Mendelova univerzita v Brně
Provozně ekonomická fakulta

Implementace moderních multidotykových aplikací s využitím knihovny Qt

Diplomová práce

Vedoucí práce:

Ing. David Procházka, Ph.D. **Bc. Bronislav Ryba**

Brno 2014

Rád bych poděkoval vedoucímu diplomové práce Ing. Davidu Procházkovi, Ph.D. za příkladné vedení této práce. Děkuji za připomínky, cenné rady, zkušenosti a nápady, které mi poskytl během vypracování této práce.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Implementace moderních multidotykových aplikací s využitím knihovny Qt**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 29. prosince 2014 _____

Abstract

Ryba, B. *The implementation of modern multitouch applications by using Qt library*. Diploma thesis. Brno: Mendel University, 2014.

This thesis deals with the issue of creating GIS type multitouch applications by using Qt library. Introductory part of this work discusses the current desktop and mobile solutions in the field of GIS with evaluations of user interfaces. It also contains a description of the available map services and options of the Qt library in the area of modern applications creation for touch screens. The following part is devoted to defining user and designing GIS with attention to user-friendly interface suitable for touch screens. The outcome of this work is the implementation of the proposed GIS having a modern multitouch user interface.

Keywords

GIS, user interface, Qt, QML

Abstrakt

Ryba, B. *Implementace moderních multidotykových aplikací s využitím knihovny Qt*. Diplomová práce. Brno: Mendelova univerzita v Brně, 2014.

Diplomová práce se zabývá problematikou tvorby multidotykových aplikací typu GIS pomocí knihovny Qt. Práce v úvodní části pojednává o současných desktopových a mobilních řešeních v oblasti GIS s hodnocením jejich uživatelských rozhraní. Dále obsahuje popis dostupných mapových služeb a možností knihovny Qt v oblasti tvorby moderních aplikací pro dotykové obrazovky. Následující část práce je věnována definování uživatele a návrhu vlastního GIS s pozorností kladenou na přívětivé uživatelské rozhraní vhodné pro dotykové obrazovky. Výstupem práce je pak implementace navrženého GIS disponujícího moderním multidotykovým uživatelským rozhraním.

Klíčová slova

GIS, uživatelské rozhraní, Qt knihovna, QML

Obsah

Seznam obrázků

1 Úvod a cíl práce

1.1 Úvod

Už tomu bude více než 30 let, kdy doktor George Samuel Hurst na univerzitě v Kentucky představil světu svůj první dotykový senzor, který si nechal o několik let později patentovat (Espacenet.com, 2014). Nejednalo se sice o moderní dotykovou obrazovku, jakou máme dnes například u svého mobilního telefonu, byl to však vynález, jenž položil základy budoucího dynamického rozvoje této technologie až do podoby, jak ji zná, a v drtivé většině i využívá, snad každý z nás. Tato technologie a její následný rozvoj také umožnil nástup trendu mobilních zařízení všeho druhu.

Nacházíme se v době, kdy se můžeme s dotykovými obrazovkami různých typů setkat téměř všude počínaje mobilním telefonem, nebo tabletem a domácími spotřebiči, jakým je například pračka či mikrovlnná trouba, konče. Všechna tato zařízení však mají něco společného. Měla by sloužit svému uživateli. A aby mohla dobře sloužit, je nutné zajistit to, aby uživatel, jenž zařízení obsluhuje, dokázal zařízení správně ovládat. Měla by tedy být uživatelsky přívětivá. K tomu, aby byl uživatel schopen zařízení či aplikaci správně ovládat, slouží jednak informace, kterých se uživateli dostává obvykle právě z obrazovky daného zařízení, ale především způsob, jakým jsou tyto informace uživateli zprostředkovány, tedy zobrazeny na onom zařízení. Ke zmiňované uživatelské přívětivosti pak také obrovskou měrou přispívá i způsob, jakým uživatel s daným zařízením komunikuje. A tuto podstatnou funkci plní dnes již povětšinou dotykové obrazovky s možností využití gest, která jsou pro člověka mnohem přirozenější.

Technologii dotykových obrazovek se dnes těší větší pozornosti ve stále více oblastech a také díky tomu, že dnešní zařízení již disponují i více dotykovým ovládáním, stávají se lukrativnější také pro větší firmy. Zejména podniky věnující se GIS v současné době zaměřili svou pozornost na tento způsob ovládání. Důkazem tomu mohou být například společnosti *TouchShare* se svým 46 palcovým displejem (TouchShare.com, 2013) a *Esri* se svým projektem *Landsat Touch* (Esri, 2014). Oba tyto projekty se zaměřují na využití geodat a jejich manipulaci prostřednictvím vícedotykových velkoplošných obrazovek. Už i z těchto 2 zmíněných ukázek je názorně vidět, jaký přínos může spojení geografických informačních systémů (GIS) a dotykových obrazovek přinést do svých oborů. Navíc v dnešní době, kdy máme k dispozici vysokorychlostní internetové připojení, se přímo nabízí jejich využití zejména v oblasti jednotlivých záchranných složek, kde by potenciál těchto technologií mohl být využit například pro plánování, rozmísťování jednotek či vzájemnou kolaboraci jednotlivých týmů či jako nástroj pro podporu manažerského rozhodování (Machalová, 2007). Správně navržený GIS dokáže při kvalitní realizaci významně urychlit a zjednodušit činnosti, které by si za jiných okolností vyžadovaly například složitou manipulaci s mapami v papírové podobě, nebo paralelní využití několika různých zařízení.

V této práci se budu této problematice věnovat o něco blíže. Pokusím se zde zachytit výhody realizace GIS pro dotyková zařízení oproti jejich realizacím určeným primárně pro klasické počítače, u kterých je obvykle nutné k jejich obsluze ještě využívat periferie typu klávesnice a myš. Blíže se zaměřím také na několik již existujících řešení a provedu základní zhodnocení současné situace z pohledu uživatelské přívětivosti, možnosti využití pro dotykové obrazovky a podobně. Pokusíme se zachytit výhody i nedostatky jednotlivých řešení. Po provedení analýzy těchto existujících realizací vytvořím návrh vlastního GIS, který bude určen primárně pro využití dotykové obrazovky. K následné implementaci bude využita knihovna Qt, u které se také nejdříve podívám na možné způsoby toho, jak lze podobné aplikace vytvářet (Summerfield, Blanchette, 2008). Po samotné implementaci navrhovaného řešení provedu zhodnocení samotné realizace a navrhnu případně další kroky, které by bylo možné udělat proto, aby se výsledná aplikace mohla dále rozšiřovat a tím dostat blíže samotnému koncovému uživateli, pro kterého bude určena.

1.2 Cíl práce

Cílem práce bude prozkoumat možnosti knihovny Qt s ohledem na tvorbu aplikací pro dotykové obrazovky a demonstrovat možnosti knihovny na aplikaci zpracovávající geodata. Pro splnění tohoto cíle bude nutné prozkoumat stávající situaci v oblasti tvorby a řešení aplikací určených pro zpracování a manipulaci s geodaty. Následně také nastudovat možnosti knihovny Qt v oblasti tvorby multidotykových uživatelských rozhraní. Na základě tohoto průzkumu následně navrhnout aplikaci pro dotyková zařízení včetně moderního uživatelského rozhraní, která poskytne základ pro další vývoj a rozšiřování této aplikace.

2 Dostupné GIS

Geografický informační systém neboli GIS je, jak už z definice samotného informačního systému vyplývá, systém pro získávání, ukládání, analýzu a vizualizaci dat, která však v tomto případě mají prostorový vztah k zemskému povrchu. Existuje celá řada softwarových řešení, která jsou schopna vykonávat s geodaty uvedené operace. Tato řešení mohou být založená na různých přístupech, obvykle odvozeného od hlavního účelu využití, jako je například vytvoření aplikace přístupné primárně prostřednictvím webového prohlížeče (*OpenLayers, GeoMoose,…*), nebo jako desktopová či mobilní aplikace.

Tato část práce je zaměřena na desktopové a mobilní řešení GIS. A jelikož existuje nepřeberné množství tzv. open source (dále svobodný software) aplikací, nejsou zde uvedena pouze známá komerční řešení, ale také několik rozšířených a často využívaných svobodných software, jenž jsou ještě doplněny o moderní mobilní verze pro dotykové obrazovky. V rámci této kapitoly je proveden stručný popis jednotlivých aplikací, včetně krátkého přehledu základní funkcionality a především analýza a hodnocení uživatelských rozhraní.

2.1 Svobodné GIS

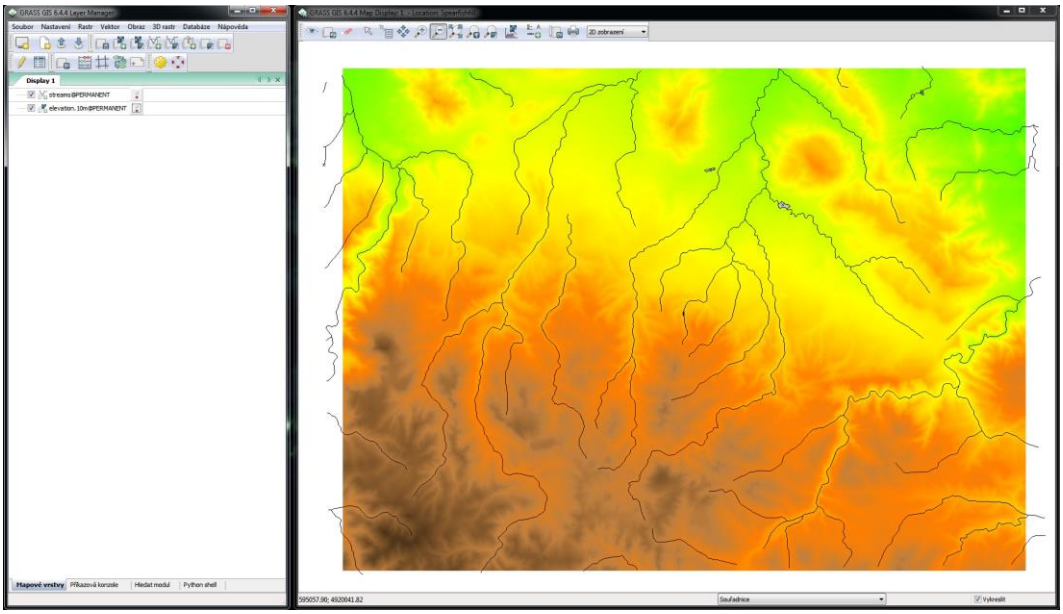
Tato část práce pojednává o GIS, které jsou k dispozici zdarma a v rámci svých licencí nabízí otevřeně i svůj kód. Svobodných GIS existuje celá řada a mají velmi širokou oblast využití. Níže uvedené jsou ty, které by se dalo považovat za nejrozšířenější.

2.1.1 GRASS GIS

Geographic Resources Analysis Support System (GRASS) je oficiálním projektem nadace *Open Source Geospatial* a je nejstarším a největším dostupným svobodným GIS, který byl původně vyvinut *U. S. Army Construction Engineering Research Laboratories* jako nástroj pro hospodaření s půdou a plánování v oblasti životního prostředí. Jedná se o otevřený multiplatformní software pod GNU *General Public License* (GPL) licencí, který má dnes obrovské spektrum využití v oblastech aplikací a vědeckého výzkumu. GRASS je v současné době využíván v akademické i komerční sféře po celém světě, stejně tak v mnoha vládních agenturách, včetně *NASA*, *NOAA*, *DLR*, *CSIRO*, a v mnoha poradenských firmách v oboru životního prostředí.

GRASS čítá více než 400 vestavěných analytických modulů a přes 100 dalších modulů a nástrojů je k dispozici zdarma na wiki stránkách komunity, kterou dnes tvoří členové z mnoha míst. GRASS obsahuje nástroje pro prostorové modelování, vizualizaci rastrových a vektorových dat, správu a analýzu geoprostorových dat a zpracování družicových a leteckých snímků. Poskytuje možnosti pro tvorbu sofistikovaných prezentačních grafických výstupů a tištěných map. Je přeložen do dvaceti jazyků a podporuje širokou škálu formátů dat. GRASS lze také navíc použít jako zásuvný modul do některých z ostatních GIS. (Grass, 2014)

- Výčet vybraných funkcí:
    - Široká škála podporovaných formátů
    - Rastrové, vektorové i 3D rastrové analýzy
    - Podpora *SQL*
    - Zpracování obrazu (UAV, satelitní snímky, detekce hran,…)
    - Vizualizace
- Některé podporované standardy (pro přehlednost budou uváděny pouze zkratkou a budou vysvětleny v následující kapitole):
    - WMS
    - WFS
    - WPS



Obr. 1 Grafické uživatelské rozhraní dvouoknového GRASS GIS

GRASS GIS je možné spouštět s různým uživatelským rozhraním. První z možností je spustit aplikaci pouze s textovým uživatelským rozhraním, tzn. komunikovat s aplikací prostřednictvím příkazové řádky. Dále je možné zvolit rozhraní *Tcl/Tk* či starší verzi *old Tcl/Tk*. Na obrázku Obr. 1 je však ukázka nejnovějšího typu řešení uživatelského rozhraní, které je v GRASS GIS k dispozici. Jedná se o grafické rozhraní *wxPython*, u kterého byl zvolen přístup 2 samostatných oken, kde první z nich slouží jako okno pro veškerá nastavení, dále pak jako tzv. manažer vrstev a také jako středisko všech dostupných nástrojů pro manipulaci s geodaty. Výsledky jednotlivých operací v podobě map se pak zobrazují v okně druhém, ve kterém jsou k dispozici pouze nástroje mající vliv pouze na konečnou podobu zobrazeného výstupu a sloužící pro manipulaci se zobrazovaným výstupem. Rozdělení aplikace

na 2 samostatná okna se jeví jako poměrně logický krok vzhledem k tomu, jakému účelu aplikace slouží. Uživatel tak má možnost, především využívá-li více monitorů, zobrazit si své výstupy přes celou obrazovku aniž by mu nějakým způsobem zavazela jiná okna či nástroje, které v danou chvíli nepotřebuje.

Jelikož se jedná o pokročilý nástroj, předpokládá se od uživatele určitá znalost jednotlivých nástrojů a postupů, které je nutné vykonat k dokončení daných úkolů. Z této skutečnosti zřejmě vychází množství a především rozložení jednotlivých nabídek, kterých je opravdu mnoho a jsou velmi často řešeny prostým seznamem, což poměrně znesnadňuje práci. Pokud tedy uživatel hledá jeden konkrétní nástroj, který se zrovna nenachází mezi hlavními nástroji přímo na obrazovce, a navíc danou aplikaci třeba příliš nezná, může strávit poměrně dost času procházením veškerých nabídek. K urychlení tohoto hledání byla do dolní části hlavního okna aplikace umístěna možnost *Hledat modul*, kde může uživatel hledat na základě klíčového slova, příkazu či popisu požadovaný nástroj a velmi si tak usnadnit hledání. Velmi častým prvkem jsou také tzv. karty, které jsou zde využívány jak pro správce stromu vrstev, tak pro jednotlivý dialogová okna pro přidání či editaci vrstev.

Dané prostředí uživateli nenabízí také příliš mnoho možností, jak si uzpůsobit rozhraní tak, aby mu co nejlépe vyhovovalo. Lze si pouze libovolně rozmístit jednotlivé skupiny nástrojů, ne však přidávat či odebírat nástroje v rámci těchto definovaných skupin.

Shrnutí: grafické uživatelské rozhraní GRASS GIS není jeho příliš sinou stránkou a zůstává klasickým příkladem statického desktopového uživatelského rozhraní. Vzhledem ke zmiňovanému množství a rozložení jednotlivých nabídek není vhodné ani pro dotykové obrazovky, kde jsou prvky jako karty a seznamy poměrně nepraktické, především pak v podobě, v jaké jsou realizovány v této aplikaci. Je zjevné, že při tvorbě tohoto rozhraní bylo hlavním cílem pouze nahradit dřívější zadávání příkazů do příkazové řádky a nebylo příliš dbáno na uživatelskou přívětivost

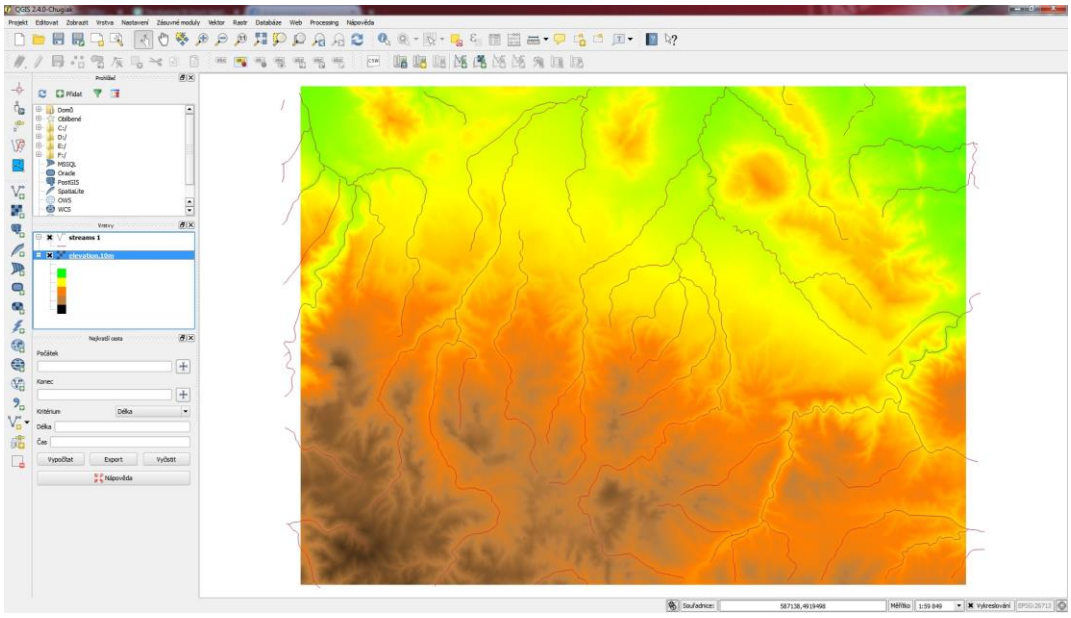
2.1.2 QGIS (Quantum GIS)

Jedná se o GIS s otevřeným kódem, který se začal vyvíjet již v roce 2002. Je dostupný pro všechny běžné platformy *Windows*, *Linux*, *OS X*, ale vzniká i verze pro mobilní zařízení se systémem Android. Pro jeho tvorbu je využíván programovací jazyk *C++* a knihovna Qt. QGIS si zakládá na tom, aby byl přívětivý a zvládli jej používat i uživatelé, kteří mají pouze základní znalosti v oblasti počítačů.

Základním cílem projektu bylo poskytnout uživateli prohlížeč geografických dat. Momentálně se dosáhlo bodu ve vývoji, kdy QGIS obsahuje velké množství funkcí, které obsáhnou většinu našich potřeb pro práci s vektorovými i rastrovými geodaty. Podobně jako u ostatních GIS, tak umožňuje vytvářet mapy s větším množstvím vrstev a různými způsoby zobrazení bez toho, aby byl nutný nějaký dodatečný převod do vnitřního formátu.

Distribuce tohoto GIS probíhá pod licencí GNU GPL, která zajišťuje, že další vývoj QGIS bude mít vždy přístupný zdrojový kód, bude zdarma a bude moci být dále upravován. (QGIS, 2014)

- Výčet vybraných funkcí:
    - Prohlížení dat
    - Analýzy dat
    - Tvorba map
    - Tvorba, úprava, správa a export dat
    - Sdílení map prostřednictvím internetu
- Některé podporované standardy:
    - WMS
    - WMTS
    - WFS
    - WFS-T
    - WCS



Obr. 2 Grafické uživatelské rozhraní hlavní obrazovky QGIS

Aplikace QGIS disponuje stejně jako GRASS GIS velkým množstvím nástrojů, což je i na první pohled možné vyčíst z grafického uživatelského rozhraní (Obr. 2). Na rozdíl od předcházejícího řešení, je však rozhraní čitelnější a vizuálně přitažlivější. Seznamových nabídek je zde výrazně méně. Také nejsou natolik rozsáhlé, nýbrž jsou rozdělené do více kategorií a podkategorií. Tyto seznamy a jednotlivé položky v nich jsou také často doplněny o ikony, což velmi usnadňuje orientaci v těchto nabídkách. Velké množství zobrazených skupin nástrojů je možné si také velmi jednoduše rozmístit podle vlastních představ, případně si některé schovat či naopak zobrazit nabídky nové. Skrývat a zobrazovat skupiny nástrojů je možné přímo

z jednotlivých panelů nebo záložky nastavení na hlavní liště. QGIS také disponuje širokými možnostmi nastavení aplikace. Pro úpravu vzhledu jsou však tato nastavení pouze kosmetická a velmi podobná těm z GRASS GIS (změna sady ikon, změna celkového stylu). Užitečné jsou však možnosti pro nastavení velikosti písma a zejména pak ikon. Jako jeden z nejvíce rušivých elementů se naopak jeví velmi častý výskyt rozbalovacích nabídek ve všech částech aplikace.

Shrnutí: rozhraní QGIS je stále především desktopové, ale mnohem příjemnější než je rozhraní, kterým disponuje předešlá aplikace. Dalším plusem je možnost využití aplikace na dotykových obrazovkách, jelikož umožňuje například přiblížení a pohyb s mapu pomocí gest (ikona umístěna také na hlavním panelu). Pokud si však chceme užít větší komfort na dotykové obrazovce, vyplatí se projít si možnosti nastavení a aplikaci si lépe nastavit.

2.1.3 OpenJUMP

OpenJUMP je také otevřený svobodný GIS, založený na programovacím jazyce *Java*, který je primárně určen pro práci s vektorovými daty. Byl vyvinut a nyní stále udržován skupinou dobrovolníků z celého světa a podporuje platformy *Windows*, *Linux* a *OS X*.

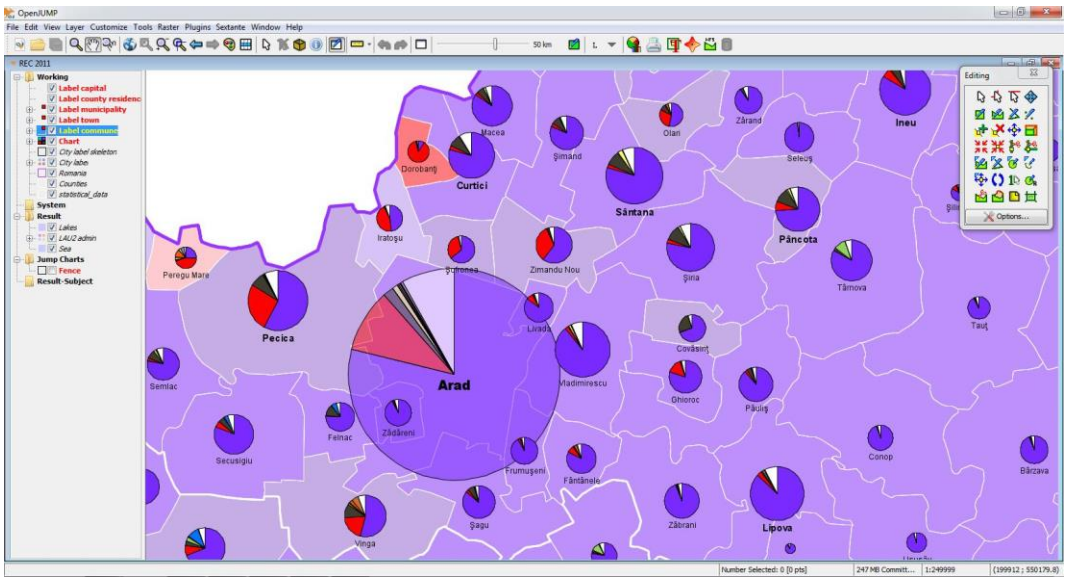
V aktuální verzi je možné číst a vtvářet shapefily[1] a jednoduché GML soubory. Disponuje kvalitní podporou pro zobrazení dat obdržených z webových služeb WFS a WMS. Lze jej tedy použít i jako prohlížeč dat. Jeho hlavní síla ovšem spočívá v úpravách geometrie a atributů. Můžeme také upravovat podobu dat v mapovém displeji OpenJUMP a následně exportovat do SVG. K dispozici je také stále rostoucí počet analytických nástrojů pro topologické analýzy a operace překrytí. Vydání verze 1.4 přineslo nové možnosti zpracování rastrových dat a lepší použitelnost EZ tlačítek (tvorba zkratek), inverzní výběr, uživatelsky definovaný vzhled výběru a spoustu oprav menších i větších chyb.

Limity má OpenJUMP například při čtení velmi velkých datových souborů, disponuje pouze omezenou podporu kartografických projekcí a ne všechna funckionalita je dostupná v základní verzi aplikace a je tedy nutné si stáhnout některé zásuvné moduly.(OpenJUMP GIS, 2011)

- Výčet vybraných funkcí:
    - Podpora mnoha formátů pro zápis i čtení
    - Široká škála možností úprav a manipulace s vektorovými daty
    - Analýzy a dotazování
    - Skriptování pomocí *BeanShell* a *Java Python*

[1] Shapefile je otevřený formát vyvinutý společností pro vektorová data. Ukládá geometrie a hodnoty atributů pro elementy v prostorových datech. Geometrie pro jednotlivé prvky jsou uloženy jako tvary, které obsahují sadu vektorových souřadnic. (Esri, 2008)

- Některé podporované standardy:
    - WFS
    - WFS-T
    - WMS



Obr. 3 Grafické uživatelské rozhraní OpenJUMP s Pie-Chart zásuvným modulem
Zdroj: Jump, 2014.

