Katedra mapování a kartografie Stavební fakulty
ČVUT

Diplomová práce

Téma:

Automatizovaná tvorba prostorových modelů map

2002 **Jan Havrlant**

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím literatury uvedené v seznamu.

V Praze, dne 6. prosince 2002 *podpis*

Chtěl bych tímto poděkovat svému vedoucímu diplomové práce Ing. Petru Soukupovi, PhD. za jeho rady a připomínky k diplomové práci. Dále bych chtěl poděkovat Zeměměřičskému úřadu za poskytnutá data a PhDr. Jaroslavu Kuchařovi, CSc. za to, že mi věnoval mapy pro orientační běh, které jsem využil ve své diplomovou práci.

Obsah

Zkratky

Bpv	Výškový systém baltský – po vyrovnání.
DMT	Digitální model terénu.
DPI	Dot per inch – počet bodů na palec.
EAI	External Authoring Interface. Rozhraní, které umožňuje předávat data mezi VRML světem a externí aplikací.
GIS	Geografický informační systém. Software spojující prostorově orientovaná grafická data s databázovými daty a umožňující provádět analýzy.
HSB	Hue, saturation brightness. Odstín, sytost, jas. Třísložkový barevný systém.
html	HyperText Markup Language. Označovací jazyk pro hypertext. Jazyk používaný pro vytváření hypermediálních dokumentů pro službu WWW internetu.
ISO	International Standards Organization. Mezinárodní organizace pro normalizaci.
RGB	Red Green Blue. Červená, zelená, modrá. Třísložkový barevný systém.
RZM	Rastrová reprezentace Základní mapy ČR.
S-JTSK	Souřadnicový systém Jednotné trigonometrické sítě katastrální.
VAG	VRML Architecture Group. Skupina programátorů a návrhářů zabývající se vývojem jazyka VRML.
VRML	Virtual reality modeling language. Modelovací jazyk pro popis virtuální reality.
WWW	World Wide Web. Celosvětová síť (pavučina). Distribuovaný soubor dokumentů v internetu, lze jej hypertextově prohledávat.
ZABAGED	Základní báze geografických dat.

Typografická konvence

V této diplomové práci používám následující konvence:

`Funkce` Tímto písmem jsou označovány všechny výpisy programů, názvy funkcí a parametry funkcí.

Název Tímto písmem jsou zapsány citace, názvy programových produktů, jména firem, původní anglické a jiné názvy.

Tlačítko Bezpatkové písmo používám pro názvy tlačítek a příkazy nabídek programů.

Odkaz Tímto písmem jsou označovány internetové adresy.

1 Úvod

Důvodů, proč jsem si vybral toto téma, je hned několik. V prvé řadě se už dlouho zajímám o počítače, rád programuji a v neposlední řadě mě zaujala diplomová práce mého kolegy Karola Jandy o prostorových modelech terénu ve virtuální realitě na internetu. Proto bych rád navázal na jeho práci, využil jeho poznatků a zkušeností a pokusil se dále rozvést toto téma směrem k automatizované tvorbě prostorových modelů map.

Jazyk VRML 97 je dnes již standardem pro zobrazování prostorových dat na internetu, jeho výhody a otevřenost tohoto formátu mě proto vedly k jeho použití. Stále více programů zabývajících se tvorbou a zobrazením prostorových modelů, např. *MicroStation* nebo *3D Studio*, obsahuje možnost exportu do VRML. Co se týče jejich vztahu k VRML, nepracují s ním jako se stavebním kamenem, ze kterého by vycházely, ale berou ho jako prostředek pro zápis výsledné scény. Z toho vyplývá, že VRML je pro ně jen okrajovou záležitostí, a proto je také export do formátu VRML u nich špatný. Nevyužívají všech možností, které VRML nabízí, a tím snižují rychlost a kvalitu zobrazení. Dalším problémem, kterým se v této diplomové práci chci zabývat, jsou možnosti zobrazení větších územních celků ve VRML. Problém spočívá ve velkém množství dat, které musí prohlížeč VRML modelů zobrazit najednou, čímž se sníží rychlost zobrazení.

V první části této práce bych chtěl popsat základní prvky jazyka VRML, které jsem použil při tvorbě prostorových modelů terénu. V pořadí třetí kapitola se věnuje možnostem zobrazení modelů map ve VRML a vlastnostem, které přispívají k větší reálnosti zobrazovaného modelu. Čtvrtá kapitola se zabývá možnostmi zrychlení zobrazení prostorových modelů ve VRML. Pátá kapitola naznačuje možné využití vytvořených modelů. V šesté kapitole se pokusím nastínit možnosti automatizace tvorby modelů rozdělené do základních kroků, pro které jsem napsal čtyři rozsáhlejší programové moduly vykonávající tyto kroky. Automatizovaný postup jsem navrhl pro tvorbu prostorových modelů z klasických papírových map, ale i map v digitální formě, a to jak rastrové, tak i vektorové. Sedmá kapitola popisuje možné zdroje dat pro tvorbu modelů map.

Zdrojové kódy programů a jejich spustitelné verze si lze prohlédnout na adrese http://gama.fsv.cvut.cz/~soukup/dip/havrlant, kde je také vystavena celá má diplomová práce spolu s VRML modely, které jsem vytvořil s použitím mnou napsaných programů.

2 Úvod do VRML

Na začátku bych chtěl shrnout základní vlastnosti jazyka VRML, který se používá k popisu virtuálního světa.

V této kapitole uvedu nejdůležitější prvky jazyka VRML, které jsem použil pro tvorbu prostorových modelů map, dále jejich výhody, nevýhody a vlastnosti důležité pro tvorbu DMT.

2.1 Historie

Historie jazyka VRML 97 začíná v roce 1995 ve firmě *Silicon Graphics* definicí VRML 1.0 jako rozšíření aplikační knihovny *OpenInventor* (pro použití grafické knihovny *OpenGL*) o využívání prostorových dat ze sítě WWW. Současně vzniká skupina vývojářů VAG (*VRML Architecture Group*), která se obrací na všechny významné tvůrce systémů virtuální reality, ale i veřejnost s výzvou k vytvoření specifikace budoucího jazyka VRML 2.0. V roce 1996 je z 8 různých návrhů vybrána specifikace firem *Silicon Graphics* a *Sony*, ta se stává základem specifikace VRML 2.0. Tato specifikace byla během jednoho roku měněna a doplňována v otevřené diskusi probíhající na internetu. Koncem roku 1997 je specifikace jazyka VRML oficiálně přijata za standard ISO a nyní se používá pod označením VRML 97. Dále v textu budu pod označením VRML chápat definici VRML 97.

2.2 Prohlížení virtuálních světů

K prohlédnutí virtuálního světa potřebujeme prohlížeč VRML, který provádí převod textového popisu virtuálního světa do grafické podoby. Nejčastěji se tento prohlížeč instaluje jako doplněk internetového prohlížeče (např. programu *Internet Explorer* nebo *Netscape Communicator*). Nejznámější prohlížeče pro VRML jsou *CosmoPlayer* od firmy *Silicon Graphics*, *WorldView* od firmy *Intervista Software* a *Corona* od firmy *ParallelGraphics*[1]. Pomocí prohlížeče se dá virtuální svět dokonce vnímat prostorově, např. při použití speciálních brýlí.

[1] Výše zmíněné VRML prohlížeče jsou v současné době vyvíjeny pouze pro operační systém *Microsoft Windows* popřípadě *Macintosh MacOS*. Seznam dalších prohlížečů je uveden např. v přehledu VRML prohlížečů: http://www.web3d.org/vrml/bro2.htm.

Prohlížeč VRML nám umožňuje pohyb ve virtuálním světě a případnou interakci s virtuálními předměty.

2.3 Zobrazení virtuálního světa na internetu

Při zobrazení VRML modelu máme dvě možnosti. Lze ho umístit na internetové stránce jako odkaz přímo na VRML model, který se pak zobrazí v celém okně prohlížeče. Další možností je zobrazit ho jako součást html stánky pomocí příkazu `EMBED`.

Příklad použití:

```
<EMBED SRC="file.wrl"
WIDTH="300"
HEIGHT="300"
PLUGINSPAGE="http://www.parallelgraphics.com/cortona"
VRML_DASHBOARD="FALSE"
VRML_BACKGROUND_COLOR="#000077"
CONTEXTMENU="FALSE"
MASK="0 0, 150 10, 300 0, 290 150, 300 300, 150 290, 0 300,
10 150">
```

Kde

`SRC` specifikuje jméno VRML modelu, který má být zobrazen.

`PLUGINSPAGE` navádí uživatele na stránky, které ho informují, jak nainstalovat plug-in, který umí zobrazit VRML soubor, jestliže ho uživatel ještě nemá nainstalován.

`VRML_DASHBOARD`[2] "TRUE" – zapne zobrazení ovládacího panelu; "FALSE" – zobrazení vypne.

`VRML_BACKGROUND_COLOR`[3] "#rrggbb" specifikuje barvu pozadí modelu (hexadecimálně), pokud ovšem není pozadí již definováno ve VRML souboru uzlem `Background`.

[2] Jméno parametru platí pro prohlížeč *Corona*, prohlížeč *CosmoPlayer* používá označení `VRML-DASHBOARD`.

[3] Jméno parametru platí pro prohlížeč *Corona*, prohlížeč *CosmoPlayer* používá označení `VRML-BACKGROUND-COLOR`

`CONTEXTMENU`[4] "TRUE" – umožňuje použít kontextové menu v okně modelu; "FALSE" – kontextové menu zablokuje.

`MASK`[4] umožňuje specifikovat tvar oblasti, ve které bude model zobrazen.

2.4 Popis VRML

Soubor s popisem v jazyce VRML se skládá z uzlů a tvoří hierarchickou (stromovou) strukturu. V roli parametrů uzlů se mohou objevovat jiné uzly. Podle vzájemné polohy jednotlivých uzlů mluvíme o vztazích, jako je rodič, potomek nebo sourozenec. Podobně jako rodič v rodině předává svým potomkům své vlastnosti, předává rodičovský uzel vlastnosti svým potomkům (např. poloha nebo rotace).

Soubor s popisem virtuálního světa můžeme rozdělit do několika základních částí. Na začátku je vždy hlavička. Její tvar je neměnný a informuje prohlížeč, o jaký typ souboru se jedná. Tento řádek je povinný. Za hlavičkou většinou následují úvodní informace o virtuálním světě. Jsou to informace o souboru a jeho tvůrci (`WorldInfo`), seznam zajímavých míst uvnitř virtuálního světa (`Viewpoint`) a způsob procházení světem (`NavigationInfo`). Třetí, nejrozsáhlejší část tvoří vlastní popis virtuálního světa. Pořadí těchto částí není závazné, ale je dobrým zvykem ho kvůli čitelnosti dodržovat.

Soubory, které popisují VRML-světy, mají příponu wrl a lze je pro úsporu místa zkomprimovat programem *gzip,* přičemž si zachovají příponu wrl, prohlížeč sám rozpozná takový soubor a automaticky ho dekomprimuje.

Abychom mohli s VRML pracovat, je zde zaveden systém prostorových souřadnic. Směr vzhůru je totožný s kladnou osou Y a země je totožná s rovinou XZ. Kladná poloosa X běžně míří doprava a osa Z míří směrem k nám.

Řádek začínající znakem # je chápán jako komentář (s výjimkou prvního řádku).

Parametry podle vztahu k událostem se dělí na 4 druhy. `field` je statická veličina a lze ji změnit pouze zapsáním nové hodnoty do souboru VRML. `eventIn` je parametr schopný přijmout událost daného datového typu. `eventOut` je parametr schopný vyslat událost v okamžiku, kdy dojde ke změně jeho hodnoty. Parametr `exposedField` má iniciální hodnotu a je schopen přijmout události měnící jeho hodnotu a také po změně této

[4] Tento parametr podporuje pouze prohlížeč *Corona*

hodnoty události vysílat. Protože tvorba dynamických světů jde nad rámec této diplomové práce, není v dalším popisu uzlů uveden význam dynamických parametrů.

Názvy uzlů začínají velkým písmenem a názvy parametrů malým. Je důležité si na to dávat pozor při psaní, neboť jazyk VRML je citlivý na velká a malá písmena.

Délky se ve VRML zadávají v metrech, čas v sekundách a úhly v radiánech. Barva se popisuje trojicí hodnot RGB v rozsahu 0-1.

Na pořadí parametrů uvnitř uzlu nezáleží. Každý parametr má definovanou iniciální hodnotu. Stačí tedy definovat jen ty parametry, jejichž hodnoty jsou odlišné do iniciálních. Zápis všech hodnot snižuje čitelnost programu.

Textové řetězce se zapisují do uvozovek. U každého parametru je stanoveno, zda do něj lze zapsat jednu nebo více hodnot. Vkládá-li se více hodnot, jsou uzavřeny do hranatých závorek a odděleny bílými znaky (konec řádku, tabulátor, mezera nebo čárka). Každý parametr má jednoznačně dáno, jaký typ dat do něj lze ukládat. Je definováno několik datových typů, obvykle ve dvou variantách, buď s předponou SF (*Single Field*), která dovoluje ukládat pouze samostatnou hodnotu, nebo s předponou MF (*Multiple Field*), do které se zadává seznam hodnot, jehož délka není obecně omezena.

2.4.1 Avatar (Návštěvník)

Pojem avatar[5] označuje uživatelova dvojníka, který prochází virtuálním světem. To, co avatar vidí, je zobrazeno v okně prohlížeče. Avatar má své rozměry, které mu např. brání procházet malými průchody.

V interaktivně navržených světech lze detekovat, v jakém místě virtuálního světa avatar stojí, co všechno vidí, zda naráží na překážku nebo zda vstupuje do nějaké hlídané zóny. O veškerou tuto detekci se starají senzory, speciální uzly definované ve VRML.

Ovládání avatara je zajištěno prostřednictvím tzv. velitelského stanoviště, které je na obrazovce reprezentováno ovládacími prvky prohlížecí aplikace. Jak vlastnosti avatara, tak i některé prvky velitelského stanoviště lze zadat pomocí uzlu `NavigationInfo`.

[5] Jméno avatar pochází z hinduistické mytologie, kde označuje dočasnou tělesnou schránku, do které se vtěluje Bůh při své návštěvě Země.

2.4.2 Hlavní uzly

Uzly, které jsem použil při tvorbě prostorového modelu, vyjmenuji v následujících podkapitolách.

2.4.2.1 Viewpoint (Stanoviště)

Umístění a pohledové vlastnosti stanoviště.

`Viewpoint {`

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFFloat	fieldOfView	0.785398	Zorný úhel
exposedField	SFBool	jump	TRUE	Povolení plynulého přechodu na stanoviště
exposedField	SFRotation	orientation	0 0 1 0	Natočení avatara
exposedField	SFVec3f	position	0 0 10	Poloha avatara
field	SFString	description	""	Jméno stanoviště uváděné prohlížečem
eventIn	SFBool	set_bind		
eventOut	SFTime	bindTime		
eventOut	SFBool	isBound		

`}`

Tento uzel definuje stanoviště pozorovatele. Je důležité, aby bylo správně umístěno a zajištěno zacílení pohledu pozorovatele na pozorovaný objekt. V opačném případě se může stát, že pozorovatel vůbec nenajde zobrazený model, zvláště tehdy, je-li tento model jednostranně zobrazen a ze stanoviště je vidět pouze rubová strana modelu.

Pomocí tohoto uzlu je také možné zajistit prohlídku virtuálního světa definováním více stanovišť, kdy při přepínání dochází k plynulému přechodu ze stanoviště na stanoviště.

První stanoviště definované v souboru nastavuje iniciální pohled při vstupu do virtuálního světa.

2.4.2.2 NavigationInfo (ovládání Avatara)

Geometrické, světelné a pohybové charakteristiky avatara.

`NavigationInfo {`

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	MFFloat	avatarSize	[0.25, 1.6, 0.75]	Trojice čísel určující rozměry avatara

exposedField	SFBool	headlight	TRUE	Zapnutí svítilny na čele avatara
exposedField	SFFloat	speed	1.0	Rychlost pohybu avatara v m/s
exposedField	MFString	type	["WALK", "ANY"]	Seznam povolených způsobů pohybu avatara
exposedField	SFFloat	visibilityLimit	0.0	Dohled avatara
eventIn	SFBool	set_bind		
eventOut	SFBool	isBound		

}

Při zobrazování krajiny je vhodné vypnout avatarovu svítilnu, aby se zvýšila reálnost prostředí. Dále je vhodné zvýšit rychlost pohybu avatara, implicitní hodnota je pro pohyb v krajině dosti nízká. Jako způsob pohybu je nejlépe použít typ WALK, při kterém se avatar pohybuje po zemi a působí na něj gravitace, dále je možné použít typ FLY, kdy se ve světě pohybuje jako pták, anebo použít typ ANY; pak je možné použít jakýkoliv způsob pohybu. Dohled avatara je dobré ponechat nastavený na nulu, což znamená nekonečno.

2.4.2.3 Background (pozadí, panorama)

Definice pozadí, reprezentovaná vnitřkem krychle s panoramatickým obrazem či vnitřkem koule s plynulými barevnými přechody, obklopující virtuální svět.

Background {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	MFFloat	skyAngle	[]	Rostoucí posloupnost úhlů, pro které jsou dány barvy oblohy
exposedField	MFColor	skyColor	0 0 0	Seznam barev pro postupné přechody na sférické obloze
exposedField	MFColor	groundColor	[]	Seznam barev pro postupné přechody na sférické zemi
exposedField	MFFloat	groundAngle	[]	Rostoucí posloupnost úhlů, pro které jsou dány barvy země
exposedField	MFString	backUrl	[]	Obrázek pro zadní stěnu obklopující krychle
exposedField	MFString	bottomUrl	[]	Obrázek pro spodní stěnu obklopující krychle
exposedField	MFString	frontUrl	[]	Obrázek pro přední stěnu obklopující krychle
exposedField	MFString	leftUrl	[]	Obrázek pro levou stěnu obklopující krychle
exposedField	MFString	rightUrl	[]	Obrázek pro pravou stěnu obklopující krychle
exposedField	MFString	topUrl	[]	Obrázek pro horní stěnu obklopující krychle
eventIn	SFBool	set_bind		
eventOut	SFBool	isBound		

}

Při prezentaci modelu dodá pozadí virtuálnímu světu větší iluzi prostoru, nehledě na to, že bez definování tohoto uzlu se mapa zobrazí s černým pozadím. Jsou dvě možnosti, jak definovat pozadí. Jednou z možností je definovat pozadí jako přechod barev měnících se s výškou nad horizontem, další možností jsou panoramatické obrázky obklopující virtuální svět ze všech stran.

2.4.2.4 PointLight (Bodový zdroj světla)

Charakteristika světelného zdroje, který vysílá paprsky z jednoho bodu. Jejich intenzita klesá v rámci zadaného rozsahu.

`PointLight {`

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFFloat	ambientIntensity	0	Příspěvek světelného zdroje k nepřímému osvětlení světelného zdroje
exposedField	SFVec3f	attenuation	1 0 0	Trojice koeficientů pro výpočet útlumu světla
exposedField	SFColor	color	1 1 1	Barva paprsků
exposedField	SFFloat	intensity	1	Intenzita paprsků
exposedField	SFVec3f	location	0 0 0	Umístění světelného zdroje
exposedField	SFBool	on	TRUE	Vypnutí nebo zapnutí světelného zdroje
exposedField	SFFloat	radius	100	Dosah osvětlení

`}`

Definice VRML obsahuje několik druhů světel. `PointLight` je nejvhodnější pro osvětlení světa, protože se jím dá definovat světelný zdroj podobný slunci.

2.4.2.5 Fog (Mlha)

Definice mlhy, která je míchána s barvou objektů podle rostoucí vzdálenosti od avatara.

