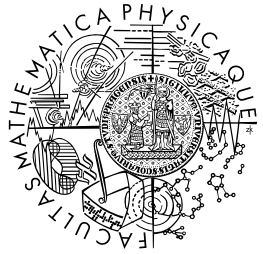


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Oleg Petruny

Vícežánrová příběhová počítačová hra s podporou načítání nového obsahu za běhu

Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: Mgr. Martin Mirbauer
Studijní program: Informatika (B0613A140006)
Studijní obor: IPP6 (0613RA1400060010)

Praha 2025

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Rozhodně děkuji svému vedoucímu Mgr. Martinu Mirbauerovi, za jeho čas, ochotu, trpělivost a cenné rady. K tomu i spoustě dalších lidí zapsaných v titulkach hry, s pomoci, testováním a podporou. Bez všech těchto lidí by táto práce nemohla vzniknout nebo by nenabyla takového rozsahu či kvality.

Název práce: Vícežánrová příběhová počítačová hra s podporou načítání nového obsahu za běhu

Autor: Oleg Petruny

Katedra: Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: Mgr. Martin Mirbauer, Katedra softwaru a výuky informatiky

Abstrakt: V dějinách herního průmyslu je opravdu malé množství her, schopných načítat obsah za běhu, natož když hry jsou s příběhem a určené pro jednoho hráče. Tato práce poskytuje vzorovou implementaci obsahlé hry v Uneral Engine, 3D modelů a dynamického soundtracku. Výstupem je dohratelný celistvý příběh probíhající v pět žánrově a dynamicky odlišných úrovních. Projekt zavádí rozhraní pro načítání nového obsahu za běhu, systémy dialogů, cutscén, quick time eventů a objektové interakce. Spolu s tím jsou podrobně řešeny problémy a úskalí při tvorbě hry a grafiky v Unreal Engine 5.

Klíčová slova: Počítačová hra Unreal Engine dynamické načítání obsahu 3D grafika

Title: Multi-genre game with support for loading new content in real-time

Author: Oleg Petruny

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Martin Mirbauer, Department of Software and Computer Science Education

Abstract: In the history of the gaming industry, there are really few games capable of loading content on the fly, let alone when the games are story-driven and designed for a single player. This thesis provides an example implementation of a comprehensive game in Unreal Engine, 3D models, and a dynamic soundtrack. The output is a playable, coherent story pieced into five genre and dynamic different levels. The project introduces an interface for loading new content on the fly, systems of dialogues, cutscenes, quick time events, and object interaction. Along with this, in detail are addressed problems and pitfalls in creating a game and graphics in Unreal Engine 5.

Keywords: Computer game Unreal Engine dynamic content loading 3D graphics

Obsah

Úvod

Běžně příběhové hry jsou navrhnuté předem a zůstávají neměnné. Existují i hry výjimky například žánru rogue-like, kde ale stále generování obsahu je založeno na použití speciálně vytvořeném a odladěnem algoritmu. Tento algoritmus různé složitosti stále používa pouze prostředky předem založené a odladěné vývojáři. To vše má podstatný důvod, jelikož vývojáři mají za úkol tzv. postavit správný herní zážitek a zaručit, že hra bude umělecky správně interpretovaná hráčem. Bohužel nezávisle na kvalitě výsledné hry, počet přehrání hry v nejlepším případě klesá nebo v tom horším je pouze jeden.

Celý problém v branži příběhových her je řešen tzv. módy neboli modifikacemi hry a to různých druhů jako například grafické, kódové a další. Samotné vývojáři poskytují potřebné prostředky pro podporu nebo tvorbu modů. Dokonce i technické zdatné hráče dokážou oblíbené hry úpravit, aby ony jakousi podporu měli. V jakémkoliv případě modifikace značně zvětšují počet přehrání her, kde klasickým příkladem je hra The Elder Scrolls: Skyrim od studia Bethesda.

Přestože celé řešení zdánlivě funguje, spoléhá se na ochotu samotných hráčů módy vytvářet. Jestliže hra nebude mít dostatečnou fanouškovskou bázi, potom taktéž nebude mít dostatek nového fanouškovského obsahu a výsledně počet přehrání neroste. Klasickým příkladem je hra Starfield od již zmíněného studia Bethesda. Navíc ke spoustě technických omezení samotných módů, hráči musí mít nejen ochotu, ale i být technický a umělecky zdatní, aby něco vytvořili. Minimálně musí přidat úsilí k manuálnímu vyhledání, stažení a instalaci modů.