Rozhraní aplikace OpenJUMP (Obr. 3) se řadí opět do skupiny klasických desktopových uživatelských rozhraní. Nachází se zde hlavní nabídka v podobě seznamů s nástroji, dále pak klasická lišta nejpoužívanějších nástrojů, kterou je možné pouze přemístit, ale nelze nijak editovat. V rámci hlavního okna je pak možné vytvářet vlastní projekty, kde každý projekt je řešen vlastním děleným podoknem, přičemž levá část zastupuje správce stromu vrstev a pravá slouží jaké prostor pro zobrazení mapových výstupů. Stejně jako GRASS, ani OpenJUMP nedisponuje rozhraním, které by bylo vhodné pro dotykové obrazovky. Není zde ani žádná možnost si rozhraní libovolně přizpůsobit či alespoň změnit velikost zobrazovaných ikon, jako je tomu např. u QGIS.

Shrnutí: OpenJUMP také nedisponuje rozhraním, u kterého by byl kladen větší důraz na uživatelskou přívětivost. Orientace pro neznalé uživatele není příliš intuitivní a především základní úkoly, jako přidat libovolnou vrstvu a upravit si její parametry, zabere poměrně dost času.

2.1.4 uDig

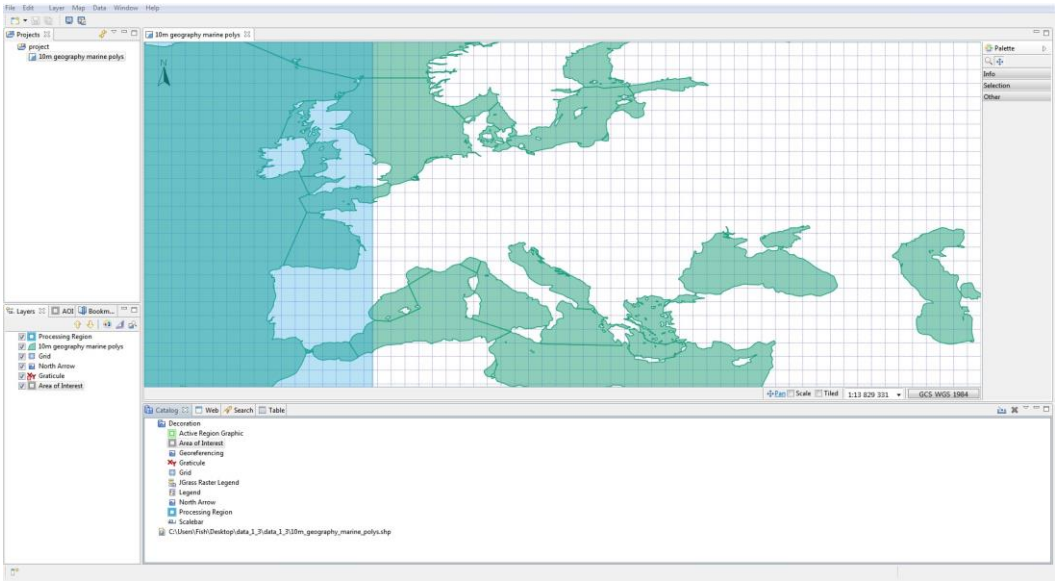
Jedná se je volně šiřitelný otevřený framework (pod licencemi EPL a BSD) na stolní počítače vytvořený pomocí technologie *Eclipse Rich Client* (RCP[2]).

uDig (*User-friendly Desktop Internet GIS*) je internetově orientovaný na webové služby (zejména WMS, WFS, WPS) a de facto se jedná o geoprostorovou webovou službu (GeoRSS, KML, dlaždice). Poskytuje rámec, na kterém lze vytvářet složité analytické funkce, jenž mohou být následně zahrnuty do hlavní aplikace. Lze jej použít také jako samostatnou aplikaci, rozšířit ji o zásuvné moduly RCP, nebo může uDig sloužit jako zásuvný modul v existující RCP aplikaci. Cílem uDig je umožnit kompletní řešení přístupu ke geodatům, jejich úpravě a prohlížení v GIS pro stolní počítače v jazyce *Java*. Uživatelům GIS poskytuje známé grafické prostředí a je spuštěn lokálně jako tlustý klient. Provozovat jej je možné pod systémy *Windows*, *OS X* a *Linux*. (UDig, 2008)

- Výčet vybraných funkcí:
    - *Drag and drop* integrace do průzkumníku či prohlížeče
    - Práce s běžnými formáty dat, ale i pokročilými rastrovými formáty
    - Podpora webových mapových služeb
    - *Style layer descriptor* pro publikování map
- Některé podporované standardy:
    - WMS
    - WFS
    - WFS-T
    - WMS-C

Na obrázku níže (Obr. 4) je ukázka uživatelského rozhraní posledního z uvedených svobodných GIS v této práci. Při realizaci rozhraní této aplikace byl zvolen jiný přístup, než je tomu u aplikací předešlých. Hlavní okno aplikace je realizováno jako dělené okno. Na první pohled by se mohlo zdát, že jednotlivých podoken je příliš mnoho, ale okna slouží skvěle sému účelu a je navíc možné si libovolně upravit jejich rozložení a zobrazení v rámci hlavního okna. Hlavní lišta je ve srovnání s ostatními desktopovými aplikacemi velmi zredukována, stejně jaké množství nástrojů na hlavní liště. Vše je lépe rozmístěno a kategorizováno. Při vykonávání jednotlivých úkolů uživateli také vždy napomáhá přehledný průvodce jednotlivými kroky, což výrazně usnadňuje práci a nijak neodrazuje uživatele od využívání této aplikace.

[2] RCP neboli *Rich Client Platform* je sada zásuvných modulů potřebných k vytvoření *rich client application*.



Obr. 4 Grafické uživatelské rozhraní aplikace uDig

Shrnutí: na první pohled nepřehledná dělená okna se ukazují jako velmi účelná a ve výsledku uživateli velmi usnadňují, zpřehledňují a tím i urychlují práci. Uživatelské přívětivosti zde by věnován výrazně větší prostor než u předešlých aplikací. Stále se ale nedá říci, že by rozhraní bylo ideální pro využívání na dotykových obrazovkách. Opět je zde problém s velikostí ikon a realizace některých nabídek.

2.2 Komerční GIS

Komerčních produktů zaměřených na manipulaci s geodaty existuje také poměrně velké množství. Spousta velkých společností těží z potenciálu, který se v této oblasti technologií skrývá. V komerční sféře jednotlivé podniky obvykle nenabízí pouze desktopové řešení GIS, ale obvykle nabízí k dispozici také webové mapové servery. Některé svou nabídku doplňují také o systém řízení báze dat či mobilní verze svých aplikací.

2.2.1 ArcGIS for Desktop

ArcGIS je zřejmě nejznámějším řešením na poli GIS. Jedná se o balík profesionálních nástrojů vyvinutých americkou společností *Esri*, který je dostupný pod třemi úrovněmi licencí na základě šíře funkcionality, která je uživateli k dispozici. Těmito úrovněmi jsou *Basic*, *Standard*, *Advanced* mající stejné základní integrované aplikace, uživatelské rozhraní a vývojové prostředí.

- **Basic** (dříve ArcView): umožňuje zobrazení a základní manipulace s prostorovými daty a včetně základních prostorových analýz. Tvorba map na základě vrstev.

- **Standard** (dříve ArcEditor): funkcionalita *Basic* licence + pokročilé kartografické nástroje, možnosti geodatabází a širší možnosti geoprocessingu[3].
- **Advanced** (dříve ArcInfo): obsahuje funkcionalitu *Standard* doplněnou především o pokročilé nástroje geoprocessingu. (Esri, 2014)

Jak bylo zmíněno, jedná se o balík vzájemně propojených aplikací, který je navíc možné rozšířit o mnohé další nástroje či funkcionality dostupné v podobě zásuvných modulů. Hlavní aplikace a funkce, kterými ArcGIS disponuje, jsou především:

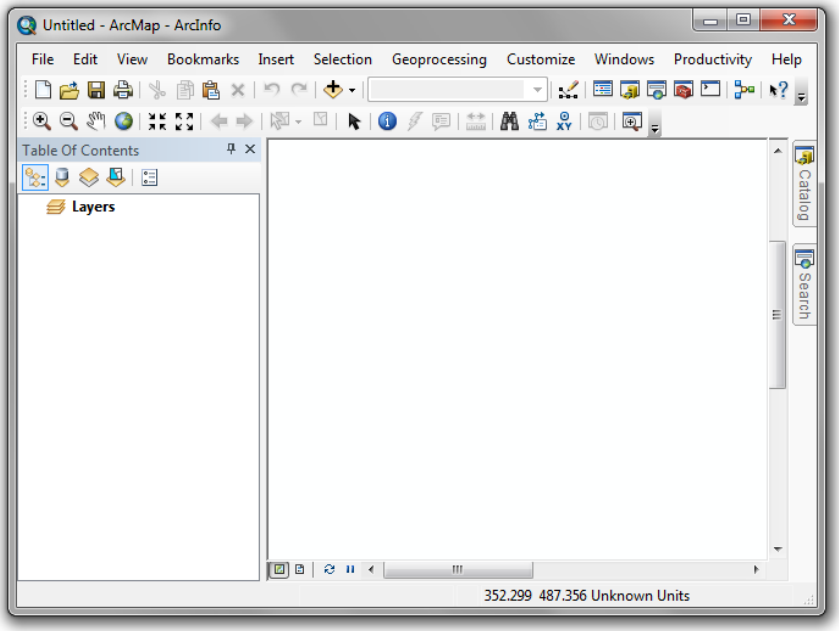
- **ArcMap**: prohlížení, editace, dotazování nad geodaty a tvorba výstupů v podobě map.
- **ArcCatalog:** správa dat a metadat.
- **ArcToolbox:** nástroje pro zpracování prostorových dat a jejich konverze, analytické nástroje.
- **ArcExplorer**: prohlížeč geodat umožňující také přístup k mapovým službám.

Samozřejmostí u komerčního GIS je i podpora standardů mapových služeb, kterou disponuje teké ArcGIS. (ArcGIS Resources, 2014)

- Některé podporované standardy:
    - WMS
    - WMTS
    - WFS
    - WCS

Níže uvedený obrázek (Obr. 5) představuje základní obrazovku aplikace ArcMap, která je centrální aplikací ArcGis společnosti *Esri*. Rozvržení prvků v aplikaci je věrně známo i z jiných desktopových aplikací, kdy hlavní okno je řešeno jako dělené a kterému dominuje pravá část sloužící pro zobrazení prostorových dat. Horní část okna je tvořena statickým menu s libovolně rozmístitelnými lištami obsahující ovládací prvky a celou řadu další funkcionality. Stejně jako u většiny jiných aplikaci využívající toto rozložení hlavní nabídky uživatel celou řadu nástrojů na ní téměř nepoužije. V levé části děleného okna je ve stromové struktuře zachycen obsah zobrazovacího okna. Ve výchozím nastavení se zde nachází seznam vrstev. Vpravo od grafického okna jsou umístěny odkazy pro vyhledávání a také odkazy do ArcCatalogu, který slouží k organizaci a správě potřebných dat.

[3] Proces či množina procesů pro zpracování geodat



Obr. 5 Grafické uživatelské rozhraní aplikace ArcMAP

Shrnutí: rozhraní je řešeno standardním desktopovým způsobem, který není vhodný pro dotykové obrazovky. Vychází ze zažitých principů a postupů rozložení prvků v aplikaci. Toto rozvržení velmi připomíná známé textové a grafické editory. Lišta s hlavní nabídkou je rozsáhlá a obsahuje komplexní funkcionalitu. Možnosti uzpůsobení rozhraní spočívá opět především ve volném rozmístění oken a jednotlivých nástrojových lišt.

2.2.2 Další komerční GIS

Kromě uvedeného ArcGIS od společnosti *Esri* existuje několik dalších dodavatelů komerčních GIS, které se v praxi velmi často využívají. Jsou to například:

- **AutoCAD Map 3D**: tato spíše CAD aplikace společnosti *Autodesk* je zaměřená především na modely. Poskytuje mnoho nástrojů pro analýzu dat, podporu plánování, návrh a správu dat.
- **Cadcorp SIS**: jedná se o integrovanou skupinu nástrojů (podobně jako ArcGIS) pro kompletní správu a manipulaci s geodaty. *Cadcorp* nabízí jak desktopová řešení, tak i například servery, webové či mobilní nástroje.
- **MapInfo Pro:** jedná se především o analytický nástroj kladoucí důraz na kvalitní a přehlednou vizualizaci výstupů.
- **Maptitude**: aplikace společnosti *Caliper* zaměřující se především na vizualizací analyzovaných dat a jejich následnou manipulaci Je určena především do podnikové sféry k usnadnění plánování a rozhodování. *Caliper* nabízí ještě další GIS a tím je *TransCAD*, který je jako jediný GIS zaměřen na dopravní data.

- **Netcad 6 GIS**: přednostně CAD aplikace, která umožňuje profesionální technická řešení v širokém oblasti využití.
- **SuperMap Desktop GIS**: snadno použitelný GIS ve stylu Microsoft Office, který poskytuje funkce pro zpracování dat, tematické mapování a pokročilou analýzu prostorových dat.
- **TNTMips**: kompletní GIS společnosti *MicroImages*, který poskytuje plnohodnotné GIS, RDBMS a systém pro automatizované zpracování obrazu s CAD, TIN, modelováním povrchu, rozložení map atd.

2.3 Mobilní GIS

V posledních letech se objevuje také mnohem více aplikací, které fungují spíše na principu tenkého klienta. Existuje již celá řada úspěšných navigací či mapových aplikací. Dalo se tedy očekávat, že se vývojáři pokusí zeštíhlit dosavadní GIS řešení v podobě tlustých klientů do podoby aplikací, které jsou povětšinou určeny primárně pro různá mobilní zařízení. Přestože nedisponují natolik rozsáhlou funkcionalitou jako klasické desktopové alternativy, staví na potenciálu současných moderních webových technologií, stávajících možnostech mobilních technologií (integrace GPS apod.) a také na výhodách, které přináší dotykové obrazovky. U mobilních aplikací se obvykle vychází ze skutečnosti, že jsou více úkolově orientované a slouží obvykle k usnadnění či přímo splnění konkrétního specifického cíle, který leží před daným uživatelem.

2.3.1 Wolf-GIS

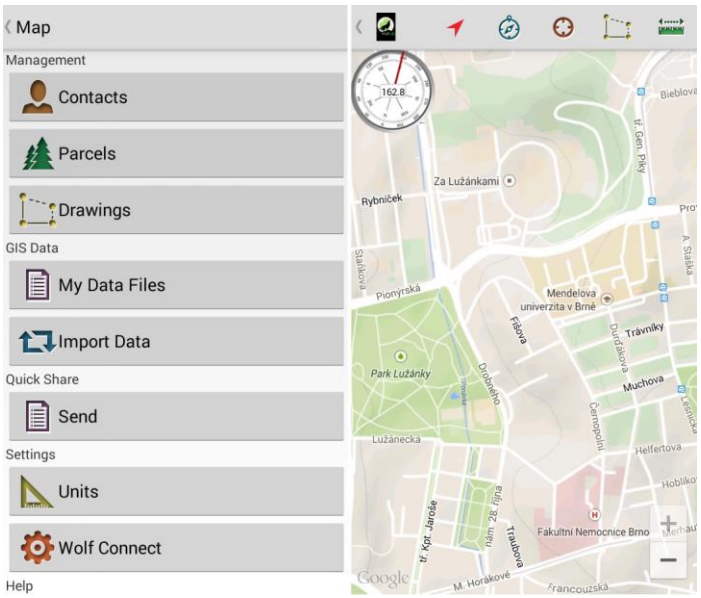
Jedná se o produkt společnosti *Wolf-Tek*, která se touto cestou snaží o propojení silných stránek GIS a mobilních zařízení. Za svou dobu existence si Wolf-GIS získal své uživatele z různých oblastí, a to především díky své patentované *Track-Lock* technologii, která upozorní uživatele, pokud se blíží k hranicím vyznačené oblasti. V současné době tak můžeme najít uživatele Wolf-GIS například v oblasti zemědělství, obchodování s nemovitostmi, hornictví, lesnictví či rekreace.

Aplikace je dostupná pro *iOS* a *Android*. Uživatelé *iOS* si mohou také vybírat mezi licencí *Basic* a *Pro*. Uživatelé Android mají ekvivalent k *Pro* verzi dostupný prostřednictvím měsíčního poplatku za přístup k *Wolf-Connect* účtu.

- Výčet vybraných funkcí:
    - **Basic**: ukládání a sdílení dat, podpora shapefile, měření vzdáleností, detailní informace o parcele, vyhrazené 2 GB na uložišti.
    - **Pro**: veškeré možnosti *Basic* licence doplněné o neomezený prostor na uložišti, *Track-Lock*, kreslení, živé GPS sledování, neomezený počet shapefilů či rastrů a další možnosti. (WolfGIS, 2014)

- Podporované standardy:

  Aplikace Wolf-GIS neumožňuje připojení k veřejným mapový službám. Veškeré vkládání dat je řízeno prostřednictvím zmíněného *Wolf-Connect* účtu popřípadě je možné svá data spravovat pomocí *Wolf-Direct* služby rovnou z vašeho počítače.



Obr. 6 Grafické uživatelské rozhraní základní obrazovky a menu aplikace Wolf-GIS

Aplikace Wolf-GIS disponuje poměrně příjemným uživatelským rozhraním (Obr. 6) s velmi přehledným menu rozděleným do logických celků a jednoduchou hlavní obrazovkou. Povedený jednotný grafický styl velmi napomáhá aplikaci k tomu, aby byla velmi intuitivní i pro ty, kteří se s ní dříve nesetkali. Uživateli je k dispozici GPS k určení jeho polohy. Pokud využije této funkce, nástroje pro měření vzdálenosti a kreslení se budou vztahovat k tomuto bodu. Pokud GPS nevyužijete, lze měřit vzdálenost pomocí gesta. Toto řešení se jeví jako velmi praktické, avšak složitější pro menší obrazovky, kde je nutné pro menší vzdálenosti nejdříve velké přiblížení.

Shrnutí: první z mobilních GIS aplikací se může chlubit velmi příjemným grafickým uživatelským rozhraním. Aplikace je velmi přehledná a orientaci v ní snadná a intuitivní. Jedním z mála záporů je nabídka dalších nástrojů, které nejsou umístěny přímo na hlavní obrazovce a její umístění, které by mohlo svádět k pocitu, že se jedná o hlavní menu aplikace, do které se vchází přes tlačítko umístěné v levém horním rohu.

2.3.2 ArcGIS for Mobile

Společnost *Esri* se tímto produktem pokouší svou desktopovou verzi přenést do mobilní sféry a uspokojit tak požadavky na možnost využívat ArcGIS také v terénu,

synchronizovat získaná data z dané lokality a následně je mít k dispozici pro další práci i v režimu bez připojení k internetu.

ArcGIS je dostupný kompletně zdarma pro platformy *iOS*, *Android* i *Windows phone*. Stejně jako desktopová verze umožňuje sbírat a analyzovat data a vytvářet z nich mapové výstupy, které je možno posléze také sdílet s dalšími uživateli. (ArcGIS, 2014)

- Výčet vybraných funkcí:
    - Sdílení a prohlížení map online
    - Shromažďování, úprava a aktualizace prvků a atributů
    - Vyhledávání podle klíčových slov a získání relevantních výsledků
    - Zobrazení a přiblížení na aktuální polohu
    - Možnosti pro měření či dotazování
- Podporované standardy:

  Co se mapových služeb u ArcGIS týká, tak ani zde nejsou podporovány standardy OGC. WMS vrstvy nejsou vůbec podporovány a WFS či jiných OGC vrstev lze využít pouze jako rozšíření vrstev jiných. Primárně je zde využíván ArcGIS server.

Obr. 7 Grafické uživatelské rozhraní aplikace ArcGIS for Mobile s nástrojem pro měření

Příjemné minimalistické rozhraní aplikace ArcGIS for Mobile (Obr. 7) nabízí na hlavní obrazovce uživateli pouze hlavní čtyři nástroje (vyhledání místa, měření, přidání elementu a GPS), kterými aplikace disponuje a k tomu možnost přidat mapový podklad. Nabídky jsou poměrně přehledné a celková orientace v aplikaci in-

tuitivní. Pozitivním prvkem je také kvalitní výběr jednotlivých ikon popřípadě náhledů, které uživateli pomáhají k tomu, aby věděl, jaká data má k dispozici, respektive jaký úkol se chystá vykonat. Velmi příjemná je manipulace s nástrojem měření. Naopak nejméně příjemná, respektive mírně matoucí je stránka, která se uživateli zobrazí při přidání elementu do mapy (stejná je pro následném zobrazení detailu prvku na mapě).

Shrnutí: druhá z uvedených mobilních aplikací disponuje velmi jednoduchým, intuitivním a přehledným uživatelským rozhraním. Také po estetické stránce je rozhraní poměrně povedené bez rušivých elementů.

2.3.3 SuperSurv

Tato mobilní GIS aplikace byla navržena společností *Supergeo* především pro platformy *Android* a *iOS*. Těží zejména z integrace GIS a GPS technologie. Hlavním cílem aplikace je sběr dat, zobrazování map a navigace. Pomocí GPS je možné velmi snadno zachytit údaje o libovolných bodech, liniích či polygonech. Veškeré trasy a body na trase mohou být zaznamenávány a ukládány pro snadnější orientaci. Aplikace také podporuje nastavení globálního systému souřadnic a ukládání dat do formátů SHP, GEO či KML.

Aplikace je k dispozici pro obě výše zmíněné platformy ve 2 edicích (*SuperSurv* a *SuperSurv M3*), kde každou edici můžete pořídit zdarma jako týdenní zkušební verzi s kompletní funkcionalitou, nebo za poplatek jako oficiální verzi. (Supergeo Technologies, 2014)

- Výčet vybraných funkcí:
    - Podpora vektorových formátů
    - GPS a e-kompas
    - *Offline* mód pro zobrazování a úpravu dat
    - Správa tras
    - *SuperSurv* edice disponuje také navíc *SuperGIS* mapovou službou
- Podporované standardy
    - WMTS

Obr. 8 Grafické uživatelské rozhraní základní obrazovky a menu aplikace SuperSurv Lite

Pokud jsem označil grafické rozhraní aplikace ArcGIS for Mobile za zdařilé, tak rozhraní tohoto mobilního GIS musíme označit za to méně povedené (Obr. 8). Už na první pohled i nezaujatý uživatel musí uznat, že po grafické stránce je rozhraní poměrně neatraktivní a především po dotyková zařízení s menší obrazovkou také nepraktické. U některých nástrojů není vždy úplně snadné zjistit, jak je správně použít. Navíc je možné aktivovat více nástrojů současně a potom často dochází k překrývání jednotlivých panelů. Při zobrazení příliš velkého množství nástrojů pak není úplně jednoduché přijít na to, jak jednotlivé nabídky a panely opět skrýt.

Shrnutí: aplikace nedisponuje uživatelsky příliš intuitivním rozhraním. Nekonzistentnost grafického stylu napříč jednotlivými částmi aplikace, umístění některých nástrojů do nabídky s nastavením, překrývání panelů na hlavní obrazovce jsou jen některé z problémů, se kterými se tato aplikace potýká.

3 Mapové služby

Prostorová data, která mohou být zpracovávána prostřednictvím jednotlivých koncových aplikací, tedy pomocí GIS, lze získat z různých zdrojů. Jedním z velmi významných zdrojů těchto dat jsou mapové služby, které jsou dostupné prostřednictvím distribuované sítě internetu. Tyto služby obvykle fungují na principu architektury klient-server, tzn., že data jsou umístěna na serverech a odesílána na základě požadavků na pracovní stanice koncových uživatelů či dalších serverů, a kromě zmiňovaných datových sad mohou uživateli poskytovat také mnohé funkce a především jejich následné výstupy.

Jelikož mapové služby uchovávají data v různých formách a samotný rozvoj GIS s postupem času přinesl celou řadu nových avšak uzavřených formátů, vznikly tak problémy například s nepřenositelností jednotlivých formátů. Z tohoto důvodu se v současné době převážná část zpracování a sdílení geodat řídí specifikacemi a standardy mezinárodního konsorcia OGC (*Open Geospatial Consortium*), které k dnešnímu dni tvoří komunitu o více než 500 členech z řad společností, univerzit či vládních agentur. (OGC, 2014)

Jednotlivých OGC standardů, jenž existují v podobě volně dostupných dokumentů, které jsou k dispozici ke stažení a nahlédnutí na oficiálních stránkách organizace (OGC, 2014), existuje celá řada. Nejběžnějšími standardy jsou především:

3.1 WMS (Web Map Service)

Jedná se zřejmě o nejběžnější OGC standardní protokol, který byl v roce 2005 také příslušným úřadem přijat pod označením ISO 19128 jako mezinárodní norma.

Služba na základě požadavku uživatele na jím definovanou tematickou mapu či množinu map a zájmovou oblast vrací v odpovědi relevantní georeferencovaná obrazová data nalezená v některé z dostupných databází a vyexportovaná do některého z rastrových formátů (JPEG, PNG, TIFF, GIF,…).