`Fog {`

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFColor	color	1 1 1	Barva mlhy
exposedField	SFString	fogType	"LINEAR"	Způsob houstnutí mlhy
exposedField	SFFloat	visibilityRange	0	Vzdálenost maximálního dohledu
eventIn	SFBool	set_bind		

eventOut	SFBool	isBound

}

Pro zlepšení iluze prostoru můžeme použít mlhu, která je dnes všudypřítomná.

2.4.2.6 ElevationGrid (Výšková mapa)

Definice sítě ploch pokrývajících terén.

ElevationGrid {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFNode	color	NULL	Seznam barev v uzlu Color
exposedField	SFNode	normal	NULL	Seznam normál v uzlu Normal
exposedField	SFNode	texCoord	NULL	Seznam souřadnic textury v uzlu TextureCoordinate
field	MFFloat	height	[]	Pole výšek všech vrcholů sítě
field	SFBool	ccw	TRUE	Přivrácená strana mapy je vidět při pohledu shora
field	SFBool	colorPerVertex	TRUE	Barvy v parametru color se vztahují na vrcholy (jinak na plochy sítě)
field	SFFloat	creaseAngle	0	Mezní úhel, do kterého jsou dvě sousední plochy považovány za oblé
field	SFBool	normalPerVertex	TRUE	Normály v parametru normal se vztahují na vrcholy (jinak na plochy sítě)
field	SFBool	solid	TRUE	Mapa je jednostranná
field	SFInt32	xDimension	0	Počet vrcholů sítě v ose X
field	SFFloat	xSpacing	1.0	Vzdálenost mezi vrcholy sítě v ose X
field	SFInt32	zDimension	0	Počet vrcholů sítě v ose Z
field	SFFloat	zSpacing	1.0	Vzdálenost mezi vrcholy sítě v ose Z
eventIn	MFFloat	set_height		

}

Tento uzel je jedením ze dvou, který se používá pro definici modelu terénu. Skládá se ze sítě, nad kterou se definují výšky. Tento model se někdy také označuje jako rastrový model terénu. Plocha je standardně umístěna do roviny XZ, tedy tam, kde je očekávána pevná zem. Do parametru `height` se zapisuje pole výšek nad jednotlivými vrcholy sítě. Pomocí parametru `creaseAngle` se dá nastavit zaoblení terénu. Tak definujeme mezní úhel, při kterém jsou dvě plochy spojeny hladce.

2.4.2.7 IndexedFaceSet (Množina ploch)

Definice geometrie a vzhledu skupiny ploch.

IndexedFaceSet {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFNode	color	NULL	Seznam barev v uzlu Color
exposedField	SFNode	coord	NULL	Seznam vrcholů v uzlu Coordinate
exposedField	SFNode	normal	NULL	Seznam normál v uzlu Normal
exposedField	SFNode	texCoord	NULL	Seznam souřadnic textury v uzlu TextrueCoordinate
field	SFBool	ccw	TRUE	Plochy jsou zadávány proti směru hodinových ručiček
field	MFInt32	colorIndex	[]	Posloupnost indexu barev pro jednotlivé vrcholy nebo plochy
field	SFBool	colorPerVertex	TRUE	Barvy v parametru colorIndex se vztahují na vrcholy (jinak na plochy)
field	SFBool	convex	TRUE	Všechny plochy jsou konvexní
field	MFInt32	coordIndex	[]	Posloupnost indexů vrcholů ploch zakončená -1
field	SFFloat	creaseAngle	0	Mezní úhel, do kterého jsou dvě sousední plochy považovány za oblé
field	MFInt32	normalIndex	[]	Posloupnost indexů normál
field	SFBool	normalPerVertex	TRUE	Normály v parametru normalIndex se vztahují na vrcholy (jinak na plochy)
field	SFBool	solid	TRUE	Plochy se zobrazují jednostranně
field	MFInt32	texCoordIndex	[]	Posloupnosti indexů souřadnic textury pro jednotlivé vrcholy
eventIn	MFInt32	set_colorIndex		
eventIn	MFInt32	set_coordIndex		
eventIn	MFInt32	set_normalIndex		
eventIn	MFInt32	set_texCoordIndex		

}

Další uzel pro definici obecné plochy pomocí množiny ploch. Pomocí malých plošek se definuje libovolné těleso nebo plocha. Každá plocha se definuje pomocí pořadových čísel vrcholů. Tím se docílí úspory paměti.

2.4.2.8 Coordinate (Souřadnice prostorových bodů)

Definice souřadnic bodů v prostoru.

Coordinate {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	MFVec3f	point	[]	Seznam bodů

}

2.4.2.9 TextureCoordinate (Souřadnice textury)

Definice bodů v rovině, které slouží jako vztažné body pro nanesení textury v ploškových geometrických útvarech.

TextureCoordinate {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	MFVec2f	point	[]	Seznam bodů v rovině

}

Tento uzel jsem použil pro správné nanesení textury mapy v modelu vytvořeném uzlem `IndexedFaceSet`. Implicitní mapování textury na model vytvořený uzlem `ElevationGrid` je správné, a proto se při něm nemusí tento uzel používat.

2.4.2.10 TextureTransform (Transformace rovinné textury)

Nastavení posunu, otočení a měřítka rovinné textury.

TextureTransform {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFVec2f	center	0 0	Poloha vztažného bodu
exposedField	SFFloat	rotation	0	Úhel otočení vzhledem ke vztažnému bodu
exposedField	SFVec2f	scale	1 1	Změna měřítka
exposedField	SFVec2f	translation	0 0	Posunutí

}

Při nanášení textury na model vytvořený uzlem `ElevationGrid` je implicitně nanesená textura zrcadlově převrácená (vztažný bod výškové mapy je v levém horním rohu a vztažný bod textury v levém dolním rohu), proto je třeba texturu převrátit pomocí parametru `scale` (`scale` 1 -1).

2.4.2.11 ImageTexture (Textura určená obrázkem)

Určení obrázku a jeho případného opakovaného nanášení jako textury na povrch.

ImageTexture {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	MFString	url	[]	Seznam adres s umístěním obrázku
field	SFBool	repeatS	TRUE	Povolení opakování textury ve vodorovném směru
field	SFBool	repeatT	TRUE	Povolení opakování textury ve svislém směru

}

V tomto uzlu se definuje textura mapy, která se potom nanáší na prostorový model. Obrázek může být ve formátu JPEG a PNG. Některé prohlížeče podporují také formát GIF a vektorový formát CGM.

2.4.2.12 Transform (umístění skupiny)

Nastavení společné polohy, měřítka a otočení potomků.

Transform {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFVec3f	center	0 0 0	Vztažný bod, vůči kterému se provádí otočení
exposedField	MFNode	children	[]	Seznam potomků
exposedField	SFRotation	rotation	0 0 1 0	Osa a úhel otočení
exposedField	SFVec3f	scale	1 1 1	Změna měřítka ve směru os
exposedField	SFRotation	scaleOrientation	0 0 1 0	Osa a úhel otočení provedeného před změnou měřítka
exposedField	SFVec3f	translation	0 0 0	Posunutí
field	SFVec3f	bboxCenter	0 0 0	Souřadnice středu pomocné obálky ve tvaru kvádru
field	SFVec3f	bboxSize	-1 -1 -1	Velikost pomocné obálky
eventIn	MFNode	addChildren		
eventIn	MFNode	removeChildren		

}

Při tvorbě modelu jsem používal tento uzel hlavně pro spojování jednotlivých modelů do jednoho, nebo posunutí jednotlivých částí modelu vedle sebe.

2.4.2.13 Shape (Zobrazitelný objekt)

Provázání definice geometrického tvaru a vlastností povrchu jednoho objektu.

Shape {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFNode	appearance	NULL	Vzhled povrchu objektu
exposedField	SFNode	geometry	NULL	Geometrie objektu

}

2.4.2.14 Appearance (Vzhled povrchu)

Přiřazení barevných vlastností a textury, případně jejich kombinace, geometrickému objektu, který je sourozencem tohoto uzlu.

Appearance {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFNode	material	NULL	Barevné vlastnosti povrchu
exposedField	SFNode	texture	NULL	Textura na povrchu
exposedField	SFNode	textureTransform	NULL	Umístění a natočení textury

}

Důležitý uzel pro nanesení textury povrchu a definování vlastností odrazu světla od povrchu, které se zadávají v parametru `material` uzlem `Material`.

2.4.2.15 Group (Skupina)

Sdružení více uzlů do jednoho stromu.

Group {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	MFNode	children	[]	Seznam potomků
field	SFVec3f	bboxCenter	0 0 0	Souřadnice středu pomocné obálky ve tvaru kvádru
field	SFVec3f	bboxSize	-1 -1 -1	Definice délky stran pomocné obálky ve tvaru kvádru.
eventIn	MFNode	addChildren		
eventIn	MFNode	removeChildren		

}

2.4.2.16 LOD (Stupeň detailu)

Variabilní reprezentace jednoho objektu v různých detailech přesnosti.

LOD {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	MFNode	level	[]	Seznam reprezentací modelu s postupně klesající přesností
field	SFVec3f	center	0 0 0	Bod, vůči kterému se měří vzdálenost objektu od avatara
field	MFFloat	range	[]	Rostoucí posloupnost vzdáleností indikujících přepnutí reprezentace

}

Pomocí tohoto uzlu se dá dosáhnout zrychlení zobrazování modelu na počítači. Využívá se toho, že vzdálenější objekty nepotřebují být zobrazeny tak detailně jako objekty v popředí. Jestliže definujeme tentýž objekt s různým stupněm detailu, můžeme potom definovat, v jaké vzdálenosti od avatara má být zobrazen jaký model.

2.4.2.17 Inline (Vložení)

Vložení dalšího světa nebo objektu z vnějšího souboru do aktuálního světa.

Inline {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	MFString	url	[]	Seznam adres, kde se nachází soubor s virtuálním světem
Field	SFVec3f	bboxCenter	0 0 0	Velikost pomocné obálky
Field	SFVec3f	bboxSize	-1 -1 -1	Střed pomocné obálky

}

Tento uzel se používá k vložení virtuálního světa. Vložený soubor musí být platným virtuálním světem. Umožňuje rozdělit větší svět na několik menších souborů, což zrychluje zobrazení virtuálního světa. Svět v aktuálním souboru může být již zobrazen, ale zároveň se načítají z vnějších souborů jeho další části.

2.4.2.18 Material (Barevné vlastnosti)

Nastavení barevných charakteristik povrchu podle schopnosti odrazu barevných složek nepřímého a přímého světla. Nastavení průhlednosti nebo vlastní zářivosti objektu.

Material {

typ parametru	typ dat	název	iniciální hodnota	význam
exposedField	SFFloat	ambientIntensity	0.2	Světlost povrchu získaná odrazem od okolního světa bez ohledu na úhel dopadu
exposedField	SFColor	diffuseColor	0.8 0.8 0.8	Barva povrchu ovlivněná úhlem dopadu světla na povrch
exposedField	SFColor	emissiveColor	0 0 0	Vyzařovaná barva, luminiscence
exposedField	SFFloat	shininess	0.2	Rozmazané nebo ostré odlesky od přímých zdrojů světla
exposedField	SFColor	specularColor	0 0 0	Barva světla odraženého od přímých zdrojů světla
exposedField	SFFloat	transparency	0	Průhlednost

}

Nemá smysl nastavovat všechny parametry najednou. Barva povrchu je ovlivňována všemi světelnými zdroji a mlhou.

3 Možnosti zobrazení map pomocí VRML

Jak již bylo zmíněno v předchozí kapitole, základními stavebními kameny modelu terénu ve VRML jsou uzly `ElevationGrid` (Výšková mapa) nebo `IndexedFaceSet` (Množina ploch).

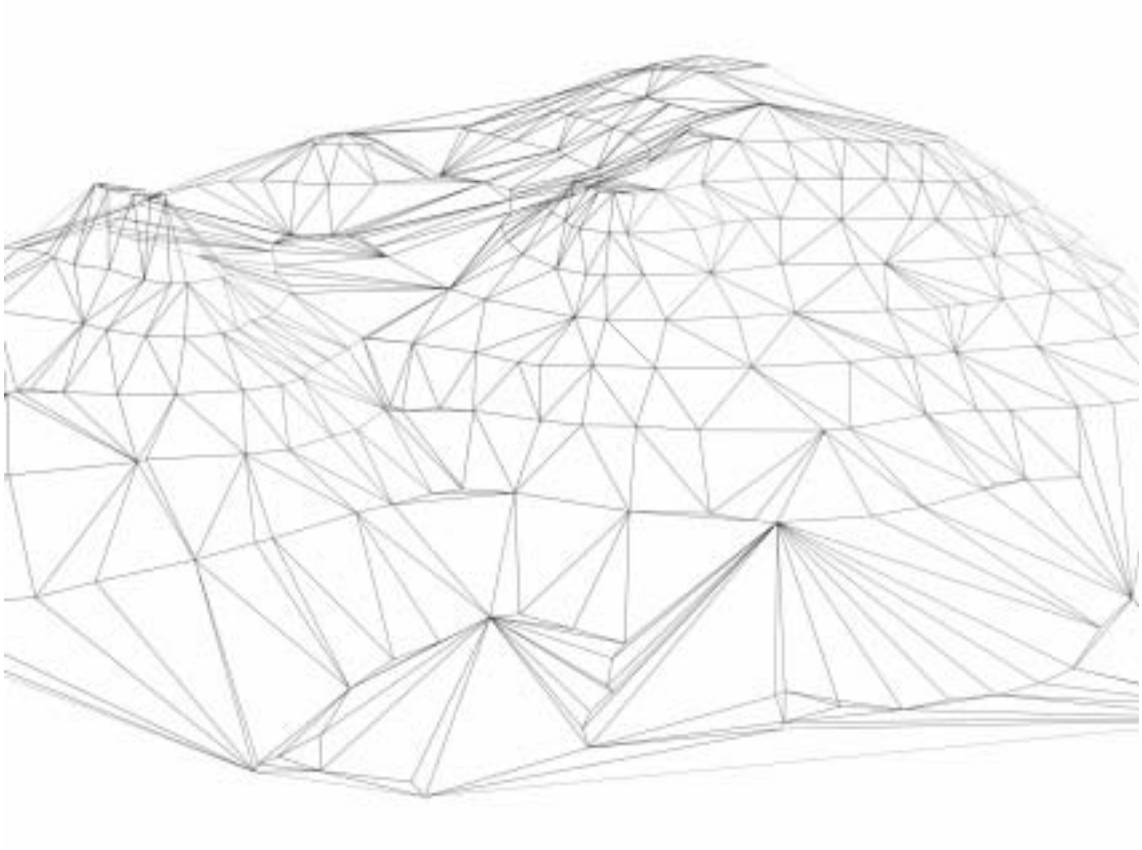
3.1 Uzel IndexedFaceSet

Tvoří model známý jako polyedrický model terénu, který je tvořen rovinnými ploškami trojúhelníkového tvaru s tím, že uzel `IndexedFaceSet` dovoluje tvořit nejen trojúhelníky, ale i obecnější n-úhelníky, které mohou, ale nemusí být rovinné. Každá plocha z množiny je definována zapsáním pořadových čísel vrcholů. Jestliže je jeden vrchol sdílen více plochami, stačí ho zapsat jen jednou a zapamatovat si jeho pořadí v seznamu vrcholů. Plochy se potom zadávají v parametru `coordIndex` jako indexy prostorových souřadnic v parametru `texCoord` zakončené „-1“. Pořadí vrcholů v parametru `coordIndex` se zadává proti směru hodinových ručiček, čímž se definuje orientace ploch. Ta hraje roli při zobrazení plochy, protože plocha je implicitně chápána jako jednostranná.

Dalším důležitým krokem při zobrazení modelu je nanášení textury na jeho povrch. Jelikož implicitní nanesení textury mapy na povrch terénu není správné, je nutné, aby byl definován parametr `texCoord`, ve kterém se definuje uzel `TextureCoordinate` se seznamem souřadnic. Tyto rovinné souřadnice vztažené k nanášené bitmapě potom odpovídají prostorovým souřadnicím modelu. Jestliže pořadí bodů uvedených v uzlu `TextureCoordinate` odpovídá pořadí bodů modelu uvedenému v uzlu `Coordinate`, není potřeba definovat posloupnost v parametru `texCoordIndex` (který má obdobnou strukturu jako parametr `coordIndex`).

Zajímavý parametr, který je výhodné použít, je parametr `creaseAngle`. Je to úhel sevřený mezi dvěma sousedními plochami, který určuje ostrou hranu nebo naopak hladké napojení těchto ploch. Jestliže definujeme tento úhel, model pak vypadá daleko realističtěji.

Většinu dalších parametrů v uzlu `IndexedFaceSet` (jako např.: `color`, `normal`, `ccw`, `colorIndex`, `colorPerVertex`, `convex`, `normalIndex`, `normalPerVertex`, `solid`) není třeba pro tvorbu modelů terénu definovat.

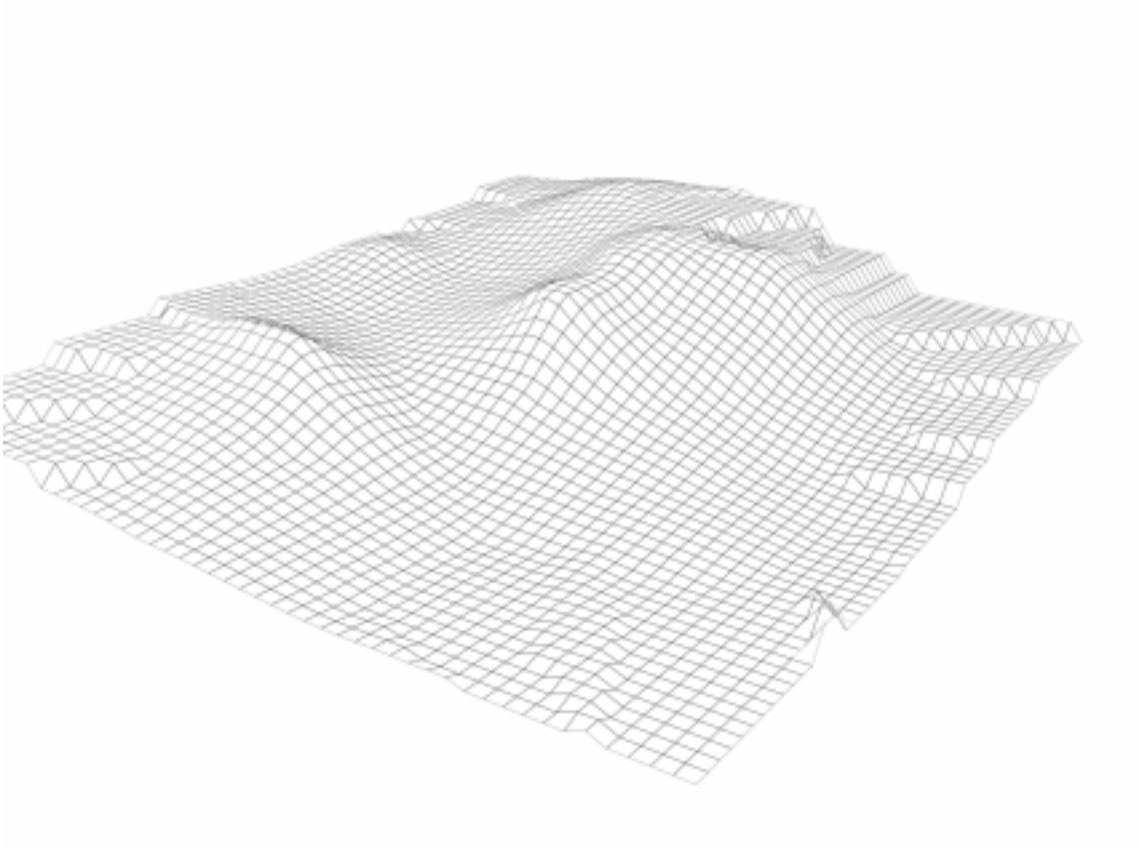


Obr. 3.1 Model vytvořený uzlem `IndexedFaceSet`.

