

UNIVERZITA KOMENSKÉHO V BRATISLAVE  
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ZUI PREZENTAČNÝ SOFTVÉR

UNIVERZITA KOMENSKÉHO V BRATISLAVE  
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

3c392266-ae72-46bc-b578-132afbb15638

ZUI PREZENTAČNÝ SOFTVÉR

Bakalárska práca

Študijný program: Aplikovaná informatika  
Študijný odbor: 9.2.9 Aplikovaná informatika  
Školiace pracovisko: Katedra aplikovanej informatiky  
Školiteľ: Mgr. Matej Novotný, PhD.

Bratislava 2011

Milan Lajtoš

---

## ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Milan Lajtoš  
**Študijný program:** aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)  
**Študijný odbor:** 9.2.9. aplikovaná informatika  
**Typ záverečnej práce:** bakalárska  
**Jazyk záverečnej práce:** slovenský

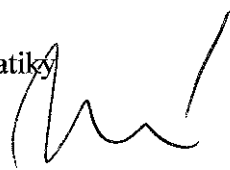
**Názov:** ZUI prezentačný softvér

**Cieľ:** Vytvoriť nástroj pre vytváranie a prehrávanie prezentácií na báze Zoomable User Interfaces. Nástroj bude umožňovať vytvorenie prezentácie na veľkom plátne, navigáciu v prezentácií, vytvorenie cesty po prezentácii, vkladanie textovej a grafickej informácie, ukladanie do vlastného formátu a export celého plátna prezentácie do vybraného grafického formátu.

**Literatúra:** <http://prezi.com/>

**Vedúci:** Mgr. Matej Novotný, PhD.  
**Katedra:** FMFI.KAI - Katedra aplikovanej informatiky  
**Dátum zadania:** 04.10.2010

**Dátum schválenia:** 12.10.2010

  
doc. RNDr. Mária Markošová, PhD.  
garant študijného programu

  
.....  
študent

  
.....  
Vedúci

# Čestné vyhlásenie

Čestne vyhlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím citovaných zdrojov a za odborného vedenia môjho vedúceho bakalárskej práce.

.....

Milan Lajtoš

# Podakovanie

Týmto by som sa chcel poďakovať vedúcemu mojej bakalárskej práce Mgr. Matejovi Novotnému, PhD. za jeho cenné rady a pripomienky, ktoré mi počas odborného vedenia poskytol.

# Abstrakt

LAJTOŠ, Milan: ZUI prezentačný softvér [Bakalárska práca]. Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. Vedúci bakalárskej práce: Mgr. Matej Novotný, PhD. Bratislava: FMFI UK, 2011. 47 s.

Táto práca sa zaoberá netradičným typom prezentačného softvéru, ktorý namiesto bežných slajdov používa na zobrazovanie prezentovaných informácií tzv. Zooming User Interface – grafické prostredie, kde používateľ môže meniť stupeň priblíženia prehliadanej oblasti, za účelom zobrazenia viac alebo menej detailov. Výsledkom práce je návrh a funkčná implementácia prototypu prezentačného programu takéhoto druhu, s využitím frameworku QtQuick.

Kľúčové slová: prezentácia, ZUI, QtQuick

# Abstract

LAJTOŠ, Milan: ZUI presentation software [Bachelor thesis]. Comenius University in Bratislava. Faculty of Mathematics, Physics and Informatics; Department of Applied Informatics . Bachelor thesis supervisor: Mgr. Matej Novotný, PhD. Bratislava: FMFI UK, 2011. 47 p.

This thesis deals with non-traditional type of presentation software, that instead of usual slides uses Zooming User Interface – graphical environment where user can change the scale of the viewed area in order to see more detail or less. The result of this thesis is design and functional implementation of prototype presentation program of this kind, utilizing QtQuick framework.

Keywords: presentation, ZUI, QtQuick

# Predhovor

Už od nepamäti ľudia medzi sebou komunikovali rôznymi spôsobmi. Nech už sa jednalo o priamu komunikáciu, dymové signály, či šifrované e-maily, cieľ tejto komunikácie bol stále ten istý – výmena informácií.

Dnes, kedy svetu vládne Internet na čele so sociálnymi sieťami, je vyhlasovanie svojich názorov a myšlienok časté ako nikdy predtým. Aby sa však jednalo o skutočnú komunikáciu, myšlienka nemôže ostať nevypočutá. Je potrebné ju dostať do cudzej mysle a eventuálne vyprovokovať odozvu. Replikácia týchto informácií však neprebíha automaticky, ako by sa mohlo zdať. Aby sme zaistili prežitie našich myšlienok, názorov, či nápadov aj do budúcnosti, je nevyhnutné ich správne *prezentovať*.



# Obsah

<b>1</b>	<b>Úvod</b>	<b>1</b>
1.1	Cieľ a obsah práce . . . . .	2
<b>2</b>	<b>Prezentačný softvér</b>	<b>3</b>
2.1	Popis . . . . .	3
2.2	Existujúce riešenia . . . . .	3
2.2.1	Microsoft PowerPoint . . . . .	3
2.2.2	OpenOffice.org Impress . . . . .	4
2.2.3	Apple Keynote . . . . .	5
2.2.4	L <sup>A</sup> T <sub>E</sub> X Beamer . . . . .	5
<b>3</b>	<b>Zooming User Interface</b>	<b>6</b>
3.1	Popis . . . . .	6
3.1.1	WIMP . . . . .	6
3.1.2	Post-WIMP . . . . .	6
3.1.3	ZUI . . . . .	7
3.1.4	Sémantické zväčšenie . . . . .	7
3.2	Implementácie . . . . .	8
3.2.1	Virtuálne plochy . . . . .	8
3.2.2	Počítačové mapy . . . . .	10
3.2.3	Pad . . . . .	10
3.2.4	Pad++ . . . . .	11
3.2.5	Zoomworld . . . . .	11
3.2.6	Raskin . . . . .	13
<b>4</b>	<b>ZUI prezentačný softvér</b>	<b>14</b>
4.1	Popis . . . . .	14
4.1.1	Nelineárnosť prezentácie . . . . .	14
4.2	Existujúce riešenia . . . . .	15
4.2.1	pptPlex . . . . .	15
4.2.2	CounterPoint . . . . .	15

4.2.3	Ahead . . . . .	16
4.2.4	Prezi . . . . .	16
<b>5</b>	<b>Návrh práce</b>	<b>19</b>
5.1	Editačná časť . . . . .	19
5.1.1	Prvky . . . . .	20
5.2	Prezentačná časť . . . . .	20
5.2.1	Cesta prezentácie . . . . .	20
5.2.2	Animácia prechodu . . . . .	20
5.3	Architektúra . . . . .	24
5.3.1	ZuiView . . . . .	25
5.3.2	ZuiItem . . . . .	25
5.3.3	ZuiHelper . . . . .	26
5.4	Používateľské rozhranie . . . . .	26
<b>6</b>	<b>Implementácia</b>	<b>28</b>
6.1	Qt . . . . .	28
6.1.1	Graphics View framework . . . . .	28
6.1.2	Qt Meta-object Language . . . . .	29
6.1.3	QtQuick . . . . .	30
6.2	Popis implementácie . . . . .	31
6.2.1	Animácia prechodu . . . . .	31
6.2.2	Export plátna . . . . .	32
6.2.3	Používateľské rozhranie . . . . .	32
<b>7</b>	<b>Zhodnotenie</b>	<b>33</b>
	<b>Literatúra</b>	<b>34</b>
	<b>Príloha</b>	<b>36</b>

# Zoznam obrázkov

2.1	Microsoft PowerPoint pod Mac OS X . . . . .	4
3.1	Ukážka sémantického zväčšenia . . . . .	8
3.2	Spaces – virtuálne plochy v Mac OS X . . . . .	9
3.3	Pad++ – prehliadač HTML stránok . . . . .	12
3.4	Raskin – pohľad na časť plochy . . . . .	13
4.1	Prezi . . . . .	17
5.1	Ukážka $u, w$ diagramu . . . . .	23
5.2	Schéma architektúry prezentačnej vrstvy . . . . .	25
5.3	Používateľské rozhranie . . . . .	27
6.1	Ukážka bežiacej prezentácie . . . . .	32

# Kapitola 1

## Úvod

Porozumenie cudzím myšlienkam býva občas zložité. Správne podať svoj názor alebo ideu je o to ťažšie, ak sme odkázaní len na verbálne vyjadrovacie prostriedky. Nie každý sa narodil s darom reči, či s dostatočnou dávkou sebavedomia pre verejné vystupovanie. Hoci sa tieto schopnosti dajú nadobudnúť dlhodobým tréningom, niekedy aj najskúsenejší rečník potrebuje použiť inú formu vyjadrovania. Veď nie nadarmo sa hovorí, že obrázok je hodný tisíc slov.

Ešte pred pár desiatkami rokov si rečník musel vystačiť sám, avšak s nástupom nových technológií sa toto obmedzenie vytratilo. Premietanie priesvitiek na plátno dodalo prezentáciám nový rozmer. Rečník tak mohol lepšie a rýchlejšie zachytiť podstatu prezentovanej myšlienky, než by to bolo možné predtým. Hoci sa priesvitky pomaly vytrácajú, ako väčšina zastaralých technológií si však stále nájdu svojich priaznivcov.

S príchodom osobných počítačov a projektorov sa situácia opäť zmenila. Priesvitky sa zdigitalizovali, zmenili meno, vznikli programy určené na ich jednoduchú tvorbu a odrazu všetci mohli podať vizuálne dokonalú prezentáciu. O rečníckej časti sa to žiaľ povedať nedá.

Slajdy, ako sa dnes nazývajú novodobé priesvitky, dostali mnoho funkcií. Možnosť vložiť multimediálny obsah ako obrázky, videá, grafy, či tabuľky je dnes samozrejmosťou. Krásne vizuálne témy idú ruka v ruke s oku lahodiacimi animáciami, či typograficky dokonalým textom.

Hoci počítačové prezentácie nadobudli veľa moderných prvkov, stále sa držia pôvodného konceptu priesvitiek. Tie majú pevný rozmer, ktorý obmedzuje vyjadrenie myšlienky maximálne na pár slajdov. Tento nedostatok je možné prekonať nahradením množstva malých slajdov jedným veľkým plátnom, kde budú umiestnené prezentované informácie. Samotnou prezentáciou bude postupné zobrazovanie výrezov tohto plátna.

## 1.1 Cieľ a obsah práce

Cieľom tejto bakalárskej práce je popis a funkčná implementácia nového typu prezentačného softvéru, ktorý využíva na prezentáciu informácií tzv. *Zooming User Interface* – grafické prostredie, kde sú objekty rozmiestnené na nekonečnej virtuálnej ploche, pričom používateľ môže meniť stupeň priblíženia prehliadanej oblasti za účelom zobrazenia viac alebo menej detailov. Aplikácia bude vytvorená s využitím frameworku *QtQuick*.

Tému pre túto záverečnú prácu som si zvolil z osobného presvedčenia, že ZUI používateľské rozhrania sú v mnohých ohľadoch prirodzenejšie, než tie tradičné, ktoré sa v dnešnej dobe využívajú. Hoci o prezentačné programy nejavím rovnaký záujem, fakt že existuje iba veľmi málo programov na tvorbu prezentácií, ktoré využívajú *Zooming User Interface*, ma presvedčil, že toto bude vhodná téma na záverečnú prácu. Dôvodom pre voľbu frameworku *QtQuick*, bola chuť naplno využiť deklaratívny prístup tvorby aplikácií, ktorý tento nástroj ponúka.

V prvej kapitole s názvom „Prezentačný softvér“ sa venujeme tradičným prezentačným programom, ktoré sa v dnešnej dobe používajú. V kapitole „Zooming User Interface“ popisujeme a uvádzame významné príklady tohto druhu používateľských rozhraní. Novým typom prezentačného softvéru sa zaoberáme v kapitole „ZUI prezentačný softvér“, kde vymenujeme a popíšeme vybrané implementácie. Kapitola „Návrh práce“ obsahuje detailný popis návrhu ZUI prezentačného programu, ktorého implementáciu opíšeme v nasledujúcej kapitole „Implementácia“. V poslednej kapitole „Zhodnotenie“, zhrnieme dosiahnuté výsledky a popíšeme budúce smerovanie projektu.

# Kapitola 2

## Prezentačný softvér

### 2.1 Popis

Prezentačný softvér alebo jednoducho prezentačný program, je aplikácia, ktorá umožňuje používateľovi vytvoriť prezentáciu. Tá sa typicky skladá z jedného alebo viacerých snímkov, ktoré sa nazývajú slajdy (z anglického *slide*). Samotná prezentácia je teda len prepínanie jednotlivých slajdov, ktoré nasledujú za sebou.