Tato práce se zaměřuje na tvorbu hry a sytemu, který umožní výše popsaný lidský faktor a nedostatky eliminovat. Přináší tak příběh rozdělený na 5 žánrově odlišných úrovní a zavádí high-level API pro Unreal Engine na stahování a načítání obsahu do hry přímo za běhu. Specifický příběh snižuje ludonarativní disonanci při vzniku nového obsahu ve hře a více žánru umožňuje otestovat toto řešení, jestli v každém z nich dostatečně funguje. Zároveň táto práce přenechává samotné generování obsahu pomoci AI modelů, práci s API a testování výsledků až pro další práci. Čili primárně se zaměřuje na kostru samotné hry, aby potom byl prostor kam nový obsah začlenit.

Hlavními tématy na které se práce zaměřuje jsou:

- Práce s Unreal Engine, jeho reálná omezení, obcházení/vyrovnání se s těmito omezení a tipy.
- Postupy tvorby různého druhu grafiky pro 3D hry zejména na Unreal Engine.
- Postupy tvorby audia pro hry.
- Ukázkové příklady tvorby herních systému a mechanik pro Unreal Engine.
- Tvorba generativního obsahu a jeho načítaní na Unreal Engine.

Kapitola 1

Important first chapter

First chapter usually builds the theoretical background necessary for readers to understand the rest of the thesis. You should summarize and reference a lot of existing literature and research.

You should use the standard *citations*.

Use \emph command like this, to highlight the first occurrence of an important word or term. Reader will notice it, and hopefully remember the importance.

This footnote is an acceptable way to 'cite' webpages or URLs. Documents without proper titles, authors and publishers generally do not form citations. For this reason, avoid citations of wikipedia pages.

Obtaining bibTeX citation Go to Google Scholar[1], find the relevant literature, click the tiny double-quote button below the link, and copy the bibTeX entry.

Saving the citation Insert the bibTeX entry to the file `refs.bib`. On the first line of the entry you should see the short reference name — from Scholar, it usually looks like `author2015title` — you will use that to refer to the citation.

Using the citation Use the `\cite` command to typeset the citation number correctly in the text; a long citation description will be automatically added to the bibliography at the end of the thesis. Always use a non-breakable space before the citing parenthesis to avoid unacceptable line breaks:

```
Trees utilize gravity to invade ye
noble sires~\cite{newton1666apple}.
```

Why should I bother with citations at all? For two main reasons:

- You do not have to explain everything in the thesis; instead you send the reader to refer to details in some other literature. Use citations to simplify the detailed explanations.

[1] `https://scholar.google.com`

- If you describe something that already exists without using a citation, the reviewer may think that you *claim* to have invented it. Expectably, they will demand academic correctness, and, from your perspective, being accused of plagiarism is not a good starting point for a successful defense. Use citations to identify the people who invented the ideas that you build upon.

How many citations should I use? Cite any non-trivial building block or assumption that you use, if it is published in the literature. You do not have to cite trivia, such as the basic definitions taught in the introductory courses.

The rule of thumb is that you should read, understand and briefly review at least around 4 scientific papers. A thesis that contains less than 3 sound citations will spark doubt in reviewers.

There are several main commands for inserting citations, used as follows:

- Knuth [1] described a great system for typesetting theses.
- We are typesetting this thesis with LaTeX, which is based on TeX and METAFONT [1].
- TeX was expanded to LaTeX by Lamport [2], hence the name.
- Revered are the authors of these systems! [1, 2]

1.1 Some extra assorted hints before you start writing English

Word order Strictly adhere to the English word order rules. The sentences follow a fixed structure with a subject followed by a verb and an object (in this order). Exceptions to this rule must be handled specially, and usually separated by commas.

Sentence structure Do not write long sentences. One sentence should contain exactly one fact. Multiple facts should be grouped in a paragraph to communicate one coherent idea. Both the sentences and paragraphs should include various hints about their relation to the other ideas and paragraps. These are typically materialized as adverbs or short sentence parts that clarify the cause–outcome and target–method–result relationship of the sentences in a paragraph. Such 'word glue' helps the readers to correctly draw the lines that hold their mental

images of your thesis together, and ideally see the big picture of what you were trying to convey right from the first read.