Tímto mezinárodním standardem jsou definovány dvě třídy WMS. Prvním z nich je *Basic WMS* a druhou pak *Queryable WMS*. Jednotlivé třídy jsou vymezeny mimo jiné pomocí základních operací, které musí vykonávat. Jsou jimi *GetCapabilities* (dotaz na metadata o službě) a *GetMap* (dotaz na mapu) pro základní WMS a pro *Queryable WMS* k těmto 2 ještě *GetFeatureInfo* (dotaz na informace o prvcích mapy). (OGC, 2006)

3.2 WMTS (Web Map Tile Service)

Tento standardizovaný implementační protokol doplňuje výše uvedený WMS a služba využívající tohoto standardu umožňuje sdílení předrenderovaných georeferencovaných dat v podobě rozdělení každé mapové vrstvy do dlaždic. Tento přístup umožňuje zejména zvýšení výkonu ze strany serveru. Servery dříve využívaly

vlastní struktury, které však limitovaly požadavky na protokol. Vývojáři byli nuceni vytvářet speciální klienty pro každý server.

Standard specifikuje WMTS ve dvou částech. Jednak popisuje sémantiku zdrojů poskytovanými servery a požadovanými klienty (sémantika sokumentu *ServiceMetadata*, jednotlivé snímky dlaždic – *Tile*, popisky map *FeatureInfo*), ale také definuje výměnný mechanismus mezi klientem a serverem, tedy jednotlivé operace (*GetCapabilities, GetTile, GetFeatureInfo*), jenž jsou založené na použití zpráv v různém kódování (KVP, XML zprávy, SOAP, aj.). Také je definován mechanismus žádostí a koncová publikační strategie založená na URL, která zjednodušuje klientům žádat o zdroje v podobě dokumentů. (OGC, 2010)

3.3 WFS (Web Feature Service)

Jde o standard, který poskytuje rozhraní umožňující sdílení geografických vektorových dat prostřednictvím internetu. Uživatelé mohou pracovat s daty, která obdrží ve formátu GML. Tento formát vychází z XML a v rámci přenosu geodat uchovává geometrii i topologii přenášených dat. Uživatel tak dostává k dispozici diskrétní geodata. Veškeré vektorové prvky jsou umístěny v prostoru na základě použitého souřadnicového systému. Nejčastěji využívanou množinou dat pro označování souřadnicového referenčního systému je *EPSG Geodetic Parameter Set*, který v sobě nese databázi zeměpisných a kartografických souřadnicových systémů, měrných jednotek, geodetických dat i například zemských elipsoidů. WFS se dělí na 3 základní typy podle toho, jaké dotazy lze v rámci služby využívat:

- **Basic WFS** zahrnující dotazy:
    - *GetCapabilities* - jde o popis dostupných vlastností a operací podporovaných pro každý typ.
    - *GetFeature* - jde o dotaz pro obdržení instancí prvků z WFS.
    - *DescribeFeatureType* - na požádání je nutné, aby WFS dokázal popsat strukturu jakéhokoli prvku.
- **XLink WFS** jehož součástí jsou dotazy z *Basic WFS* doplněné o:
    - *GetGmlObject* - funkce zajišťující vrácení instancí prvků pomocí *XLinks*, které se odkazují na jejich XML ID.
- **Transactional WFS** umožňující dotazy z *Basic WFS* rozšířené o operace *insertFeature*, *updateFeature*, *deleteFeature* a *LockFeature*. (OGC, 2005)

3.4 WCS (Web Coverage Service)

Tato služba umožňuje přenos dat v původním formátu spolu s metadaty, která jsou využívána pro jejich interpretaci. V porovnání s WMS nepracuje tedy pouze se statickými snímky. Významným znakem této služby je to, že jednak pracuje s 2D i 3D daty, ale také dokáže pracovat se čtvrtou dimenzí, kterou tvoří čas. Díky této sku-

tečnosti je možné provádět nad těmito daty velmi složité a především přesné analýzy jako jsou například analýzy týkající se změn povrchu Země, vývoje klimatu atd. Proto se tedy také velmi často využívají pro přenos například satelitních snímků, digitálních výškových dat a podobně. (OGC, 2012)

WCS také definuje tři operace, které mohou být v rámci požadavku klienta zaslány na WCS server. Těmito operacemi jsou:

- *GetCapabilities*
- *DescribeCoverage*
- *GetCoverage*

3.5 WPS (Web Processing Service)

Web Processing Service poskytuje pravidla standardizace jednotlivých vstupů a výstupů nutných pro jejich další zpracování. Norma definuje, jak může klient požádat o provedení procesu a jak je nakládáno s následným výstupem. Data z WPS pak mohou být uživateli dodána po síti nebo mohou být k dispozici dostupné na serveru opět prostřednictvím některé OGC služby.

WPS mimo jiné umožňuje přístup klienta prostřednictvím sítě k předem naprogramovaným výpočtům nebo výpočetním modelům, které fungují nad prostorově orientovanými daty. Výpočty mohou být velmi jednoduché, jako je odečtení jedné množiny prostorově orientovaných dat od jiné (např. stanovení rozdílu v případě chřipky mezi dvěma různými ročními obdobími), nebo velmi složité, jako je globální model změny klimatu. Je možné počítat s libovolným počtem datových vstupů a výstupů. Můžeme tedy říct, že WPS zprostředkovává na straně serveru provádění výpočtů nad vektorovými i rastrovými daty, které jsou jinak příliš složité na to, aby byly prováděny na lokálních zařízeních. (OGC, 2007)

Kromě uvedených OGC standardů existuje i řada firemních technologií, které jsou však obvykle zaměřeny pouze na produkty daného dodavatele. To je jeden z mnoha důvodů, proč OGC standardy zažívají v posledních letech obrovský rozmach. Velkou měrou k tomu přispívá také dohoda o vzájemné spolupráci na rozvoji, užívání, údržbě a podpoře těchto otevřených mezinárodních standardů, kterou spolu uzavřelo evropské společné výzkumné středisko (JRC, *Joint Research Centre*) a konsorcium OGC v rámci vývoje a implementace direktivy INSPIRE (*Infrastructure for spatial information in Europe*).

Každý z těchto standardů má svou specifickou oblast užití, ale i přesto jsou nejčastěji jednotlivé mapové služby nabízeny především jako standardní WMS, případně jako dlaždicové WMTS.

4 Možnosti knihovny Qt

Knihovna Qt dnes tvoří jeden z nejpopulárnějších nástrojů pro tvorbu multiplatformních jak konzolových aplikací, tak především aplikací s grafickým uživatelským rozhraním. Jedná se o komplexní vývojový rámec, jehož nativním programovacím jazykem je *C++* a relativně nově také *QML* (*Qt Meta/Modeling Language*). Podporuje však celou řadu dalších programovacích jazyků. Možnosti Qt jsou velmi široké a jelikož většinou potřebné základní funkcionality disponuje samotná knihovna (včetně vývojových aplikací), není obvykle nutné používat žádné externí knihovny. Knihovna Qt obsahuje 15 základních modulů (viz Tab. 1), které je možné rozšířit o početnou skupinu přídavných modulů.

Tab. 1 Dostupné základní moduly knihovny Qt 5

Modul	Popis
Qt Core	I Základní třídy bez grafiky využívané jinými moduly.
Qt GUI	Základní třídy pro grafické uživatelské rozhraní.
Qt Multimedia	Třídy pro funkcionalitu zvuku, videa, radia či kamery.
Qt Multimedia Widgets	Třídy založené na ovládacích prvcích pro implementaci multimédií.
Qt Network	Třídy pro snadnější síťové programování.
Qt QML	Třídy pro *QML* a *JavaScript*.
Qt Quick	Deklarativní rámec pro vytvoření dynamických aplikací s vlastním uživatelským rozhraním.
Qt Quick Controls	Prvky uživatelského rozhraní Qt Quick pro vytvoření klasického desktopového stylu.
Qt Quick Dialogs	Typy pro vytváření a interakci pomocí dialogů.
Qt Quick Layouts	Rozvržení prvků uživatelského rozhraní.
Qt SQL	Třídy pro integraci databází využívající *SQL*.
Qt Test	Třídy pro testovací jednotku Qt aplikací a knihoven.
Qt WebKit	Třídy pro implementace založené na *WebKit2* a *QML* API.
Qt WebKit Widgets	Třídy z Qt4 založené na *WebKit1* a QWidget.
Qt Widgets	Třídy rozšiřující Qt GUI o *C++* ovládací prvky.

Zdroj: Qt Project, 2014.

Pomocí Qt je možné využívat různých technologií k tvorbě uživatelského rozhraní aplikace. Základní přístupy jsou tři: tvorba pomocí jazyka QML, pomocí ovládacích prvků anebo je možné zobrazovat obsah aplikací webových.

Každé z uvedených řešení má své výhody i nevýhody. Obvykle je vhodné zvolit pouze jeden z přístupů podle typu výsledné aplikace. Samozřejmě je však možné jednotlivé přístupy kombinovat a využít například nabídky či dialogy jako *Qt Widgety* doplněné o vlastní elementy vytvořené pomocí *QML* (*Qt Quick*) a například

dokumentaci zpřístupnit jako webový obsah (*Qt WebKit*). Tímto způsobem je vytvořeno například rozhraní ve vývojovém nástroji *Qt Creator*.

4.1 Qt Widgety

Jedná se o tvorbu uživatelské rozhraní pomocí tradičních desktopových ovládacích prvků neboli widgetů. Tento přístup se nejlépe hodí pro aplikace určené zejména na klasické stolní počítače, kde se vyžaduje především tradiční uživatelské rozhraní, jako jsou například různé kancelářské aplikace.

Ovládací prvky disponují skvělou integritou v rámci jednotlivých platforem, což poskytuje přirozený vzhled a intuitivitu v systémech *Windows*, *Linux* i *OS X*. Obsahují velké množství prvků pro uživatelská rozhraní vhodná pro z větší části statická uživatelská rozhraní.

4.2 Qt WebKit

Knihovna Qt poskytuje vlastní engine *WebKit* pro zobrazení obsahu webových stránek ve vlastní aplikaci. Tím je zajištěna podpora široké škály standardních webových technologií, jako je *HTML*, *CSS* a *JavaScript*.

Qt WebKit využívá pro zobrazení obsahu webových stránek typu *WebView QML*, zatímco *Qt WebKit widgety* využívají *C++* API. Hlavním rozdílem mezi těmito dvěma rozhraními spočívá ve využití multiprocesní architektury *WebKit2* respektive architektury jednoho procesu *WebKit*.

4.3 QML

QML je v rámci knihovny Qt relativně nový přístup k tvorbě uživatelského rozhraní deklarativním způsobem, podobně jako je tomu např. v případě *CSS*. Nespornou výhodou je možnost sestavit rozhraní z vlastních jednoduchých prvků a jejich vzájemných vazeb. Jednotlivé prvky navíc mohou tvořit komponenty, které lze využít na více místech realizované aplikace. Na rozdíl od ovládacích prvků lze jednotlivé elementy libovolně uzpůsobit pro dotykové obrazovky a navíc zajistit plynulost přechodů a změň pomocí animací, různých přechodových efektů a podobně. Lze tak docílit jednoduchého moderního uživatelského rozhraní, které je navíc díky *Qt Quick* modulu a jazyka *C++* možné napojit na libovolnou *back-end* aplikaci. (Qt Project, 2014)

Jak bylo zmíněno výše, volba vhodného přístupu není vždy úplně jednoduchá a může záviset na mnoha kritériích. Následující tabulka (Tab. 2) znázorňuje jednoduché srovnání možností jednotlivých přístupů a může tak posloužit jako nástroj pro usnadnění rozhodování o tom, kterou technologii při tvorbě uživatelského rozhraní zvolit. Tučně jsou zvýrazněna kritéria, která hrála největší roli při volbě vhodného přístupu vzhledem k cílům této práce.

Tab. 2 Srovnání jednotlivých přístupů tvorby UI v Qt

	Qt Quick	**Qt Widgets**	**Qt WebKit**
Použité jazyky	QML/JS	C++	HTML/CSS/JS
Nativní vzhled	X	X	
Vlastní vzhled	X		(X)
Plovoucí animované GUI	X		
Dotykové obrazovky	X		
Standardní ovládací prvky		X	
Programování model/view	(X)	X	
Zrychlený UI vývoj	X		(X)
Grafická HW akcelerace	X		
Grafické efekty	X		
Široké možnosti práce s textem	X	X	
Existující webový obsah			X

Zdroj: Qt Project, 2014.

5 Metodika práce

Jedním z hlavních cílů této práce je vytvořit aplikaci s moderním grafickým uživatelským rozhraním a také prozkoumání knihovny Qt. Konkrétně možností, které přináší v oblasti tvorby moderních uživatelských rozhraní. Z toho vyplývají 2 základní požadavky na danou aplikaci:

1. Aplikace bude vytvořena pomocí knihovny Qt.
2. Bude disponovat moderním uživatelským rozhraním (podpora gest).

Základní 3 principy tvorby uživatelského rozhraní pomocí knihovny Qt byly uvedeny v kapitole 4.1 a na základě těchto informací, především pak z výsledků uvedených ve srovnávací tabulce (Tab. 2) na konci této kapitoly, jsem pro realizaci výsledné aplikace zvolil možnosti jazyka *QML*.

Před samotnou realizací aplikace je však nejdříve nutné učinit ještě několik dalších neméně důležitých kroků:

1. Definovat samotného uživatele a účel aplikace – bude nutné určit, kdo bude především aplikaci využívat, jaké jsou jeho případné znalosti či dovednosti a také na základě tohoto následně stanovit, jakému hlavnímu účelu bude daná aplikace sloužit.
2. Určit základní funkcionalitu, kterou bude aplikace disponovat – primárně na základě účelu aplikace definovat prvotní množinu funkcí, kterou bude aplikace disponovat a názorně ji zachytit pomocí diagramu.
3. Na základě provedených analýz stanovit architekturu uživatelského rozhraní – hlavním bodem tohoto kroku je vyhotovení počátečního návrhu grafického uživatelského rozhraní aplikace a jeho postupné vyspecifikování až do finální podoby.
4. Navrhnout architekturu samotné aplikace – zde bude přihlíženo především na existující možnosti a realizace zejména na poli svobodných systémů a jejich potenciálního využití i pro výslednou aplikaci.

Při návrhu jak uživatelského rozhraní, tak architektury samotné aplikace bude přihlíženo na skutečnosti, které byly objeveny v předchozí části této práce. To znamená, že například při návrhu uživatelského rozhraní bude přihlíženo především na uživatele a konečnou uživatelskou přívětivost daného řešení.

6 Vlastní práce

6.1 Účel aplikace a definice uživatele

Pro vytvoření vhodného návrhu výsledné aplikace, je velmi důležité nejdříve určit její hlavní účel a samozřejmě také uživatele, pro které daná aplikace bude sloužit.

V tomto případě se jedná o aplikaci, která představuje jednoduchý geografický informační systém určený pro zařízení disponující dotykovou obrazovkou. Hlavním účelem tedy bude především prohlížení a snadná manipulace s geodaty a možnost tvorby jednoduchých mapových podkladů. Typickým uživatelem pak bude člověk, u kterého se předpokládá určitá pokročilejší znalost užívání informačních technologií. Jelikož se jedná konkrétně o jednoduchý GIS, uživatel bude disponovat alespoň základními znalostmi pojmů a technik nutných k bezproblémovému užívání takového systému.

6.2 Základní funkcionalita aplikace

Vzhledem k výše uvedeným informacím týkající se účelu aplikace je nyní nutné vyspecifikovat množinu klíčové funkcionality výsledné aplikace. Výsledná aplikace by tedy měla uživateli pro základní práci s geodaty nabídnout tyto možnosti:

- Správa projektu
    - Nový projekt
    - Otevřít projekt
    - Uložit projekt
    - Uložit projekt jako
- Správa vrstev
    - Přidat vrstvu
    - Odstranit vrstvu
    - Editovat vrstvu
- Správa mapových elementů
    - Přidat prvek
    - Editovat prvek
    - Odstranit prvek
- Manipulace s výstupy

Na základě prvotního výčtu klíčové funkcionality uvedeného výše, byla posléze provedena konkrétnější specifikace včetně zachycení do přehlednějšího schématu. Zejména na základě porovnání s aplikacemi v rešerši a vytvoření určité ucelené formy aplikace byly konkretizovány například typy vrstev, jenž budou podporová-

ny, určeny typy a základní vlastnosti mapových prvků a především definovány manipulace jak s výstupy, tak jednotlivými elementy. Podrobnější kompozice jednotlivých funkcí, včetně barevně rozlišené úrovně důležitosti, je zachycena na obrázku níže. Uvedené schéma (Obr. 9) může sloužit také jako základní mapa průchodu aplikací a bylo vytvořeno pomocí zdarma dostupného webového nástroje *MindMup*[4].

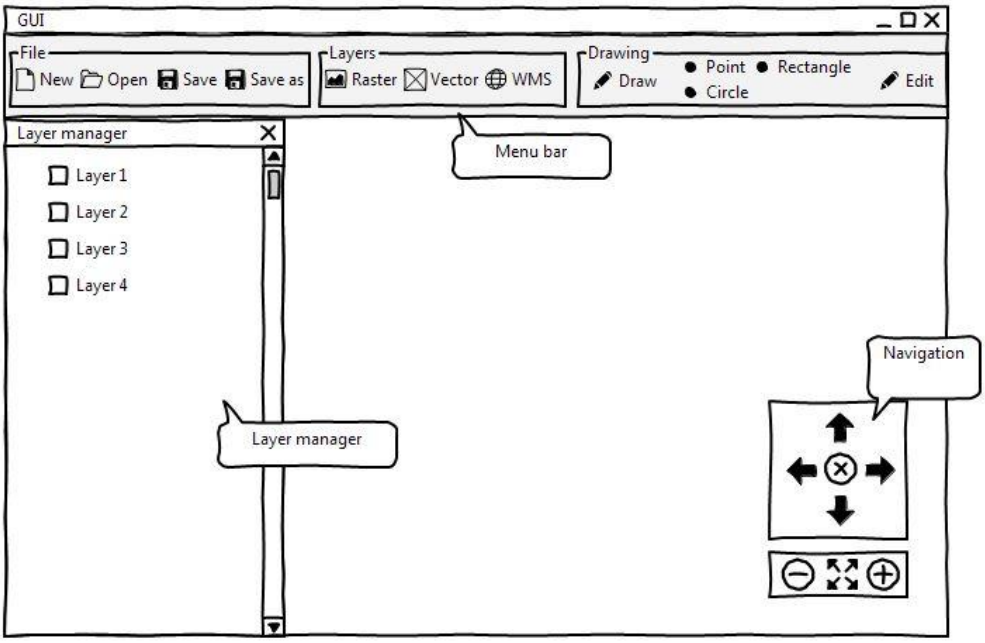
Obr. 9 Funkcionalita navrhovaného GIS

6.3 Návrh uživatelského rozhraní

Po provedení specifikace funkcionality aplikace byl realizován prvotní návrh grafického uživatelského rozhraní. Tato původní verze (Obr. 10) vychází z výše uvedené funkční specifikace a jednotlivé funkční větve schématu jsou také v návrhu seskupovány do logických funkčních celků. Jednotlivé skupiny funkcí jsou umístě-

[4] MindMup je zdarma dostupná svobodná webová služba pro tvorbu mind map a jim obdobných diagramů. Služba je dostupná na webových stránkách www.mindmup.com.

ny na hlavním horním panelu a jsou tak uživateli kdykoliv k dispozici. V tomto návrhu bylo hlavní okno aplikace navrženo jako dělené, kde levá část je tvořena správcem vrstev, ze kterého je možná veškerá manipulace a editace vrstvy, a pravá dominantní část je tvořena oknem vykreslujícím veškeré výstupy, tedy mapy. Do pravého dolního rohu pak byla umístěna ještě navigační tlačítka pro manipulaci se zobrazenými mapami.



Obr. 10 Prvotní návrh uživatelského rozhraní aplikace

Tento počáteční návrh vycházel především z inspirace klasickými rozhraními aplikací uvedených dříve, včetně rozdělení a seskupení jednotlivých funkčních celků. Bylo myšleno především na to, aby měl uživatel k dispozici veškerou definovanou funkcionalitu. Silně tedy připomínal běžná statická desktopová uživatelská rozhraní a nesplňoval tak jeden z hlavních požadavků, kterým bylo moderní rozhraní určené především pro dotykové obrazovky. Proto byl tento návrh postupně přepracován a posloužil spíše jako základ pro rané fáze implementace.

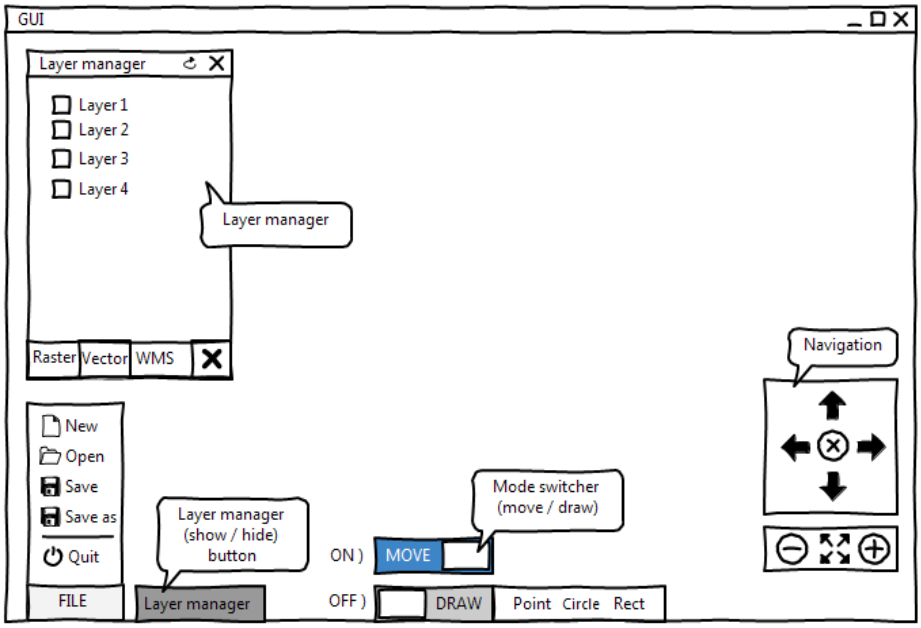
Výsledný přepracovaný návrh (Obr. 11) byl realizován zejména jako úkolově orientovaný návrh s ohledem na požadavek větší uživatelské přívětivosti a také ovládání aplikace primárně pomocí dotykové obrazovky. Inspirován byl tedy více mobilními verzemi GIS a větší důraz byl také kladen na aplikování jednotlivých návrhových vzorů (Tidwell, 2010), které definují jakási typická chování uživatelů. Aplikování těchto postupů pak napomáhá uživateli efektivněji a rychleji dosáhnout kýžených výsledků.

Nejdříve tedy byla více rozebrána požadovaná funkcionalita z hlediska definování aktivit a úkolů, z nichž se jednotlivé aktivity skládají. Příklad rozboru některých aktivit, které představují povětšinou určitou množinu funkci (ve schématu výše oranžovou barvou), a úkolů vypadá přibližně následovně:

- **Přidat vrstvu**: zvolit typ vrstvy → zvolit příslušné hodnoty → potvrdit / zrušit přidání vrstvy.
- **Smazat vrstvu**: volba vrstvy pro smazání → vykonání akce pro smazání → potvrdit / zrušit smazání vrstvy.
- **Přidání mapového elementu**: zvolit typ elementu → vybrat umístění → umístit prvek → nastavit vlastnosti a atributy elementu.

Na základě původního návrhu, stanoveným požadavků a provedeného rozboru aktivit tedy byla v přepracovaném návrhu nejdříve odstraněna statická horní hlavní nabídka obsahující skupiny s dostupnou funkcionalitou. Nahradila ji skupina tlačítek v dolní části obrazovky, kde jsou především na větších obrazovkách dostupnější a toto umístění je tedy vhodnější. Tato tlačítka jsou také realizována s větším přihlédnutím na úroveň důležitosti dané funkcionality. To znamená, že uživateli nejsou zbytečně nabízeny veškeré funkce hned na hlavní obrazovce, ale například až po stisknutí příslušného tlačítka a podobně. Nejen tato tlačítka jsou ale ponechána v podobě logických svázaných skupin (*Button Groups*). Další takovou skupinou je nabídka pro přidání mapových prvků. Tato nabídka je navržena také podle vzoru *Canvas Plus Palette*, kdy po zvolení typu elementu jej lze přidat na plátno tvořené mapovým výstupem. Vzhledem k tomu, že jak s mapovými výstupy, tak následně s přidanými prvky lze manipulovat pomocí gest, byl pro aktivaci této nabídky navržen přepínač. Tento přepínač slouží pro přepínání mezi módy pro manipulaci s mapou a právě kreslením elementů. Umožní to jednoznačné určení toho, s jakými prvky lze v danou chvíli manipulovat.