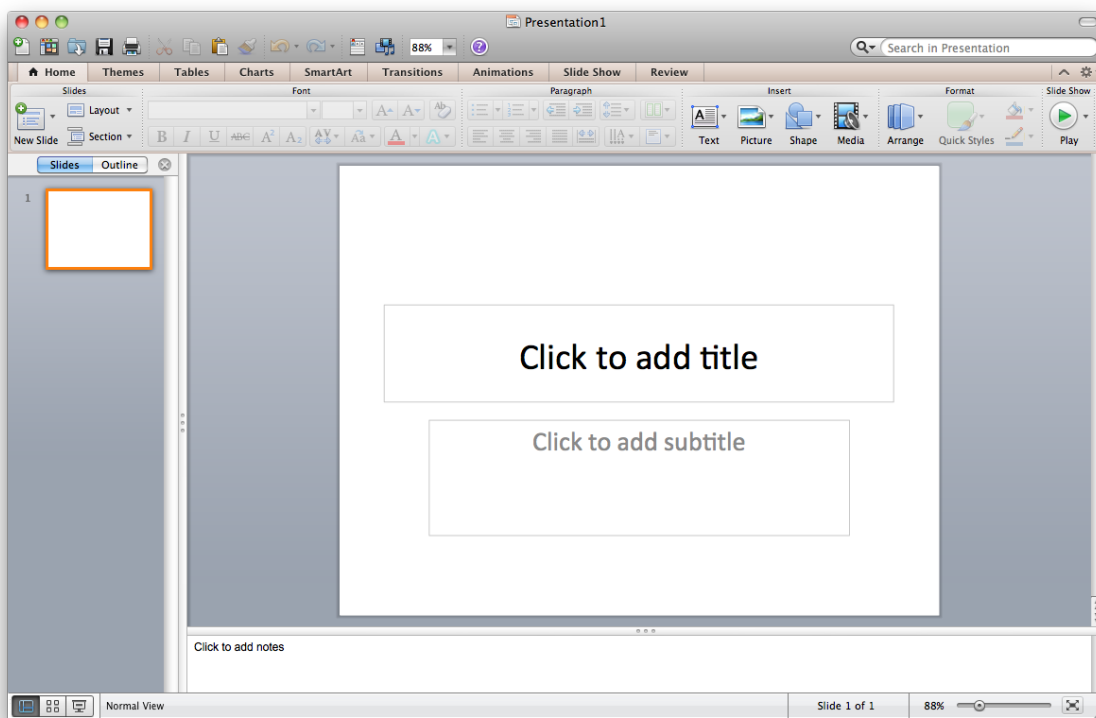
Ako budeme môcť vidieť, súčasné prezentačné programy obsahujú rôznorodú funkcionality. Avšak používanie slajdov robí z týchto aplikácií doslova len moderné priesvitky.

### 2.2 Existujúce riešenia

Na svete sa dá nájsť nespočetné množstvo prezentačného softvéru. Máme na výber z komerčne úspešných produktov, ale aj z projektov s otvoreným zdrojovým kódom, ktoré im celkom úspešne konkurujú. V nasledujúcej časti si popíšeme štyri najpoužívanéjšie prezentačné systémy.

#### 2.2.1 Microsoft PowerPoint

Jednoznačne najpopulárnejším a najpoužívanejším programom na tvorbu prezentácií je *Microsoft PowerPoint* z kancelárskeho balíku *Microsoft Office*. Tento prezentačný program pôvodne vznikol v roku 1984 pod názvom *Presenter* ako aplikácia pre počítače *Macintosh* od spoločnosti *Apple*. V roku 1987 bol z dôvodu problémov s obchodnou značkou premenovaný na *PowerPoint* a ešte v tom istom roku odkúpený spoločnosťou *Microsoft*.



Obr. 2.1: Microsoft PowerPoint pod Mac OS X

*PowerPoint* ponúka bohatú funkcionálnosť, ktorá zahŕňa definovanie vlastných animácií na slajde, animácie prechodu slajdov, veľmi pokročilé formátovanie textu, vkladanie obrázkov, videí, tabuliek, grafov a veľa ďalšieho. Používateľ si tiež môže vybrať z mnoho preddefinovaných grafických tém a vyrobiť tak profesionálne vyzerajúcu prezentáciu podstatne rýchlejšie.

Aj napriek tomu, že *Microsoft PowerPoint* ponúka veľké množstvo funkcií a je jednoznačnou dominantou v oblasti prezentačného softvéru, nemá žiadnu podstatnú výhodu oproti konkurencii.

## 2.2.2 OpenOffice.org Impress

*Impress* patrí do slobodného kancelárskeho balíku *OpenOffice.org*, ktorý vznikol na základoch kancelárskeho balíku *StarOffice* od spoločnosti *Sun Microsystems*. Dôvodom vzniku mala byť snaha zapojiť komunitu do vývoja a lepšie konkurovať spoločnosti *Microsoft* s ich balíkom *Office*.

Aplikácia používa na ukladanie prezentácií, tzv. formát *OpenDocument*, ktorý bol schválený a prijatý ako štandard ISO. Okrem podpory natívneho formátu, *Impress* podporuje aj import prezentácií z ostatných prezentačných programov vrátane *Microsoft PowerPoint*.

Po kúpe spoločnosti *Sun Microsystems* v roku 2009 spoločnosťou *Oracle*, sa vyskytli obavy o budúcnosť celého kancelárskeho balíku. Poprední vývojári, ktorý pracovali na balíku, opustili spoločnosť a založili organizáciu *The Document Foundation*, ktorá vytvorila čisto komunitnú vetvu tohto kancelárskeho balíku s názvom *LibreOffice*.

### 2.2.3 Apple Keynote

*Apple Keynote* patrí do kancelárskeho balíku *iWork* od spoločnosti *Apple*. Podobne ako predchádzajúce dva príklady prezentačných programov, aj *Keynote* používa rovnaký spôsob prezentovania – slajdy. Hoci sa v tejto základnej črte oproti konkurencii nelíši, obsahuje pár vylepšení, ktoré robia tento produkt zaujímavejším.

Pravdepodobne najzvláštnejšou funkciou, ktorú *Keynote* ponúka je možnosť využiť *iPhone* (smartfón od spoločnosti *Apple*) ako dodatočný displej, ktorý je synchronizovaný s prezentáciou a zobrazovať na ňom poznámky k aktuálnemu slajdu. Samozrejmosťou je možnosť prechádzať medzi jednotlivými snímkami prezentácie jednoduchými gestami na ploche telefónu.

Ďalšou vlastnosťou tohto produktu je možnosť používať pokročilé animácie. Príkladom môže byť funkcia *Magic Move*, ktorá interpoluje pozíciu, veľkosť a rotáciu pre objekt, ak sa nachádza na dvoch snímkach prezentácie za sebou.

### 2.2.4 L<sup>A</sup>T<sub>E</sub>X Beamer

Veľmi nezvyčajný spôsob tvorby prezentácií používa systém L<sup>A</sup>T<sub>E</sub>X *Beamer*. Narozdiel od predchádzajúcich programov, *Beamer* nie je tzv. WYSIWYG (*What You See Is What You Get*, v doslovnom preklade „čo vidíš, to dostaneš“) editor, ale balíček makier pre L<sup>A</sup>T<sub>E</sub>X, ktoré umožňujú vytvorenie dokumentu určeného na prezentáciu. Používateľ upravuje zdrojový kód prezentácie, ktorý sa neskôr skompiluje do zobraziteľnej podoby.

Narozdiel od predchádzajúcich programov, *Beamer* nepracuje s pojmom slajd. Používateľ vytvára tzv. rámy (z anglického *frame*), do ktorého umiestňuje informácie. Používateľ má však možnosť nezobraziť všetky informácie v ráme naraz, ale postupne, čím vytvorí sadu slajdov z jedného rámu. Existuje množstvo funkcií, ktoré ovplyvňujú zobrazenie istej informácie na jednotlivých slajdoch, čím sa dá dosiahnuť veľmi komplexné správanie, ktoré by sa v ostatných prezentačných programoch vytváralo zložito.

Okrem zadefinovania slajdov si používateľ môže vybrať z viacerých predpripravených animácií, ktoré spríjemňujú zmenu rámu. Podobne ako pri slajdoch, používateľ má plnú slobodu nad tým, ktorá animácia sa má spustiť pri prechode na ľubovoľný slajd.



# Kapitola 3

## Zooming User Interface

### 3.1 Popis

V nasledujúcej kapitole si popíšeme netradičný typ používateľského rozhrania – *Zooming User Interface*, ktoré vo veľkej miere používa priblíženie a oddialenie pre zobrazovanie informácií.

#### 3.1.1 WIMP

Väčšina dnešných grafických používateľských rozhraní (*Graphical User Interface* - *GUI*) používaných na interakciu medzi človekom a počítačom patrí do skupiny tzv. WIMP používateľských rozhraní. Skratka WIMP označuje „*window, icon, menu, pointing device*“, teda „okno, ikona, kontextová ponuka, ukazovacie zariadenie“, čo vyjadruje prvky, ktoré sa používajú v tomto type používateľských rozhraní. Kvôli veľkému rozšíreniu býva WIMP nesprávne označované ako GUI.

Prvé WIMP používateľské rozhranie bolo vyvinuté v roku 1973 vo výskumnom centre PARC spoločnosti Xerox rámci projektu Xerox Alto. Aj napriek revolučnej myšlienke, WIMP nebolo považované za správny smer vo vývoji užívateľských rozhraní a dlho ostalo nepovšimnuté. O popularizáciu sa postarala až spoločnosť Apple so svojím počítačom Macintosh v roku 1984. Od toho momentu sa WIMP paradigma stala hlavným prúdom tvorby používateľských rozhraní a za posledné tri desaťročia nepodstúpila takmer žiadne zmeny.

#### 3.1.2 Post-WIMP

Hoci WIMP používateľské rozhrania sú pre začiatočníkov s počítačmi veľmi intuitívne vďaka dobrej analógií s objektami zo skutočného sveta [9], tak významnú úlohu začí-

najú zohrávať aj tzv. post-WIMP používateľské rozhrania, prvýkrát popísané Jakobom Nielsenom v *Noncommand User Interfaces* [13]. Ako z ich názvu vyplýva, snažia sa odpútať od WIMP paradigmy a využívať alternatívne ovládacie prvky pri interakciách s používateľmi. Typickými príkladmi post-WIMP používateľských rozhraní sú napr. počítačové hry, systémy využívajúce virtuálnu realitu, používateľské rozhrania ovládané gestami, či rečou.

Vďaka rozmachu múdrych telefónov tzv. *smartfónov* a najmä ich operačných systémov, sa začínajú post-WIMP používateľské rozhrania používať omnoho častejšie, aj napriek tomu, že na stolových a prenosných počítačoch ostáva WIMP stále štandardom.

### 3.1.3 ZUI

*Zooming User Interface*<sup>1</sup>, skrátene označované ako ZUI, sa zaraďuje medzi post-WIMP grafické prostredia. Charakteristickou črtou rozhrania sú objekty rozmiestnené na nekonečnej virtuálnej ploche, ktorej prehliadanie je možné vďaka zmene stupňa priblíženia, pohybu po ploche a častokrát aj jej natočeniu.

Používateľ má úplnú slobodu nad navigáciou – nie je obmedzený veľkosťou plátna, ktoré je nekonečné vo všetkých smeroch. Používateľ sa teda môže pohybovať po ploche podľa svojich potrieb bez akýchkoľvek obmedzení.

Obmedzenie nenastáva ani pri priblížení, resp. oddialení zobrazovanej plochy. Rozlíšenie tejto virtuálnej plochy je podobne ako jej veľkosť tiež nekonečná. Táto skutočnosť umožňuje vtesnať do priestoru medzi dva objekty akékoľvek množstvo objektov, čo ponúka naozaj široké možnosti použitia.

