem, který ukázal, že se add-on uživatelům líbí a jsou za něj rádi. Uživatelský průzkum ukázal, že i uživatelé skrze všechny úrovně zkušeností s Blenderem a zkušenosti s různými vizualizačními nástroji dokáži add-on intuitivně použít pro vizualizaci svých dat. Příklady vizualizací vytvořených add-onem se nachází v příloze C.

Snahou bylo add-on vytvořit rozšiřitelně, aby se dobře vyvíjel i následně mimo tuto práci. Mezi hlavní možnosti rozšíření patří přidání dalších typů podporovaných grafů, přidání funkce, jenž natočí popisky grafu na vybranou kameru pro renderování, vizualizace grafů matematických funkcí, další podporované formáty dat nebo načtení více datových sad a následný výběr mezi nimi.

Literatura

[1] *Adobe Flex 3 Advanced Data Visualization Developer Guide* [online]. 2008 [cit. 2020-02-16]. Dostupné z: http://few.vu.nl/~eliens/assets/flex3/datavis_flex3.pdf.

[2] *Blender 2.81 Documentation* [online]. 2020 [cit. 2020-01-10]. Dostupné z: https://docs.blender.org/.

[3] Burget, R. *Informační systémy* [Materiál k přednášce]. FIT VUT Brno.

[4] Elliot, A. J. a Maier, M. A. Color Psychology: Effects of Perceiving Color on Psychological Functioning in Humans. *Annual Review of Psychology*. 2014, sv. 65, č. 1, s. 95–120. DOI: 10.1146/annurev-psych-010213-115035. PMID: 23808916.

[5] Harvey, G. *Excel 2019 all-in-one for dummies*. 1st. Hoboken, New Jersey: John Wiley & Sons, Inc., [2019]. ISBN 978-1119517948.

[6] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*. IEEE COMPUTER SOC. 2007, sv. 9, č. 3, s. 90–95. DOI: 10.1109/MCSE.2007.55.

[7] Khan, M. a Shah, S. Data and Information Visualization Methods, and Interactive Mechanisms: A Survey. *International Journal of Computer Applications*. Prosinec 2011, sv. 34, s. 1–14.

[8] Kirk, A. *Data visualisation: a handbook for data driven design*. 2nd Edition. Los Angeles: Sage Publications, 2016. ISBN 978-1-5264-6892-5.

[9] Koffka, K. *Principles Of Gestalt Psychology*. Mimesis International, 2014. ISBN 978-8857523934.

[10] MATLAB. *Documentation of version 9.8.0 (2020a)*. Natick, Massachusetts: The MathWorks Inc., 2020.

[11] Microsoft. *Available chart types in Office* [online]. 2020 [cit. 2020-04-10]. Dostupné z: https://support.office.com/en-us/article/available-chart-types-in-office-a6187218-807e-4103-9e0a-27cdb19afb90.

[12] Nasution, M., Aulia, I. a Elveny, M. Data. In:. Institute of Physics Publishing, 2019, sv. 1235, č. 1. ISSN 17426588.

[13] Přibyl, O. a Přikryl, J. *Úvod do analýzy dat* [Materiál k přednášce]. FD ČVUT Praha [cit. 2020-03-15]. Dostupné z: https://zolotarev.fd.cvut.cz/static/mamy/mamy-2016-03-slides.pdf.

[14] Schroeder, W., Martin, K. a Lorensen, B. *The Visualization Toolkit–An Object-Oriented Approach To 3D Graphics*. Fourth. Kitware, Inc., 2006. ISBN 978-1-930934-19-1.

[15] Sochor, J., Beneš, B., Felkel, P. a Žára, J. *Vizualizace*. 1. Praha: FEL ČVUT Praha, 1997. ISBN 80-01-01582-3.

[16] Tableau. *Data visualization beginner's guide: a definition, examples, and learning resources* [online]. 2019 [cit. 2019-12-26]. Dostupné z: https://www.tableau.com/learn/articles/data-visualization.

[17] Tufte, E. R. *The Visual Display of Quantitative Information*. USA: Graphics Press, 1986. ISBN 978-0961392109.