Z původního návrhu tedy byla ponechána pouze navigační tlačítka v pravém dolním rohu a také správce vrstev, který však není navržen tak, aby pevně dělil hlavní okno aplikace, ale je možné jej libovolně přemístit a měnit jeho velikost. Struktura správce vrstev však dále vychází z klasické podoby, kterou uživatelé znají z běžných aplikací. Po vyhotovení návrhu podoby hlavního okna aplikace byly vytvořeny také návrhy pro zbývající části uživatelského rozhraní. Bylo vytvořeno několik skic pro navigační tlačítka (Obr. 38), pro správce vrstev, včetně možností realizace jednotlivých položek seznamu, kde jejich detail bude realizován podle návrhového vzoru *One Window Drilldown*. Vlastnosti vrstvy se tak uživateli zobrazí přímo uvnitř správce vrstev namísto seznamu vrstev. Dále byl realizován návrh úvodní spouštěcí obrazovky (Obr. 35), u které se předpokládá využití návrhových vzorů pro vstupní data, jako jsou například *Same-Page Error Message* pro informování o vyplnění nesprávných vstupních hodnot a *Input Prompt* pro popisky umístěné přímo uvnitř vstupních polí. V neposlední řadě byl navržen vzhled dialogu pro přidání jednotlivých typů vrstev (Obr. 37), kde se předpokládá realizace v podobě modálního okna podle vzoru *Modal Panel*. Kromě uvedených vzorů se také počítá s využitím pokročilých animací (podle vzoru *Animated transition*) pro veškeré pohyby či změny prvků v rámci hlavního okna, *Progress Indicator* pro indikaci průběhu činností, které neproběhnou okamžitě a případně dalších vzorů, jenž budou v dalších částech práce zmíněny u případného konkrétního využití. Veškeré zmíněné návrhy jsou umístěny v příloze této práce.



Obr. 11 Přepracovaný návrh uživatelského rozhraní

6.4 Návrh architektury aplikace

Posledním krokem, který je nutné učinit před zahájením samotné implementace je návrh architektury realizované aplikace a specifikace nefunkčních požadavků.

V části práce, ve které byly provedeny rešerše jednotlivých v současnosti existujících a využívaných GIS, bylo prezentováno několik možných řešení disponujících v podstatě kompletní funkcionalitou, která byla výše zmíněna v rámci specifikace funkčních požadavků. Bylo by tedy zbytečné a také poměrně náročné implementovat kompletní vlastní GIS zpracovávající geodata. V jedné z předešlých kapitol byl mimo jiné zmiňován také svobodný systém GRASS GIS, jenž disponuje možností komunikace pomocí zadávání příkazů do příkazové řádky. A jelikož je GRASS GIS velmi komplexní systém, tak samozřejmostí jsou i veškeré požadované funkční požadavky. Především z tohoto důvodu bude tato aplikace sloužit jako nosný bod nově navrhovaného GIS pro dotykové obrazovky. Dalším důvodem je skutečnost, že GRASS GIS nedisponuje grafickým uživatelským rozhraním, které by splňovalo stanovené požadavky, jak bylo také v uvedeno v příslušné kapitole.

Navrhované řešení tedy bude realizováno jako *front-end* aplikace, která bude fungovat právě nad GRASS GIS, se kterým bude komunikovat prostřednictvím zmiňované příkazové řádky. GRASS GIS tak bude poskytovat a zpracovávat požadovaná geodata a jeho výstupy budou následně zpracovávány a vykreslovány pomocí nově navrženého uživatelského rozhraní.

- **Souhrn nefunkčních požadavků:**
    - Systém GRASS GIS jako *back-end*.

- *Front-end* vytvořen pomocí knihovny Qt (implementace v jazyce *QML*, *C++,* případně *JavaScript*,…).
- Moderní rozhraní pro dotykové obrazovky.
- Realizováno pro platformu *Windows*.

6.5 Implementace

6.5.1 Zprovoznění výsledné aplikace

Pro spuštění výsledného řešení je nutné mít na svém zařízení nainstalován v první řadě GRASS GIS. Jelikož bude aplikace nativně pro platformu *Windows*, tak i verze zmiňovaného systému musí být určena pro tuto platformu. GRASS GIS určený nativně pro Windows nese označení WinGRASS. K dispozici je hned několik verzí. Nejaktuálnější je momentálně 7.0, která je však právě ve vývoji a vydávány jsou pouze beta verze. Pro výslednou aplikaci je tedy zapotřebí některá ze stabilních verzí 6.4, které jsou k dispozici ke stažení zdarma na oficiálních stránkách tohoto svobodného systému (GRASS Development Team, 2012) nebo na přiloženém CD v podobě samostatného instalátoru. Při vývoji a testování aplikace byla využita především verze 6.4.3 a také poslední svn verze 6.4.5svn. K získání plné funkcionality a funkčnosti WinGRASS je vhodné instalovat aplikaci s právy administrátora. Po nainstalování tohoto systému je doporučeno jej rovnou vyzkoušet. Po spuštění se totiž mohou vyskytnout drobné problémy či nedostatky. Nejčastěji se pro danou platformu vyskytují potíže s okamžitým pádem po startu aplikace, případně chybová hlášení o chybějících knihovnách *MSVCR71.dll* a *MSCVP100.dll* a podobně. Tyto problémy lze obvykle vyřešit instalací *.NET Frameworku* společnosti *Microsoft*. Další známé vyskytující se problémy s instalací či používáním GRASS GIS na platformě *Windows* jsou popsány na příslušných wiki stránkách systému i s případnými možnými postupy, jak dané problémy řešit. (GRASS-Wiki, 2014)

Po instalaci je vhodné dále mít k dispozici alespoň nějaká testovací geodata ve formátu GRASS. Tato data lze opět získat z různých zdrojů. Nejjednodušší možností je nechat si data rozbalit přímo při instalaci WinGRASS. Tato data jsou rovněž dostupná opět na oficiálních stránkách produktu GRASS GIS (GRASS GIS, 2014), případně lze stáhnout také například množinu volně šiřitelných geografických dat České republiky *FreeGeoDataCZ* (FreeGIS, 2014). Množiny dat, které byly pro práci využity mnou, jsou opět umístěny také na přiloženém CD.

Poslední úpravy, které je zapotřebí udělat k tomu, aby bylo možné WinGRASS využívat způsobem, jaký byl uveden výše, tzn. pomocí komunikace přes příkazovou řádku, jsou drobné maličkosti ve spouštěcích dávkových souborech. První úpravou je odstranění či alespoň zakomentování řádku pro přesunutí do adresáře s aktuálním uživatelským profilem *cd "%USERPROFILE%"*, kam se velmi často ukládají data aplikačních nastavení. Tento příkaz se nachází v dávkovém souboru *grass64.bat* umístěného v instalačním adresáři WinGRASS. Pro jistotu je dobré zkontrolovat, zda je v tomto dávkovém souboru správně nastavena hodnota proměnné prostředí *GISBASE*. Tato proměnná by měla obsahovat správnou cestu do

instalačního adresáře WinGRASS. V tomto dávkovém souboru jsou také vyvolávány další dva dávkové soubor, konkrétně jsou to *env.bat* a *Init.bat* nacházející se v podadresáři */etc*. Právě v souboru *Init.bat* je nutné udělat poslední změnu. Jedná so opět o odstranění jednoho příkazu. Tento příkaz se nachází přibližně na 165. řádku této dávky a konkrétně je to příkaz *"%GISBASE%\etc\set_data"*, který slouží k tomu, že v příkazové řádce vyvolá jakýsi dialog (Obr. 12), kde čeká na uživatelem specifikovanou množinu dat (*Mapset*), lokaci (*Lacation*) a adresář s geodaty (*Database*), se kterými chce v rámci aplikace pracovat. Tři zmíněné pojmy budou upřesněny v další části text. Jelikož dialog vyvolaný programem *set_data.exe* nebylo možné vhodně zpracovat a zadat požadované vstupy, byla volba jednotlivých požadovaných parametrů nahrazena načítáním z inicializačního souboru, o kterém bude více informaci uvedeno také později.

Obr. 12 Dialog vyvolaný programem set_data.exe v příkazové řádce

Pro využití možnosti přidání WMS vrstvy je pak nutné mít tento modul správně zprovozněn. Doporučuji si nejdříve vyzkoušet modul *r.in.wms* přímo ve WinGRASS, ideálně z příkazové řádky. S využitím tohoto modulu je spojena celá řada problémů týkajících se chybějících nástrojů či knihoven. Některé z nich jsou k dispozici na přiloženém CD, zejména ty, které jsou v manuálech k danému modulu na webových stránkách vypsány mezi požadavky. Na příslušných stránkách projektu existuje také několik stále otevřených defektů k této problematice (GRASS GIS Tracker and Wiki, 2009), zejména 820 a 2010. Jako alternativa k uvedenému modulu může sloužit i novější verze *r.in.wms2*. Tento modul lze získat jako rozšíření. Stažení se provádí v grafickém rozhraní pomocí nabídky *Nastavení – Addons extensions – Install extension from addons*. Stejně jako předchozí modul, je i tento nutné vyzkoušet. Je velmi pravděpodobné, že také nebude správně fungovat bez určitých opravných kroků. Pro fungování aplikace však tyto moduly nejsou nutností. Rastrové a vektorové vrstvy tyto požadavky nemají a lze je bez problému využít.

Celá aplikace je pak spustitelná pomocí *.exe* souboru podadresáře *release* ve složce *grass_demo* na přiloženém CD. Nebo je možné otevřít soubor projektu *grass_demo.pro* a celý projekt sestavit znovu.

6.5.2 Realizace uživatelského rozhraní

Výsledné realizované grafické uživatelské rozhraní vychází z návrhů, které byly uvedeny v jedné z předešlých podkapitol. Oproti návrhu většina prvků nedoznala

příliš mnoha změn. Pouze některé funkce například nejsou dostupné přes klasická tlačítka, ale pomocí gesta atd. Typicky například mazání vrstvy ve správci vrstev.

- **Možnosti tvorby prvků rozhraní**
  Knihovna Qt umožňuje několik přístupů, jak návrhy rozhraní přenést do reálné podoby. Jednou z možností je využít nástroj Qt Quick Designer. Jedná se standardní editační nástroj této knihovny. Tvorba je založená na tom, že si z části dostupných typů sestavíte vlastní jednotlivé obrazovky či kompletní rozhraní jejich přetažením na plátno. Uživateli je k dispozici přehledná struktura přidaných typů a možnosti snadného nastavení jejich veškerých vlastností. Jako velkou přednost tohoto přístupu vidím především možnost přehledného a jednoduchého definování stavů objektu a přechody mezi nimi. Při realizaci výsledné aplikace však bylo nutné doplňovat funkcionalitu určitými funkcemi a podobně. Bylo by tedy potřeba neustále se přepínat mezi tímto grafickým editorem a editorem kódu, což není příliš efektivní. Proto tento přístup byl využíván jen sporadicky.

  Další možností realizace konečné podoby uživatelského rozhraní je vytvoření návrhu v grafickém editoru *Adobe Photoshop* nebo *GIMP* a jeho následný export přímo do *QML* souboru. Jedná se o velmi efektivní a rychlý způsob tvorby rozhraní. Jsou-li správně vytvořeny objekty v grafickém editoru, to znamená především správně rozvrstvené, budou tyto vrstvy po exportu tvořit jednotlivé *QML* položky, včetně jména či stylů.

  Základním způsobem realizace návrhu rozhraní je tvorba jednotlivých komponent. V tomto spočívá i největší výhoda této knihovny, konkrétně modulu *QtQuick* a jazyka *QML*. Proto jsem tento způsob zvolil jako primární pro mou realizaci návrhu uživatelského rozhraní. Princip tvorby spočívá v tvorbě jednotlivých komponent skládajících se ze základních *QML* typů nebo případně dalších komponent. Vše lze jednoduše psát přímo v editoru kódu. Tyto komponenty lze samozřejmě vytvářet také pomocí zmiňovaného Designeru, ale opět se jedná o vizuální podobu toho, co se dá velmi snadno a především rychleji vytvořit pouze pomocí kódu. Proto veškeré prvky této aplikace jsou vytvořeny pouze na základě dostupných *QML* typů a možností této knihovny.

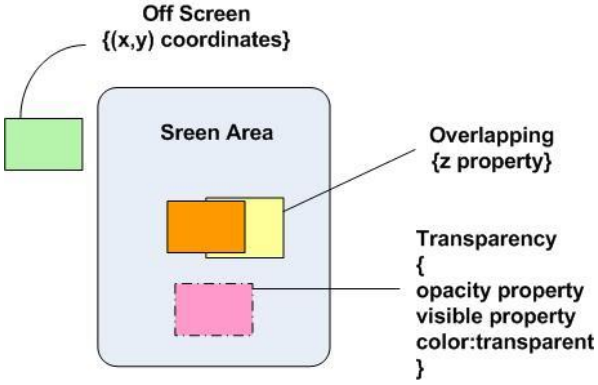
- **Řízení vizualizace prvků**

  Pro správnou vizualizaci jednotlivých prvků uživatelského rozhraní je velmi důležité jeho řízení. To znamená kontrola toho, kdy a jak se jednotlivé prvky zobrazují. Techniky, které jsou v této knihovně k dispozici lze opět rozdělit do 3 kategorií.

  1. **Nastavení hodnot příslušných parametrů**: každý *QML* typ obsahuje parametry jako *opacity*, souřadnice *z*, *visible*, *color*, pomocí nichž lze řídit zobrazení jednotlivých elementů z vlastních stylů nebo i prostřednictvím jiných objektů. Nedochází zde k vytváření či ničení prvků, pouze k jejich zviditelnění a zneviditelnění, tedy způsobu renderování.

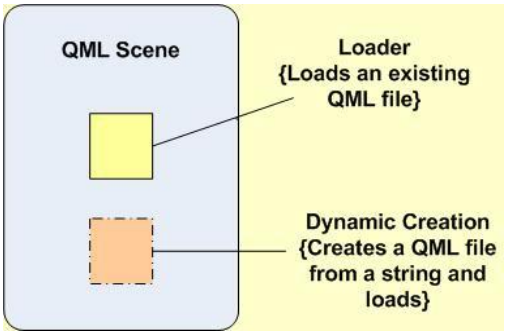
2. **Nastavení velikosti a pozice objektu**: jednotlivé objekty lze také skrýt například pomocí nastavení hodnot souřadnic *x* a *y*. Nebo také pomocí změny tvaru či velikosti objektu pomocí parametrů *width* a *height*. Opět zde nedochází k vytváření či ničení jednotlivých objektů.

   Díky tomu, že u těchto dvou metod nedochází k tvorbě či zániku jednotlivých objektů (Obr. 13), jsou výhodně z hlediska využití paměti. Jejich použití je také velmi intuitivní a jednoduché. V aplikaci jsou tyto způsoby hojně využívány, zejména při nastavení stavů objektu a přechodu mezi nimi.



Obr. 13 Princip řízení vizualizace prvků (kategorie 1 a 2)
Zdroj: Qt Project, 2014

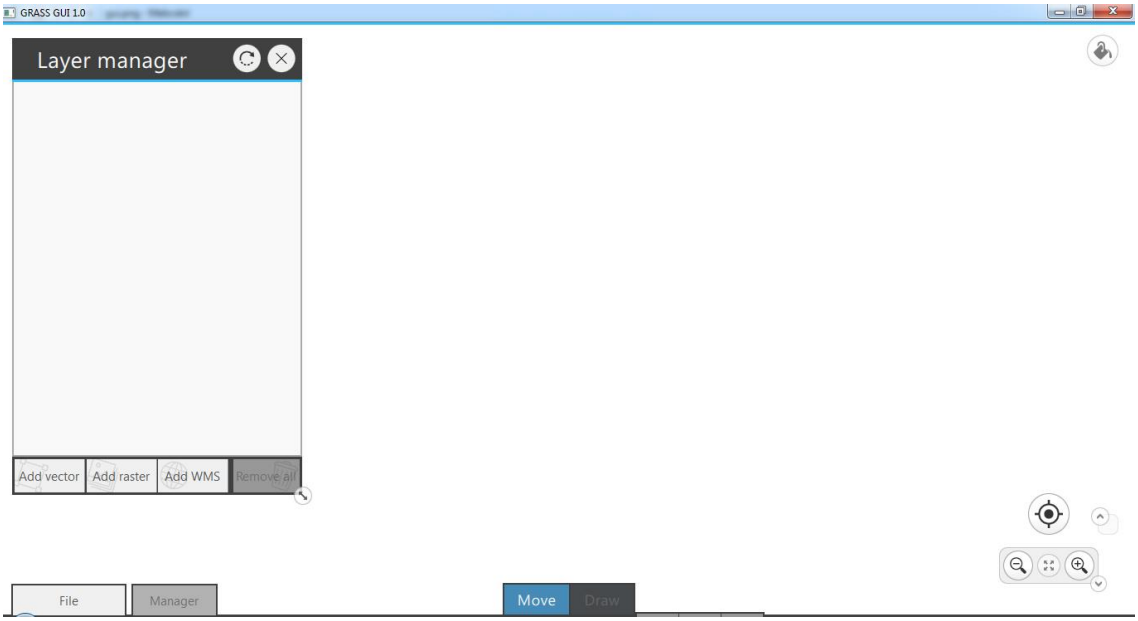
3. **Dynamická tvorba objektů**: jednotlivé objekty jsou vytvářeny za běhu aplikace. Tento mechanismus (Obr. 14) je založen na využití typu *Loader* a na dynamickém vytváření objektů například pomocí skriptů. Tento přístup má však již dopad na paměť. V aplikaci je tento princip využit pro přidání nových mapových elementů.



Obr. 14 Princip řízení vizualizace prvků (dynamická tvorba objektů)
Zdroj: Qt Project, 2014

Ukázka výsledné realizace grafického uživatelského rozhraní je zachycena na obrázku níže (Obr. 15). Lze vidět, že v aplikaci nejsou klasická desktopová menu, jak

bylo zmíněno v předešlých částech textu, ale tlačítka jsou zredukována a více strukturována. Jsou řešena převážně animovanými rozbalovacími seznamy či skupinami dalších tlačítek. Pro možnost přepnutí mezi přidáním mapového elementu a manipulací s mapou byl použit přepínač. Pokud uživatel zvolí možnost kreslení, jsou mu nabídnuty prvky, které lze přidat. Pro editaci přidaného elementu jej zle editovat gesty (*drag and drop* pro přemístění, kliknutí přenese prvek do popředí, dvojklik zobrazí atributy elementu). Popis implementovaných gest a možnosti navigace v aplikaci jsou také detailněji rozebrány v jedné z následujících podkapitol. Správce vrstev je možné libovolně přesouvat, upravovat jeho velikost a také jej skrýt. Správce je také automaticky skryt, pokud velikost hlavního okna překročí definované hodnoty. Jednotlivé vrstvy ve správci je možné opět gesty mazat, přesouvat a zobrazovat vlastnosti vrstvy (*swipe* směrem doleva pro mazání, kliknutí pro zobrazení a skrytí vlastností vrstvy, dlouhý stisk pro přesunutí vrstvy). Navigační tlačítka jsou sice stále umístěna v pravém dolním rohu, ale v počátečním stavu jsou skryta, aby uživateli zbytečně nepřekážela. Zůstalo zobrazeno pouze tlačítko pro vycentrování mapy a tlačítka pro změnu měřítka. Jednotlivé skupiny navigačních tlačítek a tlačítek pro změnu měřítka lze libovolně skrýt či zobrazit.



Obr. 15 Výsledná realizace grafického uživatelského rozhraní aplikace

6.5.3 Propojení s GRASS GIS

Jelikož navrhovaná aplikace slouží jak *front-end* nad GRASS GIS, je potřeba nějakým způsobem danou aplikaci na pozadí spustit a navázat spojení. Přesně pro tyto účely, tedy spouštění externích procesů a následná komunikace s nimi, poskytuje knihovna Qt nástroj v podobě třídy *QProcess*. Třída nabízí hotovou implementaci celé řady užitečných funkcí, signálů a slotů. Na základě této třídy je v aplikaci od-

vozen potomek *Process*. Nejdůležitější předimplementované funkce z třídy *QProcess*, které jsou využívány i jejím zmiňovaným vytvořeným potomkem, jsou:

- **void start(program, arguments, mode)**: tato veřejná procedura umožňuje spustit daný program jako nový proces se zadanými argumenty a v požadovaném vstupně-výstupním módu. Ve výsledné aplikaci je tímto procesem *cmd.exe*, tedy příkazová řádka spuštěna v módu pro čtení i zápis. Pokud je proces v pořádku spuštěn vyvolá signál *started()*, pokud nastane při spouštění chyba, vyvolá signál *error()*. Protože jsou jednotlivé procesy spuštěny touto funkcí asynchronně, mohou být i jednotlivé signály zpožděné, proto je někdy vhodné využít jinou veřejnou metody *waitForStarted()*, která ve výsledku vrací pravdivostní hodnotu a blokuje veškeré operace, dokud není vyvolán signál *started()*, nebo dokud neuběhne zadaný čas pro blokování. Tato funkce však může způsobit zamrznutí aplikace.
- **void kill():** okamžitě ukončí daný proces. Tato procedura bude zavolána před ukončení *front-end* aplikace.
- **qint64 writeData(data)**: funkce sloužící k zápisu jednotlivých příkazů do příkazové řádky, které jsou následně vykonány.
- **QByteArray readAllStandardError()** a **readAllStandardOutput()**: tyto dvě veřejné funkce slouží jako zpětná vazba komunikace s příkazovou řádkou. Vrací veškeré výstupy spuštěného externího procesu.

V rámci vlastní implementace a experimentováním s třídou *QProcess* byly či jsou ve výsledné aplikaci použity některé její další metody. Pro spuštění externího procesu jsem například v počátcích zkoušel, kromě stávajícího řešení prostřednictvím *start()*, také funkci *startDetached()*, která také spustí externí proces, ale tento proces tzv. oddělí. To znamená, že nově spuštěný proces běží nezávisle na procesu, který jej vyvolal. Z toho může být poměrně zřetelné, že takový proces se například o něco hůře ukončuje. Proto při volání *startDetached()* je jedním z parametrů identifikační číslo tohoto procesu, které je mu přiděleno systémem po spuštění a na základě tohoto *pID* je následně při zavolání funkce *killDetached()* ukončen příslušný proces. Tyto funkce byly při využívání občas poměrně problematické, proto jsou nyní v aplikaci pro spuštění a ukončení příkazové řádky využívány výše zmíněné funkce *start()* a *kill()*, které svůj účel plní mnohem lépe.

V implementační části třídy *Process* byla naimplementována také metoda, která nahrazuje volání výše zmiňovaného *set_data.exe* programu. Tato metoda nese název *initEnv()* a jako parametry dostává při svém volání právě onu *Database*, *Mapset* a *Location*. *Database* není v podstatě nic jiného, než cesta k adresáři s dostupnými geodaty (viz Obr. 12), *Location* je jiný název pro jednu konkrétní množinu geodat a *Mapset* pak představuje něco jako projekt nad vybranou množinou dat. Tato metoda tedy slouží k tomu, že v adresáři, kde jsou uložena aplikační data (APPDATA), vytvoří inicializační soubor s uživatelem prostřednictvím grafického rozhraní definovanými zmíněnými parametry. Její konečná implementace je vidět na Obr. 16, kde lze také vyčíst, že aby bylo možné určit cestu ke správnému

adresáři, to znamená zjistit hodnotu proměnné prostředí *%APPDATA%*, je nutné nejdříve využít například příkaz *getenv("APPDATA").*

Kromě dosud uvedených metod obsahuje třída *Process* také implementace několika dalších metod, signálů či slotů a samozřejmě také parametrů nutných pro správnou funkčnost aplikace.

```
156 void Process::initEnv(QString gisDbase, QString locName, QString mapset){
157     QString gisDB = "GISDBASE: " + gisDbase + "\n";
158     QString locality = "LOCATION_NAME: " + locName + "\n";
159     QString mSet = "MAPSET: " + mapset + "\n";
160     QString data = gisDB + locality + mSet + "\n";
161     QString appdata = getenv("APPDATA");
162
163     QFile file;
164     file.setFileName(appdata + "\\GRASS6\\grassrc6");
165     if (file.open(QIODevice::WriteOnly | QIODevice::Truncate)){
166         file.write(data.toStdString().c_str(), qstrlen(data.toStdString().c_str()));
167         file.close();
168     } else {
169         qDebug() << "File " << file.fileName() << "can not be opened";
170     }
171 }
```

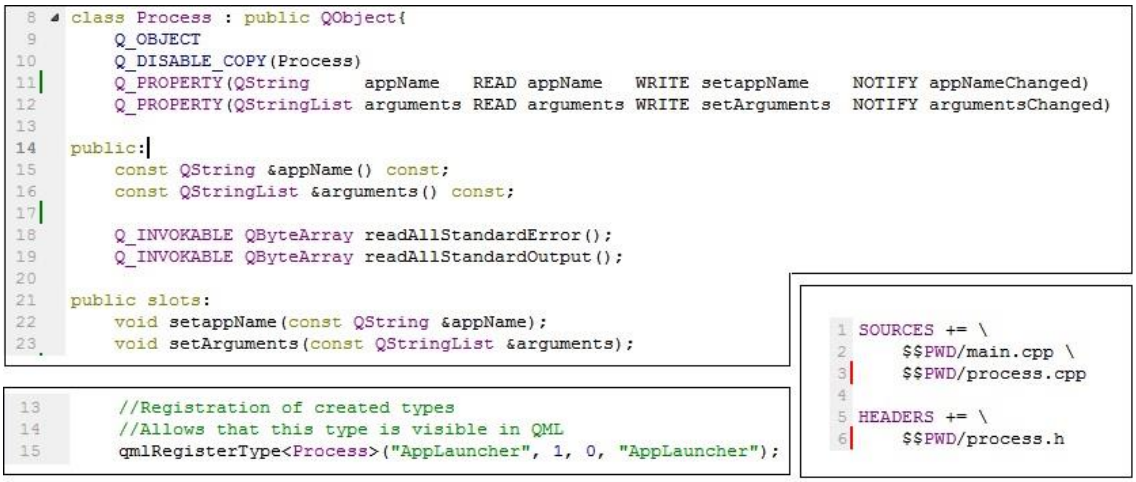
Obr. 16 Implementace metody initEnv()

6.5.4 Provázání C++ a QML

Veškerá zmíněná implementace a použití třídy *QProcess* je napsáno v jazyce *C++*. K tomu, aby mohly být jednotlivé konstrukce dostupné a použitelné také v uživatelském rozhraní psaném v jazyce *QML*, je potřeba tyto implementace zpřístupnit. K těmto účelům poskytuje knihovna Qt několik API, prostřednictvím kterých lze v jazyce *C++* vytvářet širokou škálu rozšíření od definování nových *QML* typů až po vytvoření komplexních zásuvných modulů.