3.2 Uzel ElevationGrid

Tento uzel je speciálně definován pro tvorbu krajiny ve VRML. Vytváří model, známý jako rastrový model terénu, definovaný pomocí pravoúhlé sítě vrcholů s příslušnými výškami. Mapa je umístěna do roviny XZ. Počet vrcholů sítě je definován v parametrech `xDimension` a `zDimension`. Vzdálenost vrcholů v ose X a Z se definuje v parametrech `xSpacing` a `zSpacing`. Seznam výšek, reprezentující souřadnice Y v jednotlivých vrcholech sítě, je zapsán v parametru `height`. Tento seznam je uspořádán po řádcích. V jednom řádku je proto tolik hodnot, kolik udává hodnota zadaná v parametru `xDimension`. Řádky se zapisují postupně ve směru osy Z.

Nanášení textury na výškovou mapu je mnohem jednodušší než u množiny ploch. Před nanášením textury je pouze nutné texturu svisle převrátit. Tato úprava je důležitá kvůli způsobu zadávání bodů výškové mapy a kvůli umístění počátečního bodu textury do levého horního bodu, zatímco vztažný bod obrázku je ve skutečnosti v levém dolním rohu. To lze udělat buď převrácením připraveného obrázku, anebo jednodušeji pomocí transformace textury v uzlu `TextureTransform` nastavením parametru `scale` 1 -1.



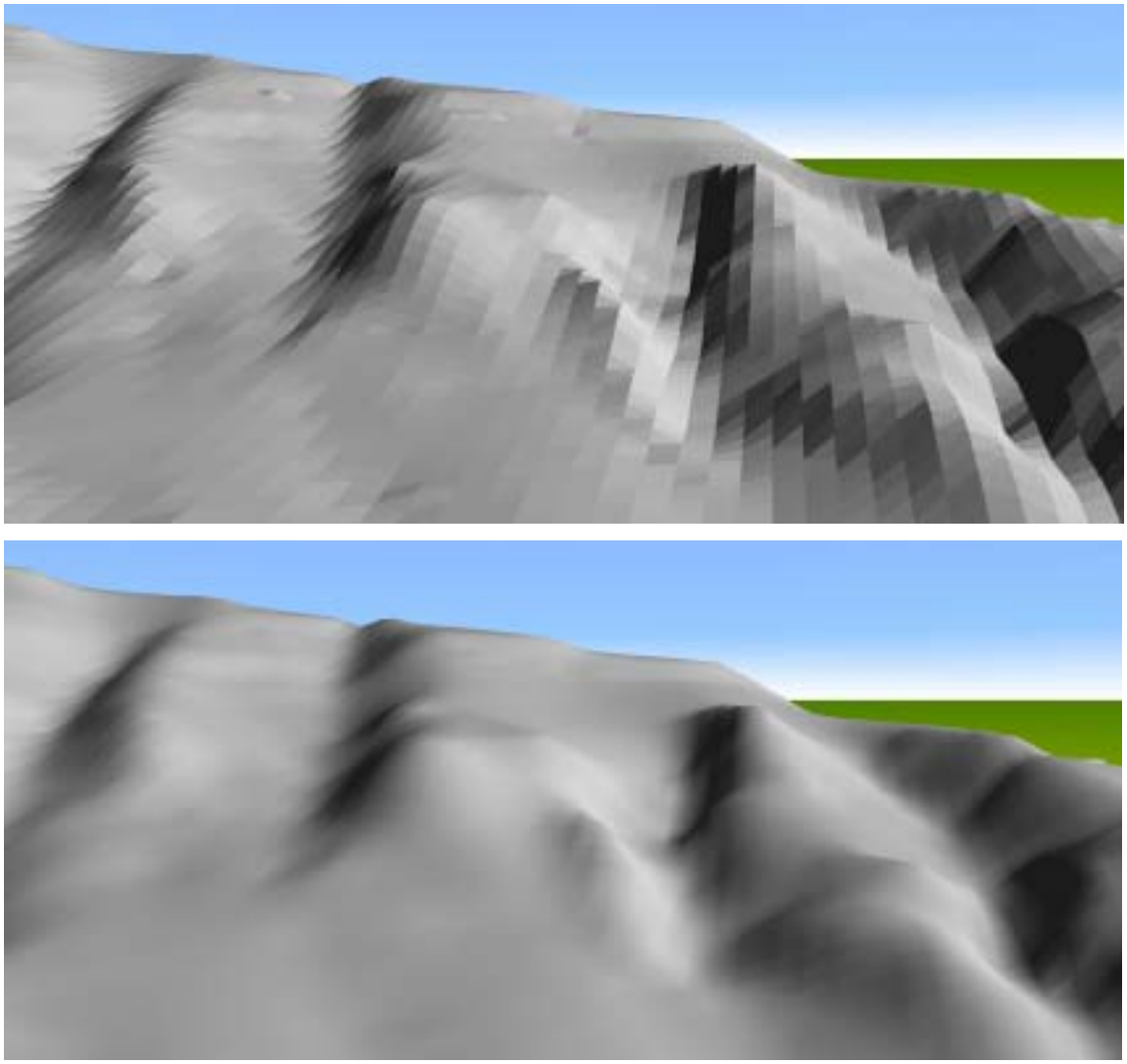
Obr. 3.2 Model vytvořený uzlem `ElevationGrid`.

3.3 Výhody a nevýhody jednotlivých typů modelů

Na závěr této kapitoly bych chtěl shrnout hlavní výhody a nevýhody zobrazení modelů terénu těmito uzly.

1. Pomocí uzlu `IndexedFaceSet` lze zobrazit mapu jakéhokoli tvaru. Pomocí uzlu `ElevationGrid` lze zobrazit pouze mapu tvaru obdélníkového.
2. Uzel `IndexedFaceSet` může zobrazit jakýkoli tvar plochy, např. převis, což u uzlu `ElevationGrid` nelze.
3. Uzel `ElevationGrid` používá pro popis jak složitějšího, tak jednoduchého terénu stejné množství dat. Velikost souboru popisujícího členitější terén bude stejná jako velikost souboru popisujícího rovinu.
4. Uzel `ElevationGrid` má jednodušší nanášení textury - nemusí se definovat souřadnice textury. V uzlu `IndexedFaceSet` se musí vždy definovat souřadnice textury pro správné nanesení mapy na model.

5. Model tvořený uzlem `ElevationGrid` se snadněji generalizuje.



Obr.3.3 Model bez parametru `creaseAngle` (nahoře). Model s nastavením parametru `creaseAngle` na úhel 2 rad (dole).

3.4 Iluze prostoru

3.4.1 Světlo

Aby byl prostorový model vůbec vidět, je třeba ve virtuálním světe definovat nějaké světlo. VRML má definováno několik druhů světla. Přehled je uveden v tabulce

uzel	popis
`NavigationInfo`	čelní svítilna avatara
`DirectionalLight`	zdroj rovnoběžných paprsků

uzel	popis
`PointLight`	bodový zdroj
`SpotLight`	reflektor, směrový zdroj

Pro vytvoření iluze virtuální krajiny však stačí jeden bodový zdroj světla, který nám nahradí slunce. Z tohoto důvodu se nebudeme zabývat zbývajícími druhy světel. Počet světel ve virtuálním světě není omezen, ale je zbytečné nebo spíše škodlivé definovat světel více, protože bychom mohli ztratit velice pěkné stínování krajiny.

Zdroj světla je definován polohou (`location`), dosahem paprsků (`radius`) a barvou paprsků (`color`) o dané intenzitě (`intensity`). Světelný zdroj můžeme vypnout (`of`) nebo zapnout (`on`) a lze zadat, do jaké míry přispívá k celkovému projasnění celého virtuálního světa (`ambientIntensity`). Čím jasnější je svět, tím jsou jasnější plochy odvrácené od zdrojů světla.

Zdroj světla ve virtuálním světě není vidět. Jestliže chceme ve virtuálním světě vidět „virtuální sluníčko“, musíme ho definovat např. jako zářící kouli. Jinou důležitou vlastností světelného zdroje ve virtuální realitě je, že žádný objekt nemůže zastavit světelný paprsek na jeho dráze a zaclonit tak objekt jiný. V praxi to pak vypadá tak, že tělesa nevrhají stíny.

3.4.2 Pozadí

Další vlastnost, která dodá virtuální krajině větší reálnost, je pozadí (uzel `Background`). VRML poskytuje několik možností, jak definovat pozadí. Nejjednodušší možnost je vyplnit pozadí jedinou barvou, tato možnost ale není vhodná pro zobrazení krajiny. Lepší je možnost použít škálu barev měnících se s výškou nad horizontem.

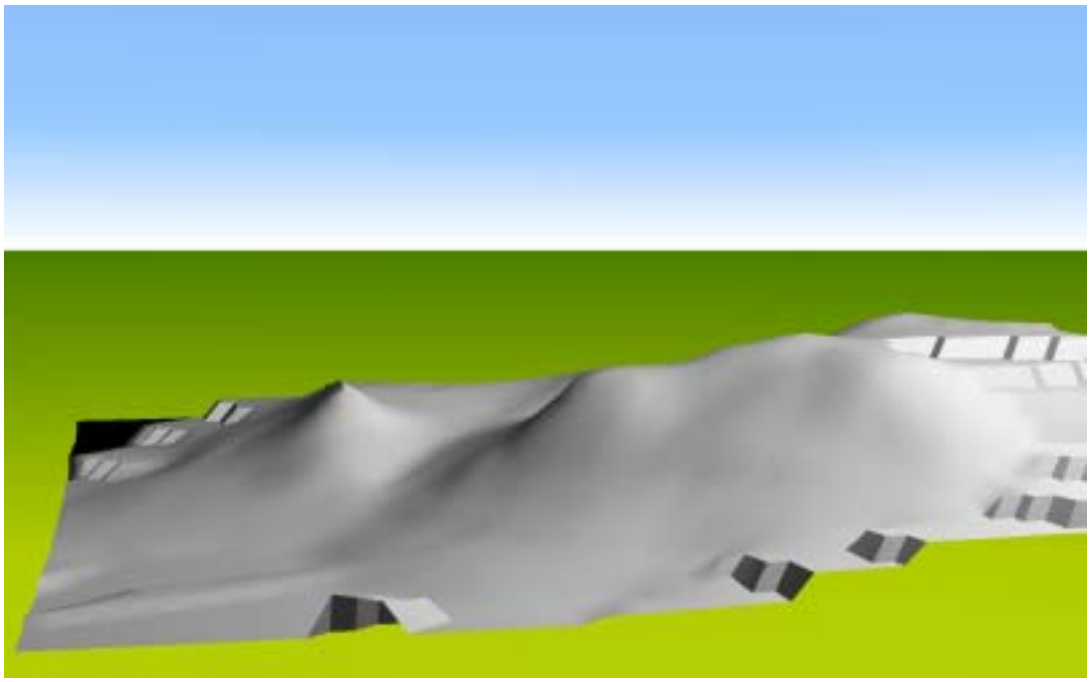
Mezi základní vlastnosti pozadí patří to, že je vykreslováno bez ohledu na světelné zdroje, mlhu či dohled avatara. Pozadí je samostatný prvek, který je vykreslen jako první, a poté jsou před něj umisťovány virtuální objekty. K pozadí se nemůžeme přiblížit. Můžeme si ho představit jako těleso, které obklopuje celý virtuální svět, kde se avatar vždy nachází uprostřed.

Uzel `Background` definuje dvě tělesa, která mohou obklopovat virtuální svět – kouli a krychli. Každé těleso má jiné využití. Koule se používá pro jednodušší pozadí, na kterém pouze přecházejí barevné odstíny ve vodorovných pruzích. Krychle se používá jako podklad pro panoramatické obrázky.

3.4.2.1 Pozadí v podobě koule

U koule se rozlišuje obloha (*sky*) a zem (*ground*). Zem nemusí být definována, obloha potom pokrývá celou kouli, což může vypadat jako vesmírný prostor. Je-li definována i zem, překryje její barva dolní polokouli oblohy.

Barevné přechody se zadávají jako posloupnost barev v parametru `skyColor`, resp. `groundColor` a ke každé barvě je definován úhel v parametru `skyAngle`, resp. `groundAngle`. Úhly jsou odměřovány pro oblohu od nadhlavníku v intervalu 0 – π a pro zem od nadiru v intervalu 0 – π/2. Počet barev musí být vždy o jednu vyšší než počet úhlů, neboť první barva v seznamu se použije pro nadhlavník, resp. nadir, tj. pro úhel rovný nule.



Obr. 3.4 Pozadí v podobě koule.

3.4.2.2 Pozadí v podobě krychle

Daleko lepšího vzhledu docílíme použitím panoramatických obrázků nalepených na pomyslné krychli. Problém je získat kvalitní panoramatické obrázky, které na sebe navazují. Ještě větším problémem, než návaznost obrázků na sebe, je při zobrazení krajiny rozumná návaznost obrázku na model. Nejde totiž o přímou návaznost, obrázky v tomto případě dokreslují horizont, a je proto třeba, aby tomuto faktu odpovídaly.

Obr. 3.5 Pozadí v podobě krychle.

3.4.2.3 Kombinace obou druhů pozadí

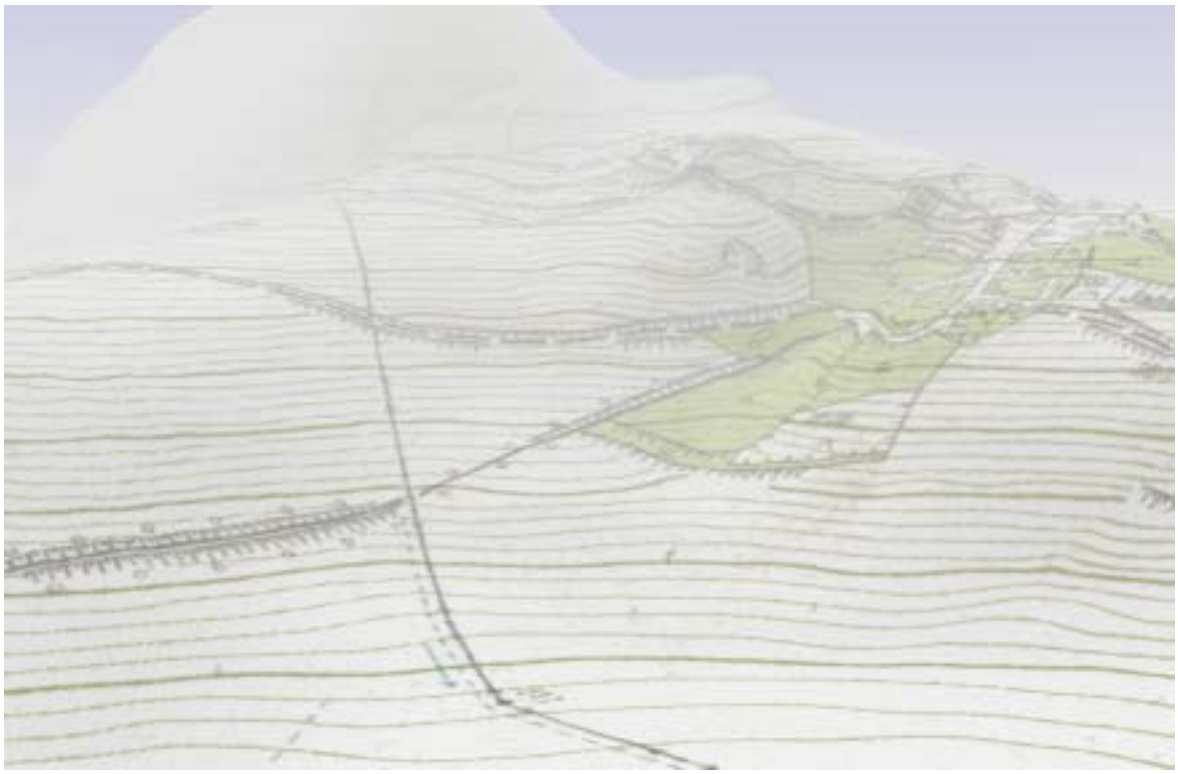
Poslední možností je kombinovat oba druhy pozadí. V tomto případě je krychle vepsána do koule. Smysl spočívá v tom, že můžeme vynechat některé části krychle, např. horní a dolní stěnu. Místo vynechaných stěn bude vidět pozadí definované na kouli.

Další možností je použití poloprůhledných obrázků, jimiž prosvítají barvy pozadí na kouli.

3.4.3 Mlha

Pro dokreslení iluze lze ve virtuálním světě definovat mlhu (uzel `Fog`). Má tři parametry, a to barvu, způsob houstnutí a vzdálenost, za kterou již objekty v mlze nejsou vidět. Mlha se ve virtuálním světě projeví smícháním s barvou objektů. Množství původní barvy, kterou si objekt uchová, závisí na vzdálenosti. U avatara mají objekty pouze svou barvu, dále od avatara se původní barva čím dál tím více mísí s barvou mlhy.

Při současném použití pozadí a mlhy je vhodné barvu pozadí nastavit v oblasti horizontu na barvu mlhy. Další barva nad horizontem pak jakoby vystupuje z mlhy.



Obr. 3.6 Krajina v mlze.

4 Zrychlení zobrazování světů ve VRML

Se vzrůstající rychlostí počítačů se zvyšuje i rychlost zobrazení virtuálních světů a je možné zobrazovat stále více objektů. Zobrazení modelu krajiny ale klade extrémní nároky na počítač z důvodů velkého množství dat. Aby se návštěvník mohl pohybovat po krajině virtuálního světa a měl dojem plynulého pohybu, měl by počítač vykreslit 24 snímků za sekundu. Mezní rychlost, kdy pohyb není ještě příliš trhaný, je 10 snímků za sekundu. Rychlost zobrazení lze ovlivnit jednak hardwarově, ale především optimalizací souboru s VRML daty.

4.1 Hardwarové zrychlení

Koupit si kvůli virtuální realitě nejvýkonnější počítač je sice pěkné řešení, ale pro většinu lidí řešení nedostupné. Řekl bych, že v současnosti vyráběné počítače střední cenové úrovně už dokážou virtuální realitu zobrazit velice slušně. Je nutné pamatovat na několik věcí. Nejdůležitější pro zobrazení VRML v počítači je grafická karta, nebo spíše grafický 3D akcelerátor. Dnes má většina grafických karet již tento 3D akcelerátor integrován. 3D akcelerátor zvyšuje rychlost zobrazení 2-5x. Prohlížeč VRML komunikuje s akcelerátorem prostřednictvím rozhraní DirectX nebo OpenGL. Je proto nutné zapnout v prohlížeči VRML zobrazování pomocí jednoho z těchto rozhraní. Aby pracoval grafický systém co nejrychleji, je dobré mít nainstalovány nejnovější ovladače grafické karty, popřípadě 3D akcelerátoru.

4.2 Optimalizace VRML souboru

Optimalizace se dá rozdělit do dvou kategorií. První je zrychlení načítání souborů s popisem virtuálního světa po síti a druhá je zrychlení zobrazování a pohybu. Některé prvky optimalizace ovšem zasahují do obou kategorií.

4.2.1 Načítání souborů

Snahou je, aby čas od chvíle, kdy prohlížeč dostane pokyn k zobrazení virtuálního světa, do chvíle, kdy se objeví první objekt, byl co nejkratší. V případě zobrazení virtuálních modelů terénu toho lze docílit následujícími způsoby:

4.2.1.1 Rozdělení světa do několika souborů

Jestliže návštěvník vstoupí do virtuálního světa vždy v jednom místě, je vhodné definovat v hlavním souboru jen objekty v jeho nejbližším okolí a vzdálenější objekty vkládat pomocí uzlu `Inline`. Další objekty jsou potom postupně zobrazovány a návštěvník se může zatím věnovat objektům v jeho bezprostřední blízkosti.

Při zobrazování map toho lze využít tak, že každý mapový list je uložen v samostatném souboru a pro zobrazení celku se pak vkládají jednotlivé listy do hlavního souboru prostřednictvím uzlu `Inline`.