Paragraphs are grouped in labeled sections for a sole purpose of making the navigation in the thesis easier. Do not use the headings as 'names for paragraphs' — the text should make perfect sense even if all headings are removed. If a section of your text contains one paragraph per heading, you might have wanted to write an explicit list instead.

Mind the rules for placing commas:

- Do not use the comma before subordinate clauses that begin with 'that' (like this one). English does not use subordinate clauses as often as Slavic languages because the lack of a suitable word inflection method makes them hard to understand. In scientific English, try to avoid them as much as possible. Ask doubtfully whether each 'which' and 'when' is necessary — most of these helper conjunctions can be removed by converting the clause to non-subordinate.

  As an usual example, *'The sentence, which I wrote, seemed ugly.'* is perfectly bad; slightly improved by *'The sentence that I wrote seemed ugly.'*, which can be easily reduced to *'The sentence I wrote seemed ugly.'*. A final version with added storytelling value could say *'I wrote a sentence but it seemed ugly.'*

- Use the *Oxford comma* before 'and' and 'or' at the end of a longer, comma-separated list of items. Certainly use it to disambiguate any possible mixtures of conjunctions: *'The car is available in red, red and green, and green versions.'* Remember that English 'or' is typically understood more like 'either this or that, but not both,' and the use of 'and' is much more appropriate in cases such as possibility overviews and example listings (like in this sentence).

- Consider placing extra commas around any parts of the sentence that break the usual word order, especially if they are longer than a single word.

Nouns Every noun needs a determiner ('a', 'the', 'my', 'some', ...); the exceptions to this rule, such as non-adjectivized names and indeterminate plural, are relatively scarce. Without a determiner, a noun can be easily mistaken for something completely different, such as an adjective or a verb.

Name all things with appropriate nouns to help both the reader and yourself, and do not hesitate to invent good names and labels for anything that you will refer to more than once. Proper naming will save you a lot of writing effort because you will not have to repeat descriptions such as *'the third output of the second benchmarked method of the improved set,'* instead you may introduce a

labeling that will allow you to say just something like *'output M2⁺-3'*. At the same time, this will reduce the risk that the reader will confuse the object with another one — for illustration, the long version of the previous example might very easily confuse with the second output of the third method. The same also applies to methods descriptions, algorithms, programs, testing datasets, theorems, use-cases, challenges and other things. As an example, *'the algorithm that organizes the potatoes into appropriate buckets'* shortens nicely as *'the potato bucketer'* and may be labeled as a procedure BucketPotatoes(), and *'the issue where the robot crashes into a wall and takes significant time to return to the previous task'* may be called just *'the crash–recovery lag'*.

Verbs Although English can express a whopping 65 base verb tenses and their variants, scientific literature often suppresses this complexity and uses only several basic tenses where the meaning is clearly defined. Typically, you state facts in present simple (*'Theorem 1 proves that Gadget B works as intended.'*), talk about previous work and experiments done in past simple (*'We constructed Gadget B from Gizmo C, which was previously prepared by Tinkerer et al.'*), and identify achieved results in present perfect (*'We have constructed Technology T.'*). Avoid using future tense, except for sections that explicitly describe future work — as a typical mistake, if you state that the thesis *will* describe something in later chapters, you imply that the description is not present there yet.

Do not write sentences in passive voice, unless you explicitly need to highlight that something has passively subjected itself to an action. Active voice is more preferable in the theses because it clearly highlights the actors and their contributions — typically, *'you did it'* instead of *'it was done'* by a mysterious entity, which the reviewers rarely envision as yourself. Writing in active voice additionally benefits the explanation of complex processes: There, the word order forces you to identify the acting subject as the first word in the sentence, which further disambiguates how the individual process parts are triggered and ordered.

Try to avoid overusing gerunds (verbs that end with '-ing'). It is convenient to write shorter sentences by using gerunds as adjectives, but these are typically quite hard to understand because the readers may easily confuse the intended adjectives with verbs. If your sentence contains two gerunds close to each other, it may need a rewrite.

Scientific writing resources Consult the book by Glasman-Deal [3] for more useful details and recommended terminology for writing about the scientific research. Very pragmatically, the book by Sparling [4] describes many common mistakes that Czech and Slovak (and generally Slavic) writers make when writing English.

Kapitola 2

More complicated chapter

After the reader gained sufficient knowledge to understand your problem in kapitola 1, you can jump to your own advanced material and conclusions.