Mnou vytvořená třída *Process* je pro další *QML* implementace zpřístupněna jako nový typ. Využívá se k tomu tzv. *meta object system* knihovny Qt. Tento systém poskytuje potřebný mechanismus signálů a slotů. Jako základní třídu sloužící k tomu, aby umožňovala jednotlivým objektům přístup k možnostem tohoto systému, využívá *QObject*. K navázání nově vytvořené třídy na uvedený systém je nutné v deklarační části třídy využít makro *Q_OBJECT*, které říká, že nově vytvořená třída je odvozena on třídy *QObject*. Dále je nutné využít makra *Q_PROPERTY*. Pomocí tohoto makra jsou v *QML* dostupné i jednotlivé proměnné třídy, které jsou v rámci parametrů tohoto makra současně napojené na příslušné signály a sloty. Při zpřístupnění i jednotlivých funkcí deklarovaných v rámci třídy *Process* lze využít 2 možnosti. První z nich je opět propojení s *meta object systémem* prostřednictvím makra, kdy na libovolnou metodu stačí aplikovat makro *Q_INVOKEABLE* a daná metoda je následně přístupní i v *QML*. Druhou možností je deklarovat metodu jako slot. Obě možností jsou v mé třídě *Process* využity. Poté, co jsem pomocí jazyka *C++* vytvořil nový typ, deklaroval příslušné proměnné, signály a sloty, bylo pro jejich použití v *QML* systému nutné v hlavní funkci programu provést tzv. registraci tohoto nového typu. Registrace se provádí pomocí funkce *qmlRegisterType<třída>(namespace, verze, revize, QML název)* a bez této registrace není možné

nový typ v *QML* využívat. Poté už stačilo jen zahrnout příslušný hlavičkový a zdrojový soubor do projektu a provést import v libovolném *QML* souboru. Konečnou podobu jednotlivých uvedených kroků znázorňuje následující obrázek (Obr. 17) s výběrem několika vzorových řádků příslušných zdrojových kódů. Z registrace nového typu lze vyčíst, že příslušný import do příslušného *QML* souboru provede příkaz *import AppLauncher 1.0* a v rámci celé aplikace pak bude tento typ přístupný jako *AppLauncher*, u kterého lze využívat všechny deklarované atributy a metody.



Obr. 17 Ukázka deklarace nového typu v C++ a jeho provázání s QML

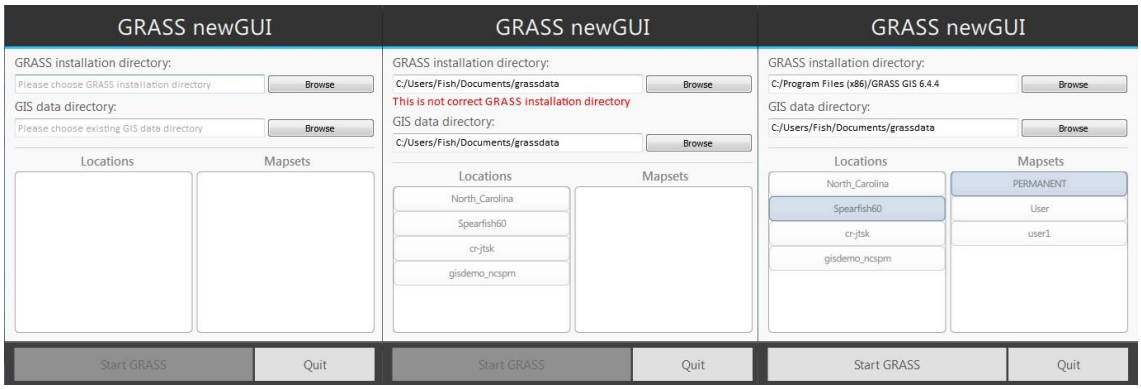
Téměř stejný postup lze praktikovat i v případě, že se nebude jednat pouze o nový typ, ale o zásuvný modul. Registrace pak neprobíhá v hlavní funkci programu, ale je zapotřebí vytvořit podtřídu třídy *QExampleQmlPlugin*, v rámci které se provede příslušná registrace nových typů a pomocí makra *Q_PLUGIN_METADATA* export celého zásuvného modulu. Posledním krokem je pak vytvoření tzv. *qmldir* souboru, který slouží k popisu daného zásuvného modulu a je nutný k následnému importu.

6.5.5 Dialog pro výběr dat a spuštění aplikace

Po spuštění samotné aplikace je uživatel, předtím než se dostane do hlavní části aplikace, vyzván k tomu, aby vybral či zadal několik parametrů, které jsou nutné ke spuštění WinGRASS na pozadí. Jsou to samozřejmě 3 výše zmíněné proměnné pro uvedený inicializační soubor. Kromě těchto hodnot ještě musí uživatel zadat cestu do adresáře, ve kterém má nainstalován WinGRASS 6.4.

Tento krok je řešen jako modální okno vykreslené nad hlavním oknem celé aplikace, které je nastaveno jako kompletně průhledné, dokud není stisknuto některé z tlačítek *Start GRASS*, nebo *Quit* (viz Obr. 18). Možností, při tvorbě tohoto okna, bylo využít některý z několika dostupných *QML* typů sloužících pro podobné účely. Nejblíže se požadavkům blížil typ *Dialog* určený přímo pro dialogová okna. Nakonec však byl zvolen obecnější typ *Window*, od kterého je uvedený typ *Dialog* odvozen, především díky možnosti kompletně si přizpůsobit obsah okna.

Při implementaci tohoto okna je použita celá řada *QML* typů, konstrukcí a postupů, které jsou využívány v mnoha dalších částech aplikace.



Obr. 18 Spouštěcí dialogové okno aplikace[5]

Obsah tohoto okna se tedy skládá z hlavičky a patičky, oddělovačů těchto částí a hlavního obsahu okna. Hlavička s patičkou jsou pouze jednoduché obdélníkové objekty ukotvené k příslušným stranám okna a nastavenou barvou a pevnou výškou. Stejně tak jednotlivé oddělovače těchto částí okna jsou pouze obdélníky, které pomáhají graficky dělit okno na jednotlivé logické části. Oddělovače jsou definovány jako vlastní typ a jsou využívány na více místech v aplikaci.

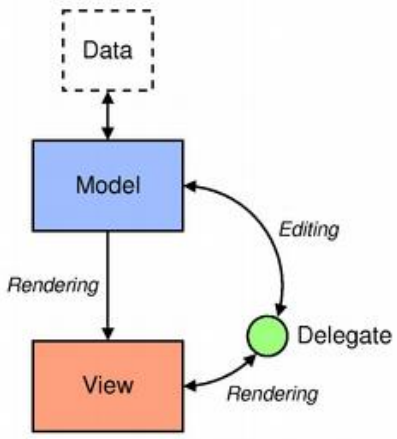
Hlavní částí tohoto okna jsou pole pro vyplnění potřebných parametrů. Prvním z nich je volba adresáře, ve kterém se nachází WinGRASS, konkrétně jeho spouštěcí dávkový soubor *grass64.bat*. Tato volba lze učinit 2 způsoby. Buď pomocí tlačítka *Browse*, které vyvolá souborový dialog a jeho prostřednictvím pak lze v průzkumníku nalézt požadovaný adresář a potvrdit ji, nebo prostřednictvím přímého zapsání cesty do příslušného textového pole. V obou případech pak cesta zůstává vyplněna v příslušném textovém poli. Pokud se uživatel rozhodne zadávat cestu do textového pole a například nezná cestu celou, stačí vyplnit i její část a stiskem klávesy *Enter* se opět zavolá souborový dialog, který se nyní bude nacházet v adresáři podle dosud vyplněné cesty. Pro ověření správnosti instalačního adresáře je aplikována kontrola vstupu. Této kontroly je dosaženo pomocí *QML* typu *FolderListModel*, který slouží k procházení zvoleného adresáře a je možné mu nastavit například filtry pro procházené typy souborů a podobně. Použil jsem tedy nastavení tohoto parametru *nameFilters: ["grass*.bat"]* k tomu, abych zjistil, zda se ve vybraném adresáři nachází alespoň jeden soubor podle zadaných kritérií. Pokud není nalezen alespoň jeden výsledek, uživateli se zobrazí červené hlášení o nevyhovujícím adresáři. Stejné postupy a principy jsou pak aplikovány na volbu adresáře s geodaty. Do obou textových polí je pro ještě zřetelnější význam těchto polí

[5] Vlevo původní stav bez vyplněných hodnot, uprostřed špatné hodnoty či nekompletní vyplnění s informačním hlášením pro uživatele a vpravo správně kompletně vyplněné hodnoty a zaktivnění tlačítka *Start GRASS*.

umístěn tzv. *placeholderText*, který uživateli napovídá, co by měl do textového pole vyplnit.

Pro volbu obou adresářů lze použít zmiňovaný souborový dialog (*FileDialog*). Aby nebylo nutné implementovat pro každé tlačítko či pole vlastní dialog, byla tomuto dialogu implementována jednoduchá metoda, která v parametru obdrží *ID* objektu, který jej vyvolal a na jeho základě pak příslušnému objektu vrátí požadované výsledky. Tato metoda zároveň slouží pro otevření daného dialogu. Při využívání souborového dialogu, především pak s manipulací s cestami, které uchovává, je nutné dát pozor na typové konverze. Dialog vrací hodnoty typu *URL* cesty, pro ostatní potřeby je obvykle nutný textový formát.

Poprvé v aplikaci je zde využit i typ *Action*, který slouží k vykonání stejného příkazu, či posloupnosti příkazů, bez ohledu na to, jak a odkud byly v aplikaci vyvolány. Typicky například vyvolání souborového dialogu stisknutím tlačítka *Browse* nebo klávesou *Enter* z textového pole. Nejčastěji jsou tedy akce v aplikaci vázány na tlačítka, ale je možné je spustit odkudkoliv pomocí signálu *trigger()*. Tlačítka s příslušnou akcí tento signál vysílají automaticky po stisknutí. Výhodou akcí je také to, že lze pro ně definovat například klávesovou zkratku, nastavit ikonu, tooltip či text, který se pak objevuje na příslušných tlačítkách, na která je daná akce nastavena, nebo připojit na tzv. exkluzivní skupinu.



Obr. 19 Schéma fungování modelů a pohledů v modulu Qt Quick
Zdroj: Qt Project, 2014

Další hodnoty, které uživatel musí zvolit před spuštěním aplikace, jsou *Mapset* a *Location*. Pro volbu těchto parametrů byl zvolen výběr ze seznamu dostupných možností. Seznamy jsou tedy 2 (*locListView* a *msetListView*) pro každý jednotlivý parametr. Přesně pro takové účely je v *QML* typ *ListView*, kterému je nutné předložit alespoň model (obsahuje požadovaná data a jejich strukturu, tedy seznam položek, které mají být vypsány) a delegáta (komponenta představující jednu položku daného seznamu, konkrétně její visuální a funkční nastavení). Princip fungování zachycuje Obr. 19. Jako model jsem zvolil opět *FolderListModel*, který podle jména seřadí obsah adresáře (pouze složky), který mu byl předložen prostřednictvím

textového pole pro určení geodat. Složky v tomto adresáři jsou pak vypsány v seznamu a díky komponentě delegáta lze vybrat libovolnou položku v seznamu. Pro zvýraznění vybrané položky je využita další komponenta, kterou je nutné přiřadit parametru *highlight* příslušného seznamu. Po vybrání položky v seznamu lokací je tato položka přidána k cestě s geodaty a v dostupných seznamu projektů je tak vypsán opět seznam obsažených složek. Jelikož jednotlivé položky v seznamech nejsou pouze text, ale jedná se o jednotlivé položky souborového modelu, je nutné k přístupu názvu těchto položek využít daný model, resp. seznam variant (položek) tohoto modelu. Proto jsem komponentu delegáta každého seznamu obohatil o proměnnou *myLocData*, respektive *myMsetData*, a přístup ke konkrétnímu názvu položky v daném seznamu se pak učiní například takto *locListView.currentItem.myLocData.fileName*.

Poslední částí tohoto okna jsou tlačítka pro spuštění (*StartGRASS*) a ukončení (*Quit*) aplikace. Tlačítka jsou ukotvena ke stranám patičky okna, které byly zmíněny výše. První tlačítko sloužící k přesunu do hlavního okna aplikace je nastaveno jako neaktivní, dokud nejsou vyplněna všechna požadovaná pole. V případě, že jsou všechna pole řádně vyplněna a tlačítko je aktivní, po jeho stisknutí se provede sekvence příkazů:

1. Nastavení proměnných *AppLauncheru* hodnotami cest k příslušným zvoleným adresářům.
2. Spuštění externího procesu (příkazové řádky) na pozadí.
3. Čekání na korektní spuštění procesu a zavolání metody *initEnv()* s příslušnými parametry.
4. Zavření úvodního okna a zviditelnění hlavního okna aplikace.

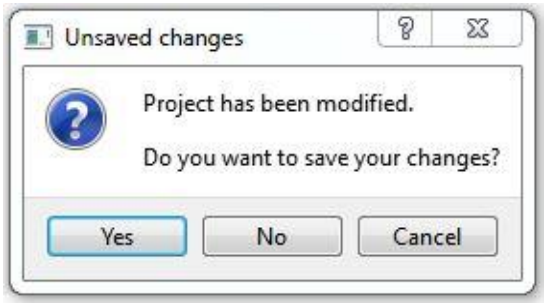
Tlačítko *Quit* pouze zavře úvodní okno a ukončí celou aplikaci. Obě tlačítka mají nataven grafický styl, kterým však bude věnována samotná podkapitola, proto se jim zde blíže věnovat nebudu.

6.5.6 Správa projektu

Aplikace disponuje možností ukládat a nahrávat vlastní projekty. Pro každou operaci s projektem (nový, otevřít, uložit a uložit jako) je definována vlastní akce. Akce pro otevření a pro uložení projektu pod určitým názvem vyvolá přímo příslušný souborový dialog. Akce pro vytvoření nového projektu zjistí, zda byly uloženy všechny dosavadní změny, pokud ano, smaže všechna uživatelská data (vyexportované mapy) a vyčistí správce vrstev. Pokud existují změny, které ještě uloženy nebyly, vyvolá okno (*MessageDialog*) s upozorněním na neuložené změny a vyzve uživatele k příslušné další akci (viz Obr. 20).

Co se týká poslední zmiňované akce pro uložení projektu, dochází nejdříve k zjištění toho, zda existuje cesta k souboru stávajícího projektu. To znamená, jestli byl projekt načten, nebo alespoň uložen pomocí možnosti *Uložit jako*. Pokud cesta existuje, dojde k uložení projektu. V opačném případě je vyvolán dialog pro zmíněnou možnost uložení projektu.

FileDialog pro možnost *Uložit jako* funguje poměrně jednoduše. Pouze do příslušného uživatelem zvoleného adresáře uloží soubor pod daným názvem. Uložení probíhá tak, že dochází k postupného vyčítání příkazu a průhlednosti pro každou vrstvu z modelu a následnému zápisu do souboru. O poznání složitější je dialog pro otevření projektu (*openFileDialog*). Zde je nutné vyčíst položky ze souboru, správně je rozdělit, naplnit příslušný model vrstev a na základě jednotlivých příkazů vygenerovat příslušné mapy. K tomu je využívána celá řada naimplementovaných metod, které korespondují s jednotlivými položkami, které jsou během chodu aplikace udržovány v modelu vrstev.



Obr. 20 Informační hlášení o neuložených změnách

Onen zápis a čtení ze souboru je zprostředkováván vlastním typem vytvořeným stejným principem, jak je tomu u typu pro externí proces *AppLauncher*, tedy implementací v *C++*, registrací pro zpřístupnění v *QML* a následným importem v libovolném souboru. Tento typ je pojmenován *FileIO* a ve své implementaci v *C++* obsahuje 3 metody - pro zápis, čtení a vyčištění souboru před uložením. Ukládání se provádí pouze do základního textového souboru. Hlavním důvodem je struktura ukládaných dat (ukládány jsou především příkazy pro generování map v GRASS) a čitelnost výstupu. Uživatel znalý GRASS GIS si díky tomu může velmi jednoduše vytvořit vlastní projekt nebo snadno upravit uložený, který následně stačí opět otevřít v aplikaci a za oment uvidí výsledné výstupy svých příkazů.

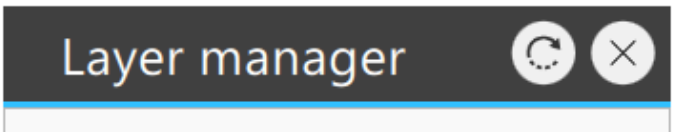
6.5.7 Správa vrstev

Klíčová část aplikace je orientována na manipulaci a správu vrstev, která je v aplikaci realizována prostřednictvím správce vrstev. Princip přidání a zobrazení vrstev lze ve zkratce popsat následovně. Jednotlivé vrstvy, které jsou uživatelem přidány, jsou zobrazeny jako vyexportované PNG soubory. Tyto soubory jsou umístěny do uživatelem zvolené složky s prostorovými daty. V určitém definovaném momentu jsou tyto obrázky přesunuty do podadresáře *user_data*, jehož obsah je pak následně vyčten a zobrazen. Všechny soubory této složky jsou reprezentovány také příslušnou položkou ve správci vrstev.

- **Manipulace se správcem vrstev**
  Tento správce vrstev je realizován jako objekt, který je umístěn v hlavním okně aplikace (Obr. 41 v příloze), avšak není ukotven k žádnému okraji, což

umožňuje jeho snadnou manipulaci v rámci tohoto okna. Celý správce se skládá z hlavičky, těla a patičky.

Hlavička (Obr. 21) slouží především k tomu, že pomocí ní a gesta *drag & drop* (princip fungování a implementace gest je umístěn v samotné podkapitole 6.5.11) je možné správce přemístit na libovolné místo v rámci hlavního okna aplikace. Možnost přemístění správce je indikována také změnou kurzoru. V hlavičce správce vrstev jsou umístěna také tlačítka pro rotaci a skrývání celého správce.



Obr. 21 Hlavička správce vrstev

Rotovat správcem vrstev je možné pouze směrem doprava o 90 stupňů. Po otočení správcem bylo nutné zajistit, aby okraje správce nepřekročily okraje hlavního okna. Při pozicích blízko od okrajů a následném otočení docházelo k tomu, že hlavička zůstala za okrajem hlavního okna a nebylo tedy možné správce znovu otočit či přemístit. Proto byla implementována metoda *checkBound()*, která na základě orientace správce udržuje tohoto správce uvnitř hlavního okna aplikace. Tato funkce může být velmi užitečná, pokud je aplikace využívána například na velkých obrazovkách, jako jsou například moderní dotykové stoly, a uživatel tak při práci nesedí klasicky pouze u stolu, ale pohybuje se kolem něj.

Dále je možné také správce s celým jeho obsahem skrýt. V počátečních fázích implementace byla zahrnuta možnost zobrazení správce vrstev a mapového prostoru do 2 samostatných oken, ale z praktického hlediska a také z důvodu snadnější manipulace byla nahrazena funkcí pro skrytí správce vrstev. Tato možnost je užitečná, pokud uživatel nemá k dispozici příliš velkou obrazovku a správce vrstev mu překáží v pohodlné práci s mapovými výstupy. Skrývání je umožněno díky stisknutí tlačítka křížku v hlavičce, které změní definovaný stav tohoto správce (změny stavu jsou blíže popsány v podkapitole 6.5.10). Implementace aplikace také obsahuje automatické skrývání správce, pokud uživatel zmenší výšku hlavního okna aplikace na hodnotu výšky správce vrstev a šířku na hodnotu, kdy šířka správce zabírá alespoň 2/3 šířky hlavního okna. K tomu jsou využity sloty hlavního okna aplikace *onWidthChanged* a *onHeightChanged*. Zobrazení či skrytí správce vrstev je indikováno a možné měnit také tlačítkem umístěném přímo na hlavní obrazovce. Při zobrazení správce zůstává tlačítko stisknuté.

V neposlední řadě je také možné změnit rozměry správce vrstev. V pravém dolním rohu správce je umístěn objekt typu *MouseArea*, díky kterému je možné pomocí stisku a následným tažením, tedy použití *drag & drop*

gesta, změnit šířku a výšku správce vrstev. Rozměry však nesmí být nižší, než definované minimální hodnoty.

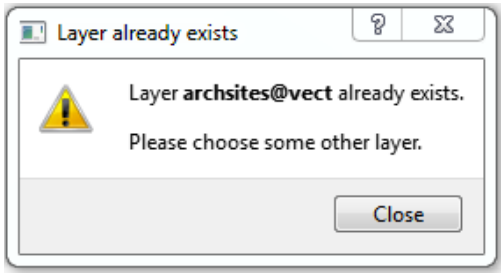
- **Obsah správce vrstev**

  Tělo správce vrstev je tvořeno seznamem přidaných vrstev, který v *QML* zastupuje typ *ListView*. Tento typ má jako model pro uchování přidaných vrstev definován *layerModel* (*LayerModel.qml*), který je typem *ListModel* a obsahuje celou řadu implementovaných funkcí sloužící k provádění operací nad příslušnými vrstvami. Jako povinný delegát je pak definována komponenta *listDelegate*, která je složena z definovaného typu *DelegateItem* a představuje vizuální podobu a funkcionalitu jedné položky daného seznamu.

    - **Přidání rastrové a vektorové vrstvy**

  Aby bylo možné naplnit model a tedy vložit položky do seznamu, je nutné nejdříve přidat nějaké vrstvy. Přidání vrstev se provádí stisknutím tlačítka pro přidání vrstvy podle jejího typu. Po stisknutí tlačítka pro přidání vektorové či rastrové vrstvy je spuštěna příslušná akce, která vyvolá vlastní definovaný modální dialog (*addLayerDialog*). Tento dialog (viz. Obr. 40 v příloze) je založen na obecném typu *Window* a jeho obsah je pak definován typem *AddLayerDialog* (*AddLayerDialog.qml*). Podobně jako správce vrstev je dialog pro přidání vrstvy sestaven z hlavičky, těla a patičky. Hlavička obsahuje titulek *Add new layer_type layer*, kde typ vrstvy je v titulku nastaven na základě toho, které tlačítko bylo stisknuto, respektive která akce vyvolala tento dialog. Akce pro přidání těchto 2 typů vrstev lze totiž vyvolat také pomocí klávesových zkratek *Ctrl + r* (rastr), *Ctrl + v* (vektor). Tělo dialogu pak pro rastrovou a vektorovou vrstvu obsahuje rozbalovací seznam, který obsahuje dostupné vrstvy. Položky tohoto rozbalovacího seznamu jsou uloženy, stejně jako seznam vrstev, v příslušném modelu (objekt typu *ListModel*). Model pro rastrové vrstvy nese *ID addRLmodel* a pro vektorové vrstvy *addVLmodel*. Naplnění těchto modelů je zajištěno pomocí tzv. *Repeateru*, který na základě počtu položek, který je mu předložen, vykoná určité operace. V tomto případě na základě položek svého modelu naplní položky modelu vrstev. Opět tedy potřebuje předložit nějaký model. Tentokrát je mu předložen model typu *FolderListModel*, který byl již zmíněn výše a slouží k zjištění obsahu konkrétní složky. Tuto složku dostupných vrstev daného typu dostane model opět prostřednictvím aktivity, jenž vyvolala daný dialog. Složka s rastrovými či vektorovými daty je součástí prostorových dat, jenž uživatel vybral při spuštění aplikace. Celkově lze říci, že přidání a správa vrstev je soustava seznamů a příslušných modelů. Pokud je rozbalovací seznam naplněn dostupnými vrstvami, může uživatel některou z nich zvolit. Poté, co je zvolena některá z vrstev, lze potvrdit stisknutím tlačítka *OK* na pravém okraji patičky, nebo naopak zrušit stisknutím tlačítka *Cancel*, které se nachází také v patičce. Zrušení přidání vrstvy zavře dialog a neprovede žádnou další akci. Potvrzení přidání vrstvy provede řadu příkazů. Nejdříve se provede zjištění toho, zda již daná vrstva není v seznamu obsaže-

na. Pokud již vrstva existuje je vyvolán *MessageDialog* s informací o tom, že není možné tuto vrstvu přidat (Obr. 22).