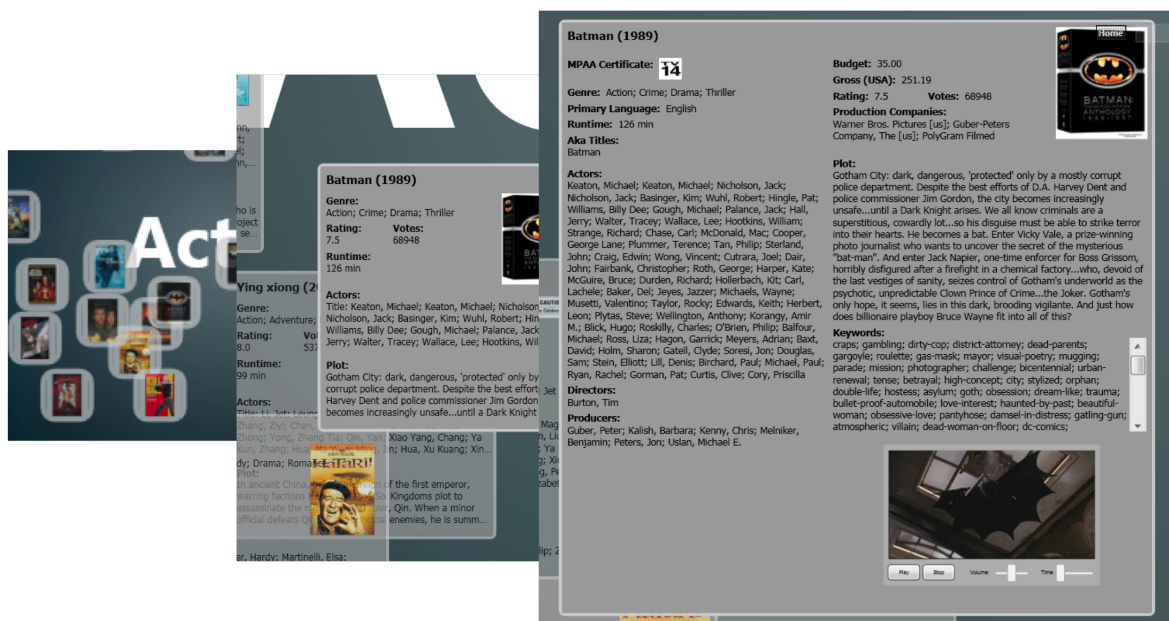
### 3.1.4 Sémantické zväčšenie

Sémantické zväčšenie, angl. *semantic zoom*, je funkcia ZUI rozhraní, ktorá umožňuje meniť obsah zobrazovaného objektu v závislosti od úrovne priblíženia. Narozdiel od obyčajného zväčšenia, ak má objekt na svoje zobrazenie iba veľmi malú plochu, nesnaží sa zobraziť všetky informácie, ktoré má v sebe obsiahnuté, ale len tie, ktoré pri danom stupni zväčšenia majú význam. Objekt teda mení svoju podobu na základe prostredia, v ktorom sa nachádza, resp. poskytuje detaily na vyžiadanie.

Ilustračným príkladom môže byť objekt, ktorý predstavuje film. Ak zobrazujeme veľké množstvo takýchto objektov, teda máme oddialený pohľad, stačí zobrazovať obrázok, ktorý reprezentuje daný film. Ak si objekt priblížime, môžeme k obrázku pridať názov filmu. Ak sa priblížime ešte viac, máme už dostatok plochy na zobrazenie textového

---

<sup>1</sup>niekde označované aj ako *Zoomable User Interface*



Obr. 3.1: Ukážka sémantického zväčšenia

popisu filmu. Ak bude na ploche viditeľný už len sám objekt, môže sa zobrazíť zoznam obsadenia, hodnotenie alebo dokonca aj ukážku z filmu.

Implementáciu popísaného príkladu sémantického zväčšenia je možné nájsť v aplikácii *MedioVis2*, ktorá je vyobrazená na obrázku 3.1.

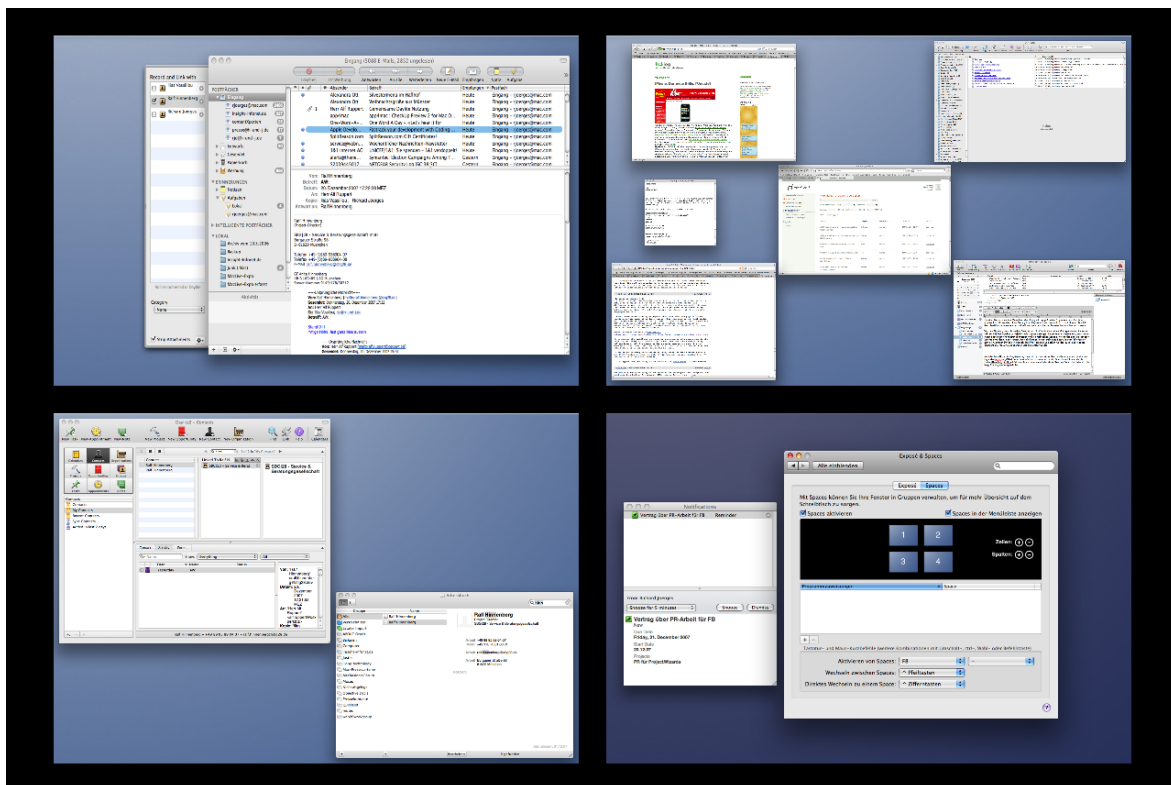
## 3.2 Implementácie

V tejto časti si popíšeme dve najpoužívanejšie implementácie ZUI, čo sú virtuálne plochy a počítačové mapy. Ďalej sa pozrieme na vôbec prvý prototyp tohto druhu rozhrania – *Pad* a jeho nasledovníka *Pad++*. Stručne spomenieme ambiciózný projekt, ktorý mal používať len *Zooming User Interface* na interakciu s používateľom – *Zoomworld*, a nakoniec aplikáciu *Raskin*, ktorá kombinuje tradičné GUI a ZUI dokopy.

### 3.2.1 Virtuálne plochy

Jednou z najpoužívanejších a najrozšírenejších implementácií *Zooming User Interface* sú virtuálne plochy, ktoré avšak ponúkajú len obmedzenú funkcionálnosť oproti ostatným existujúcim riešeniam.

Pojem „virtuálna plocha“ označuje koncept, ktorý rozširuje správu okien v desktopových prostrediach, o možnosť umiestňovať okná do samostatných skupín, resp. na rôzne plochy. Tie dokážu koexistovať, avšak aktívna je vždy len jedna plocha. Teda ak používateľ pracuje s len s pár oknami, pričom má otvorené aj ďalšie, ktoré nepotrebuje,



Obr. 3.2: Spaces – virtuálne plochy v Mac OS X

môže si nepotrebné okná premiestniť na samostatnú plochu, kde mu nebudú prekážať a môže sa tak sústrediť na prácu.

Podobnosť so ZUI nastáva pri zobrazení všetkých virtuálnych plôch zároveň. Používateľ teda získava prehľad o všetkých plochách a ich oknách súčasne. Pri tomto zobrazení má teda používateľ najlepšiu možnosť manipulovať s oknami (presun, zatvorenie), či samotnými plochami (pridávanie, odoberanie alebo zmena usporiadania).

Tento virtuálnych plôch vznikol, podobne ako celá WIMP paradigma, vo výskumnom stredisku Xerox PARC v roku 1986 pod názvom *Rooms* [12] a neskôr bol prevzatý aj do iných prostredí, kde zastáva neoddeliteľnú súčasť pri práci s oknami.

Táto funkcionality je typická pre desktopové prostredia alebo správcov okien nachádzajúcich sa na unixových systémoch, ktoré obsahujú *X Window System*. Medzi najznámejšie príklady, ktoré implementujú túto funkciu patria *KWin* (prostredie *KDE*), *Metacity* (prostredie *Gnome*), *Compiz*, *Fluxbox* a mnoho ďalších.

Iné operačné systémy nie sú výnimkou. *Mac OS X* vyvíjaný spoločnosťou *Apple* od verzie 10.5 ponúka totožnú funkciu pod názvom *Spaces* (obr. 3.2). Naproti tomu operačný systém *Windows* od spoločnosti *Microsoft* umožňuje používať funkciu len vďaka rôznym doplnkom, ktoré nie sú súčasťou samotného systému.

### 3.2.2 Počítačové mapy

Veľmi typické použitie *Zooming User Interface* sú počítačové mapy. Aj napriek tomu, že sa nejedná o ZUI podľa definície, táto paradigma slúži na prezeranie máp veľmi dobre a práca s nimi sa stáva veľmi intuitívnou aj vďaka podobnosti s reálnym svetom. Hlavnými zástupcami počítačových máp sú produkty *Maps* a *Earth* od spoločnosti *Google*, *Bing Maps* od spoločnosti *Microsoft*, *Ovi Maps* od *Nokie* alebo otvorené mapy *OpenStreetMap*.

Najväčším obmedzením väčšiny počítačových máp oproti klasickým implementáciám ZUI, je používanie rôznych dát pre rôzne stupne priblíženia. Táto technika sa používa najmä pri satelitných fotografiách zemského povrchu, ale vektorové podklady nie sú výnimkou. Hlavným dôvodom pre toto obmedzenie je najmä šetrenie prostriedkami ako pamäť, disk, či pripojenie do Internetu. Dalo by sa tvrdiť, že táto vlastnosť je akýmsi simulovaním sémantického priblíženia.

### 3.2.3 Pad

*Pad* [14] bola prvá implementácia *Zooming User Interface* rozhrania na svete, za ktorou stoja Ken Perlin a David Fox. *Pad* bol na svoju dobu (rok 1993) veľmi výnimočný projekt, ktorý išiel ďaleko za hranice tradičnej WIMP paradigmy. Okrem prvenstva medzi ZUI rozhraniami, priniesol aj veľmi revolučné myšlienky, ktoré postupne prichádzajú aj do používateľských rozhraní, ktoré denne používame.

Jednou z myšlienok, ktoré boli spolu s projektom *Pad* predstavené je sémantické zväčšenie, ktoré je podrobnejšie popísané v časti 3.1.4 Sémantické zväčšenie na strane 7. Ďalšou nemenej zaujímavou, avšak podstatne abstraktnejšou myšlienkou boli portály a tzv. „portálové filtre“.

#### Portály

Portál označuje v tomto kontexte objekt, ktorý umožňuje presun k iným častiam plochy. Ako každý portál, aj tento má dva konce – zdrojový a zobrazovací. Zdrojová časť sa umiestni na požadovanú časť plochy, ktorá má byť zobrazená cez portál. Zobrazovacia časť sa umiestni na požadovanú polohu, kde má byť zobrazená plocha zo zdroju. Veľmi vhodným umiestnením portálov by sme napríklad mohli simulovať fraktály.

#### Portálové filtre

Pojem portálové filtre označuje (nielen pri ZUI rozhraniach) spôsob vizualizácie istého zdrojového objektu iným objektom, ktorý modifikuje výzor zdrojového objektu podľa

istých pravidiel. Najhlavnejším pravidlom je, že objekt, ktorý mení zdrojový objekt, musí „rozumieť“ dátam, ktoré poskytuje zdroj – musia byť teda navzájom kompatibilné. Názorným príkladom môže byť tabuľka čísel, ktorá slúži ako zdroj dát. Ďalej máme portálový filter, ktorý rozpoznáva tabuľkové dáta a transformuje ich na koláčový graf. Ak presunieme tento portálový filter nad našu tabuľku, prekrytá časť tabuľky bude zobrazená ako graf. Ak portálový filter nemá zdroj dát, nemá žiadnu grafickú podobu, resp. nič nezobrazuje.

Tento koncept sa veľmi podobá na *Model-View-Controller* [17] architektúru, ktorá sa používa pri návrhu a implementácii aplikácií. *Model* sa dá chápať ako zdroj dát, ktoré majú byť transformované a zobrazené cez *view*. *Controller* zabezpečuje prepojenie oboch častí, prípadne zmenu dát, ktoré sú obsiahnuté v modeli. Pri portálových filtroch je však situácia mierne odlišná. Všetky tri elementy majú grafickú podobu, čo pri MVC architektúre neplatí, kde len *view* má takúto vlastnosť. Navyše, portálové filtre nemusia len zobrazovať dáta, ale môžu s nimi aj manipulovať, teda správať sa ako *controller* alebo môžu spĺňať obe funkcie súčasne.