You will need definitions (see definice 1 below in sekce 2.1), theorems (věta 1), general mathematics, algorithms (algoritmus 1), and tables (tabulka 2.1). Obrázky 2.1 a 3.1 show how to make a nice figure. See obrázek 2.2 for an example of TikZ-based diagram. Cross-referencing helps to keep the necessary parts of the narrative close — use references to the previous chapter with theory wherever it seems that the reader could have forgotten the required context. Conversely, it is useful to add a few references to theoretical chapters that point to the sections which use the developed theory, giving the reader easy access to motivating application examples.

See documentation of package `booktabs` for hints on typesetting tables. As a main rule, *never* draw a vertical line.

2.1 Example with some mathematics

Definice 1 (Triplet)**.** *Given stuff X, Y and Z, we will write a* triplet *of the stuff as* (X, Y, Z).

Věta 1 (Car coloring)**.** *All cars have the same color. More specifically, for any set of cars C, we have*

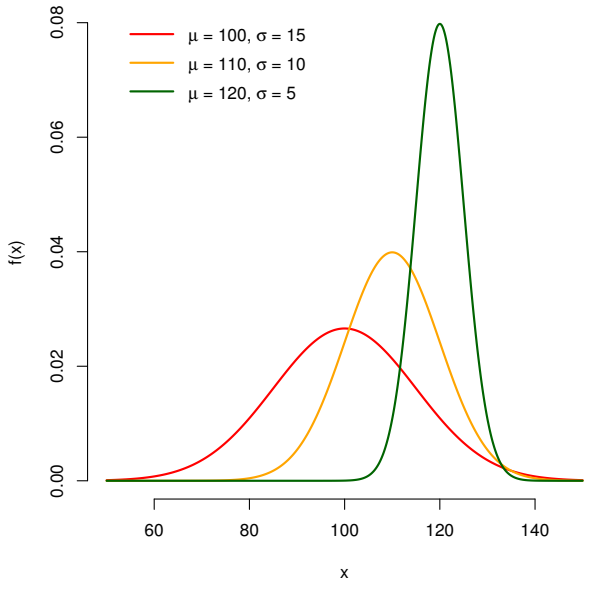
$$(\forall c_1, c_2 \in C)\ \textsc{Colour}(c_1) = \textsc{Colour}(c_2).$$

Důkaz. Use induction on sets of cars C. The statement holds trivially for $|C| \leq 1$. For larger C, select 2 overlapping subsets of C smaller than $|C|$ (thus same-colored). Overlapping cars need to have the same color as the cars outside the overlap, thus also the whole C is same-colored. □

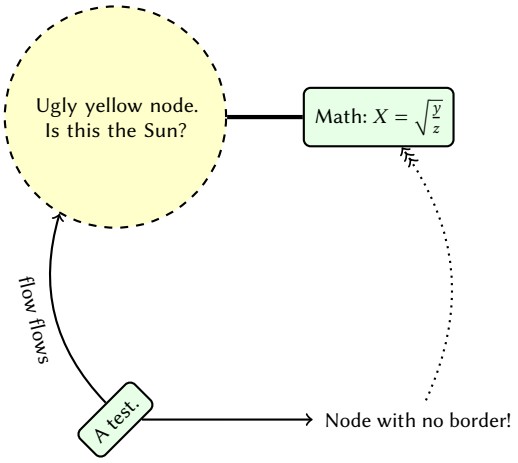
This is plain wrong though.

Column A	Column 2	Numbers	More
Asd	QWERTY	123123	–
Asd qsd 1sd	BAD	234234234	This line should be helpful.
Asd	INTERESTING	123123123	–
Asd qsd 1sd	PLAIN WEIRD	234234234	–
Asd	QWERTY	123123	–
Asd qsd 1sd	GOOD	234234299	–
Asd	NUMBER	**123123**	–
Asd qsd 1sd	DANGEROUS	234234234	(no data)

Tabulka 2.1 An example table. Table caption should clearly explain how to interpret the data in the table. Use some visual guide, such as boldface or color coding, to highlight the most important results (e.g., comparison winners).



Obrázek 2.1 A figure with a plot, not entirely related to anything. If you copy the figures from anywhere, always refer to the original author, ideally by citation (if possible). In particular, this picture — and many others, also a lot of surrounding code — was taken from the example bachelor thesis of MFF, originally created by Martin Mareš and others.