Obr. 22 Hlášení o tom, že nově přidávaná vrstva již existuje

Pokud vrstva v seznamu přidaných vrstev není, jsou nejdříve do WinGRASS na pozadí zaslány pomocí metody *writeData* vlastního objektu *appLauncher* příkazy pro nastavení jména souboru po exportování zvolené mapy a také pro získání výstupu dané mapy v podobě PNG souboru. K získání výstupu rastrové a vektorové mapy je využíván modul *d.rast* respektive *d.vect* systému WinGRASS. Následně je dialog pro přidání vrstev zavřen a je spuštěn ukazatel postupu (objekt *pBar* typu *ProgressBar*), zavolána metoda pro přidání vrstvy do modelu (*append()*) a seznamu vrstev a nastavena cesta k adresáři, do kterého jsou jednotlivé mapové výstupy přesunuty, aby mohly být následně zobrazeny (Obr. 23). K přesunutí souborů do tohoto adresáře dochází poté, co doběhne ukazatel progresu.

```
//adding layer to layer model (layer manager)
function addElement(){
    layerModel.append({
                        "name": layerCB.editText + "@" + addLayerDialog.layer,
                        "type": addLayerDialog.layer,
                        "opacity": 100,
                        "command": "d." + addLayerDialog.layer + " map=" + layerCB.editText
                    })
}
```

```
// Does not exist layer yet?
if(!layerModel.containLayer(fileName)){
    //send commands to GRASS for creation PNG files with coresponding name
    appLauncher.writeData("set GRASS_PNGFILE=" + fileName + ".png" + "\n")
    appLauncher.writeData("d." + addLayerDialog.layer + " map=" + layerCB.editText + "\n")
            :
    //add layer to model (layer manager)
    addLayerItem.addElement()
            :
```

Obr. 23 Získání mapového výstupu nové vrstvy z GRASS a přidání do seznamu

- **Přidání WMS vrstvy**

Princip přidání WMS vrstvy je velmi podobný předešlým dvěma typům. Pouze je ve WinGRASS využit modul *r.in.wms*. Alternativou může také být využití no-

vější verze v podobě zásuvného modulu *r.in.wms2*. Postup, jak nastavit nebo získat tyto moduly byl popsán v podkapitole 6.5.1.

Pro přidání této vrstvy tedy slouží opět dialog, který je zobrazen vyvoláním akce pro přidání WMS vrstvy. Dialog obsahuje pole pro adresu na požadovaný server, na který je zaslán požadavek *GetCapabilities* pro získání dostupných vrstev. Pokud je služba dostupná a jsou vrácena nějaká data, jsou vrácena v XML souboru umístěném ve složce *.tmp* adresáře zvolené sady geodat nebo v příkazové řádce. Tyto dostupné vrstvy jsou v dialogu zobrazeny opět jako seznam a zvolením a následným potvrzením je vrstva, stejně jako předešlé typy, přidána do seznamu, jsou zaslány příkazy na získání výstupu v podobě PNG souboru a dialog je zavřen. A stejně jako u předešlých typů vrstev dochází také k ověřování, zda vrstva již není v seznamu obsažena.

- **Manipulace s položkami ve správci vrstev**

V rámci správce vrstev jsou pro každou položku seznamu definovány funkce pro manipulaci s nimi. Především je zde implementována celá řada gest, jejichž princip a realizace bude detailněji popsána v dalších částech této práce.

Každá přidaná vrstva je reprezentována položkou v seznamu vrstev a kromě názvu vrstvy obsahuje tato položka také checkbox pro nastavení viditelnosti příslušného PNG souboru mezi výstupy, tlačítka pro posouvání jednotlivých položek napříč celým seznamem, ikonu dle typu vrstvy a barevné pozadí položek opět odlišené dle typu vrstvy (Obr. 24) a v neposlední řadě objekt mající *ID properties*, který slouží k zobrazení vlastností vrstvy a který je v počátečním stavu nastaven jako skrytý a je zobrazen až po provedení určité akce.

Obr. 24 Podoba položek v seznamu vrstev

Checkbox pro nastavení viditelnosti příslušné vrstvy slouží ke skrývání či zobrazování vyexportovaného souboru příslušícího dané vrstvě. Po odznačení zvolené vrstvy je nastavena hodnota opacity, která je také uchovávána v modelu vrstev, na 0. Současně je také podle názvu vrstvy zjištěn index položky v modelu a na základě tohoto indexu je pak pomocí metody *setOpacityToMap(index)* modelu vrstev nastavena hodnota také korespondujícímu mapovému výstupu.

Šipky u každé položky slouží k posouvání vrstev v rámci celého seznamu. Po každém kliknutí je snížen či navýšen index dané položky. Následně je zavolána metoda modelu vrstev *reorderDispLayers()*, která změny promítne také na mapové výstupy. Místo zobrazených šipek je přemístění vrstev napříč se-

znamem možné praktikovat pomocí *drag & drop* gesta. Implementace je aplikována na pozadí dané položky a uvedena v *DelegateSwipe.qml*. Základní princip je takový, že pro přemístění vrstvy v seznamu je nutné provést dlouhý stisk, při kterém je zaznamenána počáteční *y* souřadnice dané položky. Poté je položka přenesena do popředí a částečně zprůhledněna, což indikuje aktivaci přesunu této položky. Po přesunutí položky v seznamu a ukončení gesta dojde na základě nové hodnoty souřadnice *y*, počáteční pozice a výšky dané položky k výpočtu nového indexu dané položky v seznamu. Poté pomocí zmiňované metody je změna pořadí vrstev aplikována také na příslušné mapové výstupy. Uvedené gesto není jediným, které je aplikováno na *delegateItem*, tedy na každou položku seznamu vrstev.

Zmiňované vlastnosti vrstvy je možné zobrazit kliknutím na danou položku. Kliknutí se změní stav příslušného objektu a jsou zobrazeny možnosti, které vrstva nabízí. Návrat na seznam vrstev je možný kliknutím na hlavičku, respektive název vrstvy v jejím detailu. Tento detail vrstvy (Obr. 42 v příloze) je umístěn ve *Flickable* objektu, který zajišťuje možnost posouvání obsahu okna, pokud je tento obsah větší, než je objekt, ve kterém je umístěn (indikováno malou ikonou šipky v pravém dolním a horním rohu). Detail rastrové vrstvy oproti vektorové je velmi jednoduchý a obsahuje pouze nastavení úrovně průhlednosti vrstvy. Oproti tomu vektorová vrstva nabízí celou řadu možností. Od nastavení úrovně průhlednosti, přes nastavení prvků, které mají být zobrazeny, barev, tloušťku linií, až po nastavení symbolů. V rámci implementace vlastností vektorové vrstvy je použita celá řada objektů a *QML* typů, které již povětšinou byly v této práci zmíněny. Jednotlivé uvedené možnosti jsou seskupovány do logických celků pomocí typu *GroupBox* a rozvrženy pomocí sloupcových či řádkových objektů *ColumnLayout* a *RowLayout*. Jsou zde využiti klasické kontrolní prvky jako checkboxy, rolovací seznamy (k jejich naplnění využit stejný princip jak pro přidání vrstvy), posuvníky či spinboxy.

```
function hexToRgb(hex) {
    var bigint, r, g, b, a;
    //Remove # character
    var re = /^#?/;
    var aRgb = hex.replace(re, '');
    bigint = parseInt(aRgb, 16);

    //If in #RRGGBB format
    if (aRgb.length >= 6) {
        r = (bigint >> 16) & 255;
        g = (bigint >> 8) & 255;
        b = bigint & 255;
        var rgb = r + ":" + g + ":" + b;
    }
    return rgb;
}
```

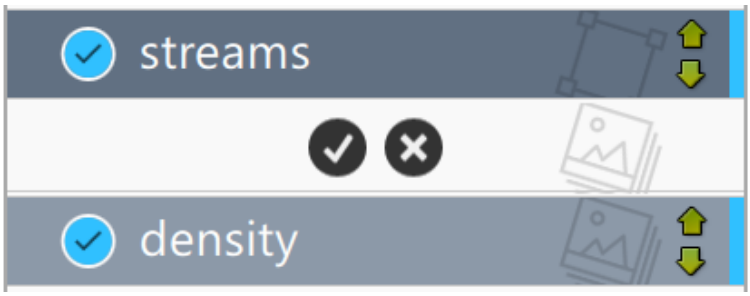
Obr. 25 Převodník vyjádření barvy z hexadecimální na RGB

Pro nastavení barev je použit opět *ColorDialog*. Aby bylo možné aplikovat změny barvy, bylo nutné převést hexadecimální vyjádření barvy, získané

z onoho dialogu, na RGB vyjádření. K tomuto účelu je využita funkce *JavaScriptu* (*converter.js*), kterou je pro její použití nutné naimportovat a přidat jí libovolný alias. Implementace této funkce je uvedena níže (Obr. 25).

V rámci vlastností vektorové vrstvy a možnostech provedení změn v jejich nastavení je pro větší uživatelské přívětivosti aplikována celá řada ošetření a omezení špatných či nesmyslných vstupů. Také jsou zachycovány změny jednotlivých položek. Pokud byly provedeny nějaké změny v nastavení vrstvy, je aktivováno tlačítko pro aplikování těchto změn, které je umístěno v dolní části zobrazeného detailu vrstvy. V případě navrácení změn na původní hodnoty, je toto tlačítko opět deaktivováno. Po stisknutí tohoto tlačítka je opět provedena posloupnost příkazů, která je velmi podobná těm pro přidání vrstvy. Je však doplněna o kroky, při kterých dojde ke smazání původního mapového PNG souboru a následném nahrazení souborem novým s aplikovanými změnami.

Jednotlivé vrstvy lze také samozřejmě odstranit. K tomuto účelu je definováno další gesto, a to *swipe*. Opět bude implementace popsána až v dalších částech této práce, ale funkčnost v rámci správce vrstev uvedu zde. Pro smazání vrstvy je nutné provést posun položky zprava doleva. Po provedení tohoto úkonu, je uživateli předložen dotaz na potvrzení smazání nebo naopak zrušení kroku pro smazání (Obr. 26). Pokud uživatel smazání potvrdí, je položka smazána ze seznamu vrstev a je smazán také příslušný výstupní soubor. Pro smazání všech vrstev v sezamu lze využít také příslušné tlačítko k tomu určené a umístěné v patičce správce vrstev. Po jeho stisknutí dojde k vyčištění seznamu pomocí metody *clear()* a smazání celého obsahu uživatelské složky s mapovými výstupy. Odstranění a přidání položek do seznamu vrstev je pro plynulost a lepší vizuální dojem doplněno o animace, jenž jsou nastaveny na příslušné sloty *onAdd* a *onRemove*.



Obr. 26 Potvrzení smazání vrstvy ze seznamu

- **Zobrazení mapových výstupů**

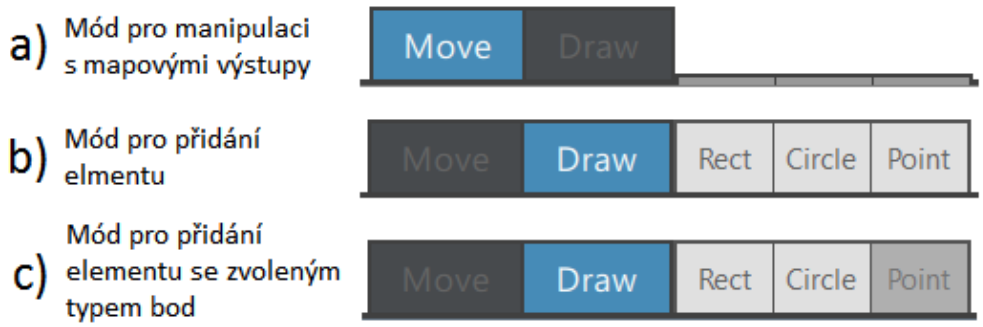
  Pro umístění a zobrazení veškerých mapových výstupů je definován objekt *mapFrame* typu *Rectangle*, který funguje na principu rámu, do kterého jsou umístěny získané mapové obrazové soubory. Tento rám je po spuštění aplikace vycentrován v rámci hlavního okna aplikace a je mu nastaven atribut *scale* určující stupeň přiblížení. Veškeré vyexportované mapové výstupy

z WinGRASS jsou po přidání do *layerModelu* uloženy v příslušné uživatelské složce. K tomu, aby mohl být obsah této složky načten a zobrazen je v první řadě využít již zmiňovaný objekt typu *FolderListModel*, který má tentokrát mezi svými atributy nastaveno, že má zahrnovat pouze soubory a tyto soubory navíc ještě filtrovat na soubory s příponou PNG. Tento model je pak následně předložen opět objektu typu *Repeater*, který podle položek v tomto modelu, zobrazí jednotlivé položky (PNG soubory) na obrazovce. Toto zobrazení lze realizovat hned několika způsoby. Jedním z nich je možnost vytvářet pro každý soubor modelu dynamický objekt nejpravděpodobněji typu *Image*, dalším způsobem je využití kontejneru, což ve výsledku není nic jiného než seznam obsahující opět jednotlivé obrazy. Dále je možné využití třídy *QDeclarativeImageProvider* knihovny Qt, pro kterou by bylo nutné vytvořit implementaci v jazyce *C++* a registrovat ji jako nový typ v *QML*. Nakonec byla použita varianta, kdy je v rámci uvedeného *Repeateru* vytvořen objekt typu *Image* vzhledem k tomu, že je jasné, že všechny mapové výstupy budou umístěny do zmiňovaného *mapFrame* rámu s nastavením příslušných atributů, které jsou známé a společné pro všechny tyto mapy. Tento princip velmi připomíná vytváření dynamických objektů, ale není nutné využívat funkcí *JavaScriptu*, jak je tomu u této metody a jak bude popsáno níže.

Pro manipulaci se zobrazenými výstupy je u *mapFrame* deklarován potomek typu *PinchArea* umožňující odchyt *pinch* gest sloužící pro rotaci a změnu měřítka mapy. Následně je deklarován také *MouseArea* objekt pro posun mapy pomocí *drag & drop* gest nebo pomocí myši. Slot *onWheel* také umožňuje změnu měřítka mapy při použití kolečka myši a její rotaci při současném stisknutí tlačítka *Ctrl*.

6.5.8 Správa mapových elementů

Pomocí módu *Draw* je možné také přidávat jednoduché mapové elementy. K dispozici je bod, kruhová a čtvercová výseč. Možnost přidání prvku je uživateli k dispozici poté, co se na hlavní obrazovce přepne z módu *Move*, sloužícího k manipulaci s mapovými výstupy, právě do módu *Draw* pro kreslení těchto mapových prvků. K přepínání mezi těmito 2 módy je použit typ *Switcher*, který pouze zapíná a vypíná (nastavuje hodnotu atributu *enabled*) zmiňovaný objekt *dragArea* typu *MouseAresa* umožňující manipulaci s mapovými výstupy. Na základě toho, zda je tento objekt zapnutý či nikoliv, je pak také vypnut nebo naopak zapnut objekt stejného *QML* typu umožňujícího přidávání mapových prvků. Tento typ je ukotven tak, že vyplňuje vlastní definovaný typ *PaintArea*, do kterého se vykreslují jednotlivé elementy a který je zároveň nastaven tak, aby pokryl rám nesoucí jednotlivé mapové výstupy. To umožňuje přidat elementy jen uvnitř mapy.



Obr. 27 Přepínač módů pro manipulaci s mapovými výstupy a přidání mapových prvků

Pomocí slotů *onPressed*, *onPositionChanged* a *onReleased*, několikrát zmiňovaného typu *MouseArea* nastaveného na objekt *PaintArea*, pak lze prvek vložit do mapy. Princip je velmi podobný chování pro *drag & drop* gesta. Po zvolení pozice nového prvku stačí kliknutím umístit prvek (je zaznamenána pozice *x* a *y* kliknutí). Pokud se nejedná o bod, uživatel tahem určí jeho počáteční velikost a poté, co slot *onReleased* obdrží příslušný signál, je prvek o nastavené velikosti vytvořen a umístěn.

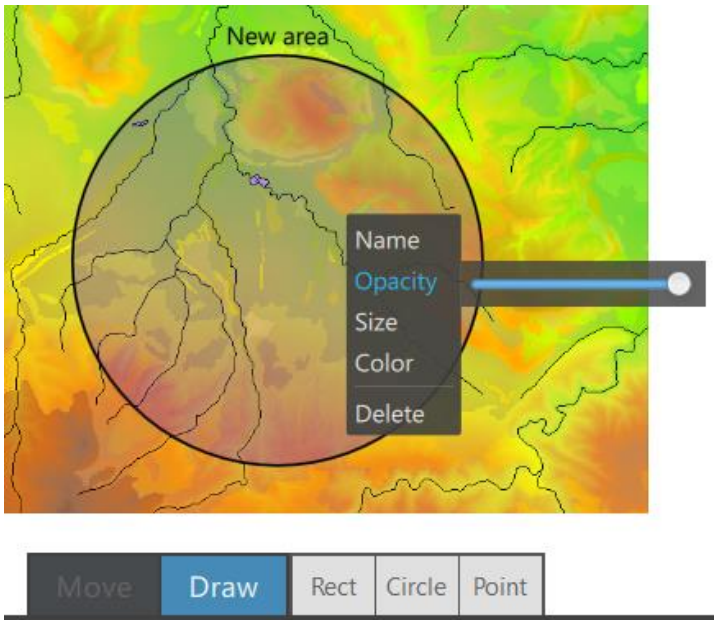
Vytváření dynamických *QML* objektů se provádí pomocí metod *JavaScriptu*. Jednou z takových je *Qt.createQmlObject(QML string, parent ID, filename).* Prvním parametrem je řetězec formátovaný jako *QML* soubor, který představuje definici daného objektu. Druhým parametrem je *ID* předka, tedy objektu, ve kterém bude nový dynamický objekt umístěn. Posledním parametrem pak je řetězec udávající název souboru pro report případných chyb. Problémem u dynamicky vytvářených objektů je skutečnost, že není možné k nim přistupovat přes jejich *ID*. K tomu, aby bylo možné k těmto objektům přistoupit, byla u předka deklarována vlastnost *child* typu *QtObject*. Níže uvedený Obr. 28 udává ukázku využití tvorby dynamických objektů v aplikaci. Odstranění těchto objektů se pak provádí pomocí metody *destroy()*. V aplikaci jeho použití vypadá následovně*: paintarea.child.destroy()*.

```
var dynObject = null
held = true
if (addElementBT.pressed == "Rect"){
    dynObject = Qt.createQmlObject(
            'import QtQuick 2.2;
            import QtQuick.Controls 1.2;
            Rectangle{
                id:rect' + parent.paintCount + ';
                objectName: "rect' + parent.paintCount + '";
                width:10;
                height:width;
                x:' + Math.round(mouseX) + ';
                y:' + Math.round(mouseY) + ';
                color:"lightblue";
                border.color: "black";
                border.width: 2;
                opacity: 0.5;
                property int defWidth: 0;
                property string defObjName: "rect' + parent.paintCount + '";
                    Text{
                        anchors.horizontalCenter:parent.horizontalCenter;
                        anchors.bottom:parent.top;
                        anchors.bottomMargin:2;
                        text:parent.objectName;
                        font.family:"Segoe UI";
                        font.pointSize:14;
                        enabled:parent.defObjName == parent.objectName ? false:true;
                        visible:enabled;
                    }
            }', parent, 'newDynamicItem')
}
```

Obr. 28 Příklad tvorby dynamických objektů

Editovat jednotlivé vytvořené dynamické objekty (mapové elementy) je možné pouze v módu *Draw* bez vybraného typu elementu pro přidání (viz Obr. 29). V první řadě je možné kliknutím na libovolný element přenést tento prvek do popředí. Po vybrání tohoto prvku je nastavena jeho souřadnice *z* na nejvyšší hodnotu v rámci vytvořených dynamických objektů. Přemístění vytvořených elementů je pak možné pomocí *drag & drop* gesta. Stačí kliknout na libovolný element v mapě, prvek se částečně zprůhlední, což indikuje jeho výběr, a následným tahem je možné jej v rámci mapy přemístit. V dřívější verzi bylo přemístění elementu možné až po provedení dlouhého stisku. Tato varianta však byla po testování z důvodu příjemnějšího použití nahrazena klasickým *drag & drop.* Na některých obrazovkách totiž dlouhý stisk nefungoval tak, jak bylo zamýšleno při implementaci v aplikaci. Dvojklikem na zvolený prvek se uživateli v místě tohoto kliknutí zobrazí nabídka atributů a vlastností, které jsou k dispozici pro daný prvek. Jsou to jméno, průhlednost, velikost, barva a možnost smazání označeného prvku. Implementace těchto vlastností prvku je obsahem souboru *ElementAtributes.qml*. Jednotlivé položky jsou umístěny ve sloupci, který je umístěn na obdélníkovém pozadí. Po kliknutí na některou z položek se vpravo od dané položky zobrazí prvek pro nastavení její nové hodnoty. Konkrétně textové pole pro nastavení jména a posuvník pro průhlednost a velikost. Po zvolení barvy se zobrazí příslušný dialog. Pro nastavení průhledné výplně obsahuje dialog možnost nastavení hodnoty alfa kanálu. Po kliknutí na

možnost smazání, dojde k odstranění dynamického objektu, jak bylo uvedené výše. Text vybrané položky je také pro přehlednost zvýrazněn.



Obr. 29 Detail a atributy mapového prvku při zapnutém módu Draw

6.5.9 Navigace v aplikaci

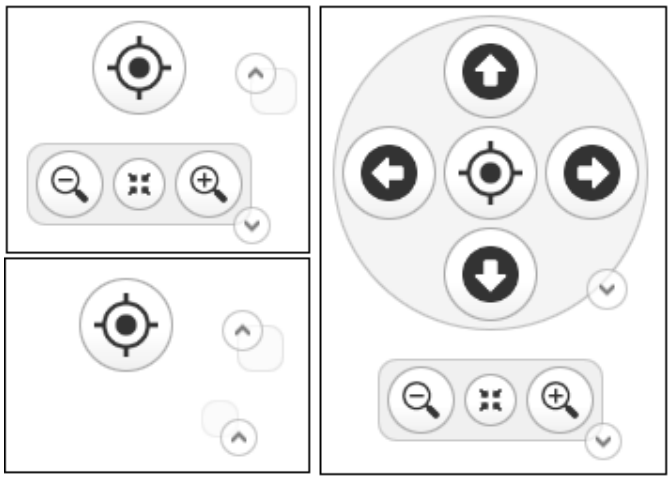
Pro snadnější manipulaci a vyšší uživatelskou přívětivost jsou v aplikaci využity různé způsoby navigace či manipulace s některými prvky.

- Veškeré aktivity v rámci celé aplikace je možné provádět myší, případně u dotykových obrazovek přímo prsty pomocí gest. Jsou implementována standardní gesta jako *swipe*, *drag & drop* či *pinch*, kterým bude věnován prostor v následující části textu. První dva typy gest lze aplikovat i pomocí myši kliknutím a tahem. Tímto způsobem zle posouvat mapou a přemísťovat některé další prvky (vrstvy ve správci vrstev, přemístění správce vrstev v okně aplikace). Přiblížení a rotace mapy se kromě *pinch* gest dá provést samotným kolečkem myši, respektive držením tlačítka Ctrl a rolováním kolečka myši.
- Druhou možností manipulace s mapou či některými dalšími objekty je klávesnice. Pro posouvání mapy jsou definovány a lze tedy využívat kurzorové klávesy. Pro změnu měřítka mapy pak fungují tlačítka „+“ a „-“. Dále jsou definovány také další klávesové zkratky jako:
    - Ctrl + v (přidat vektorovou vrstvu)
    - Ctrl + r (přidat rastrovou vrstvu)
    - Ctrl + w (přidat WMS rastrovou vrstvu)
    - Ctrl + n (nový projekt)
    - Ctrl + o (otevřít soubor)

- Ctrl + s (uložit projekt)
- Ctrl + Shift + s (uložit jako)
- Ctrl + q (ukončit aplikace)

Definování klávesových zkratek je v *QML* velmi jednoduché a provádí se prostřednictvím atributu *shortcut* pro příslušnou akci. Lze přiradit hodnotu v podobě konkrétní kombinace kláves, nebo lze využít již implementované *StandardKey* ze třídy *QKeySequence*.

- Poslední variantou, jak manipulovat se zobrazenými výsledky je grafická navigace (Obr. 30) umístěná v pravém dolním rohu hlavního okna aplikace. Tato navigace, kromě klasických tlačítek pro pohyb do všech směrů a tlačítek pro zvětšení či zmenšení, obsahuje také tlačítka pro vycentrování mapy a nastavení počátečního měřítka. Jednotlivé skupiny tlačítek pro pohyb a změnu měřítka lze případně také skrýt, s výjimkou tlačítka pro vycentrování mapy, stisknutím malého tlačítka v pravém dolním rohu každé této skupiny.



Obr. 30 Navigační tlačítka a možnosti jejich zobrazení

6.5.10 Animace, přechody a styly

Pro větší uživatelskou přívětivost jsou napříč aplikací využívány různé pokročilé vizuální prvky. *QML* disponuje propracovaným systémem animací a přechodů, které tímto umožňují dosáhnout tzv. plovoucího GUI, ve kterém prvky neskáčou, okamžitě nemizí a zase se neobjevují (návrhový vzor *Animated transition*). Uživatel tak má větší přehled o tom, kde se dané prvky při provedených změnách nachází.

- **Animace a přechody**
  Je možné využít různých způsobů animací. Od animací, které jsou vyvolány změnou hodnoty atributu, až po komplexní soustavy sekvenčně a paralelně probíhajících animací.