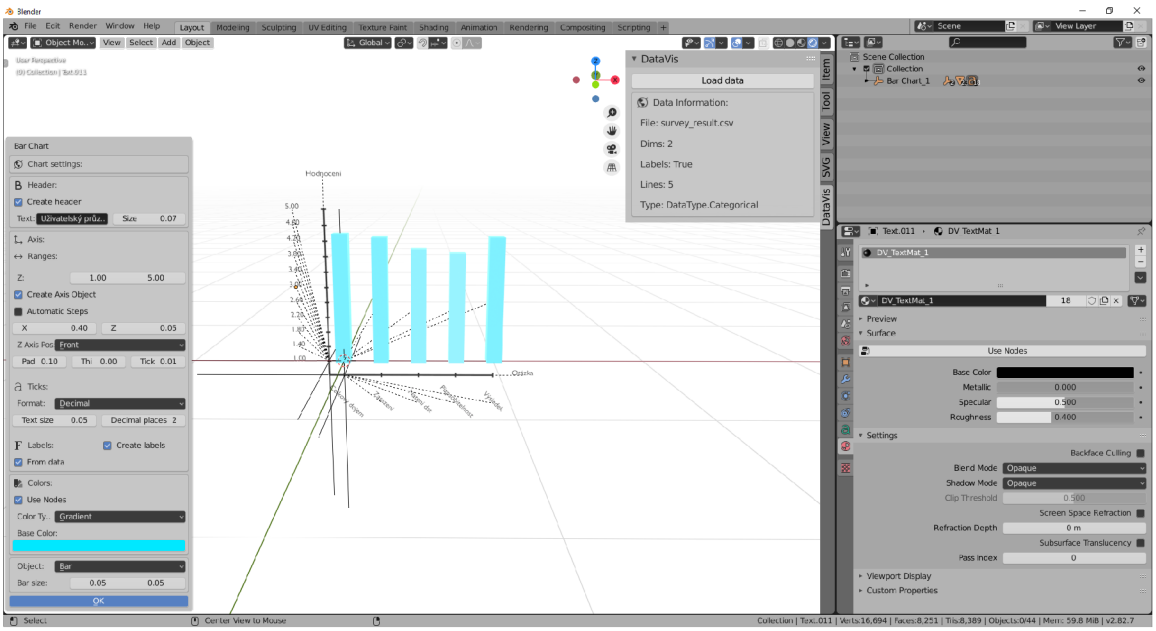
Příloha A

Obsah paměťového média

- **latex/** — zdrojový kód textu v LaTeX
- **src/** — zdrojový kód add-onu
- **data_vis_1.3.4.zip** — archív pro instalaci do programu Blender
- **README.txt** — obsah média, užitečné odkazy
- **xdolez82_bp.pdf** — text práce
- **xdolez82_video.mkv** — demonstrační video