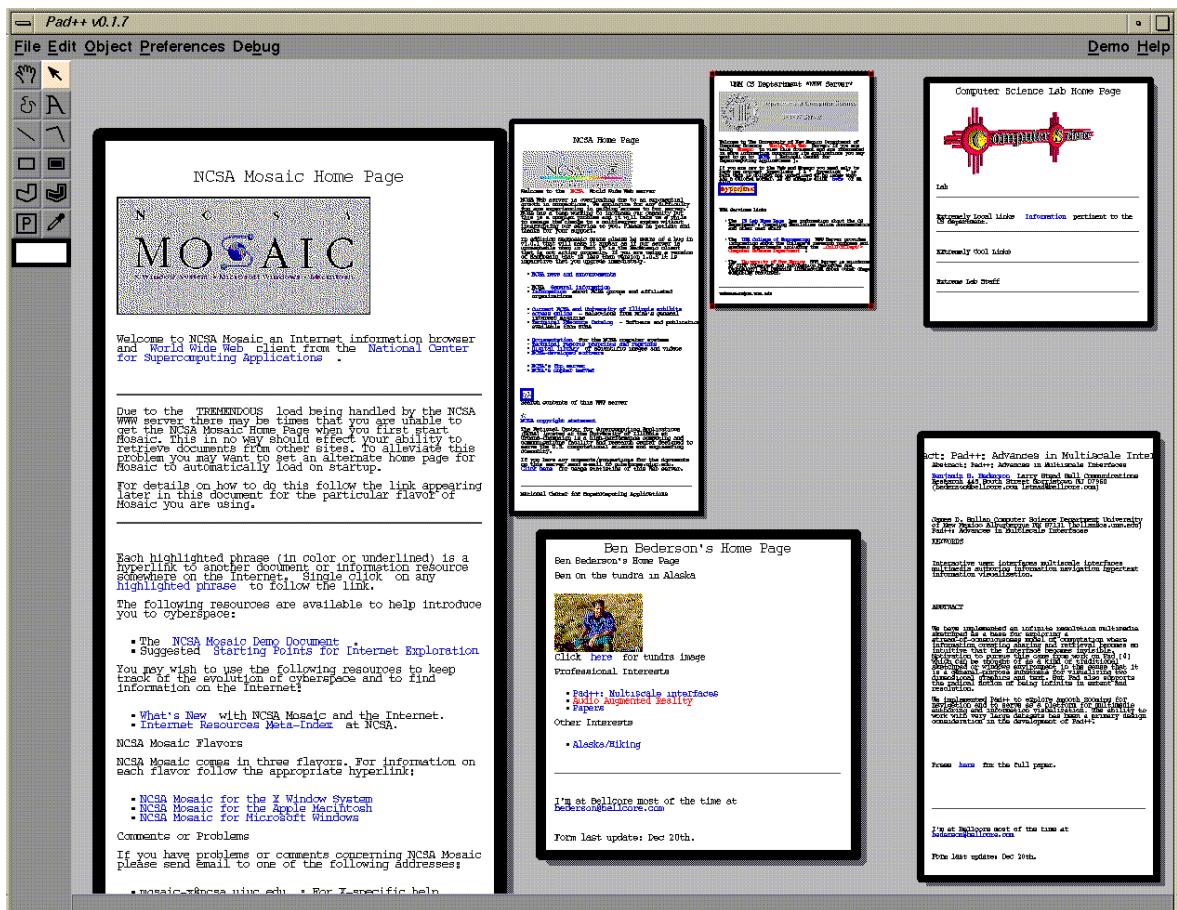
### 3.2.4 Pad++

Po úspechu s *Pad* v oblasti používateľských rozhraní, Jim Hollan a Ben Bederson začali v roku 1994 vyvíjať nasledovníka – *Pad++* [4]. Tento projekt je najdlhšie trvajúcou iniciatívou o vytvorení ZUI rozhrania, aj napriek tomu, že vývoj bol v roku 1998 oficiálne zastavený [6]. Počas svojej existencie, však posunul ZUI naozaj ďaleko, o čom svedčia Bedersonove ďalšie projekty *Jazz* [5] a *Piccolo* [3] (neskôr premenované na *Piccolo2D*), v ktorých možno nájsť veľa myšlienok z *Pad++*.

*Pad++* úspešne aplikovalo ZUI princíp na rôzne problémy, ktoré nemajú veľa spoločné. Aplikácia obsahovala prehliadač dokumentov s osnovou, kresliaci program, prehliadač obsahu disku, či dokonca prehliadač HTML stránok, ktorý je možné vidieť na obrázku 3.3.

### 3.2.5 Zoomworld

Jef Raskin, ktorý sa podieľal na vývoji počítača *Macintosh*, vo svojej knihe *The Humane Interface* [16] popisuje cestu za úplne prirodzeným používateľským rozhraním, ktoré by odstránilo dizajnové chyby WIMP paradigmy, viď. str. 6. Raskin popisuje vtedajšie dostupné používateľské rozhrania ako bludiská, v ktorých sa používateľ často stráca pri dosiahnutí nejakého cieľa. Autor preto navrhuje vyriešiť tento problém *oddialením* – nazretím naň z výšky, odkiaľ vidieť štruktúru tohto bludiska a tým pádom sa mu vyhnúť a preletieť priamo k cieľu.



Obr. 3.3: Pad++ – prehliadač HTML stránok

Raskin nazýva tento prístup *Zooming Interface Paradigm* a konkrétny prípad takéhoto rozhrania popisuje v knihe ako *ZoomWorld*. Autorovými slovami je popísaný takto:

The ZIP [Zooming Interface Paradigm] described here is called ZoomWorld and is based on the idea that you have access to an infinite plane of information having infinite resolution. The plane is ZoomWorld. Everything you can access is displayed somewhere on ZoomWorld, whether it is on your computer, on a local network to which your computer is attached, or on a network of networks, such as the Internet.

To see more of ZoomWorld, you think of yourself as flying higher and higher above it. To look at a particular item, you dive down to it. ZoomWorld also has a content searching mechanism. The overall metaphor is one of flying, climbing to zoom out and diving to zoom in. You navigate both by flying above ZoomWorld and by doing content searches.

Vidíme, že *ZoomWorld* popisuje ako nekonečnú plochu s nekonečným rozlíšením – vieme sa priblížiť, či oddaliť bez akéhokolvek obmedzenia. Toto je štandardná definícia ZUI rozhraní. Zaujímavejšie je však využitie takéhoto rozhrania – na tomto plátne by bola



zobrazená každá dostupná informácia, či už je na používateľovom počítači alebo na Internete. Touto myšlienkou sa *ZoomWorld* líši od všetkých ostatných ZUI.

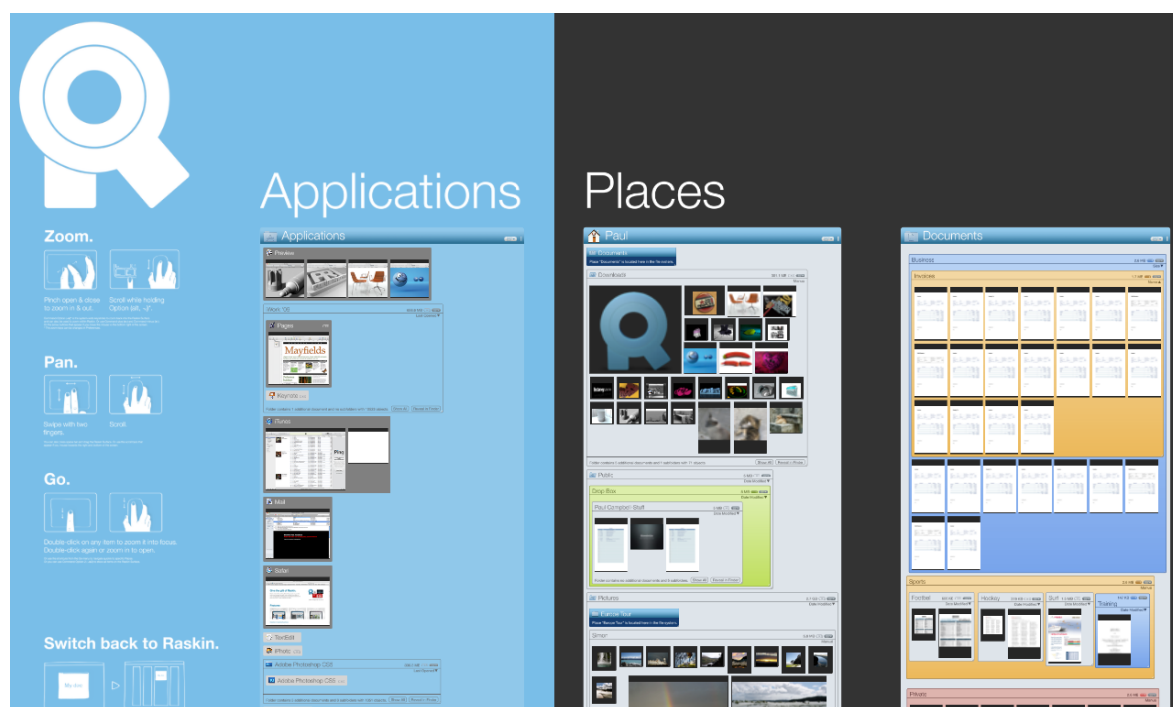
Z druhej časti citácie sa dozvedáme spôsob navigácie po ploche. Hlavným spôsobom pohybu je samozrejme priblíženie a oddialenie. *ZoomWorld* avšak ponúka aj hľadanie obsahu na tejto nekonečnej ploche.

*Zoomworld* bol len časťou vysnívaného používateľského prostredia, ktoré Raskin popísal. Snaha o implemetáciu tohto systému, pod názvom THE a neskôr *Archy*, síce skončila, avšak myšlienky uvedené v knihe stále formujú budúce generácie používateľských rozhraní. Príkladom, kde sa uplatnili Raskinove nápady nesie jeho meno...

### 3.2.6 Raskin

*Raskin* (obr. 3.4) je veľmi modernou kombináciou ZUI a tradičného WIMP rozhrania obsiahnutého v operačnom systéme Mac OS X. Štandardná pracovná plocha je nahradená zväčšovateľnou plochou, ktorá ponúka zobrazovanie dokumentov v plnom rozlíšení. Dokumenty je možné združovať do projektov, čím používateľ získava lepší prehľad nad stavom svojej práce. Spúšťanie aplikácií cez zástupcov ostalo v nezmenenej podobe – aktivácia kliknutím na ikonu.

Zaujímavou funkciou je vloženie, alebo lepšie povedané, zapustenie okien do plochy, čo umožňuje asociovať otvorenú aplikáciu s rozpracovaným dokumentom. Týmto spôsobom je tiež možné prepínať sa medzi spustenými aplikáciami.



Obr. 3.4: Raskin – pohľad na časť plochy



# Kapitola 4

## ZUI prezentačný softvér

### 4.1 Popis

Ako vyplýva z názvu, ZUI prezentačný softvér využíva na prezentovanie informácií *Zooming User Interface*, bližšie popísané v predchádzajúcej kapitole. Narozdiel od tradičných prezentačných programov, tento typ nevyužíva tzv. slajdy, opísané v časti 2.1. Namiesto toho sú jednotlivé informácie rozmiestnené na nekonečnej ploche a samotná prezentácia je zvyčajne realizovaná prechodom medzi vybratými časťami tejto plochy.

#### 4.1.1 Nelineárnosť prezentácie

Veľkou výhodou oproti klasickým prezentačným programom je možnosť nelineárnej prezentácie. Ak by sme v tradičnom prezentačnom softvéri, ktorý používa slajdy, chceli poukázať na informáciu, ktorá sa nachádzala na začiatku prezentácie, museli by sme danú informáciu (alebo celý slajd) skopírovať na požadované miesto alebo sa pri samotnom akte prezentovania vrátiť na požadovaný slajd, čo by vyžadovalo zobrazíť všetky informácie, ktoré boli spomenuté.

Pri ZUI prezentačnom softvéri môžeme uplatniť rovnakú stratégiu – vrátiť sa po preddefinovanej ceste k informácií, ktorú potrebujeme zobrazíť. To však nie je optimálnym riešením. Lepším riešením je využiť zaujímavú vlastnosť ZUI prezentácií, ktorá nám umožňujú použiť jeden objekt v tejto ceste viackrát. Teda bez toho aby sme sa museli lineárne vracieť, môžeme z aktuálneho bodu v ceste prejsť priamo na požadovanú informáciu. Hoci sa toto riešenie sa podobá na duplikáciu slajdu z tradičných prezentačných softvérov, k žiadnej duplikácii informácií nedochádza.