Obrázek 2.2 An example diagram typeset with TikZ. It is a good idea to write diagram captions in a way that guides the reader through the diagram. Explicitly name the object where the diagram viewing should "start". Preferably, briefly summarize the connection to the parts of the text and other diagrams or figures. (In this case, would the tenative yellow Sun be described closer in some section of the thesis? Or, would there be a figure to detail the dotted pattern of the line?)

Algoritmus 1 Algorithm that executes an action with high probability. Do not care about formal semantics in the pseudocode — semicolons, types, correct function call parameters and similar nonsense from 'realistic' languages can be safely omitted. Instead make sure that the intuition behind (and perhaps some hints about its correctness or various corner cases) can be seen as easily as possible.

function ExecuteWithHighProbability(A)
    $r \leftarrow$ a random number between 0 and 1
    $\varepsilon \leftarrow 0.00000000000000000000000000000000000042$
    if $r \geq \varepsilon$ **then**
        execute A ..... *We discard the return value*
    else
        print: `Not today, sorry.`
    end if
end function

2.2 Extra typesetting hints

Do not overuse text formatting for highlighting various important parts of your sentences. If an idea cannot be communicated without formatting, the sentence probably needs rewriting anyway. Imagine the thesis being read aloud as a podcast — the storytellers are generally unable to speak in boldface font.

Most importantly, do not overuse bold text, which is designed to literally **shine from the page** to be the first thing that catches the eye of the reader. More precisely, use bold text only for 'navigation' elements that need to be seen and located first, such as headings, list item leads, and figure numbers.

Use underline only in dire necessity, such as in the previous paragraph where it was inevitable to ensure that the reader remembers to never typeset boldface text manually again.

Use *emphasis* to highlight the first occurrences of important terms that the reader should notice. The feeling the emphasis produces is, roughly, "Oh my — what a nicely slanted word! Surely I expect it be important for the rest of the thesis!"

Finally, never draw a vertical line, not even in a table or around figures, ever. Vertical lines outside of the figures are ugly.

Kapitola 3

Results and discussion

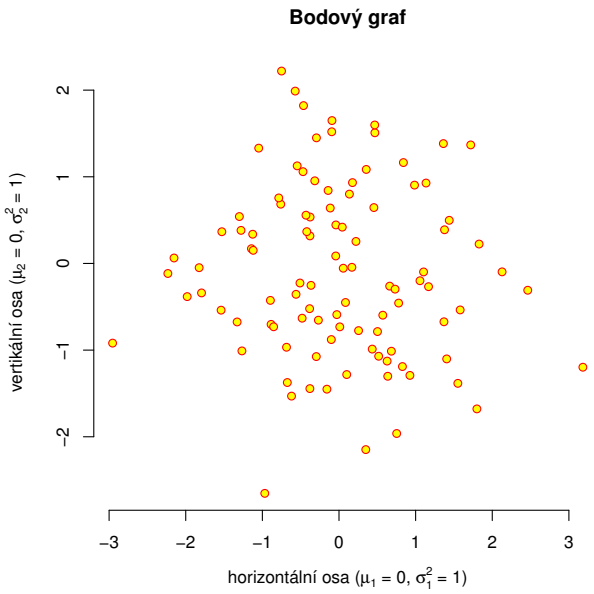
You should have a separate chapter for presenting your results (generated by the stuff described previously, in our case in kapitola 2). Remember that your work needs to be validated rigorously, and no one will believe you if you just say that 'it worked well for you'.

Instead, try some of the following:

- State a hypothesis and prove it statistically
- Show plots with measurements that you did to prove your results (e.g. speedup). Use either `R` and `ggplot`, or Python with `matplotlib` to generate the plots.[1] Save them as PDF to avoid printing pixels (as in obrázek 3.1).
- Compare with other similar software/theses/authors/results, if possible
- Show example source code (e.g. for demonstrating how easily your results can be used)
- Include a 'toy problem' for demonstrating the basic functionality of your approach and detail all important properties and results on that
- Include clear pictures of 'inputs' and 'outputs' of all your algorithms, if applicable

It is sometimes convenient (even recommended by some journals, including Cell) to name the results sub-sections so that they state what exactly has been achieved. Examples follow.

[1] Honestly, the plots from `ggplot` look much better.



Obrázek 3.1 This caption is a friendly reminder to never insert figures "in text," without a floating environment, unless explicitly needed for maintaining the text flow (e.g., the figure is small and developing with the text, like some of the centered equations, as in věta 1). All figures *must* be referenced by number from the text (so that the readers can find them when they read the text) and properly captioned (so that the readers can interpret the figure even if they look at it before reading the text — reviewers love to do that).