Příloha B

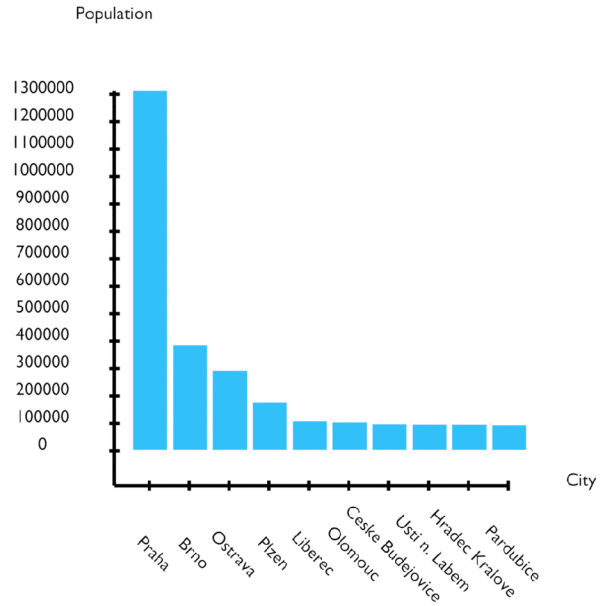
Uživatelské rozhraní



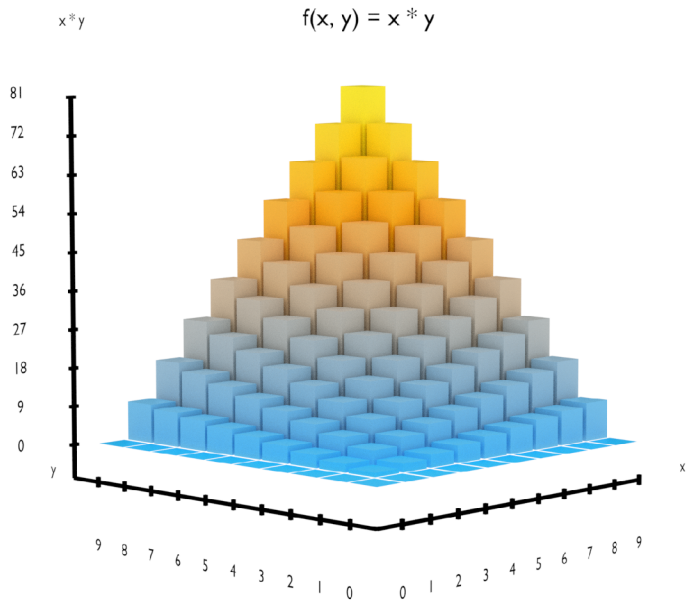
Obrázek B.1: Výsledná verze uživatelského rozhraní pro načítání dat a tvorbu grafů

Příloha C

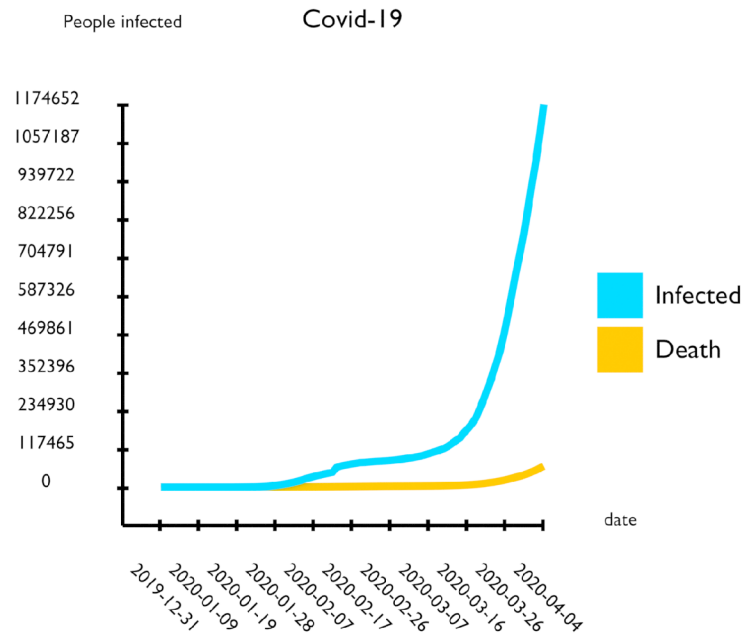
Rendery vizualizací vytvořených pomocí add-onu



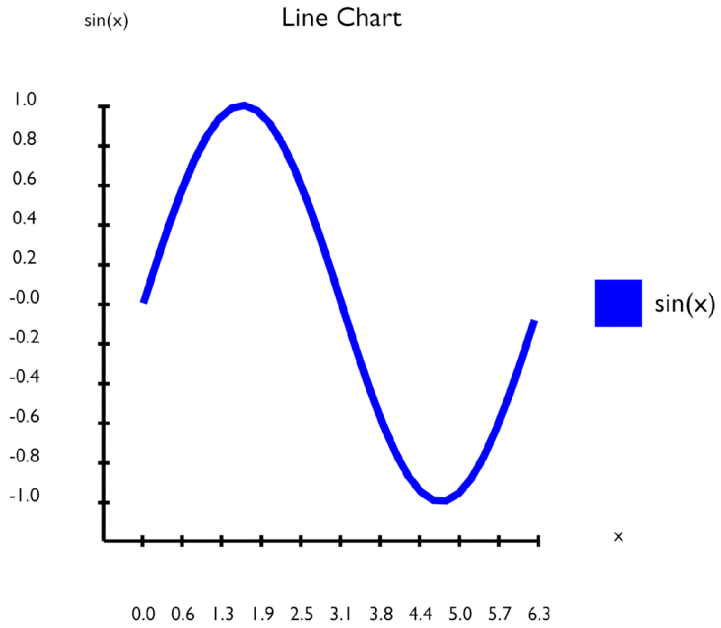
Obrázek C.1: Sloupcový graf zobrazující populaci v největších českých městech