V predchádzajúcom príklade počítame s využitím už spomenutej informácie. Problém môže nastať ak sa prezentujúci nepočíta pri tvorbe prezentácie s takouto možnosťou. Ako sme spomenuli, pri prezentáciách, ktoré využívajú slajdy je nutné lineárne prejsť

všetky informácie, ktoré sa nachádzajú medzi aktuálnym a požadovaným slajdom. Pri prezentácií, ktorá využíva ZUI je možné sa „odkloniť“ od cesty prezentácie a zobrazovať časti plochy podľa ľubovôle. Ak prezentujúci zobrazil všetky informácie, ktoré potreboval, môže pokračovať po ceste prezentácie.

## 4.2 Existujúce riešenia

Aj napriek tomu, že prvá zmienka použitia ZUI v prezentačnom softvéri je z roku 2001 [10], existuje len veľmi málo použiteľných riešení, ktoré sa stále vyvíjajú. V nasledujúcich odsekoch popíšeme štyri najvýznamnejšie implementácie tohto typu prezentačného softvéru.

### 4.2.1 pptPlex

Rozšírením *pptPlex* do Microsoft PowerPoint (popísaným na strane 3) verzie 2007, je možné dosiahnuť efektu podobného ZUI prezentačného softvéru. Toto rozšírenie dodáva do aplikácie náhľad všetkých slajdov pre pohodlný výber toho, ktorý chceme zobraziť, bez toho aby sme museli lineárne prechádzať všetky slajdy medzi. Ďalšou možnosťou je zobraziť viacero slajdov na obrazovke naraz pre ich vzájomné porovnanie.

Z popisu zreteľne vidieť, že *PowerPoint* s rozšírením *pptPlex* neponúka rovnakú flexibilitu akú by mal ponúkať prezentačný softvér, ktorý by naplno využíval *Zooming User Interface*. Projekt nezaznamenal od roku 2008 žiadnu aktivitu a teda je považovaný za mŕtvy.

### 4.2.2 CounterPoint

Iný prístup k rovnakému problému volí *CounterPoint* [10], za ktorým stojí Lance Good a priekopník ZUI rozhraní Ben Bedereson. Hoci je *CounterPoint* oveľa starší projekt ako *pptPlex*, možnosti ktoré ponúka sú podstatne väčšie. Aj keď sa nejedná priamo o doplnok do aplikácie Microsoft PowerPoint, stavia na ňom. Oproti *pptPlex* umožňoval rozložiť slajdy na ploche, za ktorú je zodpovedný *Jazz* (spomenutý v sekcii 3.2.4 o Pad++) a vytvoriť cestu, po ktorej ide prezentácia. Aj napriek tomu, že tento program ponúka lepšie využitie ZUI, stále sme obmedzení len na prechod medzi jednotlivými slajdami.

### 4.2.3 Ahead

*Ahead* je webová aplikácia, naprogramovaná v *Adobe Flash*, ktorá ponúka prezentovanie informácií na nekonečne veľkom plátne s využitím ZUI. Používateľ môže presúvať samotné objekty a nie len slajdy s informáciami ako pri predchádzajúcich dvoch príkladoch. Používateľ môže podobne ako pri aplikácii *CounterPoint* určiť cestu, po ktorej pôjde prezentácia. *Ahead* nie je určené len na prezentácie, ale môže byť využité aj ako web stránka, virtuálne múzeum, myšlienková mapa, či iné.

Hlavnou nevýhodou pri prezentovaní s *Ahead* je animácia prechodu. Priblíženie a posunutie sú lineárne interpolované, čo nevytvára dobrý dojem zo samotnej prezentácie (viď. odsek Lineárna interpolácia na strane 21).

Vidíme, že doteraz žiadna z menovaných aplikácií nevyužíva naplno možnosti, ktoré by koncept ZUI prezentačného softvéru ponúkal.

### 4.2.4 Prezi

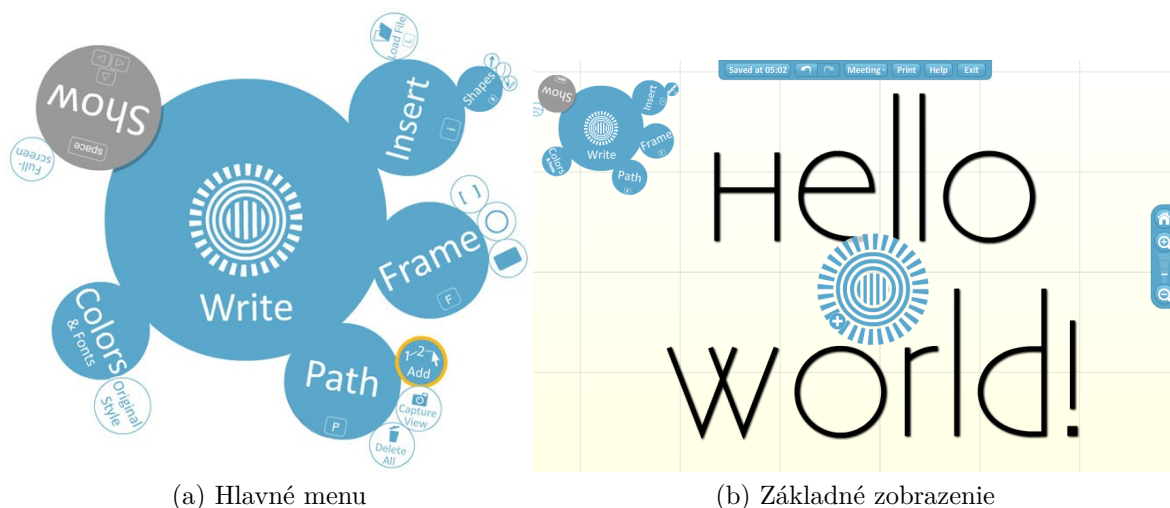
*Prezi* je doteraz najlepšou voľbou na vytváranie prezentácií využívajúcich *Zooming User Interface*. Ponúka všetky vlastnosti, ktoré mali už spomínané aplikácie a pridáva ešte mnoho ďalších inovatívnych prvkov.

#### ZUI menu

Veľmi zaujímavo je riešené hlavné menu aplikácie, viď. obrázok 4.1a, ktoré ako celá aplikácia využíva ZUI. Jednotlivé funkcie sú zobrazené ako kruhy rôznych veľkostí s textovým a niektoré aj grafickým popisom akcie. Veľkosť kruhu závisí od hierarchie funkcií, kde menej podstatnejšie funkcie majú menšiu veľkosť a opačne. Pri zvolení istej funkcie sa celá sústava kruhov priblíži, tak aby požadovaná skupina funkcií zaberala čo najväčšiu možnú plochu a ostatné funkcie boli stále prístupné. Vďaka priblíženiu sa odhalia funkcie, ktorých funkcia nebola predtým rozoznateľná kvôli ich veľkosti.

#### Zebra

Lineárne transformácie objektov na ploche, sú v *Prezi* implementované netradičnou cestou. V aplikáciách, kde je vyžadovaná podobná funkcionálna, sa zvyčajne okolo aktívneho objektu zobrazí orámovanie, ktoré má aktívne rohy. Ťahaním týchto aktívnych rohov sa aktivuje uniformné škálovanie. Body, ktoré ležia na spojnicach rohov slúžia na neuniformné škálovanie. Translácia samotného objektu je aplikovaná jednoduchým ťahaním a pustením. Na rotáciu sú zvyčajne vyhradené dva body. Ťahaním jedného prevádzame samotnú rotáciu a druhý určuje stred otáčania.



(a) Hlavné menu

(b) Základné zobrazenie

Obr. 4.1: Prezi

Naproti tomu, v *Prezi* nič podobné nenájdeme. Všetky tri transformácie sú spojené do jedného ovládacieho prvku, ktorý sa volá *Zebra*. Tento prvok sa skladá z troch sústredných kruhov, z ktorých každý predstavuje inú transformáciu. Pri ťahaní najvnútorniejšieho kruhu posúvame aktívnym objektom, pri ťahaní vnútorného kruhu uniformne škálujeme objekt a pri ťahaní vonkajšieho kruhu meníme objektu natočenie.

*Zebra* je možné vidieť v strede obrázku 4.1b prekrývajúc nápis „Hello World!“.

## Prvky na ploche

Okrem štandardných prvkov ako text, obrázkov, či video vieme do prezentácie vkladať aj iné objekty, ktoré spríjemňujú výsledný dojem z prezentácie. Najhlavnejšími sú:

- zátvorky, kruh, obdĺžnik
- šípky, čiary
- kreslenie, zvýraznenie

Prvá kategória je určená na združovanie rôznych objektov do skupín, aby mohli byť zobrazené pri prezentácii ako celok a nie len po jednom. Okrem týchto troch objektov existuje ešte štvrtý, ktorý vykonáva rovnakú funkciu, ale nemá grafickú podobu.

Šípky a čiary pomáhajú zobrazit vzťah medzi jednotlivými objektami, či poslúžiť pri referenciách na iné objekty.

Kreslením môžeme načrtnúť nejaký tvar alebo obrázok, ktorý nemáme k dispozícii a zvýrazňovačom zdôrazniť nejaký významný text alebo časť obrázku.

## Zmena štýlu

*Prezi* disponuje veľmi užitočnou funkciou, ktorá je však určená len pre pokročilých používateľov. Tým je zmena grafického výzoru prvkov na ploche, či dokonca samotnej plochy, pomocou kaskádových štýlov, angl. *Cascading Stylesheet* (CSS). Táto funkcionality dáva používateľovi neobmedzenú možnosť prispôbiť si prezentáciu do najmenších detailov. Pre používateľov, ktorí neovládajú CSS sú predpripravené témy, medzi ktorými môžu jednoducho prepínať.

## Nevýhody

Hoci je *Prezi* v kategórii ZUI prezentačných programov jednoznačne najlepšia voľba, má určité nevýhody, ktoré škodia celkovému dojmu z aplikácie:

- webová aplikácia
- platená *Pro* licencia
- implementované v *Adobe Flash*
- proprietárny softvér
- povinné zverejnenie prezentácií s normálnym používateľským kontom

Prvé bod sa dá vnímať dvojako. Na jednej strane to môžeme vnímať ako výhodu, z dôvodu dostupnosti z celého sveta, ale aj ako nevýhodu ak práve nemáme dostupné pripojenie k Internetu. Táto nevýhoda je minimalizovaná možnosťou stiahnuť si natívnu aplikáciu, ktorá je avšak dostupná len platiacim zákazníkom s *Pro* licenciou, spomenutou v druhom bode.

Obe tieto aplikácie – webová aj natívna, sú však naprogramované v *Adobe Flash*, ktorý má svoje obmedzenia. Najväčšou nevýhodou použitia tejto technológie je nutnosť mať zásuvný modul do webového prehliadača, ktorý však znižuje jeho stabilitu. Ďalšou negatívnou vlastnosťou je absencia hardvérovo-urýchľovaného vykresľovania na desktopoch.

V oboch spomínaných prípadoch sa jedná o proprietárny softvér, teda o softvér s uzavretým zdrojovým kódom. Avšak v spojení s prvými dvoma negatívami, táto vlastnosť vytvára možnosť pre finančný zisk pre spoločnosť, ktorá zastrešuje vývoj aplikácie.

Poslednou nevýhodou je povinné zverejňovanie vytvorených prezentácií na webovej stránke aplikácie. Toto obmedzenie sa však netýka študentov s edukačnou licenciou a platiacich zákazníkov.

# Kapitola 5

## Návrh práce

V nasledujúcej kapitole popíšeme návrh aplikácie určenej na tvorbu a zobrazovanie prezentácií využívajúcich *Zooming User Interface*.

### 5.1 Editačná časť

Podstatnou, aj keď nie vždy nutnou vlastnosťou prezentačného softvéru, je schopnosť vytvárať prezentácie. V prípade ZUI prezentačného programu je to nasledovné:

- pridávanie a odoberanie prvkov z plochy
- zmena ich vlastností (veľkosť, pozícia, natočenie, ...)
- vytvorenie cesty prezentácie
- navigácia po ploche

Aby bola zaistená prvá podmienka, používateľ bude mať k dispozícii objekty, z ktorých každý jeden bude predstavovať istý typ objektu, či už je to textový alebo grafický prvok. Ťahaním tejto reprezentácie prvku na plochu sa vytvorí inštancia požadovaného objektu, ktorý môže byť ďalej upravovaný. Odobratie prvku z plochy bude aplikovateľné presne opačným postupom – ťahaním objektu z plochy na jeho reprezentáciu.

Zmena veľkosti, pozície a natočenia objektu na ploche, bude možné meniť pomocou ťahania aktívnych rohov každého objektu. Tento prístup používajú takmer všetky grafické programy a používateľ sa tak nebude musieť učiť novým návykom.