4.2.1.2 Definování pomocných obálek

Pokud definujeme velikost a umístění ohraničujícího kvádru (*bbox*) pro uzly `Inline`, `Transform` a další, prohlížeč nemusí při načítání vyhodnocovat skutečné rozměry potomků těchto uzlů. Správně definovaná obálka je výhodná i pro návštěvníka virtuálního světa. Pokud prohlížeč ještě nenačetl potřebný objekt, může na jeho místě dočasně zobrazit hrany kvádru reprezentujícího obálku. Návštěvník je tak informován, že v daném místě brzy přibude do scény objekt dané velikosti.

4.2.1.3 Velikost textur

Při zobrazení prostorových modelů map je důležité určit správné rozlišení textury, která se nanáší na model. Jestliže bude mít textura velké rozlišení, bude vypadat výsledný model lépe, ale velikost souboru s texturou bude značná a rychlost zobrazení malá. Naproti tomu bude-li rozlišení malé, rychlost zobrazení a pohybu bude velká, velikost souboru s texturou malá, ale např. čitelnost nápisů na mapě bude špatná, ne-li nemožná. Proto je vhodné volit určitý kompromis mezi velikostí a čitelností a také použít uzel `LOD` s vícenásobnou prezentací textury.

Jestliže zmenšíme rozlišení obrázku na polovinu, zvýší se rychlost zobrazení o 20-50%.

4.2.1.4 Používání komprese

Každý hotový soubor s virtuálním světem by měl být nakonec zkomprimován programem *gzip*. Soubor s takovouto kompresí dokáže prohlížeč sám rozeznat a také dekomprimovat. Program *gzip* je schopen soubor zmenšit více než dvakrát, a tak významně přispět ke zrychlení načítání.

4.2.2 Rychlost zobrazování a pohybu

Tyto optimalizace mají jako hlavní úkol co nejvíce zrychlit zobrazování, aby byl pohyb po virtuálním světy plynulý.

4.2.2.1 Využití vícenásobné reprezentace

Pro každý model mapy je dobré definovat několik reprezentací s různým stupněm generalizace zapsaným v uzlu `LOD` (*Level Of Detail*). Vložíme-li detailnější reprezentace uzlem `Inline`, prohlížeč může odložit jejich přečtení až do doby, kdy má více času.

Definování více reprezentací sice zabere více času a zvyšuje množství přenášených dat, ale přináší urychlení zobrazení. Tento uzel je důležitý hlavně při zobrazení větších virtuálních světů.

Uzel `LOD` obsahuje parametr `range`, který definuje vzdálenosti objektu od návštěvníka. Podle vzdálenosti potom prohlížeč přepíná jednotlivé reprezentace. Jestliže necháme parametr `range` prázdný, potom bude prohlížeč přepínat jednotlivé reprezentace automaticky podle výkonu počítače. Automatické přepínání ale může na pomalejších počítačích znamenat, že prohlížeč bude vždy zobrazovat jen nejjednodušší model a návštěvník nikdy neuvidí detaily.

Asi nejlepší je kompromis v podobě několika uzlů LOD v hierarchickém uspořádání. Nejvyšší uzel definuje jedinou přepínací vzdálenost. Pro objekt nacházející se před touto hranicí se vybere jeden ze dvou lepších modelů. Tato volba závisí na prohlížeči. Obdobně jako u dvojice lepších modelů se postupuje u dvojice horších modelů.

```
LOD {
  range [ 30 ] #jednoduché rozdělení na blízké a vzdálené
               #modely
  level [
    LOD {  #automatický výběr ze dvou modelů pro pohled
           #zblízka
      level [
        Transform {  #model kužel...
        }
        Shape {      #model čtyřboký jehlan...
        }
```

```
      ]
    }
    LOD { #automatický výběr ze dvou modelů pro pohled zdálky
      level [
        Billboard {  #model trojúhelník na billboardu...
        }
        Group {}     #model nic
      ]
    }
  ]
}
```

Program 4.1 Kombinace automatického a doporučeného přepínání mezi reprezentacemi v uzlu `LOD`.

4.2.2.2 Šetření počtem ploch

Je jasné, že čím bude prohlížeč zobrazovat najednou méně plošek, tím bude zobrazení rychlejší. Šetřit počtem ploch při zobrazení modelu terénu je problém zvláště u větších oblastí, ale je možné vzdálenější oblasti silně generalizovat, a s použitím vícenásobné reprezentace tak ušetřit mnoho ploch.

4.2.2.3 Správná definice plošných objektů

K urychlení zobrazení také přispívá správná definice uzlů `ElevationGrid` a `IndexedFaceSet`. Tyto uzly by měly splňovat implicitní nastavení parametru `solid`, `convex` a `ccw` na hodnotu TRUE, tj. plochy by měly být zobrazovány jednostranně, měly by být konvexní a přivrácená strana plochy by měla být vidět shora.

Velkou úsporou je definování plošek výhradně jako trojúhelníků. Rozdělení plošek na trojúhelníky sice zvyšuje velikost souboru, ale celkově znamená úsporu času. Plochy obecného tvaru totiž prohlížeč vždy převádí na trojúhelníky. Tento postup je časově náročný a nemusí vždy správně modelovat požadovanou plochu.

5 Možnosti zobrazení a využití map ve VRML

5.1 Rozhraní VRML EAI

Jazyk VRML umožňuje pomocí uzlu `Script` vyvolávat externí funkce zapsané v několika programovacích jazycích, avšak virtuální svět tvoří spolu s těmito funkcemi uzavřený celek, do něhož není možné z vnějšku zasahovat. Tento svět je posléze uživateli prezentován prohlížečem nejčastěji na stránce html. Spolupráce s dalšími prvky, které se na stránkách html mohou objevovat, je v základní podobě jazyka VRML nemožná. Proto bylo vytvořeno rozhraní, které umožňuje předávat data mezi světem VRML a externí aplikací. Toto rozhraní se nazývá EAI (*External Authoring Interface*), a jeho pomocí lze např. psát program pro editaci VRML světů, přičemž o zobrazení se stará prohlížeč WWW. Ačkoli toto rozhraní může být obecně poskytováno pro různé jazyky, v současné době existuje pouze pro jazyk Java.

Rozhraní EAI tedy umožňuje ovládat VRML svět prostřednictvím html stránky, což lze využít např. při prohlídce, editaci nebo změně atributů virtuálního světa. Pro demonstraci jsem vytvořil java applet, který mění výškové zkreslení mapy.

5.2 Možnosti využití prostorových modelů map ve VRML

Prostorově zobrazené mapy sice vypadají velmi pěkně, ale jejich praktické využití jako samostatného produktu není nijak velké. Samostatně mohou sloužit snad jen jako prezentace nebo reklama na internetu. Proto si myslím, že uplatnění takových map bude sloužit spíše jako podklad pro další virtuální modely, a to jednak modely statické, nebo dynamické. Statický model by mohl sloužit jako virtuální procházka po krajině s vyznačenou turistickou značkou a zároveň rozhledem po krajině. Dynamický model může například zobrazit model záplavy v krajině se zvedající se hladinou vody. Dále by se modely terénu mohly objevit například v nabídkovém katalogu cestovních kanceláří. Kromě toho existuje možnost využití modelů také v oblasti GIS.

Samozřejmě teprve čas ukáže, jak se bude VRML dále vyvíjet a jaké budou požadavky na něj. S rostoucím výkonem počítačů by mohly vzniknout mapové servery s 3D modely map obdobně jako existují servery s klasickými 2D mapami.

6 Návrh a tvorba automatizovaného procesu pro generování prostorových map

Proces generování prostorového modelu map bych rozdělil do několika částí:

- získání výškové složky z mapy
- tvorba VRML modelů
- skládání jednotlivých modelů vedle sebe do větších celků a vytváření méně podrobných modelů pro využití v uzlu `LOD`.

Je pravda, že existují programy, které řeší dílčí úlohy, ale komplexně se tímto problémem ještě nikdo nezabýval. Existují vektorizační programy, např. *Adobe Streamline*. 3D modely lze vytvářet např. ve *3D Studiu* s následným exportem do VRML. Dokonce i *MicroStation*, který používá mnoho geodetů, již obsahuje export do VRML, což ukazuje, jak se virtuální realita rozvíjí. Tyto programy jsou ale příliš univerzální a nepodporují všechny možnosti VRML, rovněž optimalizace pro prohlížení na internetu je většinou mizivá. Zkoušel jsem například exportovat z *MicroStationu* poměrně složitý model. Export proběhl celkem bez problémů, ale výsledný soubor byl tak velký, že ani nešel pořádně zobrazit. S vektorizací to bylo obdobné. Vektorizační program sice některé čáry rozpozná, ale vznikne jen soubor krátkých čar, protože vrstevnice jsou často přerušované. Proto jsem přistoupil k vývoji vlastních programů, které jsou „šité na míru“ a které mohu podle svých představ upravovat tak, abych dosáhl co nejlepšího výsledku.

K splnění úkolů jsem napsal čtyři rozsáhlejší programové moduly v jazyce C++. Použil jsem vývojové prostředí *C++ Builder 5*, které ovládám nejlépe a už jsem v něm dříve pracoval. *C++ Builder* nabízí velice kvalitní vizuální vývojové prostředí pro návrh a tvorbu graficky orientovaných aplikací pod operačním systémem MS Windows. Celková velikost zdrojového kódu, který jsem napsal, je zhruba 270 kB.

První program nazvaný *Filtr* získává výškovou složku z naskenované mapy. Rozpoznává ji podle barvy a vytváří dvoubarevný obraz, jenž obsahuje už jen výškovou složku a který se potom použije pro vektorizaci. Druhý program s názvem *Vektorizator* vektorizuje vrstevnice a následně se je snaží pospojovat. Ve třetím programu pojmenovaném *Editor* se výsledky vektorizace opravují, zadává se výška vrstevnic a dá se vytvořit jednoduchý model VRML. Čtvrtý program nazvaný *Generator* pak vytváří celkový model VRML, který se skládá z jednotlivých modelů.

Tento postup jsem použil pro starší mapy nebo mapy, které jsem měl k dispozici pouze v papírové formě. U dat, která máme již ve vektorové formě (např. ZABAGED), stačí exportovat tato data do programu *Editor* a dále postupovat jako u mapy analogové.

Podrobný postup je popsán v následujících kapitolách.

6.1 Skenování map

K dispozici jsem měl 6 listů Základní mapy ČR v měřítku 1:10 000, ze kterých jsem vytvářel prostorový model větší oblasti, a 3 orientační mapy pro tvorbu menších modelů. V průběhu vývoje programu pro vektorizaci jsem zjistil, že při skenování je zapotřebí rozlišení minimálně 600 DPI, aby rozpoznání vrstevnic bylo úspěšné. Ke skenování jsem používal skener formátu A4, takže jsem musel každý mapový list skenovat po čtvrtinách a v počítači jsem ho spojoval. Zde jsem narazil na menší problém, protože se mi nepodařilo mapy naskenovat stejně a jednotlivé skenované části vykazovaly odlišnosti v barevných odstínech, což pak vadilo při filtrování výškové složky. Další zpracování jsem dělal po celých mapových listech. Soubory s naskenovanými mapovými listy jsem ukládal ve formátu bmp o velikost asi 300 MB, které jsem dále použil pro zpracování v programu *Filtr*. Zpracování velkých celků najednou mělo své výhody i nevýhody. Spojování částí mapových listů zabere dost času, protože počítač musí pracovat s velkým množstvím dat, na druhou stranu je stejně nutné části mapy pospojovat, protože by ve výsledném modelu nemusely tyto části navazovat. Ideální by bylo mít k dispozici skener formátu A2 pro skenování celé mapy najednou.

6.2 Filtrace výškové složky mapy

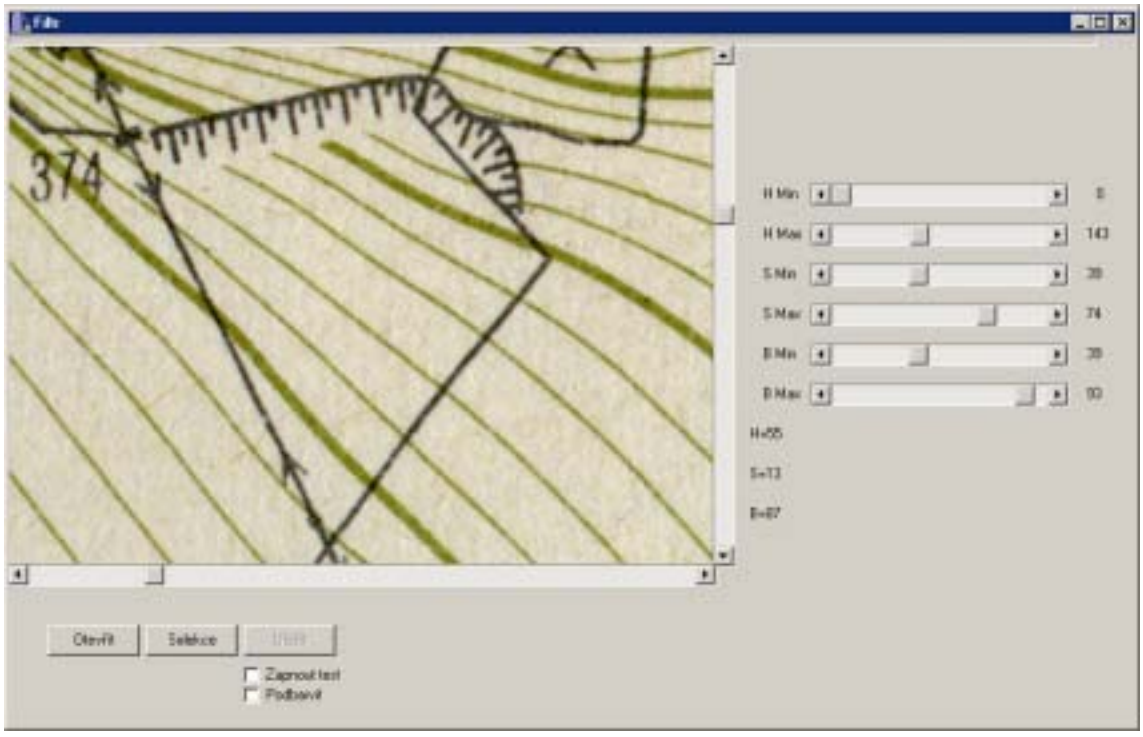
Pro tento úkol jsem již napsal první program *Filtr*, protože mě neuspokojovaly výsledky jiných programů a chtěl jsem mít plnou kontrolu nad touto operací.

K filtraci výškové složky jsem použil jednoduchou metodu, kdy jsem procházel postupně jednotlivé pixely obrázku a testoval je podle zadaného kritéria. Tak jsem vytvořil černobílý obraz, který obsahoval jen výškovou složku. Jako kritérium program testuje, zda je daný obrazový bod v zadaném rozsahu barev. Barevný interval se zadává ve třech složkách barevného modelu HSB (hue - odstín, saturation - sytost, brightness - jas), který se mi zdál nejvhodnější. Vstupní obrázek je v barevném modelu RGB, a proto jsem musel naprogramovat převod RGB do HSB. Za tímto účelem jsem vytvořil dvě třídy `RGBColor` a `HLSColor` a funkci `RGB_TO_HLS`.

Po spuštění programu *Filtr* je uživatel vyzván k otevření souboru s obrázkem, který se má filtrovat. Určení intervalu barev zadává uživatel pomocí sady posuvníků v pravé polovině okna. Aby mohl uživatel vybrat optimální interval, umožňuje program zobrazit náhled výsledného obrazu (zaškrtávací políčko Zapnout test) nebo obrazu původního v kombinaci s výsledným (zaškrtávací políčko Podbarvit + políčko Zapnout test). Dále umožňuje zobrazit hodnoty barevných složek, podle kterých se pak může upravit barevný interval, a to kliknutím na konkrétní pixel na obrázku. Jestliže jsou zaškrtnuta výše zmíněná políčka, pak při podržení tlačítka myši je zobrazena hodnota aktuálně zobrazeného pixelu a při uvolnění tlačítka je zobrazena hodnota pixelu v původním obrázku.

Jestliže je uživatel spokojen s filtrací v náhledu, spustí konečnou filtraci tlačítkem Selekce a program provede filtraci celého obrázku. Pak už zbývá obrázek jen uložit do souboru tlačítkem Uložit (výsledný soubor má příponu bis).

Pro zmenšení velikosti výsledného souboru se obrázek ukládá jako bitová mapa, kdy jeden bit reprezentuje jeden pixel, takže je výsledný soubor oproti vstupnímu 24x menší (vstupní obrázek je v barevném modelu RGB 3 x 8 bitů na pixel). Program si nastavení barevných intervalů zaznamená do svého inicializačního souboru. To je výhodné, jestliže zpracováváme více map podobných barevných odstínů.



Obr. 6.1 Program první: *Filtr*.

6.3 Příprava pro vektorizaci

Po selekci vrstevnic následuje další krok, a to vektorizace. Dříve, než bude mapa vektorizována, je nutné opravit některé chyby v obraze a také celkově obraz upravit. Když jsem přemýšlel, jak udělat nejjednodušší a nejefektivnější program pro vektorizaci vrstevnic, musel jsem zjednodušit obraz tak, aby výsledek vektorizace za něco stál. Proto jsem přistoupil ke zjednodušení, kdy se vektorizují pouze vrstevnice zdůrazněné, které se kreslí v mapě silnější čarou.

Důvodů je hned několik:

- jednodušší zpracování – menší množství dat
- mapa občas obsahuje doplňující vrstevnice, které jsou kresleny čárkovaně a mátly by program při spojování vrstevnic
- ve svažitějším terénu, kde by se základní vrstevnice slily dohromady, bývají naopak vynechány
- mapa obsahuje technické šrafy, další zdroj možných chyb při rozpoznání vrstevnic; ty jsou kresleny stejně silnou čarou jako základní vrstevnice
- zdůrazněné vrstevnice jsou silnější a lépe se rozpoznávají

Z těchto důvodů jsem se rozhodl zanedbat základní vrstevnice a generovat prostorový model jen na základě vrstevnic zdůrazněných. V méně členitém terénu tento způsob plně vyhovuje a v terénu členitějším je možno v následujícím kroku zpracování doplnit další výškové údaje. Dalším faktorem, který zvýrazňuje chyby způsobené tímto zanedbáním, je velikost výškového zkreslení. U malého výškového zkreslení se tyto chyby prakticky neprojeví, ale při větším zkreslení je tomu naopak.

Vraťme se nyní k opravě a úpravě obrazu. Oprava obrazu spočívá pouze v doplnění chybějících pixelů, které čáry obsahují. Použil jsem jednoduchý postup, kdy jsem testoval okolí všech obrazových bodů, které jsou bílé, kolik mají ve svém přímém okolí černých pixelů. Jestliže jsou tři, maximálně čtyři, bod je změněn na černý. Tím se doplní chybějící body v čarách. Samozřejmě tato oprava není příliš dokonalá, ale je rychlá a důležitá pro další úpravy.

Po základní opravě obrazu následuje úprava, kdy se mažou tenké vrstevnice. Pro tento úkol jsem použil opět velice jednoduchý algoritmus, při němž se testují vždy dva sousední pixely, a to se sousedy buď v horizontálním, nebo vertikálním směru. Jestliže jeden z nich je černý a druhý bílý, potom se změní černý pixel na bílý. Tímto postupem dochází k tomu, že jsou mazány všechny okraje čar; ve výsledku tenké čáry zmizí úplně a

silné čáry se změní v tenké. K tomuto algoritmu jsem přidal ještě jeden, který maže čáry silné jeden pixel, a to na základě testu dvou sousedů černého pixelu, které sousedí v jednom směru od něj. Jestliže jsou oba pixely bílé, potom se změní prostřední pixel také na bílý. Pomocí těchto dvou algoritmů je potom dosaženo celkem dobrého výsledku.