  Základní animace (Obr. 31) vyvolaná změnou parametru je v aplikaci využita například při otáčení správce vrstev, posunu zvýraznění vybrané polož-

ky v seznamu dostupných *Location* a *Mapset* spouštěcího dialogu aplikace, nebo u tlačítek v záhlaví správce vrstev po najetí kursoru. Využívá se k tomu element *Behavior*, který stačí připojit pomocí *on* k danému parametru a definovat si příslušnou animaci. Základní věcí je však to, že je známa počáteční a koncová hodnota daného parametru a animace se jen postarají o plynulý přechod. Ukázka, jak lze definovat základní animaci:

```
Behavior on y {
    SpringAnimation {
        spring: 2
        damping: 0.2
    }
}
```

```
Behavior on rotation{
    NumberAnimation{
        duration: 250
        easing.type: Easing.InOutQuad
    }
}
```

Obr. 31 Ukázka nastavení základních animací

Podobný princip jsem použil také při mazání vrstvy ze správce vrstev pomocí *swipe* gesta. Zde je však animace spouštěna příkazem po dokončení gesta. V tomto případě není použit element *Behavior*, ale je zde ručně nastaven atribut *running* na *true*, který udržuje informaci o stavu animace.

Složitější animace jsou obvykle spojeny s přechody mezi stavy (Obr. 32) daného objektu. Typicky se tomu tak děje ve správci vrstev, kdy po kliknutí na položku v seznamu se místo seznamu vrstev zobrazí vlastnosti vybrané vrstvy. Na základě kliknutí se provede přechod z jednoho stavu položky, kdy je zobrazen název vrstvy, do druhého, kdy jsou zobrazeny také objekty reprezentující vlastnosti dané vrstvy a provedeno několik změn barvy, pozice atd. Je tedy potřeba nejdříve definovat nový stav. V rámci tohoto stavu určit jednotlivé změny, které se mají provést a na závěr definovat také přechody, které dané změny provedou plynule.

```
//Properties of some layer is displayed
states: State {
    name: "Properties"

    // change color and z coor. of layer name
    PropertyChanges {
        target: layerName
        color: "#404040"
        y: 8
    }

    // change layer properties background color
    PropertyChanges { ...}

// Make properties visible and Fill the entire list area
    PropertyChanges {
        target: delegateItem
        propertiesOpacity: 1
        height: layerList.height
    }
```

```
transitions: Transition {
    // Make the state changes smooth
    ParallelAnimation {
        ColorAnimation {
            duration: 500
            property: "color"
        }
        NumberAnimation {
            duration: 250
            properties: "propertiesOpacity,contentY,height, y"
        }
    }
}
```

Obr. 32 Ukázka animované změny stavu prvku

- **Styly**

Velmi dobrým pomocníkem pro udržení stejného vizuálního stylu napříč celou aplikací je možnost vytvářet grafické styly pro jednotlivé prvky rozhraní.

Nejedná se o nic jiného, než komponentu skládající se z jednotlivých dostupných či vytvořených *QML* typů a s nadefinovaných chováním. Obrovskou výhodou je však to, že tuto komponentu pak lze snadno přiřadit libovolným prvkům, jenž disponují atributem style. Tento atribut obsahují především prvky zachycující určité vstupní události, jako jsou tlačítka, radio buttony, checkboxy, posuvníky a podobně.

V práci jsem tyto styly využil téměř pro všechny prvky, které se v aplikaci vyskytují vícekrát. Pro některé byly vytvořeny styl vlastní, pro některé byly aplikovány styly před vytvořené, které jsou součástí třídy *Controls* modulu *QtQUick*.

6.5.11 Podpora gest v aplikaci

Cílem práce bylo vytvořit aplikaci, která bude disponovat uživatelským rozhraním pro dotykové obrazovky a tím pádem určitou množinou gest, které zle pro manipulaci s aplikací využívat. V rámci aplikace je definována funkcionalita pro základní množinu gest. Kromě klasického klepnutí pro aktivaci či výběr prvku jsou v aplikaci implementována gesta jako je *drag & drop*, *swipe*, či *pinch* gesta.

- **Pinch gesta**
  V *QML* je pro tato gesta přímo určen typ *PinchArea*. Jedná se o neviditelný prvek, který je nutné svázat s jiným konkrétním typem, u kterého chceme tato gesta zachytávat. Možnosti, jak tento typ používat, jsou dvě. První možností je využít manipulátory resp. sloty definovaných signálů *onPinchStarted*, *onPinchUpdated* and *onPinchFinished* a druhou je určit cíl (*target*), u kterého pak bude probíhat manipulace automaticky, popřípadě podle vlastností, které lze tomuto typu nadefinovat (např. osy pohybu, maximální a minimální rotace,…).

  Ve výsledné aplikaci jsem využil druhý způsob. Konkrétně je aplikován jako jedna z možností manipulace s mapami. Typ *PinchArea* je ukotven k rámu (*mapFrame*), který v sobě nese všechny načtené mapy a který je zároveň určen jako cíl, tedy objekt, pro který je zachytáván tento typ gest a se kterým lze jejich prostřednictvím manipulovat. Byla nastavena pouze hodnota minimálního a maximálního zvětšení na hodnoty 0.1 respektive 10.

- **Drag & drop gesta**

  Tento typ gest, stejně jako předcházející, využívá určitých vlastností daného typu pro nastavení cílového prvku, který bude zachytávat daná gesta. Pro *drag & drop* se jedná o *QML* typ *MouseArea*, jenž disponuje onou vlastností *target*.

  Gesta tohoto typu jsou ve výsledné aplikaci využívány na více místech. Konkrétně jsou to například posun mapy napříč hlavním oknem, přesouvání vrstvy ve správci vrstev, přemístění správce vrstev, změna velikosti správce vrstev a posun vytvořeného mapového prvku.

  Základní princip je velmi jednoduchý. Stačí nastavit pouze onen cílový prvek. Vhodné je nastavit také osy, na kterých je možné se v rámci gesta pohybovat. Toto základní nastavení je aplikováno například na změnu velikosti správce vrstev, které je k vidění níže.

```
562  drag.target: cHandle
563  drag.axis: Drag.XAndYAxis
564  cursorShape: enabled ? Qt.SizeFDiagCursor : Qt.ArrowCursor
565  enabled: ( parent.rotation == 0 || parent.mod(parent.rotation,360) == 0 ) ? true : false
566  visible: enabled
567
568  onPressed: cHandleRect.opacity = 0.75
569  onReleased: cHandleRect.opacity = 1
570  onPositionChanged: {
571      var mouseToPlaceholder = mapToItem(parent, mouseX, mouseY)
572      var newWidth = mouseToPlaceholder.x
573      var newHeight = mouseToPlaceholder.y
574
575      // minimum width and height
576      if(newWidth >= 345 && newHeight >= 300){
577          parent.width = newWidth
578          parent.height = newHeight
579      }
580  }
```

Obr. 33 Ukázka drag & drop pro změnu velikosti správce vrstev

Pokud je požadavek ne speciální chování, je nutné definovat toto chování prostřednictvím slotů *onPressed*, *onPositionChanged* a *onReleased* (Obr. 33). Tento typ nastavení je ve výsledné aplikaci využíván ve dvou provedeních. První využívá pouze uvedených slotů a s objektem lze kdykoliv snadno manipulovat jeho přetažením. Tento způsob je použit například pro posun mapy, přemístění správce vrstev a nakresleného či přemístění elementu. Druhý z nich je, že s objektem zle hýbat až po provedení dlouhého stisku (*onPressAndHold*), který je využíván pro přesun vrstvy ve správci vrstev. Dlouhý stisk je v *QML* nastaven na interval 800 ms.

Pro lepší rozlišovací schopnost je využito například také natavení průhlednosti pro objekt, se kterým je momentálně manipulováno, nebo také změna ikony kurzoru.

- **Swipe gesta**

Tento typ gest nemá v jazyce *QML* implementován žádný speciální typ, jako je tomu u *pinch* gest. Je tedy nutné generovat manipulace tohoto typu jinak. Lze vymyslet celou řadu implementací, ale základní myšlenky budou pravděpodobně vycházet opět ze dvou způsobů, které budou z velké části vždy založeny na předešlých typech gest.

Prvním způsob lze definovat jako rychlý pohyb (tzv. *flick*) nad objektem či pohledem. Tento princip je vhodný například tehdy, pokud máme nějaký objekt, se kterým chceme hýbat v rámci menší plochy, kterou představuje jeho předek. Typicky, pokud máme například velký obrázek, kde jeho předek je rámeček. Předek pak zobrazuje pouze výřez tohoto obrázku. Jedná se v podstatě o zmiňovaný princip předešlého typu gest, ale lze jej využít také pro *swipe*, jelikož *QML* disponuje typem *Flickable*, který obsahuje řadu parametrů, jak lze docílit požadovaného výsledku. Nicméně účely, pro které je tento typ gest využíván ve výsledné aplikaci, tzn. pro mazání jednotlivých vrstev ze správce, není tato metoda příliš vhodná. Není s ní možné pohodlně měnit vlastnosti jednotlivých objektů.

Proto jsem využil opět možností typu *MouseArea,* konkrétně jeho vlastnosti *drag*, a nadefinoval si chování vlastní. Toto je zároveň druhá z možností, jak si nadefinovat vlastní *swipe* gesta. Hlavní myšlenka této implementace mazání vrstev při pohybu zprava doleva je taková, že jednotlivé položky seznamu, konkrétně jejich obdélníkové pozadí, je ukotveno k levému okraji. Všechny prvky v rámci jedné položky v seznamu vrstev, jako je *checkbox* pro viditelnost, název vrstvy a podobně, jsou pak ukotveny k pravému okraji tohoto pozadí. Prostřednictvím zmíněného typu a jeho *drag* vlastnosti je nastavena osa pohybu pouze na x a při prvním kontaktu (*onPressed*) je zjištěna a uložena souřadnice tohoto bodu. Následně při tahu prstem (*onPositionChanged*) dochází ke změně pozice současného bodu kontaktu a k porovnávání této souřadnice s počáteční hodnotou. Pokud je aktuální souřadnice menší než původní, dochází k pochybu pravého okraje pozadí směrem doleva. Přesněji řečeno je zmenšována jeho šířka a vzhledem k ukotvení prvků jsou ve stejném pohybu i všechny prvky na tomto pozadí. Délka tahu určuje, zda se šířka položky zmenší až na 0, tzn., přimkne k levému okraji správce vrstev, nebo se vrátí do původní šířky. Zlomová hodnota je nastavena na přibližně polovinu šířky celého správce. Pro plynulé přechody a příjemnější vizuální dojem byly použity také některé animace, které se spouští po přerušení gesta (*onReleased*). Kompletní implementace se nachází v souboru *DelegateSwipe.qml*.

6.5.12 Volba písma, ikon a paleta barev

Pro snadnější orientaci v aplikaci a lepší celkový estetický dojem je v rámci celé aplikace zachováván jednotný vizuální styl.

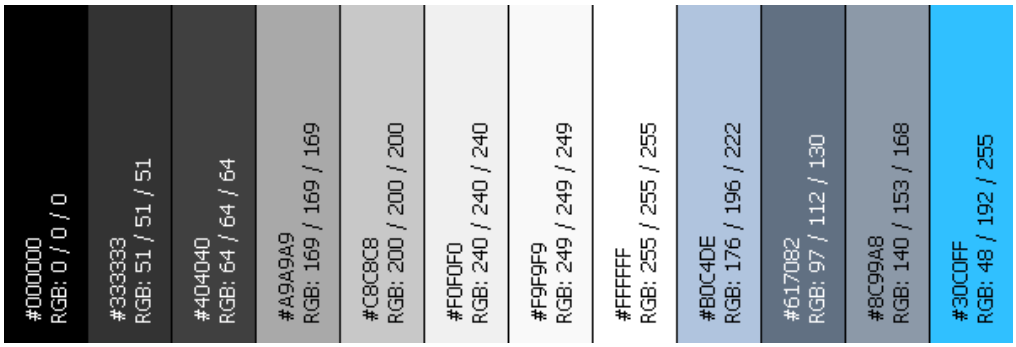
- **Písmo**
  Jako písmo využívané v celé aplikaci byla zvolena rodina *Segoe UI*. Toto písmo je v poslední době velmi známé díky využití v produktech společnosti *Microsoft*. Vyznačuje se zejména typografickou jednoduchostí a dobrou čitelností. Konkrétní řezy, stupně písma a jejich použití v aplikaci:
  - *Segoe UI*, 11 bodů – většina textu v aplikaci. Popisky jednotlivých položek, skupin, tlačítek a podobně.
  - *Segoe UI*, 9 bodů – krátké texty jako položky v seznamech či popisky položek v rámci skupin.
  - *Segoe UI*, 20 bodů – prvky, které bylo potřeba zvýraznit. Název vrstvy ve správci vrstev, titulek správce vrstev a podobně.
  - *Segoe UI Light*, 20 bodů – titulek dialogu přidáni vrstvy
  - *Segoe UI Semibold*, 11 bodů – titulky seznamů s dostupnými *Mapsets* a *Locations*

  Pro lepší rozlišovací schopnost je využita také různá míra průhlednosti písma.
  - Průhlednost 0 % – primární texty a titulky.

- Průhlednost 20 % – sekundární texty, položky měnící vlastnosti při najetí kursoru.
- Průhlednost 40 % – terciární texty.

- **Paleta barev**

  QML umožňuje pro udržení stejného grafického stylu využít například typu *SystemPalette*. Tento typ poskytuje informace o standardních barvách a lze díky němu snadno nastavit různé odstíny barvy pro aktivní, neaktivní či vypnuté prvky. Mým účelem však bylo definovat určitou množinu složenou z více barev a ne tedy pouze z různých odstínů jedné barvy. Proto jsem se rozhodl si takovou paletu vytvořit mimo *QML* a nastavovat jednotlivým prvkům konkrétní barvu v jejím hexadecimálním tvaru. Výsledná použitá paleta barev vypadá následovně:



Obr. 34 Výsledná paleta použitých barev[6]

- **Ikony**

  Veškeré použité ikony byly staženy zdarma z volně dostupných otevřených databází ikon pro webové a mobilní aplikace. Ikony byly vybírány především tak, aby zapadaly do celkového grafického konceptu aplikace a tvořili jednotný celek. Byly tady vybírány zejména takové ikony, které pokud možno pochází ze společné sady. Veškeré ikony pochází z těchto tří zdrojů:

  - www.iconfinder.com
  - www.icons8.com
  - www.flaticon.com

[6] Vytvořeno pomocí online nástroje *ColorExplorer* dostupného na webových stránkách http://www.colorexplorer.com/

7 Diskuze

Prostřednictvím této práce byly nastíněny možnosti knihovny Qt v oblasti tvorby moderních grafických uživatelských rozhraních určených primárně pro dotykové obrazovky. Výsledné řešení bylo navrhováno a realizováno tak, aby tyto možnosti prezentovalo na jednoduchém GIS. Práce potvrdila, že prostřednictvím modulu *Qt Quick* a jazyku *QML* lze vytvářet rozhraní, která splňují veškeré požadavky na moderní rozhraní. Díky tomu, že funkcionalitu implementovanou pomocí jazyka *QML* je možné doplnit o funkcionalitu implementovanou např. v jazyce *C++* nebo také o skripty v *JavaScriptu*, lze také vytvářet komplexní aplikace všeho druhu.

7.1 Hodnocení řešení

Výsledná aplikace realizovaná pomocí knihovny Qt a jazyka *QML*, splňuje požadavky, které byly na počátku vývoje stanoveny. Konečné řešení tedy není standardním desktopovým řešením, ale na základě provedených návrhů a také díky využití řady návrhových vzorů je přizpůsobeno užívání na moderních dotykových obrazovkách. Snadnější manipulaci umožnuje množina implementovaných gest a definovaných zkratek. Z tohoto hlediska lze tedy považovat navržené a následně realizované grafické uživatelské rozhraní za uživatelsky přívětivější, než tomu je u současných desktopových GIS uvedených ve 2. kapitole. Jako o něco lepší ukazatel této skutečnosti byl proveden jednoduchý uživatelský test.

7.1.1 Uživatelský test

Pro závěrečné hodnocení přívětivosti implementovaného řešení bylo využito také jednoduchého uživatelského testu. Tento test byl proveden čtyřmi vybranými osobami. Tři z těchto uživatelů již měli možnost s nějakým GIS pracovat, ať už ve škole či jinde, a jeden s podobnou aplikací nepřišel do kontaktu. Tento uživatel tímto sice není cílovým uživatelem, jak bylo uvedeno v kapitole 6.1, ale z důvodu zjištění přívětivosti řešení i pro neznalé uživatele, byl do testu zahrnut.

Test byl kromě výsledné aplikace realizován také na vybraných GIS uvedených v úvodní části této práce, konkrétně se jednalo o GRASS GIS, QGIS a OpenJUMP. Mobilní aplikace byly z testu vynechány, jelikož jejich funkcionalita jednak není stejná napříč uvedenými mobilními aplikacemi, ale také je mírně odlišná s porovnáním funkcionality výsledné aplikace. Úkolem uživatele tedy bylo provést sled určitých operací, přičemž byl měřen jeho čas, za který tyto operace zvládl. Operace, které měl za úkol uživatel vykonat:

1. Přidej alespoň 2 vrstvy libovolného typu.
2. Zobraz vlastnosti některé z vrstev.
3. Uprav nastavení některé z vrstev.
4. Změň pořadí vrstev.

5. Přidej polygon.
6. Proveď libovolnou změnu u tohoto polygonu.
7. Odstraň libovolnou vrstvu.

Test ukázal, že pro splnění těchto kroků je ve výsledné aplikaci potřeba nejméně času. Což lze považovat za ukazatel větší uživatelské přívětivosti, než je tomu u zbylých testovaných aplikací. Největší podíl na tomto výsledku má zřejmě i fakt, že v aplikaci není tolik nástrojů a nabídek, jako je tomu ve zbývajících aplikacích.

Ve všech testovaných aplikacích trvalo nejdéle přidání a upravení polygonu. Tyto kroky zabraly přibližně polovinu času plnění scénáře. Nejhorších výsledků bylo dosaženo v aplikaci *OpenJump*, kde největším problémem bylo najít vytvořené polygony, které se přidávaly pod všechny vrstvy a nebyly proto vidět. V aplikaci *QGIS* bylo největším problémem zorientování se v široké škále nástrojů zobrazených v hlavním okně aplikace. Zmiňovaný uživatel, který dříve neměl možnost pracovat s podobnou aplikací, dosahoval při plnění přibližně o třetinu až polovinu horších časů, než ostatní uživatelé, a to u všech aplikací. Výsledky testů pro jednotlivé aplikace a průměrných časů potřebných ke splnění stanovených úkolů jsou uvedeny v následující tabulce.

Tab. 3 Výsledky uživatelského testu

	GRASS GIS	**QGIS**	**OpenJUMP**	**Vlastní aplikace**
Průměrný čas	8,5 min	9 min	11 min	5,5 min

7.1.2 Shrnutí práce s aplikací

Na základě provedeného testu se také ukázalo, že by bylo vhodné vytvořit jakýsi stručný manuál, tedy souhrn možností, jak se s výslednou aplikací pracuje. V rámci předchozích kapitol byla po rozboru a vytvoření přehledu funkcionality, kterou by měla aplikace disponovat, provedena implementace. V části věnující se popisu implementace jednotlivých částí aplikace byly rozebrány také možnosti toho, jak danou funkcionalitu v aplikaci použít. Pro přehlednost zde tedy uvádím ještě stručný souhrn toho, jak lze s výslednou aplikací pracovat.

- **Správa projektu**: projekt je možné spravovat pomocí klávesových zkratek (viz kapitola 6.5.9), nebo pomocí nabídky, která se zobrazí po stisknutí tlačítka *File* umístěného trvale v levém dolním rohu hlavní obrazovky.
- **Správa mapových elementů**: pro veškerou činnost týkající se mapových elementů je nutné mít zapnutý mód *Draw*. Mód lze přepnout pomocí přepínače trvale umístěného uprostřed dolní části hlavního okna. Po zapnutí módu se uživateli zobrazí nabídka s typy prvků, které lze přidat. Pro editace je pak nutné mít stále zapnutý mód *Draw*, avšak bez zvoleného typu prvku pro přidání.
    - Přidání prvku (libovolný počet): zvolit typ, kliknutím umístit prvek a tahem nastavit počáteční velikost.
    - Pořadí přidaných prvků: přenesení dopředu kliknutím na prvek.

  - Přesouvat v rámci mapy: klasickým přetažením.
  - Editace vzhledu a atributů: dvojklikem na prvek se zobrazí nabídka s možnostmi pro daný prvek.
  - Smazání elementu: poslední položka nabídky dostupné po dvojkliku.
- **Správa vrstev**: veškerou správu vrstev lze provádět prostřednictvím správce, který je plovoucím prvkem hlavního okna. Správci lze pomocí pravého dolního rohu upravit velikost. Pomocí záhlaví a tlačítek v něm je pak možné jej přesunout, otáčet či skrýt.
  - Přidání vrstvy: pomocí tlačítek v zápatí správce vrstev nebo pomocí příslušných klávesových zkratek. Následně je zobrazen dialog, kde se stačí řídit jednotlivými kroky.
  - Editace vrstvy: přesunutí v rámci správce pomocí šipek u každé vrstvy nebo pomocí dlouhého stisku a následným přesunutím. Zobrazení a skrytí vlastností pomocí kliknutí na název vrstvy. Zobrazení či skrytí zobrazené vrstvy checkobexm vlevo u každé vrstvy.
  - Smazání vrstvy: *swipe* gesto zprava doleva a potvrzení pro smazání 1 vrstvy. Smazat všechny současně lze tlačítkem v zápatí správce.
- **Manipulace s výstupy**: pomocí navigačních tlačítek na hlavní obrazovce nebo pomocí gest (*pinch, drag & drop*), viz kapitola 6.5.9.

7.2 Omezení současného řešení

Fungování výsledné aplikace je doprovázeno množinou určitých slabších míst či omezení. Pro zprovoznění aplikace je například nutné správně nainstalovat WinGRASS a následně provést několik úprav v jeho spouštěcích dávkových souborech, jak bylo uvedeno v kapitole 6.5.1. Za určité omezení lze považovat také momentální závislost právě na WinGRASS a z toho plynoucí nutnost využívat aplikaci pouze na desktopových zařízeních. Poměrně velkým omezením je momentální řešení aplikace s ohledem na využití příkazového řádku operačního systému *Windows*. Z tohoto důvodu není možné naplno využít možností WinGRASS. Příkazová řádka ve *Windows* neumožňuje využití celé řady nástrojů v rámci tohoto GIS, především pak monitory pro správu vrstev. Z tohoto důvodu není možné s vrstvami v rámci jednoho projektu pracovat jako s celkem, ale je nutné pracovat v jeden moment pouze s jednou vrstvou. Pro další funkcionalitu WinGRASS je pak nutné instalovat řadu knihoven a nástrojů, aby ji bylo možné využívat i z příkazové řádky *Windows*.

Co se týká slabších míst implementované funkcionality, tak za určitou slabinu lze považovat to, že přidané mapové prvky se nezobrazí jako nová vrstva ve správci. Aplikace také nedisponuje souřadnicovým systémem, který by byl propojen se souřadnicovým systémem v GRASS GIS a zobrazené výstupy jsou tedy pouze PNG soubory. Není tedy možné jednoznačně určit konkrétní bod na mapě.

Někdy se může také stát, že při ukončení aplikace je aplikace ukončena dřív, než dojde ke smazání obsahu uživatelské složky *user_data*, která se nachází

v adresáři s geodaty. To obvykle způsobí, že při dalším spuštění aplikace nejsou správně zobrazeny mapové výstupy k příslušným vrstvám. Je dobré před spuštěním zkontrolovat, zda je uživatelský adresář odstraněn.

7.3 Možnosti využití aplikace a ekonomické hledisko

Současné řešení výsledné aplikace je navrženo zejména jako moderní a přívětivé grafické uživatelské rozhraní pro dotykové obrazovky, které prostřednictvím GRASS GIS jako *back-end* umožňuje tvorbu jednoduchých projektů a prohlížení základních prostorových dat. Především však umožňuje manipulaci s výstupy pomocí gest. Proto lze říct, že oblast využití této aplikace je poměrně široká.

Současný GRASS GIS lze také nahradit jiným systémem, dle daného účelu a využít tak jeho funkcionalitu. Stačí pouze, aby daný systém umožňoval nějakou komunikaci a přístup ke své funkcionalitě, například jako GRASS GIS, pomocí příkazové řádky. Následně je pak nutné v jazyce *C++* vytvořit komunikační modul a napojit jej na v jazyce *QML* implementované uživatelské rozhraní. Díky tomu, že knihovna Qt je multiplatformní, je možné využít implementované rozhraní jak na desktopové platformy, tak platformy mobilní. Ve výsledku je pak závislost na platformě odvozena od využitého GIS. GRASS GIS je momentálně k dispozici pouze pro desktopové platformy, proto i realizovaná aplikace je určena pouze pro desktopy. Vzhledem k této skutečnosti je výsledná aplikace ideální tam, kde je v současnosti využíván GRASS GIS na dotykových obrazovkách s některým ze stávajících uživatelských rozhraní, která pro tyto obrazovkou nejsou příliš vhodná, jak bylo uvedeno v rešerši.

Na základě toho, že se jedná o aplikaci pro desktopy s dotykovými obrazovkami, mohla by aplikace sloužit také jako výrazně levnější alternativa projektů využívajících GIS na velkoplošných dotykových obrazovkách, jako je např. *TouchShare* o kterém byla zmínka v úvodu této práce. Vzhledem k tomu, že výsledná aplikace je momentálně založena na svobodném GRASS GIS a také knihovna Qt je k dispozici zdarma, náklady na pořízení licencí jsou tak v podstatě nulové. Zůstávají tak náklady pouze na vývoj *front-end* aplikace, které jsou však v porovnání s náklady na pořízení kompletního řešení, nebo analýzu a vývoj řešení vlastního, zanedbatelné. Navíc by bylo možné tyto náklady pokrýt případným následným poskytnutím výsledného řešení za poplatek pod vlastní licencí.