Na vytvorenie cesty prezentácie, bude potrebné sa prepnúť do špeciálneho módu. Keď bude tento mód aktívny, výber nejakého objektu, bude mať za následok pridanie tohto

objektu do cesty. Zároveň sa bude zobrazovať aj samotná cesta v podobe spojených úsečiek, ktoré budú mať krajné body v stredoch objektov, ktoré sú obsiahnuté v ceste.

Na to aby sme vedeli previesť všetky popísané akcie tak ako chceme, musíme sa vedieť pohybovať po ploche. To pri *Zooming User Interface* docielime priblížením, resp. oddialením a posunom plochy. Zmena priblíženia bude možná cez tlačidlá v rozhraní, prípadne rolovaním kolieska myši. Posun bude možný ťahaním prázdneho miesta plochy.

### 5.1.1 Prvky

Dvomi základnými prvkami, ktoré sa budú dať pridať na plochu sú text a obrázky. Aj napriek tomu, že konkurenčné programy ako napr. *Prezi*, ponúkajú väčšie množstvo prvkov, naša aplikácia bude obsahovať systém rozšírení, vďaka ktorému bude možné neskôr doplniť do aplikácie ďalšie prvky.

## 5.2 Prezentačná časť

Pri prezentačnom softvéri, ktorý využíva zobrazovanie informácií pomocou *Zooming User Interface* je veľmi dôležitá plynulosť zobrazovaného obsahu, tak aby divák nestratil prehľad nad zobrazovanými informáciami. Preto je potrebné venovať veľkú pozornosť pri návrhu prezentačnej časti.

### 5.2.1 Cesta prezentácie

Ako sme spomenuli v predchádzajúcich častiach, ZUI prezentačný softvér môže, ale nemusí používať pri samotnej prezentácii cestu, po ktorej sa prezentácia uberá. Rozhodli sme sa, že naša aplikácia sa bude striktne držať cesty. Jediná možnosť kedy bude možné z tejto cesty zísť je opustenie prezentačného módu.

### 5.2.2 Animácia prechodu

Ako už bolo spomenuté, používateľ, resp. divák, sa musí orientovať na zobrazovanej ploche, tak aby nestratil prehľad nad zobrazovanými informáciami. To dosiahneme animáciou, ktorá zjemní prechod medzi jednotlivými objektami. Keďže zobrazované objekty sa môžu líšiť rozmermi, polohou a otočením, je nutné zabezpečiť aby animácia brala do úvahy všetky tri parametre.

Naša stratégia bola nasledovná: vždy keď sme objavili nový model akým by mohla animácia poskytovať dostatočný dojem, implementovali sme prototyp, ktorý sa správal podľa modelu a následne sme ho otestovali. Ak vyhodnotenie testovania neprinieslo požadované výsledky, proces sme zopakovali.

## Lineárna interpolácia

Naivným prístupom by sme dospeli k animácií, ktorá lineárne interpoluje všetky hodnoty medzi dvoma známymi bodmi. Aj napriek jednoduchosti implementácie by neponúkala dostatočné výsledky. Animácia by sa z pohľadu používateľa dala popísať takto:

1. pozeráme sa na prvý objekt, animácia začne
2. prvý objekt zmizne z výhľadu
3. náhodné objekty „prebehnú“ obrazovkou
4. zobrazí sa druhý objekt, animácia skončí

Z popisu animácie už tušíme, že výsledný dojem nebude pre diváka príjemný a preto treba skúsime nejaké lepšie riešenie.

## Easing curve

Možným vylepšením môže byť použitie tzv. *easing curve*, teda „uvoľňovacej krivky“. Tá funguje podobne ako lineárna interpolácia, avšak s jedným rozdielom. Ak by sme chceli animovať presun objektu z bodu  $A$  do bodu  $B$ , v oboch prípadoch by sme dostali trajektóriu pre objekt priamku z  $A$  do  $B$ . Pri lineárnej animácii by mal objekt konštantnú rýchlosť, avšak pri použití *easing curve* by mohol objekt najskôr zrýchľovať a následne spomaľovať – vieme ovplyvňovať beh času nejakou funkciou.

Na prvý pohľad sa to nezdá ako výrazné zlepšenie, avšak toto riešenie poskytuje lepší výsledný dojem z animácie. V reálnom svete sa objekty nezačnú okamžite hýbať alebo okamžite zastaviť, preto na diváka pôsobí animácia organickejšie.

Aj napriek miernemu zlepšeniu, prototyp tejto animácie, ktorý sme testovali, stále trpel rovnakými problémami ako jednoduchá lineárna interpolácia – používateľ stráca prehľad kde sa objekty nachádzajú, aká je medzi nimi vzdialenosť a pod.



## Oddialenie, posun, priblíženie

Možným riešením straty používateľovho prehľadu nad objektami, by mohla byť nasledovná stratégia:

1. oddialenie sa od prvého objektu, aby boli oba objekty viditeľné
2. presun nad druhý objekt
3. priblíženie sa k druhému objektu

Prvým bodom animácie docielime, že používateľ vidí vzájomnú polohu oboch objektov, čím sa dosiahne lepší prehľad medzi súvisom oboch objektov. Vidíme, že pri tomto riešení často využívame vlastnosť *Zooming User Interface* rozhraní – zmenu priblíženia. Obe tieto skutočnosti naznačujú, že ideme správnym smerom.

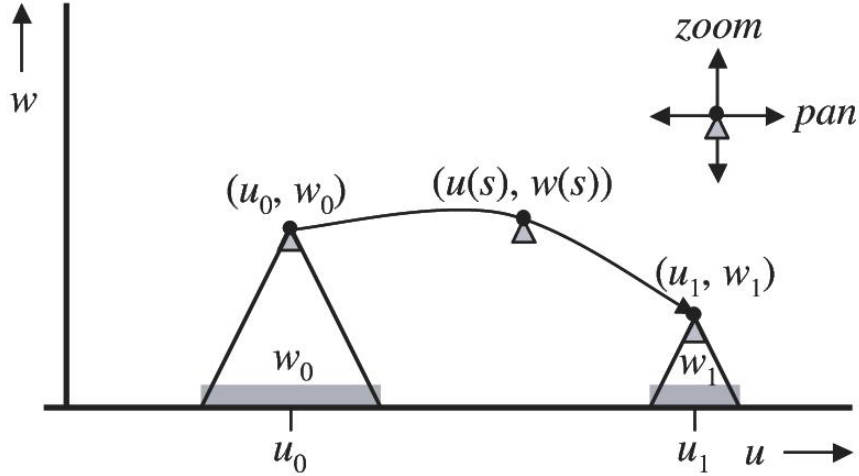
Doposiaľ sme brali do úvahy len mieru priblíženia a posun plochy – nespomenuli sme kde by sa mala odohrávať rotácia. Existujú tri použiteľné alternatívy kedy animovať rotáciu:

- počas oddialenia, resp. priblíženia
- počas presunu z jedného na druhý objekt
- počas celej animácie

Prvá možnosť predstavuje nasledovný scenár: pohľad sa z uhlu rovnému zápornej hodnote otočenia prvého objektu zmení animáciou na nulový uhol, nasleduje posun kedy sa uhol nemení a pokračuje obdobne ako na začiatku celej animácie. Pri vyhodnotení prototypu animácie tohto druhu sme neboli spokojní s výsledkom. Rotácia bola rozkúskovaná, čo vytváralo dojem nejednotnosti.

V druhej alternatíve sa rotácia pohľadu mení pri presune z objektu na objekt. Pri testovaní tejto možnosti bol výsledok lepší ako pri predchádzajúcom postupe, avšak stále sa animácia javila ako nekonzistentná, preto sme skúsili poslednú možnosť.

Posledným dostupným riešením je meniť rotáciu počas celej dĺžky animácie ako sme robili pri lineárnej interpolácii. Aj napriek prvotnému skepticizmu, rotácia s touto stratégiou dosahovala dosiaľ najlepšie výsledky a boli sme rozhodnutí zvoliť tento typ animácie.



Obr. 5.1: Ukážka  $u, w$  diagramu

### Plynulá a efektívna animácia

Hoci sme sa rozhodli použiť vyššie popísanú animáciu, neboli sme úplne spokojní s výsledkom – animácia nebola plynulá. Prezi, popísané v sekcii 4.2.4 na strane 16, poskytuje omnoho príjemnejší a plynulejší prechod medzi dvoma objektami. Preto sme sa snažili zistiť ako je ich animácia definovaná. V [15] sa nám podarilo nájsť referenciu na článok s názvom *Smooth and Efficient Zooming and Panning* [19], kde autor zavádza definíciu pre plynulú a efektívnu animáciu rovnakého typu akú potrebujeme.

Autori na lepšie znázornenie priblíženia a posunu v animácií používajú tzv. *u, w diagramy* [8] (ukážka na obrázku 5.1), kde  $u$  vyjadruje mieru posunutia a  $w$  mieru priblíženia, resp. oddialenia. Body  $(u_0, w_0)$  a  $(u_1, w_1)$  sa dajú chápať ako umiestnenia kamery, ktorá v prvom bode zachytáva objekt s veľkosťou  $w_0$ , pozíciou  $u_1$  a obdobne pre druhý bod. Trajektóriu pre kameru definujú funkcie  $u(s)$  a  $w(s)$ , kde  $s \in [0, S]$  vyjadruje čas, ak rýchlosť animácie je jednotková.

Podobne ako my, autori najskôr skúsili stratégiu „oddialenie, posun, priblíženie“, z ktorého neskôr odvodili optimálnu cestu kamery ponad plochu. Tá sa nachádza na geodetickej krivke v  $u, w$  diagrame a má jeden nastaviteľný parameter  $\rho$ . Ten ovplyvňuje mieru priblíženia – ak je  $\rho = 1$  trajektória kamery leží na polkružnici. Ak je parameter väčší,  $w$  bude mať väčšiu hodnotu, teda plocha sa oddiali viac. Ak je parameter menší, plocha sa oddiali menej. Článok uvádza optimálnu hodnotu parametru  $\rho$ , zistenú používateľskými testami, rovnú  $\sqrt{2}$ .

Pri implementácii prototypu sme avšak narazili na nedostatok popisovaného prístupu – model nepočíta s rotáciou objektov. Skúsili sme popisovanú stratégiu zlúčiť s lineárnou interpoláciou, avšak výsledok nebol konštantne uspokojivý. Dôvodom bolo nezapočítanie času potrebného na rotáciu do koncovej hodnoty animácie  $S$ . Ak bola vzdialenosť dvoch objektov minimálna, avšak ich rotácia sa výrazne líšila, výsledný čas animácie

bol veľmi krátky, teda rotácia musela prebehnúť rýchlo. Vidíme, že rýchlosť rotácie nebude vždy stála, s čím sme neboli spokojní.

## Model pre plynulé zobrazovanie a navigáciu v 2D priestoroch

Autori predchádzajúceho článku pokračovali v odvodzovaní definície pre plynulú a efektívnu animáciu v článku *A Model for Smooth Viewing and Navigation of Large 2D Information Spaces* [20]. Tam používajú rovnaký princíp odvodzovania pre ďalšie lineárne transformácie ako napr. rotáciu, či neuniformné škálovanie. Správne sme predpokladali, že na dosiahnutie konštantnej rýchlosti rotácie potrebujeme zahrnúť rotáciu oboch objektov do  $S$ .