Tento krok je vykonáván ve druhém programu, který jsem pojmenoval *Vektorizator*. Po spuštění se nejdříve musí otevřít soubor vytvořený v programu *Filtr*. Otevřený soubor se objeví v okně a lze si ho prohlédnout. V pravé části okna aplikace je sada ovládacích tlačítek. Funkce oprava se spustí tlačítkem téhož názvu. Protože je tato funkce časově náročná, je v dolní části okna umístěn indikátor průběhu (ProgressBar), který dává uživateli informaci o průběhu operace.

Pro úpravu, která odstraní silnější čáry, jsou zde dvě funkční tlačítka. Tlačítko Štíhlení auto je vhodné pro většinu map. Pokud by výsledek této funkce nebyl uspokojivý z důvodů nestandardně širokých nebo naopak tenkých čar, je možné použít tlačítko Štíhlení, které ztenčuje čáry po krocích, na rozdíl od Štíhlení auto, které dělá tyto čtyři kroky najednou. Pomocí tlačítka Štíhlení je tedy možné ztenčovat čáry v malých krocích do té doby, než je většina tenkých čar smazána, ale přitom zesílené čáry (vrstevnice) jsou stále nepřerušované. Pokud by bylo v obraze více tenkých čar, které představují základní vrstevnice, může se stát, že je dále vektorizační funkce rozpozná jako čáry. Tím by vzniklo mnoho chyb, které by bylo třeba v dalším kroku opravovat. Naopak jestliže budou zesílené vrstevnice příliš ztenčené, může se stát, že je vektorizační funkce nerozpozná a budou potom v mapě chybět. Tyto problémy jsou způsobeny složitostí tohoto úkolu, který se už dotýká programování systémů umělé inteligence, které jsou schopny se učit. Dobrým příkladem takového programu jsou např. programy pro rozpoznání písma.

Jistým nedostatkem funkcí v programu *Vektorizator* je, že aplikace nemůže tyto kroky vrátit nazpět a uživatel se musí vždy, když ztenčí čáry příliš, vrátit na začátek a otevřít soubor znovu.

Obr. 6.2 Program druhý *Vektorizator*.

6.4 Vektorizace vrstevnic

Nejobtížnější operací v tomto postupu byla vektorizace. Základní myšlenkou tohoto algoritmu je, že program postupně prochází celý obraz a hledá čáry. Jestliže na nějakou čáru narazí, snaží se ji sledovat a zároveň ji za sebou maže. Tím zabraňuje tomu, aby tutéž vrstevnici nevektorizoval vícekrát.

Hledání čar spočívá v procházení celého obrázku postupně po řádcích a zjišťování, kolik pixelů v určité čtvercové oblasti je černých. Jestliže počet černých pixelů překročí určitou mez, je toto místo pokládáno za část čáry a začíná její procházení. Když potom bude mít dostatečnou délku, uloží se do paměti pro další zpracování.

Zjišťování průběhu čáry spočívá v tom, že si program vybírá ze čtyř směrů, kam jít. Vhodný směr se vybírá podle toho, ve kterém směru je nejvíce černých pixelů. Program si pokaždé vybere nejvhodnější směr a v tomto směru postoupí o jedno políčko, znovu si vybere nejvhodnější směr, a tak postupuje až do té doby, kdy klesne počet černých pixelů v daném místě pod určitou mez. Potom uloží rozpoznanou čáru do paměti.

Tento základní postup je dále doplněn o některé kontroly, např. kontrolu větvení, kdy program zjišťuje, zda se čára nerozvětvuje, což by mohlo znamenat, že čára není vrstevnice, ale výšková šrafa.

Postup vektorizace lze shrnout do těchto základních kroků:

1) Procházíme celý obrázek po jednotlivých pixelech. Jestliže daná oblast u pixelu obsahuje větší množství černých bodů než je daná mez, považujeme toto místo za část čáry a přejdeme na bod 2).
2) Na tomto místě zjistíme směr, ze čtyř možných, ve kterém je nejvíce černých bodů.
3) V tomto směru se posuneme a zároveň v opačném směru smažeme černé body.
4) Opakujeme bod 2) a 3), dokud neklesne počet černých bodů v dané oblasti pod určenou mez.
5) Pokud je čára delší než zadané kritérium, uložíme ji do paměti a pokračujeme v prohledávání od bodu 1)

Pro práci s rozpoznanými čarami jsem vytvořil několik tříd. Protože se předem neví, kolik bude čar a bodů, je nevhodné použít pole. Proto jsem použil pro ukládání čar a bodů kontejneru `vector`. První vytvořená třída je `Hmapa`, která obsahuje popis celé rozpoznané mapy. Obsahuje velikost mapy, dále již zmíněné kontejnery, a to pro ukládaní jednak celých čar (třída `Hcara`) a dále i jednotlivých bodů (třída `Hbod3D`). Pro přístup do těchto datových struktur obsahuje tato třída také metody; dále obsahuje metody pro uložení a načtení dat této třídy do a ze souboru. Třída `Hcara` obsahuje body, které ukládá také do kontejneru třídy `Hbod2D`. Dále obsahuje nadmořskou výšku, která bude při editaci doplněna, počet bodů a informaci, zda je vrstevnice uzavřena. Třída obsahuje metody pro práci s čarou a pro uložení a načtení čáry ze souboru. Další údaj, který se ukládá spolu s těmito čarami, je pravděpodobný směr konců rozpoznaných čar, který potom slouží ke spojování vrstevnic. Další výhodou použití kontejnerů je jednodušší programování editačních funkcí.

Vektorizace se spouští tlačítkem Vektorizace, které je umístěno v pravé části okna aplikace. O průběhu funkce informuje indikátor průběhu jako v programu *Filtr*.

6.5 Spojování vrstevnic

Po vektorizaci je nutné pospojovat jednotlivé části rozpoznaných vrstevnic. To je prováděno rovněž v programu *Vektorizator*. Funkce se spouští tlačítkem Pospojovat. Program se v této funkci snaží správně spojit jednotlivé části rozpoznaných vrstevnic. V této funkci nejprve hledám konce vrstevnic, které jsou velice blízko u sebe, a spojím je. Dále hledám napojení vrstevnic podle jejich pravděpodobného směru; jestliže je v tomto směru vrstevnice, spojím je také. A do třetice hledám dvojice vrstevnic, které se v pravděpodobném směru pokračování protínají. Nakonec ještě dohledám vrstevnice na okraji mapy, jejichž konce zde pravděpodobně začínají.

Tento proces nefunguje vždy spolehlivě; problémy mu dělají zejména tenké vrstevnice, které by neměly být rozpoznány, proces si s nimi neví rady a někdy je napojí na silné vrstevnice. Tyto a jiné chyby se potom dají opravit v programu *Editor*.

Po pospojování již zbývá vrstevnice jen uložit. Program ukládá dva soubory se stejným názvem. Jeden s příponou car, kde se ukládají vlastní čáry, a druhý s příponou kon, kde se ukládají informace, jak jsou jednotlivé vrstevnice pospojované (kvůli dodatečné editaci pospojovaných vrstevnic). Soubory jsou v textovém formátu, a to hlavně z důvodu ladění vyvíjených programů.

6.6 Oprava a editace dat

Jestliže jsou vrstevnice pospojovány, zbývá už jen opravit případné chyby a zadat výšky vrstevnic, než bude možné vygenerovat první jednoduchý prostorový model. Oprava a zadání výšek vrstevnic se provádí ve třetím programu s názvem *Editor*, který jsem vytvořil. Ten umožňuje opravu a editaci vrstevnic, vložení nových vrstevnic, vložení jednotlivých bodů jako doplnění modelu, zadání výšek vrstevnic a vygenerování základního modelu terénu. Program obsahuje menu s funkcemi, klientskou oblast, kde se zobrazují vrstevnice, a několik tlačítek pro rychlý přístup k nejdůležitějším funkcím.

V první fázi je tedy třeba opravit rozpoznané vrstevnice, a to hlavně jejich špatné pospojování, doplnění nerozpoznaných částí a nakonec smazání špatně rozpoznaných vrstevnic. U kvalitní předlohy může být tento krok velice krátký, ale jestliže není předloha dobrá, může se oprava dosti protáhnout.

Po spuštění programu *Editor* je nutné nejprve otevřít soubor s rozpoznanými vrstevnicemi. Zároveň s rozpoznanými vrstevnicemi se načte soubor s popisem jejich pospojování. Vrstevnice se zobrazí v okně, kde si je můžeme prohlédnout. V okně jsou

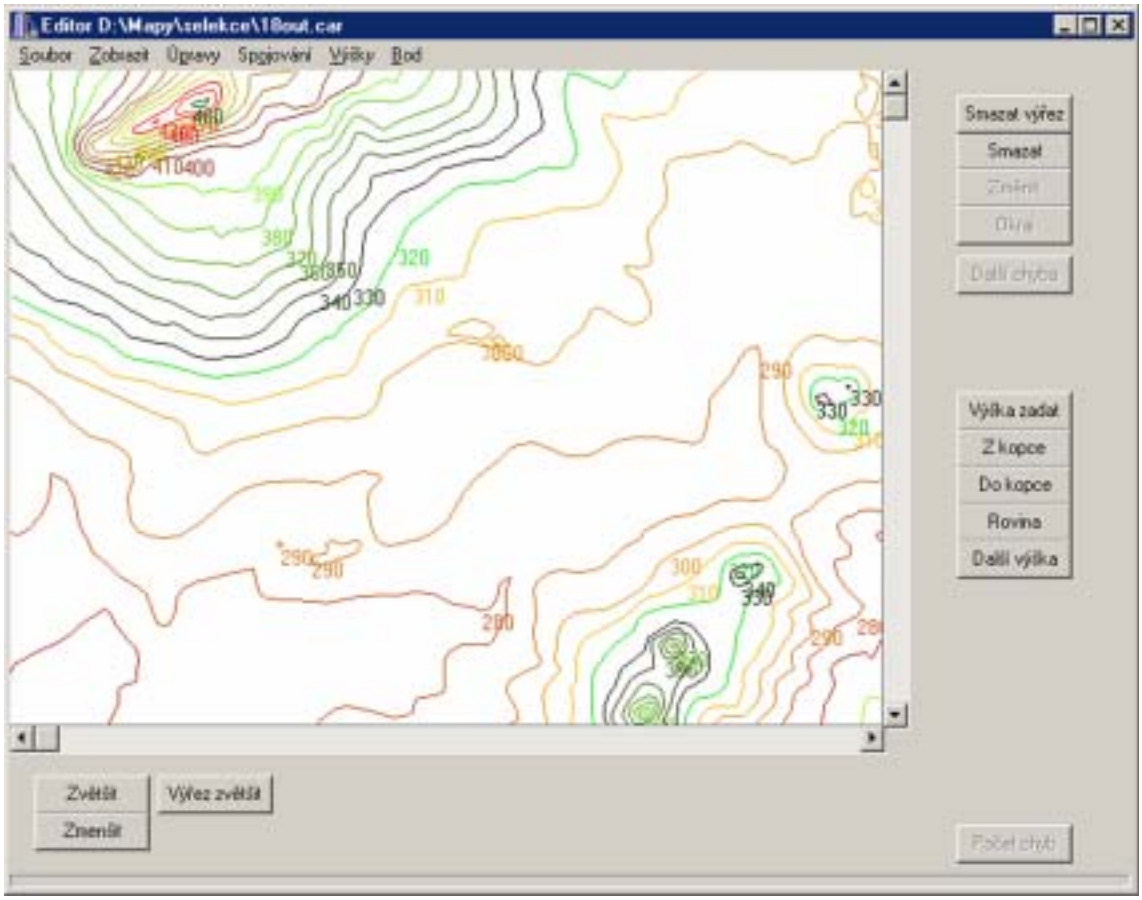
barevně rozlišeny dva typy čar. Červeně se zobrazují vlastní rozpoznané vrstevnice a světle modře se zobrazují čáry, které představují pospojování vrstevnic provedené v programu *Vektorizator*. Obraz s vrstevnicemi lze zvětšovat nebo zmenšovat. Pro zjištění pravděpodobného počtu chyb je v okně aplikace umístěno tlačítko Počet chyb. Po stisknutí tohoto tlačítka se zobrazí číslo, které představuje počet vrstevnic, jež nejsou spojené s žádnou další vrstevnicí a jejich konec není definován jako konec vrstevnice na okraji mapy. Pro procházení jednotlivých chyb je zde tlačítko Další chyba, které po zmáčknutí přesune zobrazení na další chybu, aby ji bylo možno opravit. S pomocí tohoto tlačítka je možné rychle opravit všechny chyby. Pro vlastní opravu jsou zde tlačítka, která umožňují vrstevnici smazat (tlačítko Smazat), změnit její napojení (tlačítko Změnit) anebo ji nastavit jako vrstevnici s koncem na okraji mapy (tlačítko Okraj). Hromadně lze smazat vrstevnice tlačítkem Smazat výřez.

Pro snadnější opravu je zde možnost zobrazit jako pozadí rozpoznaných vrstevnic obrázek s mapou. Tím je možné rychleji zjistit, jak správně opravit danou chybu. Po opravě všech chyb se nechají tyto části vrstevnic pospojovat (nabídka Spojování příkaz Pospojovat). Tím vzniknou už celkové čáry, které reprezentují jednotlivé vrstevnice. Dalším krokem je možné zredukovat počet bodů v modelu, což zrychlí jeho prostorové generování (nabídka Spojování příkaz Převzorkovat).

Následuje zadání výšek jednotlivých vrstevnic. Pro usnadnění tohoto úkolu je v okně aplikace 5 tlačítek. První tlačítko Výška zadat slouží pro přímé zadání výšky vrstevnice. Další tři tlačítka slouží pro hromadné zadání výšek. Vychází se vždy od vrstevnice s již známou výškou. Tažením kurzoru přes tuto vrstevnici dále k vrstevnicím o neznámé výšce je definována fiktivní spádnice. Podle toho, jakou jsme zvolili funkci (Z kopce, Do kopce nebo Rovina) se potom doplní výšky u dalších vrstevnic v zadaném intervalu. Pro tyto funkce je proto nutné zadat interval vrstevnic v dialogu Nastavení (nabídka Úpravy příkaz Nastavení). Pro zadání všech výšek je zde tlačítko Další výška, kterým se prochází vrstevnice s nezadanými výškami, a tak urychluje práci. Po zadání všech výšek je dobré soubor uložit. Program *Editor* ukládá navíc ještě soubor s jednotlivými body, kterými je možné mapu doplnit, zejména na kopcích, o výškové kóty. Soubor s čarami má příponu car a soubor se samostatnými body má příponu bod.

Jestliže máme zadané výšky, je soubor připraven na tvorbu prvního modelu. Můžeme si vybrat ze dvou základních typů modelu, a to buď model tvořený uzlem `IndexFaceSet`, nebo `ElevationGrid`. Před začátkem generace prvního modelu je ještě nutné zadat výškové zkreslení a název výsledného souboru. U modelu tvořeného

uzlem `IndexFaceSet` je třeba zadat, které vrstevnice se mají použít (použijí se pouze vrstevnice, které mají nulový zbytek po dělení zadaným číslem), a u modelu `ElevationGrid` hustotu výškové sítě (dále je nutné mít správně nastaven výškový interval v dialogu Nastavení). Rychlost tvorby modelu závisí na počtu bodů, které tvoří model, výkonu počítače, případně hustotě výškové sítě.



Obr. 6.3 Program *Editor*.

6.7 Výstup dat ve formátu VRML

Generování VRML souborů se liší podle toho, jakým uzlem je model tvořen. Co mají ale oba typy modelů společné, je použití základních uzlů nutných pro zobrazení modelu. Proto jsem vytvořil knihovnu funkcí (soubor vrmlform.cpp a vrmlform.h), která mi usnadňuje generování VRML souboru. Je to soubor tříd, které reprezentují každá jeden uzel a zabezpečují správný výstup do souboru.

Pro tvorbu vlastního modelu je potom třeba získat buď trojúhelníkovou síť, nebo množinu výšek. Funkce pro tvorbu VRML modelů jsou uloženy v souborech vrml.cpp a vrml.h, pomocné funkce v souborech geometrie.cpp a geometrie.h.

6.7.1 Trojúhelníková síť

K tvorbě trojúhelníkové sítě jsem vytvořil základní funkci `Mnozina_ploch`, která z množiny bodů a vrstevnic generuje trojúhelníkovou síť. Funkce si vytváří kvůli zrychlení práce vlastní datové struktury. Nejprve si funkce zjistí počet bodů v modelu pro alokaci paměti. Načte seznam bodů z vrstevnic a samostatných bodů a zároveň množinu čar z vrstevnic, které musí použít jako strany trojúhelníků (předurčené hrany). Pak kvůli zrychlení následuje rozdělení bodů do čtverců přibližně po padesáti bodech. Funkce nemusí při hledání vrcholů nových trojúhelníků prohledávat všechny vrcholy, ale stačí projít body v okolí. Následuje vlastní tvorba trojúhelníků. Funkce začíná uprostřed mapy, kde vznikne první trojúhelník. Potom se prochází strany tohoto trojúhelníku, tvoří se nad nimi nové trojúhelníky a nad jejich novými stranami se pak dokola tvoří další trojúhelníky. Jako základní kritérium pro hledaný bod nového trojúhelníku jsem použil součet dvou délek nových stran vytvářeného trojúhelníku. Tímto jednoduchým postupem se vytvoří trojúhelníková síť.

Postup lze shrnout do těchto základních kroků:

1) Zjistíme počet bodů a vrstevnic a alokujeme pro ně paměť.
2) Načteme do této paměti všechny body v modelu a vrstevnice, které použijeme jako předurčené hrany.
3) Model rozdělíme do přibližně čtvercových oblastí zhruba po padesáti bodech.
4) Pro každý čtverec vytvoříme seznam bodů a seznam hran, které obsahuje.
5) Vytvoříme první trojúhelník, tedy nalezneme první bod nejblíže středu mapy a k němu nejbližší dva body.

Vlastní tvorba modelu:

6) Procházíme vytvořené trojúhelníky a testujeme, zda mají ze všech stran sousedy a nemá-li se tedy tvořit nad touto stranou nový trojúhelník.
7) Nad neobsazenou hranou trojúhelníku hledáme nový bod. Procházíme všechny body v území tvořené devíti čtvercovými oblastmi.
8) Nejdříve vypočteme délku nově vytvořených stran pomocí tohoto bodu jako kritérium a porovnáme s předchozí nejkratší.
9) Dále testujeme, zda je bod ve správné polorovině.
10) Poté testujeme, zda v nově vytvořeném trojúhelníku není jiný bod.

11) Dále testujeme, zda se nově vytvořené strany trojúhelníku nekříží s jinými trojúhelníky.
12) Nakonec testujeme, zda se nově vytvořené strany nekříží s některou vrstevnicí.
13) Jestliže zkoumaný bod splňuje všechny tyto podmínky, uloží se jeho kritérium, číslo bodu a prohledávání pokračuje od bodu 8).
14) Po prohledání všech bodů se vytvoří nový trojúhelník, který je tvořen stranou původního trojúhelníku a bodu uloženého v paměti z bodu 13).
15) Dokud program neprojde všechny trojúhelníky a všechny jejich hrany, pokračuje od bodu 6).

6.7.2 Množina výšek

Pro tento úkol jsem vytvořil funkci `Vyskova_mapa`. Jejím úkolem je vytvořit množinu výšek nad body pravidelné sítě. Tato funkce – obdobně jako funkce `Mnozina_ploch` – nejdříve rozdělí body a vrstevnice do čtvercových oblastí, čímž se zvýší rychlost funkce. Výšky získává funkce pomocí lineární interpolace ze dvou nejbližších výšek, a to získaných buď z vrstevnic, nebo samostatných bodů, k hledanému bodu výškové sítě. Jestliže je požadovaný bod na okraji mapy, převezme se výška od nejbližšího samostatného bodu nebo vrstevnice.