3.1 SuperProgram is faster than OldAlgorithm

3.1.1 Scalability estimation

3.1.2 Precision of the results

3.2 Weird theorem is proven by induction

3.3 Amount of code reduced by CodeRedTool

3.3.1 Example

3.3.2 Performance on real codebases

3.4 NeuroticHelper improves neural network learning

3.5 Graphics and figure quality

No matter how great the text content of your thesis is, the pictures will always catch the attention first. This creates the very important first impression of the thesis contents and general quality. Crucially, that also decides whether the thesis is later read with joy, or carefully examined with suspicion.

Preparing your thesis in a way such that this first impression gets communicated smoothly and precisely helps both the reviewer and you: the reviewer will not have a hard time understanding what exactly you wanted to convey, and you will get a better grade.

Making the graphics 'work for you' involves doing some extra work that is often unexpected. At the same time, you will need to fit into graphics quality constraints and guidelines that are rarely understood before you actually see a bad example. As a rule of thumb, you should allocate at least the same amount of time and effort for making the figures look good as you would for writing, editing and correcting the same page area of paragraph text.

3.5.1 Visualize all important ideas

The set of figures in your thesis should be comprehensive and complete. For all important ideas, constructions, complicated setups and results there should be a visualization that the reader can refer to in case the text does not paint the 'mental image' sufficiently well. At the bare minimum, you should have at least 3

figures (roughly corresponding to the 3 chapters) that clearly and unambiguously show:

1. the context of the problem you are solving, optionally with e.g. question marks and exclamation marks placed to highlight the problems and research questions

2. the overall architecture of your solution (usually as a diagram with arrows, such as in obrázek 2.2, ideally with tiny toy examples of the inputs and outputs of each box),

3. the advancement or the distinctive property of your solution, usually in a benchmark plot, or as a clear demonstration and comparison of your results.

3.5.2 Make the figures comprehensible

The figures should be easily comprehensible. Surprisingly, that requires you to follow some common "standards" in figure design and processing. People are often used to a certain form of the visualizations, and (unless you have a very good reason) deviating from the standard is going to make the comprehension much more complicated. The common standards include the following:

- caption everything correctly, place the caption at an expectable position

- systematically label the plots with 'main' titles (usually in boldface, above the plot), plot axes, axis units and ticks, and legends

- lay out the diagrams systematically, ideally follow a structure of a bottom-up tree, a left-to-right pipeline, a top-down layered architecture, or a center-to-borders mindmap

- use colors that convey the required information correctly

  Although many people carry some intuition for color use, achieving a really correct utilization of colors is often very hard without previous experience in color science and typesetting. Always remember that everyone perceives color hues differently, therefore the best distinction between the colors is done by varying lightness of the graphics elements (i.e., separating the data by dark vs. light) rather than by using hues (i.e., forcing people to guess which one of salmon and olive colors means "better"). Almost 10% of the population have their vision impaired by some form of color vision deficiency, most frequently by deuteranomaly that prevents interpretation of even the most 'obvious' hue differences, such as green vs. red. Finally, printed colors look surprisingly different from the on-screen

colors. You can prevent much of these problems by using standardized palettes and well-tested color gradients, such as the ones from ColorBrewer[2] and ViridisLite[3]. Check if your pictures still look good if converted to greyscale, and use a color deficiency simulator to check how the colors are perceived with deuteranomaly.

Avoid large areas of over-saturated and dark colors:

- under no circumstances use dark backgrounds for any graphical elements, such as diagram boxes and tables — use very light, slightly desaturated colors instead
- avoid using figures that contain lots of dark color (as a common example, heatmaps rendered with the 'magma' color palette often look like huge black slabs that are visible even through the paper sheet, thus making a dark smudge on the neighboring page)
- increase the brightness of any photos to match the average brightness of the text around the figure

Remember to test your figures on other people — usually, just asking 'What do you think the figure should show?' can help you debug many mistakes in your graphics. If they think that the figure says something different than what you planned, then most likely it is your figure what is wrong, not the understanding of others.

Finally, there are many magnificent resources that help you arrange your graphics correctly. The two books by Tufte [5, 6] are arguably classics in the area. Additionally, you may find many interesting resources to help you with technical aspects of plotting, such as the `ggplot`-style 'Fundamentals' book by Wilke [7], and a wonderful manual for the TikZ/PGF graphics system by Tantau [8] that will help you draw high-quality diagrams (like the one in obrázek 2.2).