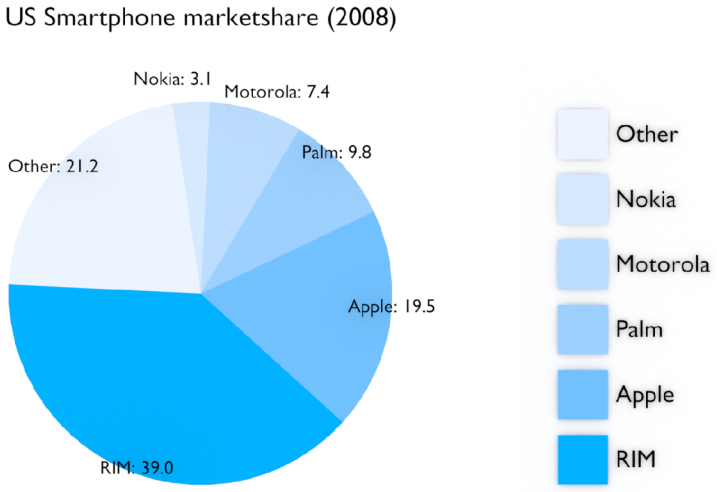
Obrázek C.2: Násobení celých čísel jako 3D sloupcový graf



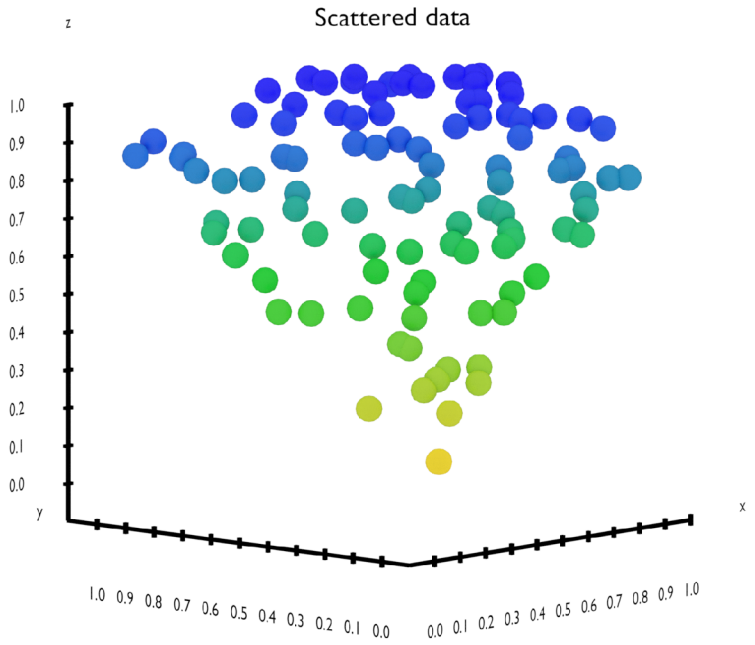
Obrázek C.3: Spojnicový graf s dvěma řadami, jenž popisují vývoj nemoci Covid-19



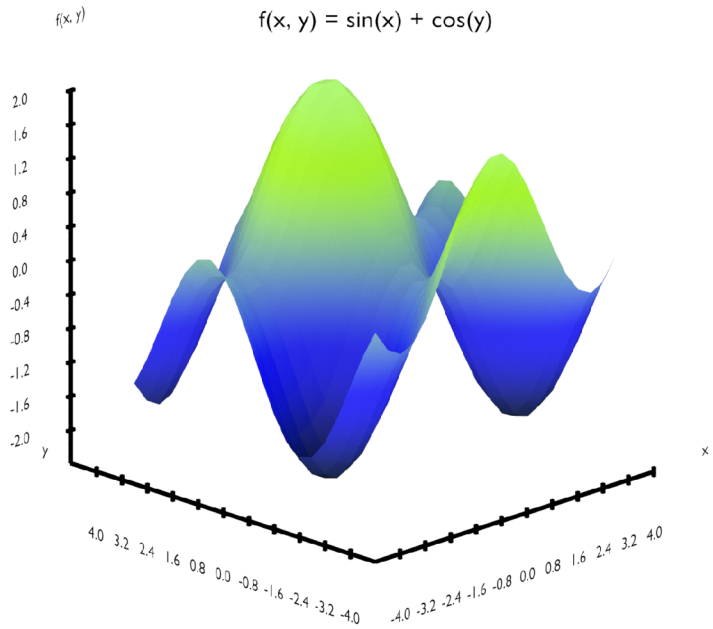
Obrázek C.4: Funkce sinus znázorněna pomocí spojnicového grafu



Obrázek C.5: Výsečový graf s legendou znázorňující podíl na trhu telefonů v roce 2008



Obrázek C.6: Bodový graf prezentující roztroušená data ve 3D



Obrázek C.7: Funkce dvou proměnných vizualizována povrchovým grafem