Postup lze shrnout do následujících kroků:

1) Zjistíme počet bodů a vrstevnic a alokujeme pro ně paměť.
2) Načteme do této paměti všechny samostatné body v modelu a vrstevnice jako dvojice bodů (vrstevnicové úseky).
3) Model rozdělíme do přibližně čtvercových oblastí zhruba po 25 vrstevnicových úsecích.
4) Pro každý čtverec vytvoříme seznam bodů a seznam vrstevnicových úseků.

Vlastní program.

5) Funkce prochází jednotlivé vrcholy výškové sítě, aby získala jejich výšku.
6) Pro každý bod sítě funkce nejdříve hledá vrstevnici, která je nejblíže.
7) Potom hledá samostatný bod, který je nejblíže.
8) Dále hledá druhou nejbližší vrstevnici a zároveň testuje, zda mezi ní a bodem výškové sítě není jiná vrstevnice.

9) Jestliže je nalezený samostatný bod blíže k vrcholu výškové sítě než druhá nalezená vrstevnice, testuje program, zda mezi samostatným bodem a vrcholem výškové sítě se nekříží s některou vrstevnicí.
10) Pokud se samostatný bod s žádnou vrstevnicí nekříží, použije se pro výpočet výšky nejbližší vrstevnice a samostatný bod; v opačném případě se výška vypočte ze dvou nejbližších vrstevnic, a to lineární interpolací.
11) Dokud program neprojde všechny body výškové sítě, pokračuje od bodu 5).

6.8 Skládání světů

Pro tvorbu rozsáhlejšího virtuálního modelu jsem vytvořil čtvrtý program Generator. Tento program umožňuje poskládat jednotlivé modely vedle sebe, vytvořit jejich vícenásobné reprezentace s různým stupněm generalizace a také rozdělit tyto modely na menší části. Dále umožňuje nastavit základní parametry generovaného virtuálního světa.

Program se skládá z několika záložek, pomocí kterých se definují vlastnosti světa.

1. První záložka se jmenuje Základ. Zde se definují základní parametry modelu, jako je typ modelu[6], měřítko původní mapy, rozlišení použité při skenování mapy, požadované výškové zkreslení generovaného modelu a ekvidistance vrstevnic modelu. Dále zde lze nastavit rozdělení vlastního modelu na menší části, které budou v samostatných souborech a budou se při otevření modelu načítat postupně.
2. Záložka Avatar, pomocí níž se definují vlastnosti avatara, a to výška, rychlost pohybu, dohled, způsob pohybu a zapnutí svítilny na pomyslném čele avatara.
3. Na další záložce s názvem Pozadí se dá definovat pozadí modelu, a to pomocí barevného přechodu tří barev pro nebe a tří barev pro zem.
4. Důležitá je také záložka Světlo a mlha, na které se definují vlastnosti světla ve virtuálním světě, jeho barva, poloha a dosah. U mlhy se dá nastavit barva, maximální viditelnost a způsob houstnutí mlhy.
5. Na záložce Vlastnosti modelu se definují jak společné vlastnosti pro oba modely (tj. úhel vyhlazení), tak i speciální vlastnosti pro konkrétní model. U výškové mapy je to hustota výškové sítě modelu a u množiny ploch je

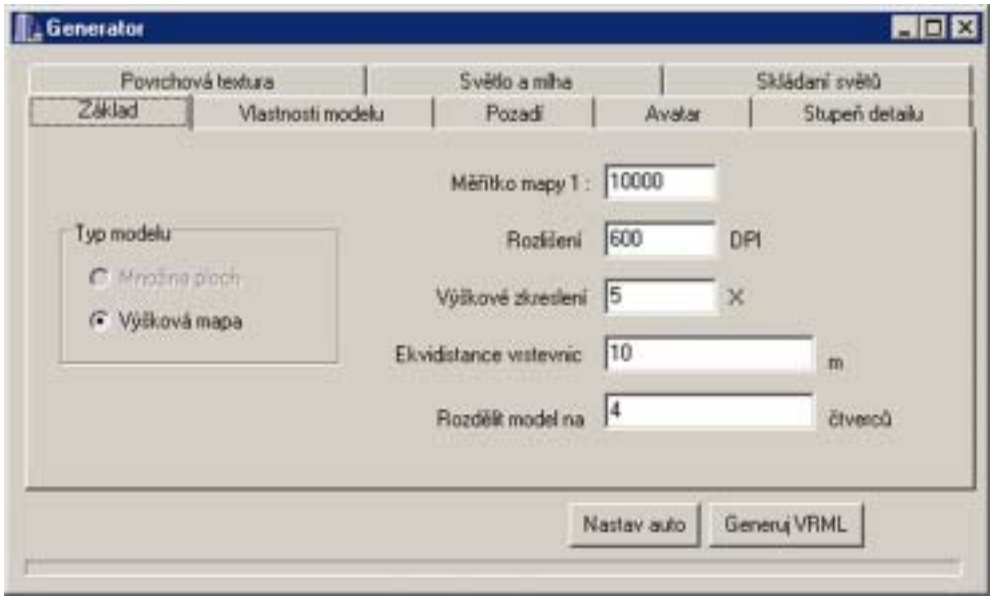
[6] V současnosti umožňuje program generovat pouze model tvořený uzlem `ElevationGrid`.

to stupeň generalizace způsobený vynecháním některých vrstevnic o dané výšce.

6. Stupeň detailu je záložka, na které se dá definovat vícenásobná reprezentace modelu vytvářená pomocí uzlu `LOD`. Je zde možnost nastavení způsobu přepínání jednotlivých reprezentací[7] ručně, automaticky nebo kombinací obou způsobů. Dále se zde nastavuje stupeň generalizace těchto pomocných modelů a to snižováním počtu vrcholů výškové sítě a také kolikrát se má zmenšit rozlišení použité textury.
7. Záložka Skládání světů slouží k poskládání jednotlivých modelů vedle sebe, např. mapových listů.
8. Do záložky Povrchová textura se zadává název souboru obsahující texturu celého modelu. Dále umožňuje spojit obrázky s texturou jednotlivých mapových listů do jednoho souboru.

Pro usnadnění nastavení stupňů detailů je zde tlačítko Nastav auto, které v závislosti na velikosti mapy, počtu menších částí, na které byl model rozdělen a na nastavení počtu stupňů detailů vypočítá a doplní hustotu výškové sítě ve Vlastnostech modelu a Stupních detailu a zároveň Vzdálenost pro přepínání jednotlivých reprezentací modelu tak aby byla rychlost zobrazení optimální.

Po zadání všech parametrů již stačí jen zmáčknout tlačítko Generuj VRML a začnou se vytvářet jednotlivé části modelu.



Obr. 6.4 Program *Generator*.

[7] V současnosti umožňuje program zvolit pouze možnosti ručně nebo automaticky.

7 Získávání dat pro VRML

Aby se dal vytvořit prostorový model z mapy, je třeba získat její výškovou složku. Ta se dá získat jednak z mapy v papírové formě, nebo z mapy digitální.

Pro tvorbu virtuálních modelů map jsem si vybral Základní mapu ČR, která pokrývá celou Českou republiku a teoreticky by se z ní dal udělat virtuální model ČR. Základní mapa ČR existuje ve formě digitální i analogové. Jako další podklad pro tvorbu virtuálního modelu jsem vyzkoušel také několik map pro orientační běh.

7.1 Z analogových map

Abychom získali výšková data z klasické mapy, musí tato mapa obsahovat výškovou složku, tedy vrstevnice a výškové kóty. Pro automatizované zpracování to znamená vektorizaci vrstevnic. Přibývají ještě další podmínky. Vrstevnice musí být vykresleny zvláštní barvou, takže zdrojová mapa musí být barevná, aby se daly ostatní složky mapy, kreslené jinou barvou, odfiltrovat. Vrstevnice musí být kvalitně vytištěné, aby se neslévaly dohromady, a co nejméně přerušované. Další podmínkou je, aby barva, kterou jsou vykresleny vrstevnice, nebyla použita i pro jiné prvky. Tato podmínka nebyla u orientačních map vždy dodržena, takže výsledky automatizované vektorizace nebyly vždy uspokojivé.

Podmínky pro automatickou vektorizaci asi nejlépe splňovala Základní mapa ČR v měřítku 1:10 000, jejíž výsledky byly dobré. Menší měřítka už automatickou vektorizaci příliš neumožňovala, protože jejich výšková složka je více generalizována a potlačená na úkor polohopisu. V takovém případě už je lepší použít mapu přímo v digitální formě nebo použít vektorizované vrstevnice z mapy většího měřítka.

7.2 Digitální data

Digitální data jsem získal na Zeměměřičském úřadě. Pro porovnání jsem si zažádal o data mapových listů stejné lokality, jako byla mapa v papírové formě. Zeměměřičský úřad v současné době poskytuje dva základní typy digitálních map. Jednak je to *„Rastrová reprezentace Základní mapy ČR“* (zkratka RZM), a dále je to *„Základní báze geografických dat“* (zkratka ZABAGED).

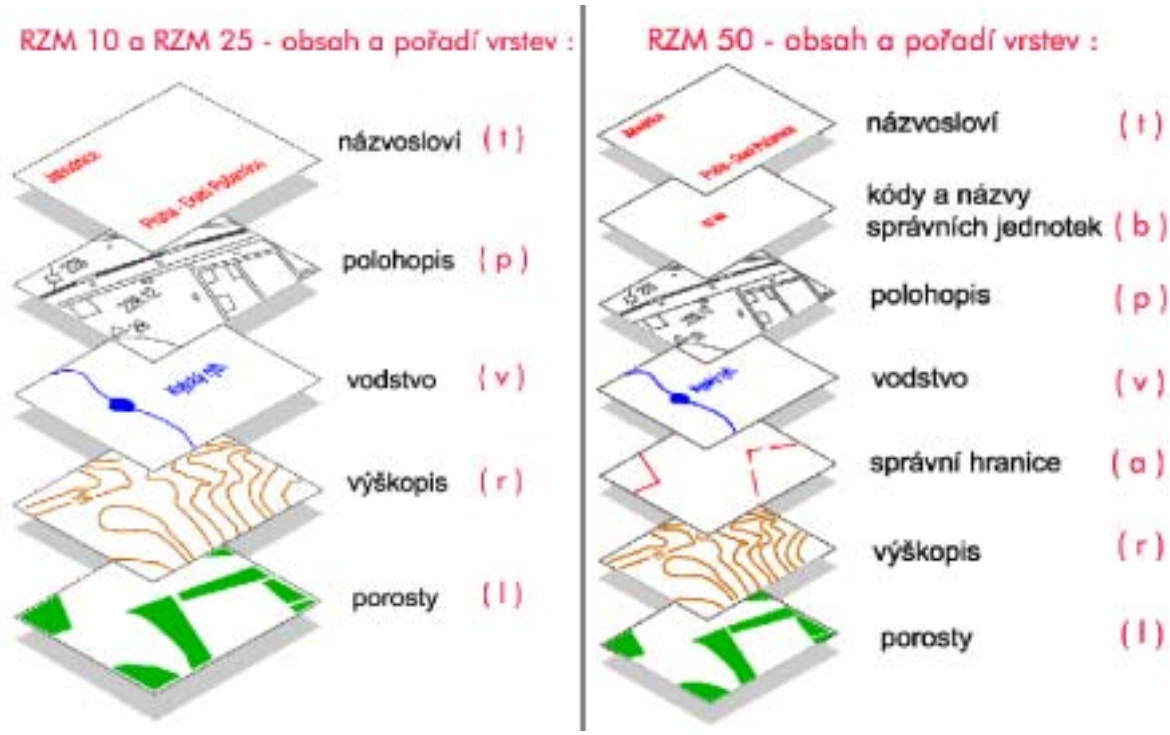
Pro zvýšení reálnosti jsem chtěl vyzkoušet použití ortofotomapy jako textury. Zeměměřičský úřad poskytuje také ortofotomapy v digitální formě. Jde o polotónový rastrový obraz v kladu listů Základní mapy ČR 1:10 000, proto jsem si také vyžádal

ortofotomapy pro zadané území. Bohužel jsou pouze černobílé, ale i tak výrazně zvyšují dojem reálné krajiny.

7.2.1 RZM

RZM jsou vlastně vrstvy rastrové mapy, které byly získány skenováním jednotlivých tiskových podkladů *Základní mapy ČR* v rozlišení 400 DPI. RZM se vydává v měřítku 1:10 000, 1:25 000, 1:50 000, a 1:200 000. Pro svou práci jsem získal mapové listy v měřítku 1:10 000, 1:25 000 a 1:50 000 stejné lokality pro porovnání. RZM se skládá z pěti barevných vrstev v měřítkové řadě 1:10 000 a 1:25 000 a sedmi vrstev v měřítku 1:50 000 viz Obr. 7.1. V této podobě lze získat data z celého území ČR. RZM jsem použil jednak jako texturu, která se zobrazuje na prostorovém modelu, a také jako zdroj dat pro vektorizaci. Výhodou těchto dat při vektorizaci je, že mapy jsou skenovány po jednotlivých vrstvách, a odpadá proto barevná filtrace, která může být zdrojem chyb. Pro vektorizaci se použije přímo výšková složka mapy, která je uložena v samostatné vrstvě. Jednotlivé vrstvy jsou uloženy v samostatných souborech ve formátu CIT. Při vektorizaci se postupuje jako při zpracování analogové mapy s tím rozdílem, že se v programu *Filtr* nastaví maximální meze při filtraci.

Pro použití RZM jako textury na prostorový model je nutné pouze složit jednotlivé vrstvy do jednoho obrázku a uložit ve vhodném formátu, nejlépe JPEG. Pro tento úkol jsem použil *MicroStation*, ve kterém jsem nejprve načetl jednotlivé vrstvy a nastavil správně pořadí jejich zobrazení. Potom už stačí jen uložit výsledný obraz funkcí v menu Pomůcky: Obrázek…- Uložit. Tato funkce ukládá otevřený pohled jako obrázek. Uložený obrázek už je třeba jen oříznout do požadovaných rozměrů v libovolném grafickém editoru.



Obr. 7.1 Obsah a pořadí vrstev RZM 1: 10 000, 1:25 000 a 1:50 000.

7.2.2 ZABAGED

Je to digitální topografický model odvozený z mapového obrazu *Základní mapy ČR* 1:10 000 v souřadnicovém systému S-JTSK a výškovém systému Bpv. ZABAGED má charakter GISu, je definován katalogem 102 typů objektů strukturovaných do 63 tématických vrstev. Výškopisná složka je tvořená vektorovým souborem vrstevnic. ZABAGED je tvořen a provozován v grafickém prostředí MicroStation. Prostorově organizačními jednotkami ZABAGEDu jsou mapové listy 1:10 000 v kladu listů Základní mapy České republiky. Proces tvorby ZABAGED započal v roce 1995 a byl dokončen v roce 2001. Data ZABAGED se dodávají po celých mapových listech jako vektorové soubory polohopisu (2D) a výškopisu (3D) ve formátu DGN.

Pro mou práci byla důležitá hlavně výškopisná složka ZABAGEDu, která je ideální pro tvorbu digitálního modelu terénu. Pro tvorbu VRML modelu je nejprve nutné přenést vrstevnice z programu *MicroStation* do formátu, který jsem vytvořil pro tvorbu VRML modelu a který potom dokáže přečíst jak program *Editor*, tak *Generator*. Pro tento úkol jsem v programu *MicroStation* vytvořil jednoduché makro, které exportuje vrstevnice do souboru v požadovaném formátu. Z tohoto souboru lze už přímo generovat VRML model. Před exportem je třeba pouze pootočit celý obraz s vrstevnicemi tak, aby byl okraj mapového listu rovnoběžný se souřadnicovými osami.

8 Zhodnocení vytvořených modelů

Po vytvoření více modelů s různým stupněm generalizace jsem dospěl k názoru, že modely tvořené pouze ze zesílených vrstevnic se většinou vyrovnají modelům tvořeným z kvalitnějších podkladů a většího množství dat, např. ZABAGED. Myslím, že je to způsobeno tím, že krajina zobrazená v měřítku 1:10 000 a menším je již tak generalizována, že její terén je jednoduchý a skládá se pouze z hladce navazujících ploch. Dalším faktorem je také subjektivní pohled na model, kdy pozorovatel nemá motivaci si model podrobněji prohlížet, protože podrobnější model by musel zobrazovat kromě dat geografických také další data, např. zobrazení budov. Naopak pro pozorovatele je daleko atraktivnější celkový pohled z výšky, kdy si může krajinu prohlédnout z ptačí perspektivy.

Jako výsledek své práce jsem vytvořil několik modelů. Některé pohledy na tyto modely si lze prohlédnout v obrazové příloze. Jednak je to několik modelů orientačních map a dále několik modelů Základní mapy ČR. Vytvořil jsem model jednoho listu ZM50 vektorizováním obrazu RZM50, kterou jsem následně použil i jako texturu. Obdobným způsobem jsem vytvořil model mapového listu RZM25. Poté jsem vytvořil model sestavený ze šesti mapových listů ZM10, na které jsem jako texturu použil jednak ortofoto, dále pro porovnání naskenovaný obraz analogové mapy a obraz RZM. Výsledky procesu automatické vektorizace obrazu RZM nebyly tak dobré, protože digitální data jsem dostal příliš pozdě a program byl již optimalizován pro práci s papírovou mapou ZM10.

Při zobrazení modelů jsem zjistil, jak je důležité správné nasvícení modelu. Je to jednak intenzita, ale hlavně správné umístění světelného zdroje. Při dobrém umístění světelného zdroje vzniknou velice pěkné stíny, které velmi zvyšují reálnost krajiny.

9 Závěr

S rostoucím výkonem počítačů se bude také zvyšovat množství dat zobrazovaných prostorově, proto si myslím, že toto téma bude stále více nabývat na aktuálnosti a bude se o ně zajímat stále více lidí. Předpokládám, že k velkému rozvoji VRML dojde hlavně v komerční sféře, a to především v oblasti reklamy, kde budou firmy hledat stále zajímavější a atraktivnější možnosti prezentace svých výrobků.

Vizuální kvalita a rychlost zobrazení, zvláště větších územních celků, není v současnosti nejlepší, ale s rostoucím výkonem počítačů se bude dále zlepšovat, což také napomůže většímu rozšíření VRML na internetu. Jsem však přesvědčen, že již dnes lze tvořit a dále prezentovat prostorové modely, dnes ještě s určitou mírou generalizace, ale v budoucnu se stále většími podrobnostmi.

Domnívám se, že prostorové modely map ve VRML budou sloužit spíše jako podklad pro další modely, a to hlavně dynamické, které jsou schopny zobrazit určitý děj, jako např. stoupající vodu při záplavě, kdy se rozlévá voda v krajině. Další možností je využití v reklamě, např. jako upoutávka na nějakou rekreační chatu zasazenou ve virtuální krajině. Modely, které jsem vytvořil, jsou ale také velice lákavé, protože je vždy zajímavé prohlédnout si krajinu z ptačí perspektivy.

Myslím, že programy, které jsem vytvořil, dostatečně řeší požadované úkoly a pro základní demonstraci postupů plně dostačují. Při jejich tvorbě jsem se vždy řídil zásadou, aby funkce, které tyto programy vykonávají, byly jednoduché, čímž se zvýší jejich rychlost a sníží pravděpodobnost chyby v programu. Vím ale, že platí Murphyho zákon „*Bezchybný program je jako kvadratura kruhu. Člověk si myslí, že je to možné, ale nikomu se to nepovedlo.*“

10 Použitá literatura

[1] **Banko, Miloslav a kol.:** *MicroStation PowerDraft V5.5 CZ:Krok za krokem*, Praha 1996.

[2] **Beran, Martin:** *Učebnice Borland C++*, Praha 1995.

[3] **Čepek, Aleš:** *Programování 23 Úvod do C++*, skriptum, Praha 1998.