Nasledujúce rovnice vyjadrujú trajektóriu kamery, ktorú implementujeme v aplikácii na animáciu prechodu z jedného objektu na druhý.

$$\alpha(s) = (\alpha_1 - \alpha_0)s/S + \alpha_0 \quad (5.1a)$$

$$u(s) = \frac{w_0}{\rho^2} \cosh r_0 \tanh((r_1 - r_0)s/S + r_0) - \frac{w_0}{\rho^2} \sinh r_0 + u_0 \quad (5.1b)$$

$$w(s) = \frac{w_0 \cosh r_0}{\cosh((r_1 - r_0)s/S + r_0)} \quad (5.1c)$$

$$S = \sqrt{\frac{(\alpha_1 - \alpha_0)^2}{\mu^2} + \frac{(r_1 - r_0)^2}{\rho^2}} \quad (5.1d)$$

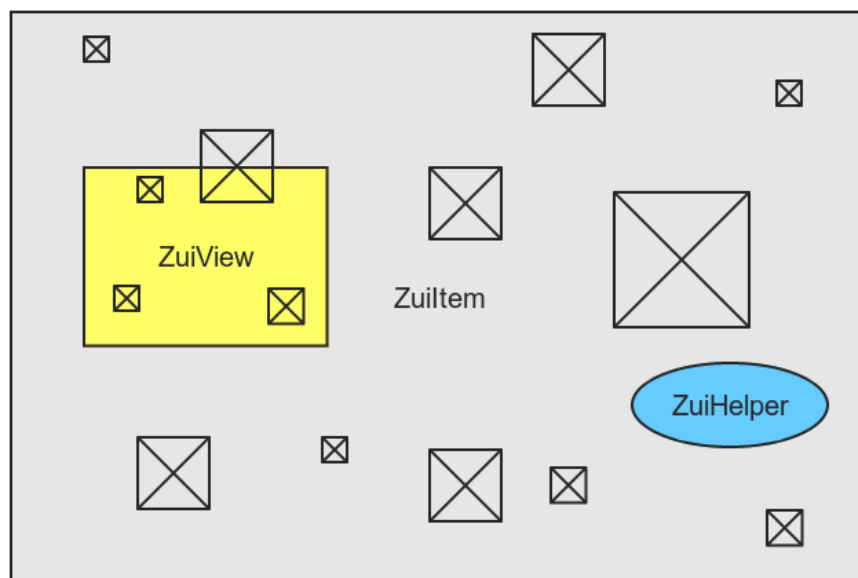
$$r_i = \ln\left(-b_i + \sqrt{b_i^2 + 1}\right), i = 0, 1 \quad (5.1e)$$

$$b_i = \frac{w_1^2 - w_0^2 + (-1)^i \rho^4 (u_1 - u_0)^2}{2w_i \rho^2 (u_1 - u_0)}, i = 0, 1 \quad (5.1f)$$

Ako vidíme, z rovníc (5.1), tak vzťah na vyjadrenie rotácie  $\alpha$  (5.1a) je lineárna interpolácia, ktorú sme skúšali aj predtým. Rozdiel je v zahrnutí  $\alpha_0$  a  $\alpha_1$  v rovnici (5.1d). Pozorný čitateľ si všimneme, že pribudol nový parameter  $\mu$ , ktorý ovplyvňuje rýchlosť rotácie a tým pádom aj celé trvanie animácie. Autori článku opäť empiricky zistili, že optimálna hodnota parametra  $\mu$  je rovná približne  $\sqrt{6}$ .

## 5.3 Architektúra

Najdôležitejšou časťou prezentačného programu je zobrazovacia vrstva. Esenciálna nie je len pri samotnom čine prezentovania informácií, ale aj pri vytváraní prezentácie – rozmiestňovaní objektov po ploche. Architektúra prezentačnej vrstvy, ktorú možno vidieť na obrázku 5.2, sa skladá z troch hlavných komponentov: `ZuiView`, `ZuiItem` a `ZuiHelper`.



Obr. 5.2: Schéma architektúry prezentačnej vrstvy

### 5.3.1 ZuiView

ZuiView je vizuálny komponent, s ktorým sa priamo interaguje, resp. ako jediný sa vkladá do používateľského rozhrania. Poskytuje vysokoúrovňové funkcie ako napr. začať a ukončiť prezentáciu, pridávanie objektov na plochu, priblíženie, či oddialenie, atď.

ZuiView nevykonáva samotné akcie, ale manažuje nízkoúrovňový komponent **ZuiItem**, ktorý zabezpečujú požadovanú funkcionality. Dôvodom pre takúto „obalenie“ funkčného komponentu je niekoľko.

Prvým dôvodom je nutnosť orezania zobrazovaného výstupu (angl. *clipping*), ktorý produkuje **ZuiItem**. Aj napriek tomu, že **ZuiView** bude zobrazený vždy na celé okno aplikácie (teda nie je nutnosť výstup orezávať), počítame so širším využitím tohto komponentu (napr. v iných aplikáciách). Preto je dobré počítať s takouto možnosťou a obaliť **ZuiItem** do **ZuiView**.

### 5.3.2 ZuiItem

Ako je vidieť na schéme architektúry (obr. 5.2), **ZuiItem** nie je obyčajný vizuálny komponent ako **ZuiView**. Aj napriek tomu, že technicky sa jedná o vizuálny prvok (existuje v súradnicovom systéme), tak nemá žiadnu grafickú podobu. Tento objekt slúži ako samotná virtuálna plocha – rodič pre všetky ostatné elementy prezentácie.

**ZuiView**, ktorý disponuje zmenou priblíženia je implementovaný v tomto komponente ako jednoduché škálovanie celého objektu, vrátane jeho detí. Rovnakým spôsobom sa prevádza rotácia a translácia.

Veľmi dôležitou podmienkou pri aplikovaní rotácie a škálovania je nutnosť udržiavať stred otáčania, resp. počiatočný bod škálovania, v strede komponentu `ZuiView`.

### 5.3.3 ZuiHelper

`ZuiHelper` je špeciálny objekt, ktorý nemá grafickú podobu a iba poskytuje informácie o lineárnych transformáciách pre `ZuiItem`. Ak je nutné pri prezentácii vykonať animáciu prechodu z jedného na druhý objekt, tento komponent vypočíta trajektóriu kamery ponad plochu podľa rovníc popísaných v časti Model pre plynulé zobrazovanie a navigáciu v 2D priestoroch na strane 24.

Tento komponent má vlastný časovač, ktorý riadi popísanú animáciu. Ak je vyžiadaný prechod na ďalší objekt počas trvania animácie, `ZuiHelper` automaticky vypočíta novú trajektóriu z aktuálneho pohľadu na cieľový objekt.

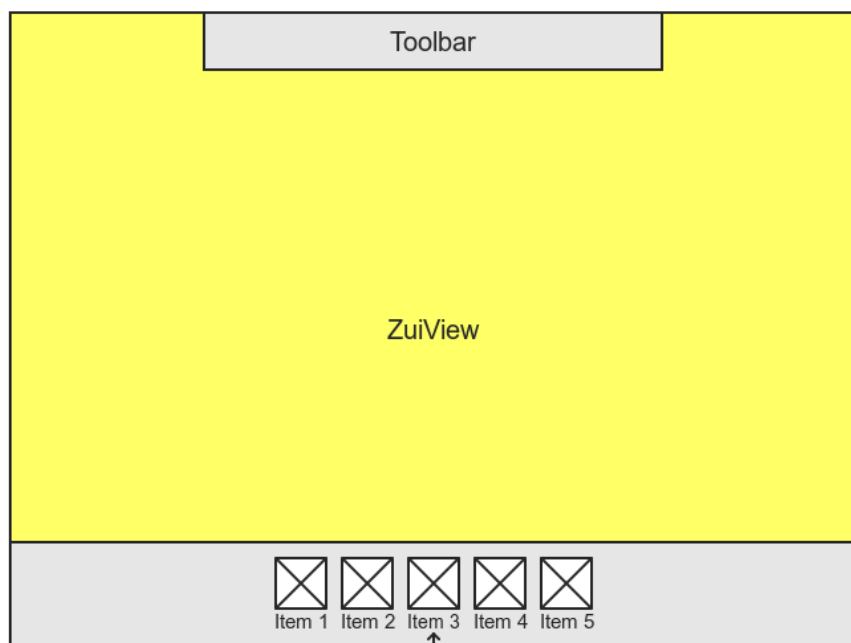
## 5.4 Používateľské rozhranie

Hoci na prezentáciu bude použité ZUI používateľské rozhranie, ostatné prvky aplikácie budú využívať tradičné rozhranie. Aj napriek tomu, že myšlienka ZUI menu v Prezi, je veľmi zaujímavá, pre ciele tejto práce bohužiaľ irelevantná.

Základným východiskom pri návrhu používateľského rozhrania, je snaha čo najviac využiť Fittsov zákon [7]. Dôsledkom tohto rozhodnutia je nutnosť zabráť aplikáciou celú obrazovku, aby okraje a rohy boli v aplikácii prístupné. Do oblastí, ktoré sú podľa tohto zákona najlepšie dosiahnuteľné, umiestnime panely s akciami. Tieto panely však budú maximálny možný čas skryté, aby neprekážali pri navigácii po ploche. Prístupné budú len po aktivácii, čo v našom prípade znamená presun kurzoru na hranu, či do rohu obrazovky.

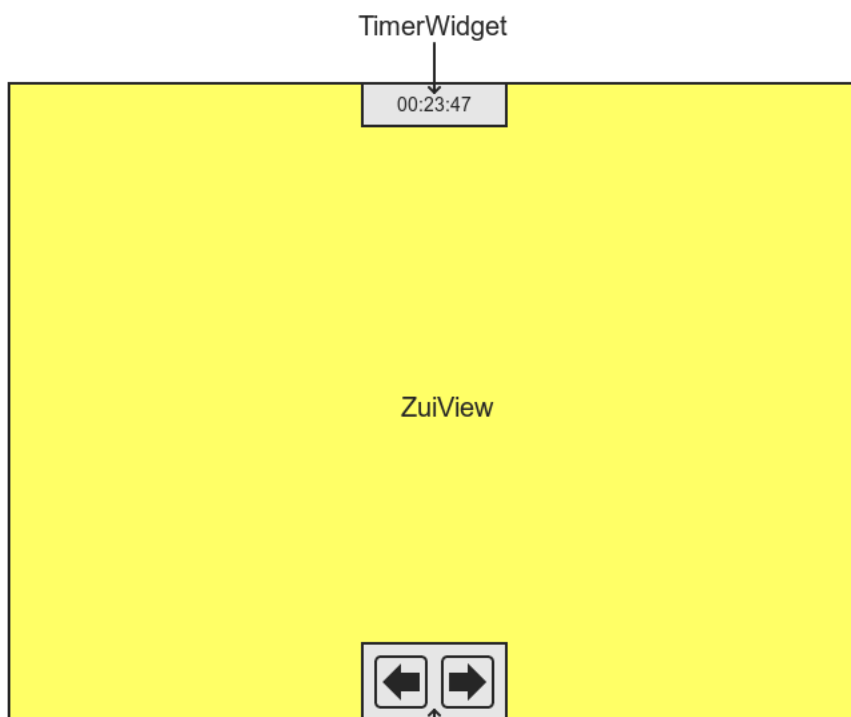
Na obrázku 5.3 môžeme vidieť návrh rozhrania pri tvorbe prezentácie (obr. 5.3a) a pri aktívnom prezentačnom móde (obr. 5.3b). Ako vidíme, používateľské rozhranie je naozaj minimálne. Je však pravdepodobné, že pri implementácii nastanú nejaké zmeny, ktoré však teraz nezohrávajú veľkú úlohu.

Pri tvorbe prezentácie máme k dispozícii hlavné menu a paletu komponentov, ktoré môžeme využiť. Ak používateľ bude využívať aplikáciu na to, na čo bola určená – prezentovanie, bude mať k dispozícii akcie, ktoré mu umožnia prechádzať po ceste prezentácie (`NavigationMenu`) a jednoduchý časovač (`TimerWidget`), ktorý meria čas od spustenia prezentácie.



PalleteMenu

(a) Editácia



NavigationMenu

(b) Prezentácia

Obr. 5.3: Používateľské rozhranie

# Kapitola 6

## Implementácia

### 6.1 Qt

Aplikáciu popísanú v kapitole Návrh práce sme sa rozhodli implementovať s využitím frameworku QtQuick. Tento framework nám ponúka vysokoúrovňové prostriedky, ktoré nám uľahčia celú implementáciu. V nasledujúcej časti popíšeme najdôležitejšie technológie, ktoré sme z tohto frameworku použili.

#### 6.1.1 Graphics View framework

Framework *Graphics View* ponúka prostriedky na vizualizáciu dát pomocou prístupu *model-view*. Základnými triedami pre manipuláciu s týmto frameworkom sú `QGraphicsScene` a `QGraphicsView`.

Framework ponúka mnoho základných prvkov, ktoré sa dajú do scény vložiť a následne vykresliť cez „pohľad“ (`QGraphicsView`). Každý prvok v scéne má vlastný súradnicový systém, do ktorého vieme vložiť ďalšie prvky. Potomkovia nejakého prvku dedia transformačnú maticu aplikovanú na rodičov a pridávajú svoju vlastnú, čím sa dá dosiahnuť vysokej flexibility.

Vykresľovanie môže byť prevedené viacerými spôsobmi. Najpoužívanějšími sú rastrové vykresľovanie, ktoré je veľmi presné, ale je ale náročné na procesorový výkon počítaču a teda nie vhodné pre animácie. Druhým spôsobom je vykresľovanie cez OpenGL, ktoré je síce rýchle (ak neberieme v úvahu celú abstrakciu architektúry frameworku), no nemusí byť podporované na každej platforme.

### 6.1.2 Qt Meta-object Language

*Qt Meta-object Language*, skrátene QML, je deklaratívny programovací jazyk inšpirovaný JavaScriptom, špeciálne navrhnutý na jednoduchý popis dynamických používateľských rozhraní. Každý QML dokument popisuje stromovú štruktúru elementov, ktoré obsahujú identifikátor, definície vlastností, funkcií, atď.