3.6 What is a discussion?

After you present the results and show that your contributions work, it is important to *interpret* them, showing what they mean in the wider context of the thesis topic, for the researchers who work in the area, and for the more general public, such as for the users.

Separate discussion sections are therefore common in life sciences where some ambiguity in result interpretation is common, and the carefully developed intuition about the wider context is sometimes the only thing that the authors

[2] `https://colorbrewer2.org`

[3] `https://sjmgarnier.github.io/viridisLite/`

have. Exact sciences and mathematicians do not need to use the discussion sections as often. Despite of that, it is nice to position your output into the previously existing environment, answering:

- What is the potential application of the result?
- Does the result solve a problem that other people encountered?
- Did the results point to any new (surprising) facts?
- How (and why) is the approach you chose different from what the others have done previously?
- Why is the result important for your future work (or work of anyone other)?
- Can the results be used to replace (and improve) anything that is used currently?

If you do not know the answers, you may want to ask the supervisor. Also, do not worry if the discussion section is half-empty or thoroughly pointless; you may remove it completely without much consequence. It is just a bachelor thesis, not a world-saving avenger thesis.

Conclusion

In the conclusion, you should summarize what was achieved by the thesis. In a few paragraphs, try to answer the following:

- Was the problem stated in the introduction solved? (Ideally include a list of successfully achieved goals.)
- What is the quality of the result? Is the problem solved for good and the mankind does not need to ever think about it again, or just partially improved upon? (Is the incompleteness caused by overwhelming problem complexity that would be out of thesis scope, or any theoretical reasons, such as computational hardness?)
- Does the result have any practical applications that improve upon something realistic?
- Is there any good future development or research direction that could further improve the results of this thesis? (This is often summarized in a separate subsection called 'Future work'.)

This is quite common.

Seznam použité literatury

[1] Donald Ervin Knuth. *TEX and METAFONT: New directions in typesetting*. American Mathematical Society, 1979.

[2] Leslie Lamport. *LATEX: a document preparation system: user's guide and reference manual*. Addison-Wesley, 1994.

[3] Hilary Glasman-Deal. *Science research writing for non-native speakers of English*. World Scientific, 2010.

[4] Don Sparling. *English or Czenglish? Jak se vyhnout čechismům v angličtině*. Státní pedagogické nakladatelství, 1989.

[5] Edward R Tufte, Nora Hillman Goeler a Richard Benson. *Envisioning information*. Graphics press Cheshire, CT, 1990.

[6] Edward R Tufte. *Visual display of quantitative information*. Graphics press Cheshire, CT, 1983.

[7] Claus O Wilke. *Fundamentals of Data Visualization*. O'Reilly Media, Inc., 2019. ISBN: 9781492031086. URL: https://clauswilke.com/dataviz/.

[8] Till Tantau. *The TikZ and PGF Packages (Manual for version 3.1.8b)*. Tech. zpr. Institut für Theoretische Informatik Universität zu Lübeck, 2020. URL: http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf.

Příloha A

Using CoolThesisSoftware

Use this appendix to tell the readers (specifically the reviewer) how to use your software. A very reduced example follows; expand as necessary. Description of the program usage (e.g., how to process some example data) should be included as well.

To compile and run the software, you need dependencies XXX and YYY and a C compiler. On Debian-based Linux systems (such as Ubuntu), you may install these dependencies with APT:

```
apt-get install \
  libsuperdependency-dev \
  libanotherdependency-dev \
  build-essential
```

To unpack and compile the software, proceed as follows:

```
unzip coolsoft.zip
cd coolsoft
./configure
make
```

The program can be used as a C++ library, the simplest use is demonstrated in výpis 1. A demonstration program that processes demonstration data is available in directory `demo/`, you can run the program on a demonstration dataset as follows:

```
cd demo/
./bin/cool_process_data data/demo1
```

After the program starts, control the data avenger with standard `WSAD` controls.

Výpis kódu 1 Example program.

```
#include <CoolSoft.h>
#include <iostream>

int main() {
  int i;
  if(i = cool::ProcessAllData()) // returns 0 on error
    std::cout << i << std::endl;
  else
    std::cerr << "error!" << std::endl;
  return 0;
}
```