[4] **Huml, Milan – Michal, Jaroslav:** *Mapování 10*, skriptum, Praha 2000.

[5] **Janda, Karol:** *Prostorové modely terénu ve virtuální realitě na Internetu,* diplomová práce FSv ČVUT, Praha 2001.

[6] **Simon, Richard J. – Gouker, Michael – Barnes, Brian C.:** *Win32 API – průvodce vývojáře, svazek 1,* Brno 1997.

[7] **Simon, Richard J. – Gouker, Michael – Barnes, Brian C.:** *Win32 API – průvodce vývojáře, svazek 2,* Brno 1997.

[8] **Simon, Richard J. – Gouker, Michael – Barnes, Brian C.:** *Win32 API – průvodce vývojáře, svazek 3,* Brno 1999.

[9] **Virius, Miroslav:** *C++ Builder 4.0 Podrobný průvodce*, Praha 1999.

[10] **Zrzavý, Jakub:** *VRML tvorba dokonalých WWW stránek. Podrobný průvodce,* Praha1999.

[11] **Žára, Jiří:** *Jazyky pro popis virtuální reality*, skriptum, Praha 2000.

[12] **Žára, Jiří:** *VRML 97 Laskavý průvodce virtuálními světy*, Praha 1999.

Další zdroje:

[13] Definice jazyka VRML 97: http://www.web3d.org/Specifications

[14] Definice VRML EAI: http://www.vrml.org/WorkingGroups/vrml-eai/Specification

[15] EAI Working Group: http://www.web3d.org/WorkingGroups/vrml-eai

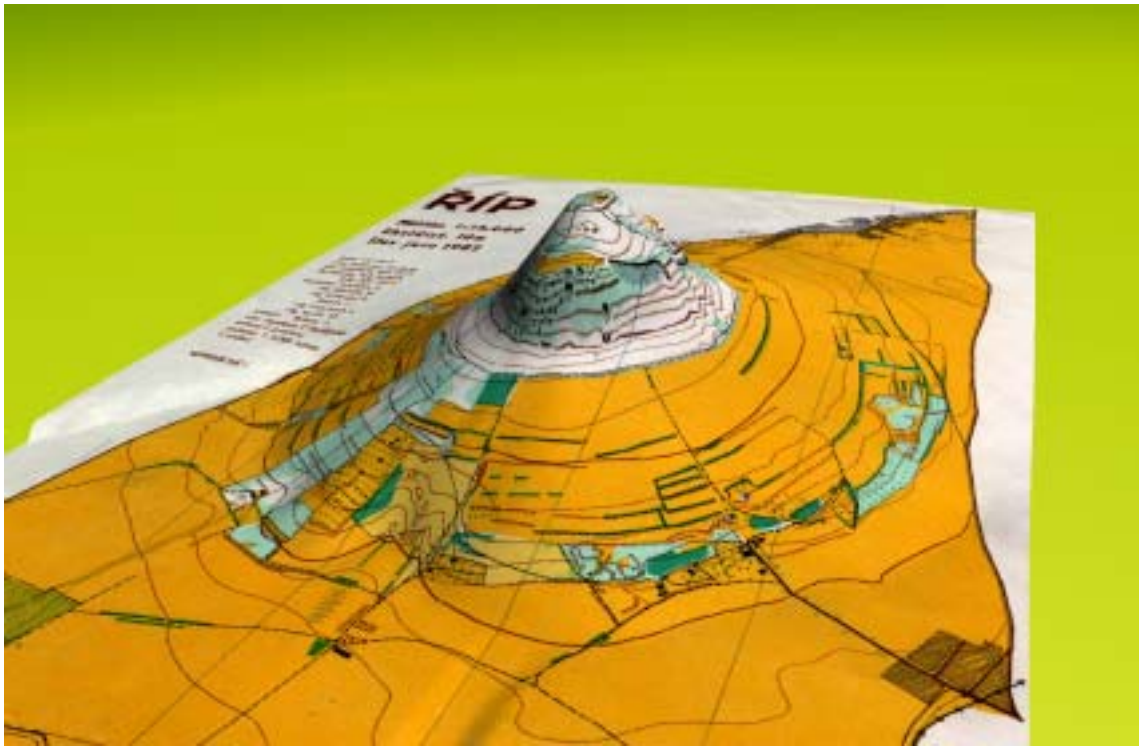
[16] Internetové stránky Zeměměřičského úřadu: http://www.vugtk.cz

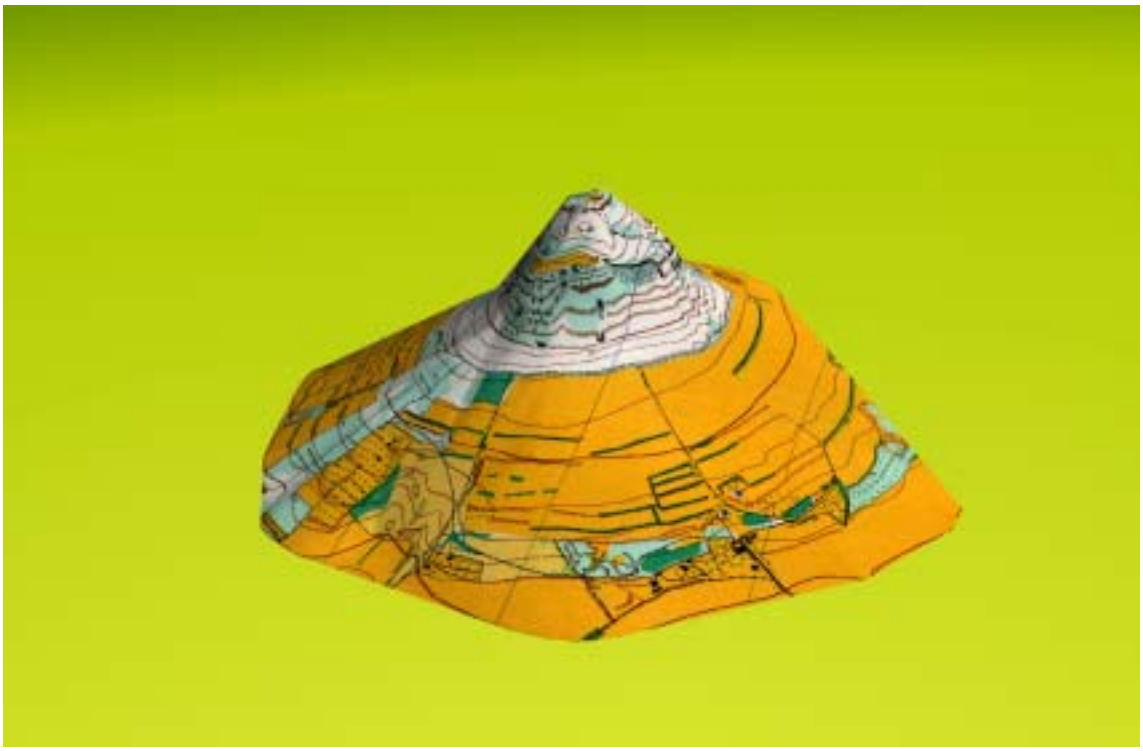
[17] Přehled VRML prohlížečů: http://www.web3d.org/vrml/bro2.htm

[18] SoftSurfer Geometry Algorithms: http://www.geometryalgorithms.com

Obrazová příloha

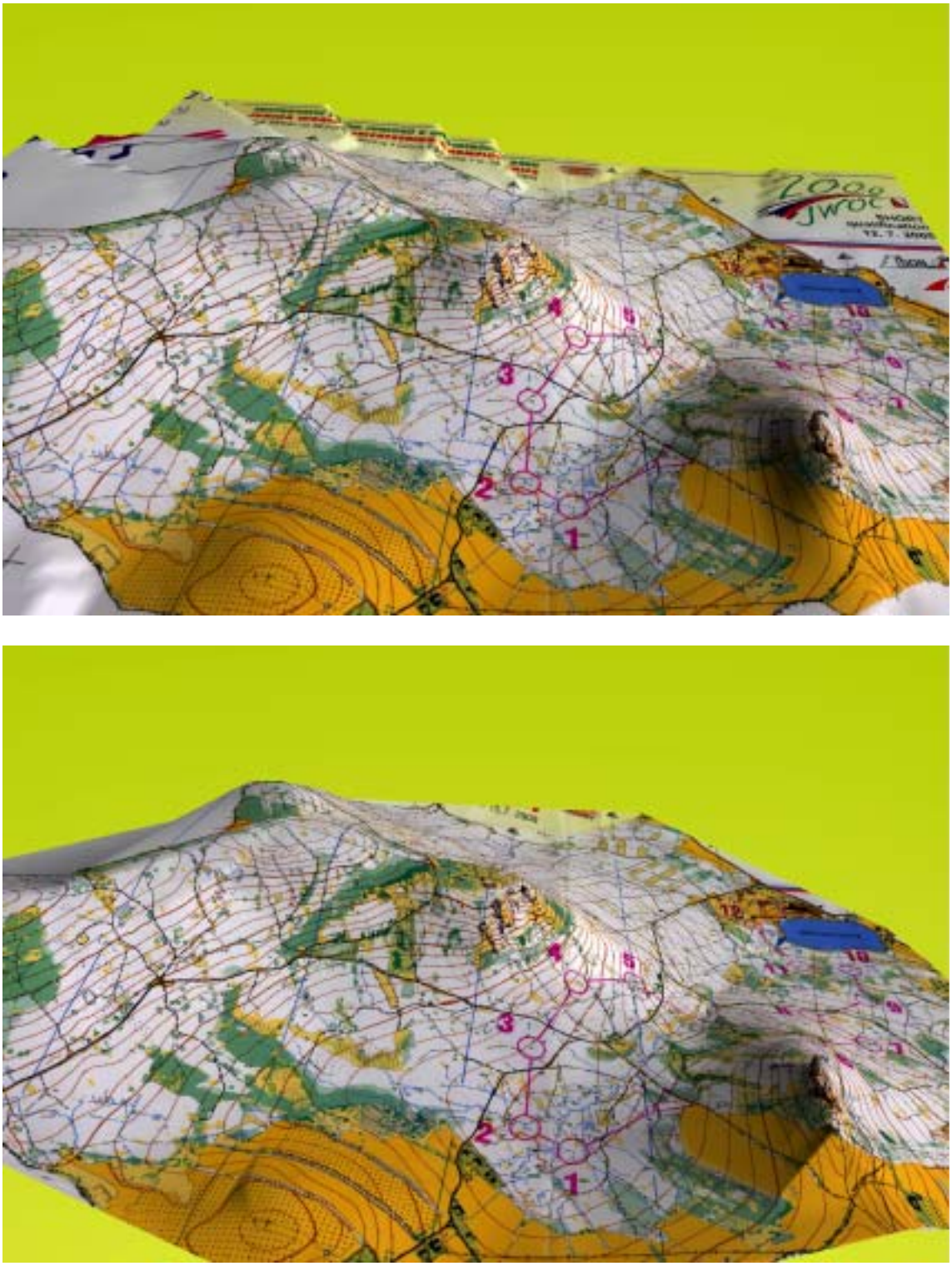
Příloha č.1





Orientační mapa lokalita Říp. Měřítko mapy 1:15 000, ekvidistance vrstevnic 10m. Výškové zkreslení 3x. Vrstevnice získány vektorizací z naskenované mapy. Interval vrstevnic použitych pro generování modelu – 50m. Nahoře model tvořený uzlem `ElevationGrid`, dole model tvořený uzlem `IndexFaceSet`.

Příloha č.2.1



Orientační mapa lokalita Nové Město na Moravě – Milovy. Měřítko mapy 1:15 000, ekvidistance vrstevnic 5 m. Vrstevnice získány vektorizací z naskenované mapy. Výškové zkreslení 3x. Interval vrstevnic použitých pro generování modelu – 25m. Nahoře model tvořený uzlem `ElevationGrid`, dole model tvořený uzlem `IndexFaceSet`.

Příloha č.2.2

Orientační mapa lokalita Nové Město na Moravě – Milovy. Měřítko mapy 1:15 000, ekvidistance vrstevnic 5 m. Vrstevnice získány vektorizací z naskenované mapy. Výškové zkreslení 3x. Interval vrstevnic použitých pro generování modelu – 25m. Nahoře model tvořený uzlem `ElevationGrid`, dole model tvořený uzlem `IndexFaceSet`.

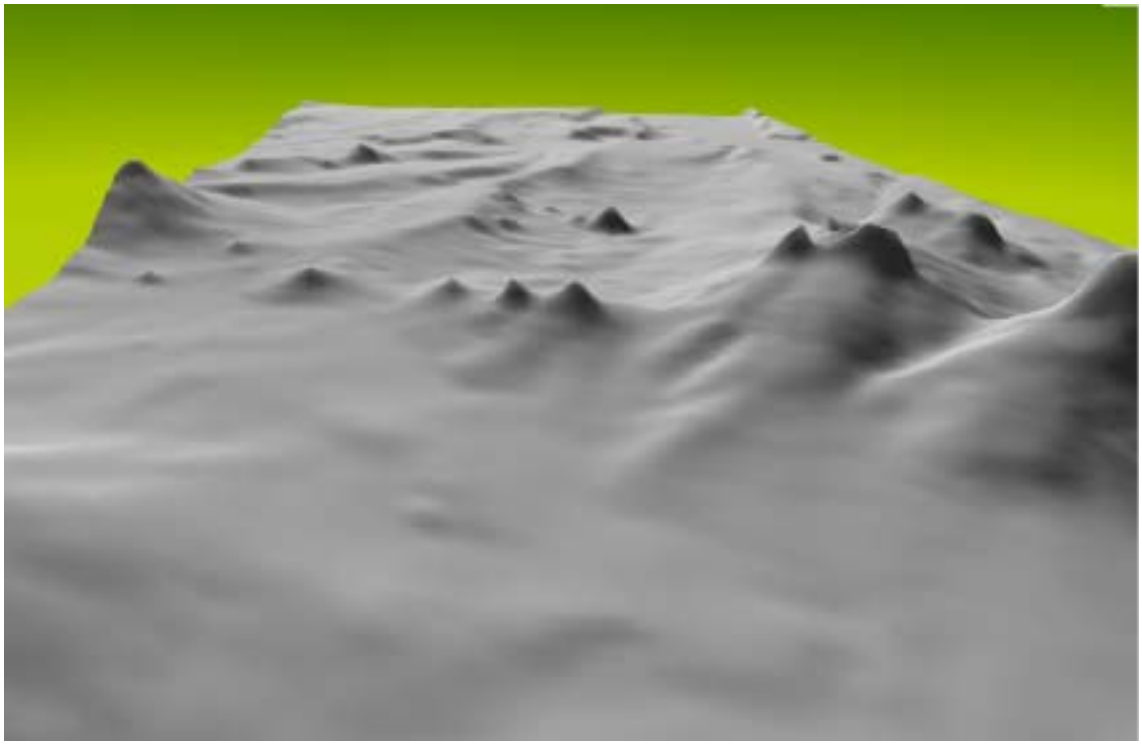
Příloha č.3.1

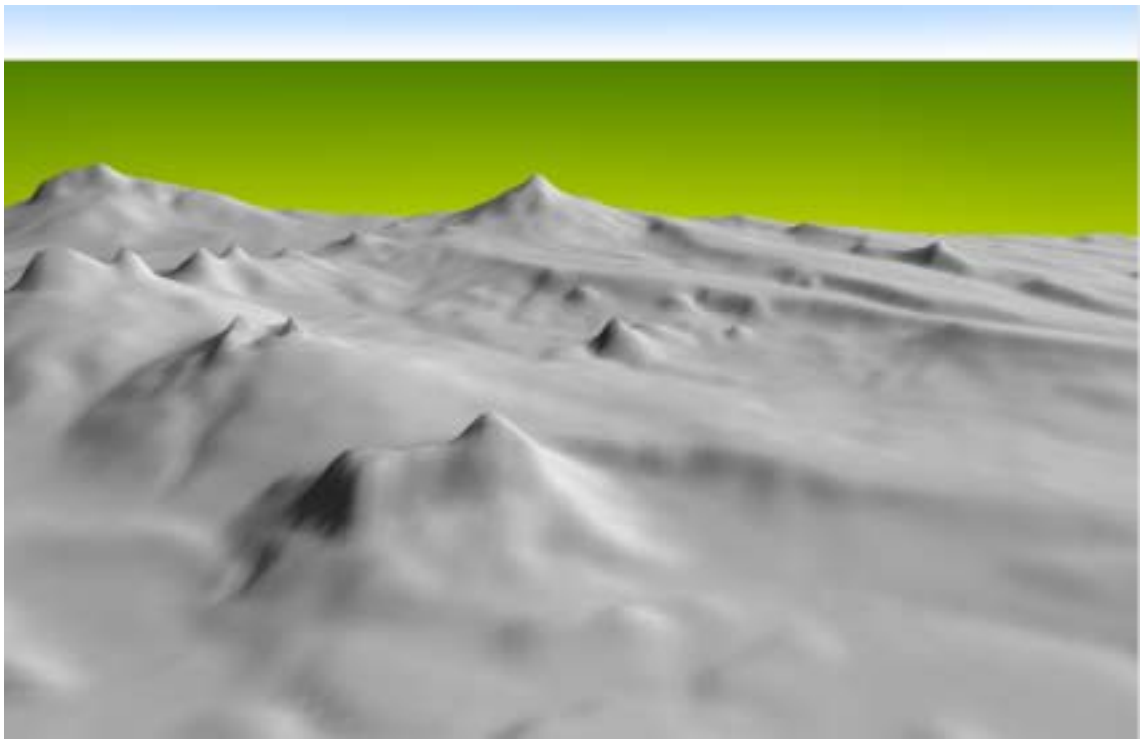
Orientační mapa lokalita Dubí – Vlčí kámen. Měřítko mapy 1:15 000, ekvidistance vrstevnic 5 m. Výškové zkreslení 3x. Vrstevnice získány vektorizací z naskenované mapy. Interval vrstevnic použitých pro generování modelu – 25m. Nahoře model tvořený uzlem `ElevationGrid`, dole model tvořený uzlem `IndexFaceSet`.

Příloha č.3.2

Orientační mapa lokalita Dubí – Vlčí kámen. Měřítko mapy 1:15 000, ekvidistance vrstevnic 5 m. Výškové zkreslení 3x. Vrstevnice získány vektorizací z naskenované mapy. Interval vrstevnic použitých pro generování modelu – 25m. Nahoře model tvořený uzlem `ElevationGrid`, dole model tvořený uzlem `IndexFaceSet`.