Realizovaná aplikace tak může ukázat cestu snadného a rychlého vývoje GIS s moderním grafickým uživatelským rozhraním, které může přispět k výraznému snížení nákladů na vývoj obdobných systémů využitím již existujících svobodných GIS. Může také posloužit jako vzorové řešení využívající možností bohaté knihovny Qt v této oblasti a tím vývoj multiplatformní aplikace také urychlit, usnadnit a především pak také snížit ony náklady na vývoj.

7.4 Možnosti dalšího rozvoje aplikace

V současné době prostor pro vývoj GIS a především pak jeho využití je obrovský. Proto i pro realizovanou aplikaci v rámci této práce je prostor pro možnosti budoucího rozšíření a využití je neméně tak velký.

Zásadním krokem pro možnosti budoucího rozšíření výsledné aplikace je rozhodnutí, zda i nadále rozvíjet tuto aplikaci jako *front-end* zvoleného GRASS GIS, nebo nahradit tento systém jiným či zvolit úplně jinou cestu odvozenou zejména na základě účelu, jemuž bude aplikace sloužit. Zvolený GRASS GIS je však velmi rozsáhlým systémem a poskytuje širokou funkcionalitu, která sama o sobě dává mnoho dalších možnosti pro rozšíření, ale ve výsledku i určitá omezení realizované aplikace.

Základním vylepšení, které by bylo velmi vhodné realizovat je rozšíření na zbylé podporované platformy. Jak již bylo uvedeno výše ve zmíněných omezeních konečného řešení, zejména využití platformy *Linux* by umožnilo výrazně bohatší možnosti komunikace se systémem GRASS GIS prostřednictvím příkazové řádky a také komplexnější manipulaci s vrstvami či využití celé řady dalších modulů, kterými GRASS GIS disponuje.

Další rozšíření současného řešení by bylo možné například v oblasti správy a manipulace s mapovými výstupy. Opět byl tento problém zmíněn také v omezeních tohoto řešení. Základem by bylo zavedení nějakého souřadnicového systému, který by korespondoval s nastavením souřadnicového systému aplikace na pozadí, případně realizovat napojení přímo na daný souřadnicový systém. Současné řešení také nedisponuje žádnými analytickými nástroji, které jsou hlavní doménou svobodného systému GRASS GIS. Bylo by tedy vhodné zpřístupnit alespoň vybranou skupinu těchto nástrojů také prostřednictvím tohoto uživatelského rozhraní. V neposlední řadě by bylo velmi užitečné integrovat modul pro export celého projektu do podoby mapy, kterou by bylo možné následně otevřít také v jiných aplikacích či rovnou vytisknout do papírové podoby.

Velmi zajímavým rozšířením by také mohlo být využití 3D zobrazení, kterým GRASS GIS také disponuje. Knihovna Qt také disponuje modulem *QtQuick3D*, který by tomuto účelu mohl vhodně posloužit. Právě kombinace 3D zobrazení a manipulace pomocí gest by byla v tomto směru velmi užitečná. Příliš mnoho existujících aplikací nedisponuje kvalitním 3D modulem a k tomu ještě navíc uživatelským rozhraním umožňující manipulaci prostřednictvím gest.

Potenciálních rozšíření lze navrhnout opravdu velké množství. V současné době, kdy se téměř veškerá činnost odehrává prostřednictvím internetového připojení, bylo by možné aplikaci také rozšířit o modul, který by například umožňoval týmovou spolupráci nad daným projektem a podobně.

8 Závěr

Cílem práce bylo prozkoumat možnosti knihovny Qt v oblasti tvorby aplikací pro dotykové obrazovky a následně na základě poznatků se pokusit navrhnout a implementovat vlastní jednoduchý uživatelsky přívětivý GIS, který bude umožňovat základní manipulaci s geodaty, jako vkládání prostorových dat, jejich editaci a především manipulaci s nimi pomocí gest. Hlavního cíle práce byly tedy dosaženo.

V úvodní části této práce bylo provedeno zmiňované nastínění současné situace na poli moderních populárních GIS. Nejdříve bylo provedeno seznámení s desktopovými verzemi těchto systémů, kde byly zahrnuty jak svobodné distribuce, tak komerční a následně bylo uvedeno i několik aplikací pro různá mobilní zařízení. U rozboru jednotlivých GIS byla uvedena základní funkcionalita a také provedeno zhodnocení uživatelského rozhraní, kterým dané aplikace disponují.

Další část diplomové práce je věnována jednotlivým standardům OGC konsorcia, jenž jsou prostřednictvím svých mapových služeb velmi častým zdrojem geodat pro jednotlivé GIS a také se těší stále větší oblibě. Zde byl zejména proveden stručný popis toho, k čemu jednotlivé standardy slouží.

V rámci metodiky byly nastíněny kroky, které bylo nutné vykonat ještě před samotnou implementací konečného řešení vlastní aplikace. Ve vlastní práci pak byl definován typický uživatel a hlavní účel aplikace. Na základě těchto zjištěných skutečnostech byla vyspecifikována množina základních funkčních i nefunkčních požadavků na konečné řešení aplikace a vyhotoveny návrhy uživatelského rozhraní včetně použití řady návrhových vzorů. Na základě hlavního cíle a pomocí nabytých informací z rešerše byl pro výslednou implementaci zvolen *Qt Quick* modul z knihovny Qt s primárním jazykem *QML* a rozhodnuto o tom, že výsledná aplikace bude fungovat jako *front-end* pro svobodný systém GRASS GIS.

Výsledná aplikace disponuje možnostmi pro přidání rastrových či vektorových vrstev. Dále pak přidání rastrových vrstev z WMS služby, mapových elementů včetně editace jak těchto mapových prvků, tak parametrů jednotlivých vrstev. Umožňuje také základní správu projektu jako například ukládání či načítaní projektu a konečně také manipulaci pomocí základní sady gest.

Závěrečná část práce je pak věnována hodnocení realizovaného řešení a diskusi o možnostech jeho využití. Dále byla navrhnuta možná vylepšení či rozšíření finální aplikace.

9 Literatura

ArcGIS. *ArcGIS app for smartphones and tablets* [online]. © Copyright 2014 [cit. 2014-11-29]. Dostupné z: http://doc.arcgis.com/en/arcgis-app/

ArcGIS Resources. *ArcGIS Help 10.2, 10.2.1, and 10.2.2* [online]. © 1995-2014 [cit. 2014-11-29]. Dostupné z: http://resources.arcgis.com/en/help/main/10.2/#/Overview_of_OGC_and_ISO_support/003700000002000000/

Espacenet. *Bibliographic data* [online]. 2014 [cit. 2014-11-25]. Dostupné z: http://worldwide.espacenet.com/publicationDetails/biblio?CC=US&NR=3911215A&KC=A&FT=D

Esri. *ArcGIS 10.2.2 for Desktop Functionality Matrix* [online]. [cit. 2014-11-29]. Dostupné z: http://www.esri.com/~/media/Files/Pdfs/library/brochures/pdfs/arcgis1022-desktop-functionality-matrix.pdf

Esri. *ArcGIS for Desktop* [online]. 2014 [cit. 2014-11-29]. Dostupné z: http://www.esri.com/software/arcgis/arcgis-for-desktop

Esri. *ESRI Shapefile Technical Description* [online]. 1998 [cit. 2014-11-28]. Dostupné z: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

Esri. *Landsat Touch* [online]. 2014 [cit. 2014-11-25]. Dostupné z: http://maps.esri.com/sldemos/landsat/default.html

EZUST, A. -- EZUST, P. *Introduction to Design Patterns in C++ with Qt4*. USA: Prentice Hall, 2006. 656 s. ISBN 0-13-187905-7.

FreeGIS. *FreeGeoDataCZ. Version: 0.3.3.* 2012. Dostupné z: http://geo.fsv.cvut.cz/data/grasswikicz/freegeodatacz/aktualni/cr-shp-jtsk-0.3.3.zip

GRASS Development Team, 2012. *Geographic Resources Analysis Support System (GRASS) Software*, Version 6.4. Open Source Geospatial Foundation. http://grass.osgeo.org

GRASS GIS. *General overview* [online]. 2014 [cit. 2014-11-28]. Dostupné z: http://grass.osgeo.org/documentation/general-overview/#GeneralInformation

GRASS GIS: The world's leading Free GIS software. *GRASS GIS sample data download* [online]. © 1998-2014 [cit. 2014-12-08]. Dostupné z: http://grass.osgeo.org/download/sample-data/

GRASS GIS Tracker and Wiki. *Active Tickets* [online]. © 2003-2009 [cit. 2014-12-02]. Dostupné z: http://trac.osgeo.org/grass/report/1

GRASS-Wiki. *WinGRASS errors* [online]. 2014 [cit. 2014-12-08]. Dostupné z: http://grasswiki.osgeo.org/wiki/WinGRASS_errors

MACHALOVÁ, J. *Prostorově orientované systémy pro podporu manažerského rozhodování*. 1. vyd. Praha: C.H. Beck, 2007. 141 s. C.H. Beck pro praxi. ISBN 978-80-7179-463-9.

OGC. *About OGC* [online]. ©1994 - 2014 [cit. 2014-11-27]. Dostupné z: http://www.opengeospatial.org/ogc

OGC. *OGC Standards* [online]. ©1994 - 2014 [cit. 2014-11-27]. Dostupné z: http://www.opengeospatial.org/standards/is

OGC. *Web WCS 2.0 Interface Standard- Core: Corrigendum. Version: 2.0.1*. Open Geospatial Consortium, Inc., 2012. Dostupné z: https://portal.opengeospatial.org/files/09-110r4

OGC. *Web Processing Service. Version: 1.0.0.* Open Geospatial Consortium, Inc., 2007. Dostupné z: https://portal.opengeospatial.org/files/?artifact_id=24151

OGC. *Web Map Server Implementation Specification. Version: 1.3.0.* Open Geospatial Consortium, Inc., 2006. Dostupné z: http://portal.opengeospatial.org/files/?artifact_id=14416

OGC. *Web Map Tile Service Implementation Standard. Version: 1.0.0.* Open Geospatial Consortium, Inc., 2010. Dostupné z: http://portal.opengeospatial.org/files/?artifact_id=35326

OGC. OpenGIS Web Feature Service Implementation Specification *. Version: 1.1.0* Open Geospatial Consortium, Inc., 2005. Dostupné z: portal.opengeospatial.org/files/?artifact_id=8339

OpenJUMP GIS [online]. © 2011 [cit. 2014-11-28]. Dostupné z: http://www.openjump.org

QGIS. *Documentation for QGIS 2.2* [online]. 2014 [cit. 2014-11-28]. Dostupné z: http://docs.qgis.org/2.2/en/docs/index.html

Qt Project. [online]. © 2014 [cit. 2014-12-01]. Dostupné z: http://qt-project.org

Sourceforge. *OpenJUMP 1.6 with Pie-Chart plugin* [obrázek]. In: *The JUMP Pilot Project* [on-line]. © 2014 [cit. 2014-12-01] Dostupné z: http://sourceforge.net/projects/jump-pilot/

SUMMERFIELD, M. -- BLANCHETTE, J. *C++ GUI Programming with Qt4*. USA: Prentice Hall, 2008. 752 s. ISBN 0-13-235416-0.

Supergeo Technologies. *SuperSurv 3* [online]. 2014 [cit. 2014-11-29]. Dostupné z: http://www.supergeotek.com/ProductPage_SuperSurv.aspx

TIDWELL, J. *Designing interfaces*. 2nd ed. Sebastopol. CA: O´Reilly, 2010. ISBN 14-493-7970-2

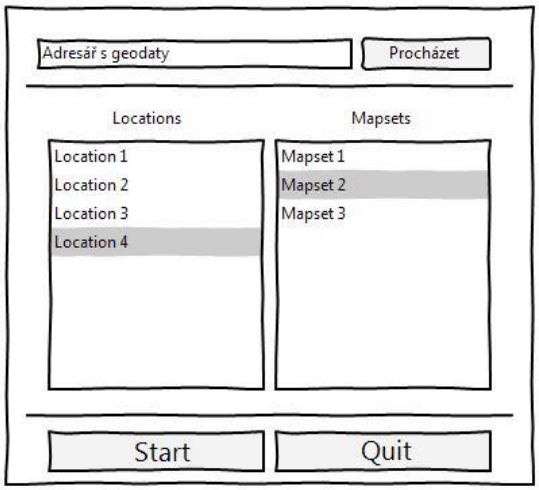
TouchShare. *TouchShare Solution* [online]. © 2013 [cit. 2014-11-25]. Dostupné z: http://touchshare.com/

UDig [online]. © 2008 [cit. 2014-11-28]. Dostupné z: http://udig.refractions.net/

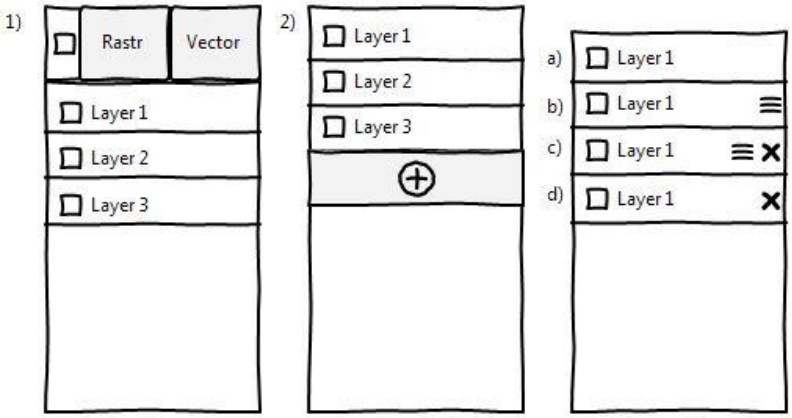
WolfGIS: Mobile Land Management GIS Technology [online]. Copyright 2014 [cit. 2014-11-29]. Dostupné z: http://wolfgis.com/

Přílohy

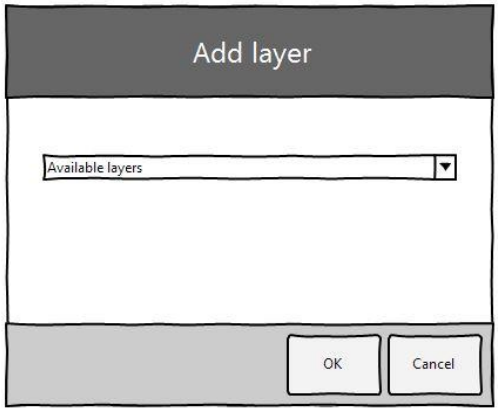
A Návrhy a realizace uživatelského rozhraní



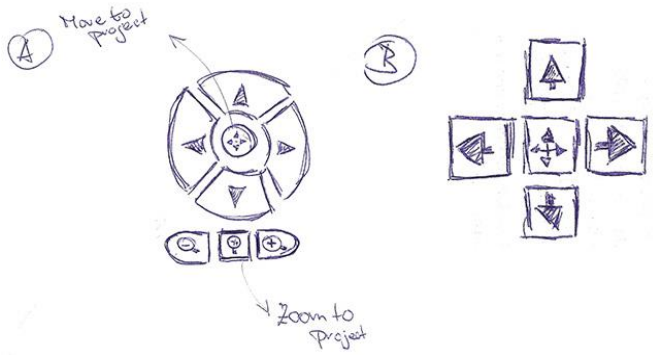
Obr. 35 Návrh úvodní spouštěcí obrazovky



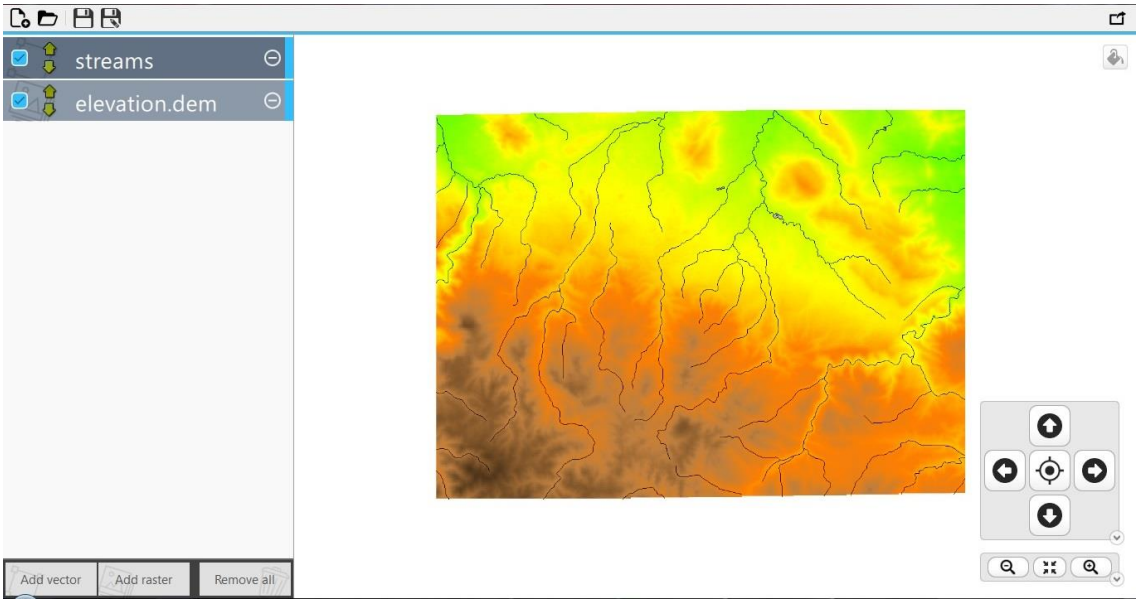
Obr. 36 Návrhy podoby správce vrstev (1 a 2) a položek v seznamu vrstev (a až d)



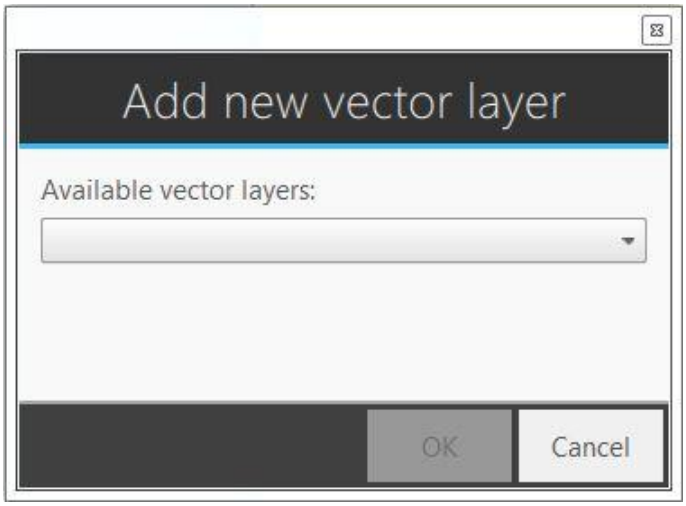
Obr. 37 Dialogové okno pro přidání rastrové či vektorové vrstvy



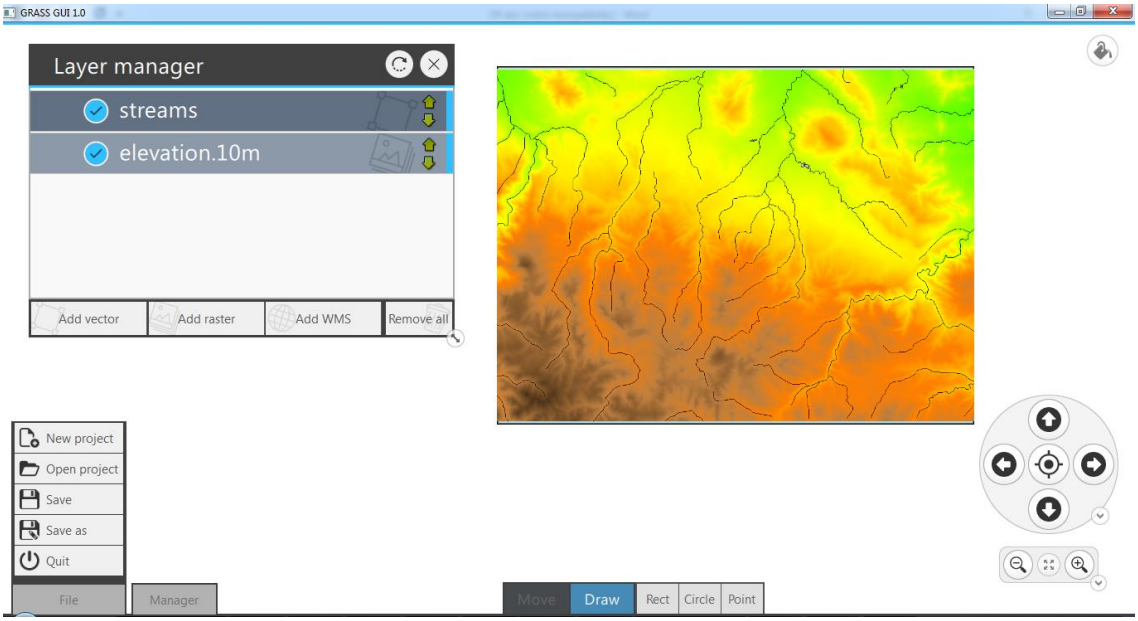
Obr. 38 Náčrtky možného vzhledu navigačních tlačítek



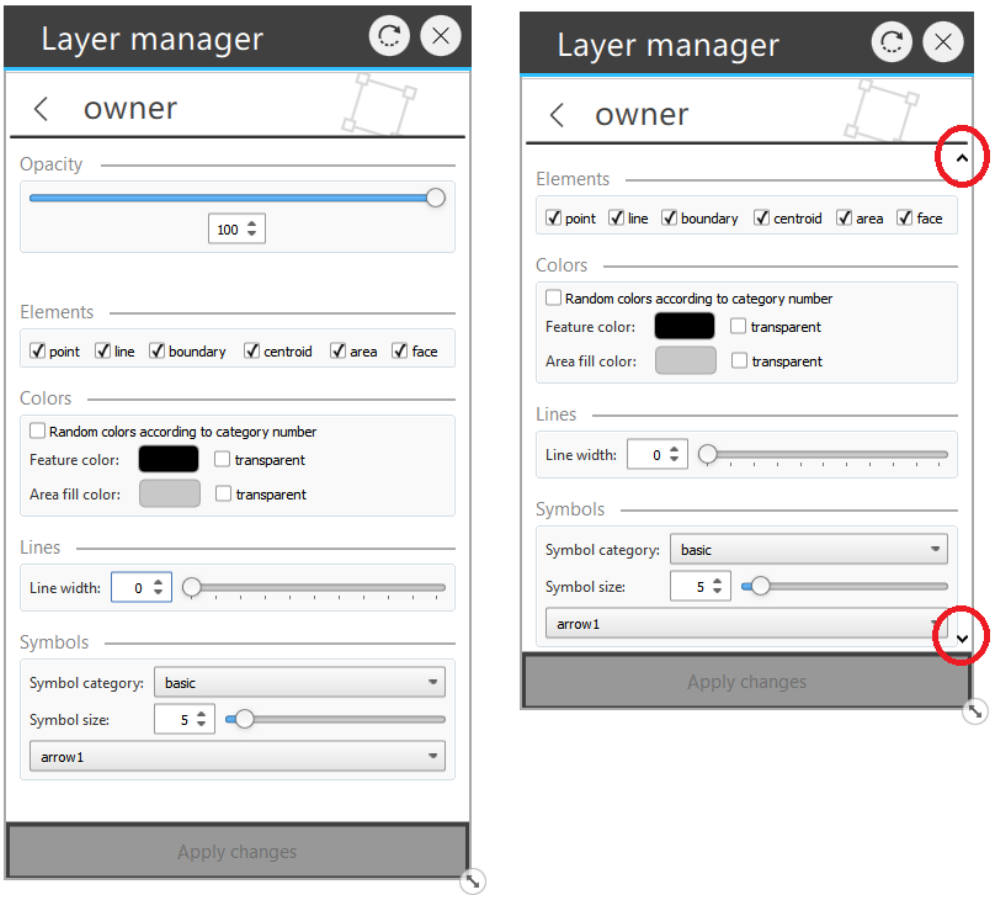
Obr. 39 Jedna z prvních realizací uživatelského rozhraní aplikace



Obr. 40 Výsledná podoba dialogu pro přidání rastrové a vektorové vrstvy



Obr. 41 Výsledná realizace rozhraní se zobrazenou navigací, nabídkami a módem pro kreslení



Obr. 42 Vlastnostmi vektorové vrstvy a zvýrazněné indikátory možnosti posouvání

B Obsah CD

Obsahem přiloženého CD je výsledná aplikace včetně zdrojových souborů projektu (zprovoznění viz 6.5.1) a použitých ikon. Dále jsou přiložena vzorová prostorová data použitá při vývoji a testování. A v neposlední řadě CD obsahuje také instalační soubory jednotlivých doplňkových aplikací a další knihovny, které jsem byl nucen pro možnost využití jednotlivých WinGRASS nástrojů (zejména WMS) nainstalovat.

- Adresář projektu *grass_demo*:
    - *content*: implementace v jazyce *QML*
    - *src*: implementace typů v jazyce *C++*
    - *other*: ostatní zdrojové soubory
    - *images*: použité ikony
    - *release*: adresář se spustitelnou aplikací
- Instalační soubory a knihovny:
    - WinGRASS GIS
    - Doplňující knihovny a nástroje
- Vzorová geodata použitá při vývoji
    - Spearfish datová sada
    - North Carolina datová sada