QML intenzívne využíva systém meta-objektov knižnice Qt. Tento mechanizmus zabezpečuje dôležitú vlastnosť jazyka, ktorým je tzv. „property binding“, teda viazanie vlastností objektov. Ak je vlastnosť `foo` objektu A naviazaná na vlastnosť `bar` objektu B, tak pri zmene hodnoty `B.bar` sa automaticky zmení aj hodnota `A.foo`. Tento systém sa na prvý pohľad podobá použitiu smerníkov z iných jazykov, avšak hodnota vlastnosti objektu v jazyku QML môže byť akýkoľvek javascriptový výraz. Tento výraz sa automaticky vyhodnotí vždy keď dôjde k zmene zdrojovej hodnoty vlastnosti. Výpis 6.1 znázorňuje jednoduché použitie tejto vlastnosti – objekt `item2` si bude udržiavať dvojnásobnú veľkosť objektu `item1`.

```
Item {
    id: item1
    width: 10
    height: 10
}
Item {
    id: item2
    width: item1.width * 2
    height: item1.height * 2
}
```

Výpis 6.1: Viazanie vlastností

Okrem viazania vlastností, QML ponúka ďalšiu zaujímavú konštrukciu, ktorá umožňuje definovať stavy objektov a prechody medzi nimi. Stav objektu je množina hodnôt jeho vlastností a prechod určuje akým spôsobom sa mení samotná hodnota vlastnosti. Vo výpise 6.2 je možné vidieť definíciu objektu, ktorý má 2 stavy – *active* a *inactive*. V aktívnom stave sa má zobrazovať ako úplne priehľadný a v normálnej veľkosti, pričom v neaktívnom stave, ktorý je prednastavený, ako polo-priehľadný a len s troj-štvrtinovou veľkosťou. Ak teda zmeníme stav objektu na jeden z definovaných, obe tieto vlastnosti zmenia svoju hodnotu na takú aká je priradená v žiadanom stave.

Nevýhodou tohto prístupu je okamžitá zmena hodnôt vlastností, čo nemusí byť vždy žiadané. Ak potrebujeme plynulú zmenu medzi definovanými hodnotami vlastnosti, môžeme využiť prechody medzi stavmi. V našom prípade vo výpise 6.2 je definovaný jeden prechod, ktorý sa automaticky použije na prechod z aktívneho na neaktívny stav a tiež opačne. Prechod obsahuje „číselnú animáciu“, ktorá zaisťuje plynulú zmenu medzi číselnými hodnotami vďaka interpoláciám medzi hodnotami vlastností z oboch stavov.



```

Item {
    id: item
    state: "inactive"

    states: [
        State {
            name: "active"
            PropertyChanges {
                target: item
                opacity: 1.0
                scale: 1.0
            }
        },
        State {
            name: "inactive"
            PropertyChanges {
                target: item
                opacity: 0.5
                scale: 0.75
            }
        }
    ]

    transitions: Transition {
        NumberAnimation {
            properties: "opacity, scale"
        }
    }
}

```

Výpis 6.2: Stavy a prechody

Definovanie animácií, nech už sú akokoľvek zložité, je v jazyku QML vskutku jednoduché. Vďaka deklaratívnejmu prístupu jazyka môžeme vytvárať hierarchiu sekvenčných a paralelných skupín animácií.

### 6.1.3 QtQuick

*QtQuick* označuje technológiu, ktorá umožňuje vytvárať vysoko dynamické používateľské rozhrania deklaratívnu cestou. Základným kameňom celého frameworku je jazyk QML, spolu s grafickými, ale aj negrafickými elementami, ktoré vieme vytvárať pomocou tohto jazyka.

#### Elementy

Medzi základné grafické prvky patria napr. `Rectangle`, `Image`, `BorderImage`, `AnimatedImage`, `Gradient` alebo aj `Text`. Interakcia s používateľom je riešená veľmi netradične,

pomocou nevizuálnych elementov ako `MouseArea` (stará sa o pohyb a tlačidlá myši), `Keys` (riadi vstup z klávesnice), `PinchArea` (využíva viacdotykové vstupné zariadenia) a ďalších. Existuje mnoho ďalších elementov, ktoré ponúka tento framework, že ich popis by bol sa samostatnú prácu.

## Verzie

Prvá verzia toto frameworku, teda *QtQuick 1.0*, je implementovaná pomocou tried z frameworku *Graphics View*, ktorý sme popísali v tejto časti. Táto skutočnosť nám dovoľuje jednoducho rozširovať funkcionality o nami definované deklaratívne objekty, ktoré môžeme používať priamo z QML. Hoci je táto implementácia aktívne podporovaná a chystá sa revízia 1.1, je známe, že v piatej verzii frameworku Qt bude nahradená verziou 2.0.

Framework *QtQuick 2.0* má s predchádzajúcou verziou len málo spoločné. Hoci sa API (*Application Programming Interface*, v preklade „rozhranie pre programovanie aplikácií“) nezmení (stačí v aplikácii importovať novú verziu), implementácia je úplne odlišná. Tá používa namiesto abstrakcie v podobe *Graphics View*, priame vykresľovanie pomocou OpenGL volaní. Hlavným dôvodom pre túto zmenu bola snaha dosiahnuť čo najlepší výkon, ktorý je žiadaný najmä na mobilných zariadeniach.

## 6.2 Popis implementácie

Vďaka kvalitnému návrhu práce, sme mali s implementáciou veľmi málo problémov. V tejto časti stručne popíšeme rozdiely oproti návrhu, ktoré nám spôsobili problémy.

### 6.2.1 Animácia prechodu

Aj napriek tomu, že sme predpokladali, že s najzložitejšou časťou aplikácie—animáciou prechodu, nastanú problémy, nič také sa neudialo. Problémy nenastali vďaka deklaratívnemu prístupu frameworku *QtQuick*, s ktorým je implementácia podobných typov algoritmov priamočiara. Jediným problémom, ktorý počas programovania tejto časti nastal, bolo nekorektné správanie animácie pri vyžiadaní animácie na ďalší objekt, ak už animácia bežala. Tento problém sme vyriešili miernou zmenou algoritmu, ktorý pri novej animácii berie v úvahu aktuálnu pozíciu kamery a pokračuje z tohto aktuálneho bodu.



Obr. 6.1: Ukážka bežiacej prezentácie

### 6.2.2 Export plátna

Pravdepodobne najväčšou prekážkou bol jeden z cieľov práce, a to export celého plátna do vybraného formátu. Ako je napísané v úvode tejto kapitoly, rozhodli sme sa túto aplikáciu implementovať vo frameworku *QtQuick* v jazyku QML. Obe tieto technológie sú ešte nevyzreté a vývoj stále prebieha, čo má za následok isté obmedzenia. Príkladom takéhoto obmedzenia je nemožnosť pristupovať k nízkoúrovňovej reprezentácii samotných objektov – pixelom. Riešením tohto problému bolo vytvorenie C++ zásuvného modulu do *QtQuick*, ktorý poskytoval QML rozhranie pre požadovanú funkcionality. Aj napriek tomu, že toto riešenie funguje, tento modul nie je kompatibilný s *QtQuick* 2.0, čo znemožňuje využiť v budúcnosti lepšie vykresľovanie, ktoré táto verzia prinesie.

### 6.2.3 Používateľské rozhranie

Vďaka *QtQuick* sme vytvorili používateľské rozhranie bez najmenších problémov. Hoci sa jedná o veľmi jednoduché rozhranie, v ktorom je dostupných pomerne málo funkcií, svojmu účelu slúži dobre. Demonštráciu rozhrania v prezentačnom móde je možné vidieť na obrázku 6.1. Ako sme uviedli v návrhu práce, panely s akciami sú normálne skryté. Až po ich aktivácii sa animáciou vysunú a po použití sa naspäť plynulo zasunú.

Nepříjemnou vlastnosťou programu je nutnosť použiť softvérové vykresľovanie na dosiahnutie prijateľného grafického výstupu.

# Kapitola 7

## Zhodnotenie

Táto práca nám poskytla mnoho teoretických, ale najmä praktických poznatkov, ktoré sme pri jej tvorení osvojili. Umožnila nám preskúmať netradičný, no o to zaujímavejší typ používateľského rozhrania – *Zooming User Interface*. Implementovali sme sofistikovanú animáciu, určenú na príjemné sledovanie a navigáciu v takýchto rozhraniach. A do tretice sme objavili výhody, avšak aj slabiny frameworku *QtQuick*.

Výsledkom tejto bakalárskej práce je jednoduchý nástroj na vytváranie a prehrávanie prezentácií na báze *Zooming User Interface*. Ciele, ktoré sme mali stanovené, sa nám podarilo zväčša naplniť. Aplikáciu sme dokonca otestovali aj v reálnych podmienkach – pri ozajstnej prezentácii. Tento test potvrdil, že aj napriek svojej jednoduchosti, je výsledná aplikácia použiteľná.

Aj napriek tomu, že aplikácia prešla testom použiteľnosti, neznamená to, že je hotová. Existuje ešte veľký priestor na vylepšenie. V budúcnosti bude možné vďaka novej verzii *QtQuick*, použiť viacvláknové vykresľovanie cez OpenGL, ktoré ponúkne lepší výkon a kvalitnejší výstup. Rozhranie pre tvorbu prezentácií by mohlo byť omnoho intuitívnejšie a obsahovať viac funkcií. Zaujímavou možnosťou by bolo preskúmať použitie ZUI pre iné typy aplikácií, či dokonca ako hlavné používateľské rozhranie operačného systému, na stolových počítačoch alebo mobilných zariadeniach.

# Literatúra

- [1] Prezi - the zooming presentation editor. <http://prezi.com/>.
- [2] Qt Reference Documentation. <http://doc.qt.nokia.com/4.7/index.html>.
- [3] B.B. Bederson, J. Grosjean, and J. Meyer. Toolkit design for interactive structured graphics. *Software Engineering, IEEE Transactions on*, 30(8):535 – 546, aug. 2004.
- [4] Benjamin B. Bederson and James D. Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, UIST '94, pages 17–26, New York, NY, USA, 1994. ACM.
- [5] Benjamin B. Bederson, Jon Meyer, and Lance Good. Jazz: an extensible zoomable user interface graphics toolkit in Java. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, UIST '00, pages 171–180, New York, NY, USA, 2000. ACM.
- [6] Benjamin B. Bederson and Jonathan Meyer. Implementing a Zooming User Interface: Experience Building Pad++. *Softw., Pract. Exper.*, 28(10):1101–1135, 1998.
- [7] Paul M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. 1992.
- [8] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams: understanding multiscale interfaces. pages 234–241, 1995.
- [9] Donald R. Gentner and Jakob Nielsen. The Anti-Mac Interface. *Commun. ACM*, 39(8):70–82, 1996.
- [10] Lance Good and Ben Bederson. Counterpoint: Creating jazzy interactive presentations. Technical report, 2001.
- [11] Lance Good and Benjamin B. Bederson. Zoomable user interfaces as a medium for slide show presentations. *Information Visualization*, 1(1):35–49, 2002.

- [12] D. Austin Henderson Jr. and Stuart K. Card. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Trans. Graph.*, 5(3):211–243, 1986.
- [13] Jakob Nielsen. Noncommand User Interfaces. *Commun. ACM*, 36(4):82–99, 1993.
- [14] Ken Perlin and David Fox. Pad: an alternative approach to the computer interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 57–64, New York, NY, USA, 1993. ACM.
- [15] Zoli Radnai. New features based on your ideas. <http://blog.prezi.com/2010/04/06/new-features-based-on-your-ideas/>, April 2010.
- [16] Jef Raskin. *The humane interface: new directions for designing interactive systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.
- [17] T. Reenskaug. Models-views-controllers. *Technical note, Xerox PARC, December*, 1979.
- [18] Andries van Dam. Post-wimp user interfaces. *Commun. ACM*, 40(2):63–67, 1997.
- [19] Jarke J. van Wijk and Wim A. A. Nuij. Smooth and efficient zooming and panning. *Information Visualization, IEEE Symposium on*, 0:3, 2003.
- [20] Jarke J. van Wijk and Wim A. A. Nuij. A model for smooth viewing and navigation of large 2D information spaces. *IEEE Trans. Vis. Comput. Graph.*, 10(4):447–458, 2004.
- [21] Wikipedia. Keynote (presentation software) — Wikipedia, The Free Encyclopedia, 2011. [Online; accessed 3-June-2011].
- [22] Wikipedia. Microsoft Powerpoint — Wikipedia, The Free Encyclopedia, 2011. [Online; accessed 3-June-2011].
- [23] Wikipedia. Openoffice.org — Wikipedia, The Free Encyclopedia, 2011. [Online; accessed 3-June-2011].

# Príloha

## Zdrojové kódy aplikácie

Kompletné zdrojové kódy aplikácie sa nachádzajú na priloženom médiu a na internetovej adrese <https://gitorious.org/zuiq>.