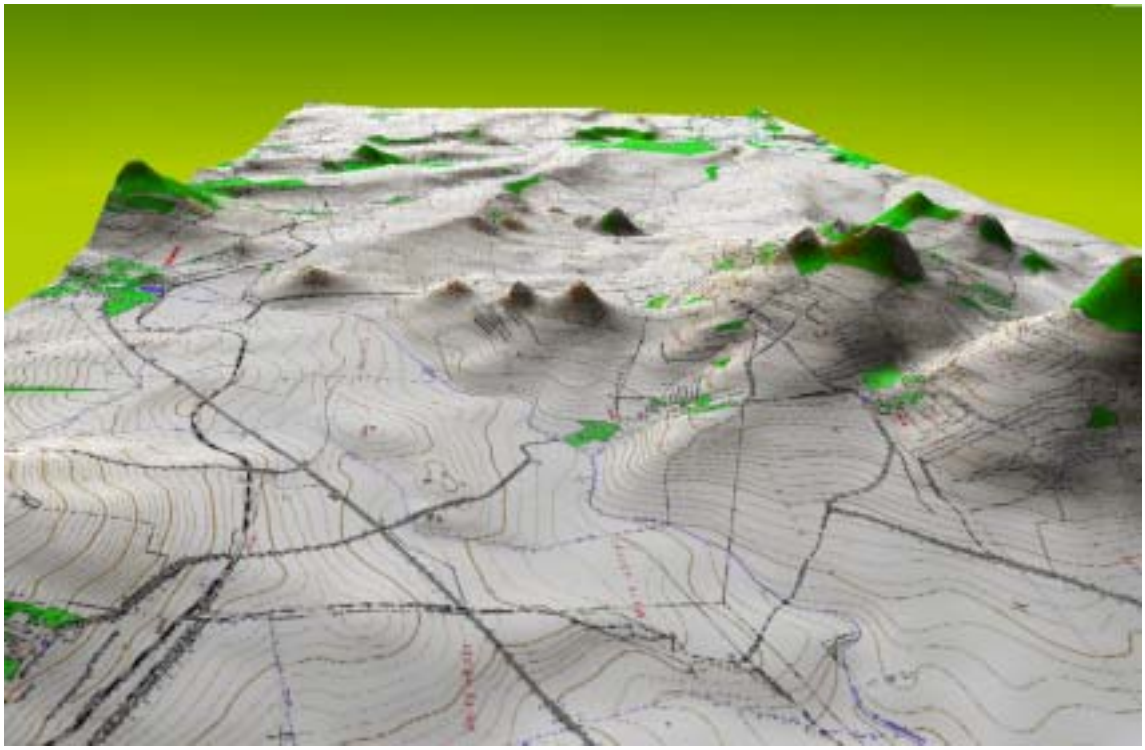
Příloha č.4.1

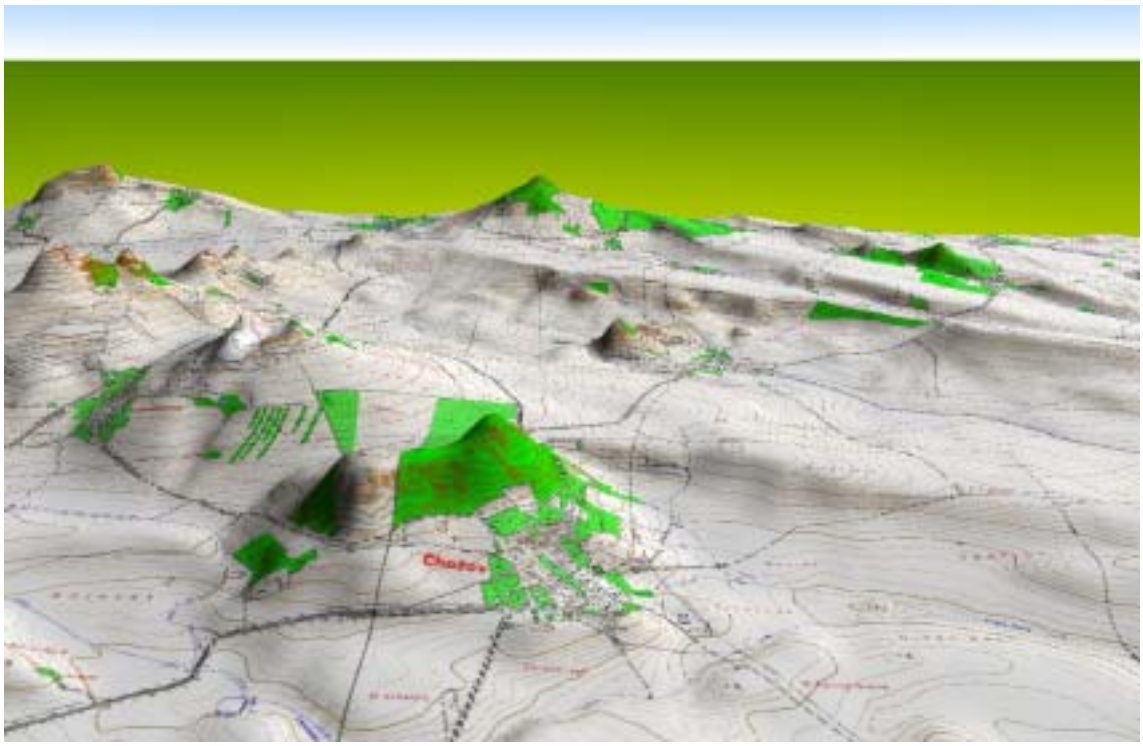




Základní mapa ČR 1:10 000. Model je složen z šesti mapových listů (02-34-18 až 20 a 23 až 25). Ekvidistance vrstevnic 2 m. Výškové zkreslení 3x. Vrstevnice získány exportem ze ZABAGEDu. Interval vrstevnic použitých pro generování modelu – 2m. Oba obrázky reprezentují model, který je tvořen uzlem `ElevationGrid`. Modely jsou zobrazeny bez povrchové textury.

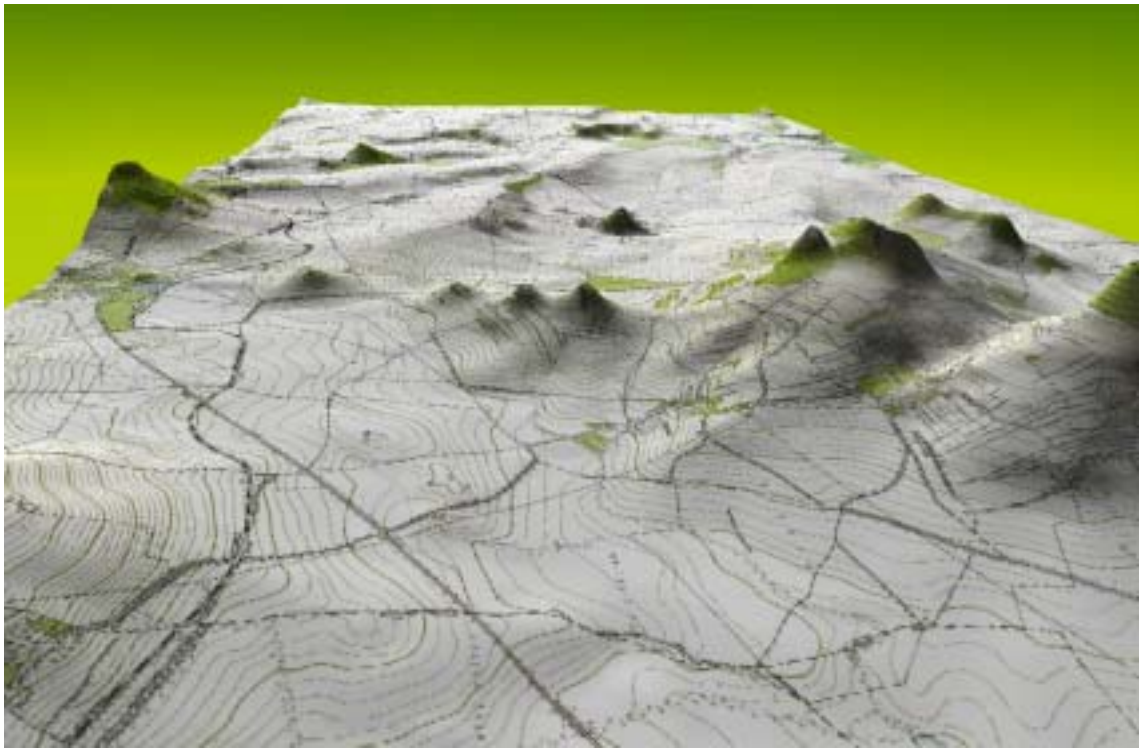
Příloha č.4.2

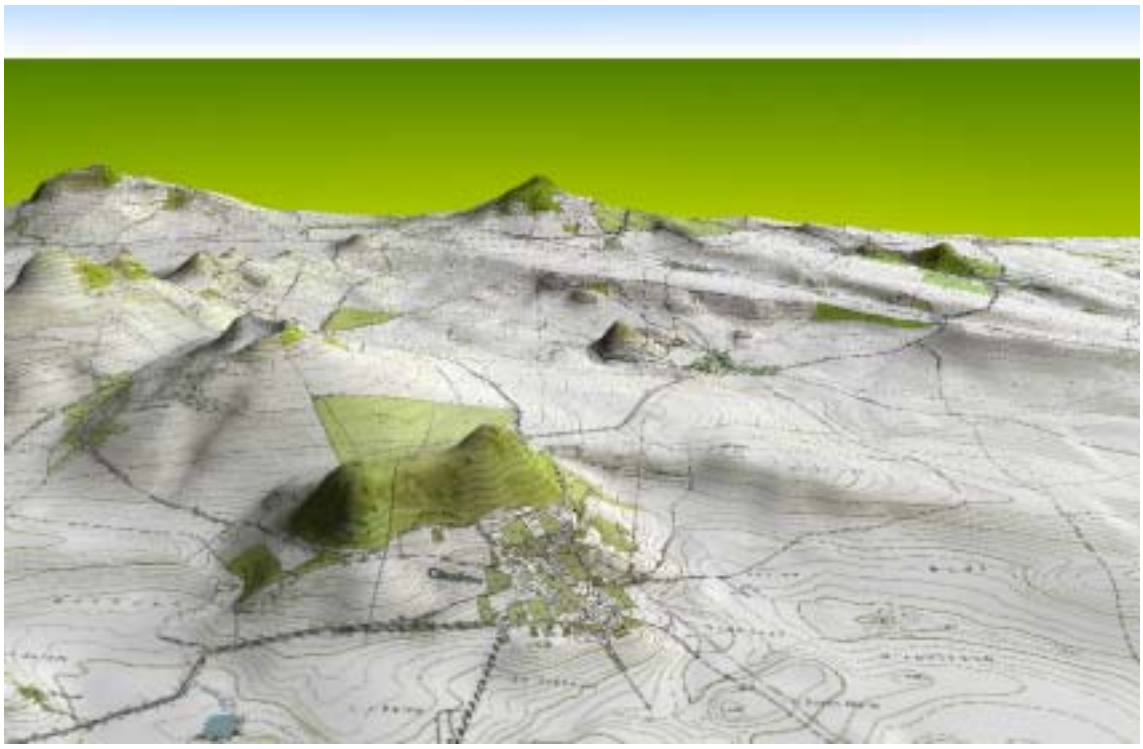




Základní mapa ČR 1:10 000. Model je složen z šesti mapových listů (02-34-18 až 20 a 23 až 25). Ekvidistance vrstevnic 2 m. Výškové zkreslení 3x. Vrstevnice získány exportem ze ZABAGEDu. Interval vrstevnic použitých pro generování modelu – 2m. Oba obrázky reprezentují model, který je tvořen uzlem `ElevationGrid`. Jako povrchová textura je použita RZM 10.

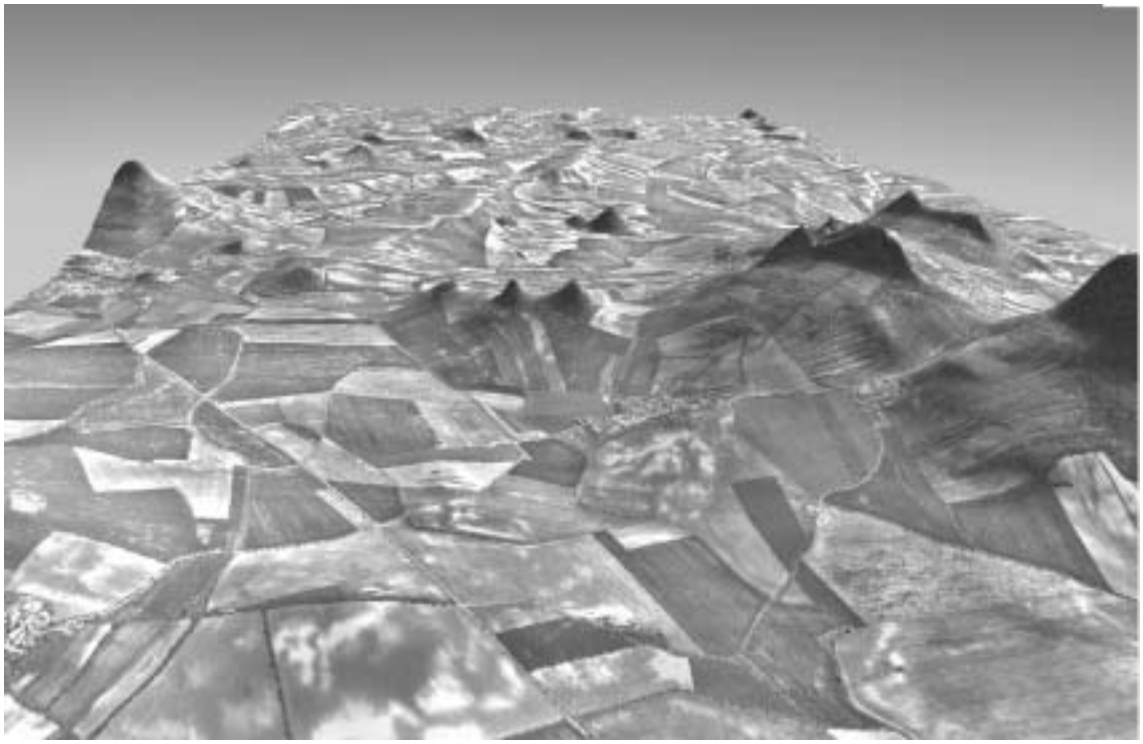
Příloha č.4.3

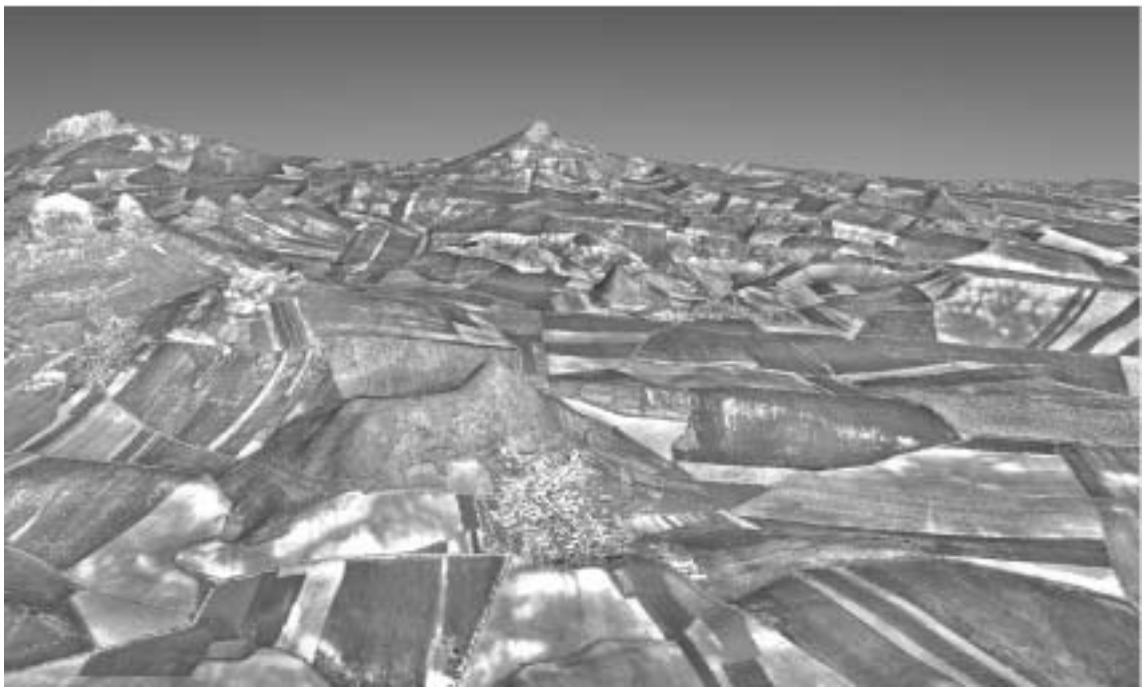




Základní mapa ČR 1:10 000. Model je složen z šesti mapových listů (02-34-18 až 20 a 23 až 25). Ekvidistance vrstevnic 2 m. Výškové zkreslení 3x. Vrstevnice získány vektorizací z naskenovaných mapových listů. Interval vrstevnic použitých pro generování modelu – 10 m. Oba obrázky reprezentují model, který je tvořen uzlem `ElevationGrid`. Jako povrchová textura je použita naskenovaná ZM 10.

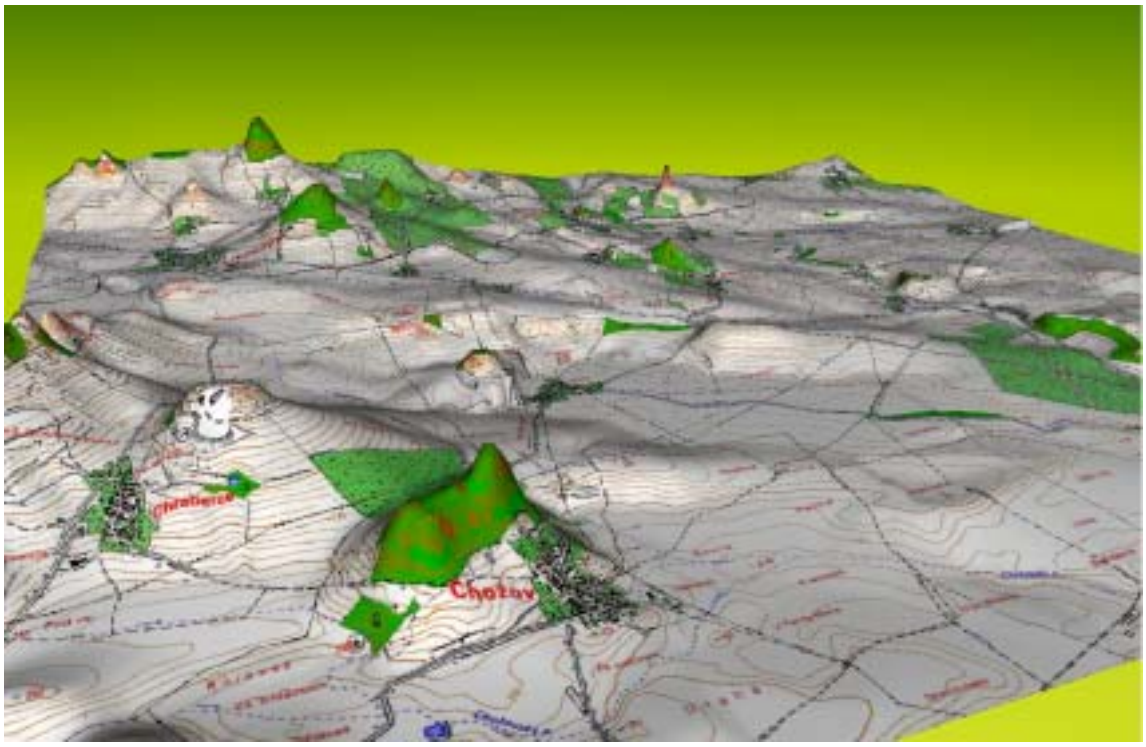
Příloha č.4.4





Základní mapa ČR 1:10 000. Model je složen z šesti mapových listů (02-34-18 až 20 a 23 až 25). Ekvidistance vrstevnic 2 m. Výškové zkreslení 3x. Vrstevnice získány exportem ze ZABAGEDu. Interval vrstevnic použitých pro generování modelu – 2m. Oba obrázky reprezentují model, který je tvořen uzlem `ElevationGrid`. Jako povrchová textura je použita černobílá ortofotomapa.

Příloha č.5



Základní mapa ČR 1:25 000, mapový list 02-344. Ekvidistance vrstevnic 5 m. Výškové zkreslení 5x. Vrstevnice získány vektorizací z RZM 25. Interval vrstevnic použitých pro generování modelu – 25m. Model je tvořen uzlem `ElevationGrid`. Jako povrchová textura je použita RZM 25.

Příloha č.6

Základní mapa ČR 1:50 000, mapový list 02-34. Ekvidistance vrstevnic 10 m. Výškové zkreslení 3x. Vrstevnice získány vektorizací z RZM 50. Interval vrstevnic použitých pro generování modelu – 50 m. Model je tvořen uzlem `ElevationGrid`. Jako povrchová textura je použita RZM 50.