Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

Diplomová práce

DYNAMICKÁ VIZUALIZACE ROZSÁHLÉHO 3D MODELU

Plzeň, 2013 Bc. Jana Strejcová

Prohlášení

Předkládám tímto k posouzení a následné obhajobě diplomovou práci zpracovanou na závěr magisterského studia oboru Geomatika na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem zadanou diplomovou práci vypracovala samostatně pod odborným vedením vedoucího práce, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí práce.

V Plzni, dne 29.5.2013 . . . . . .

Podpis

Poděkování

Na tomto místě bych ráda poděkovala vedoucímu práce Ing. Radku Fialovi, Ph.D. za ochotu, věnovaný čas a poskytnutí mnoha cenných rad a informací. Dále děkuji Ing. Karlu Jedličkovi, Ph.D. za pomoc při sestavování zadání práce. Děkuji také Ing. Janu Ježkovi, Ph.D. za ochotné poskytnutí mnoha užitečných informací.

Abstrakt

Práce se zaměřuje na řešení dynamické vizualizace rozsáhlého 3D modelu. Poskytuje přehled možných přístupů, které lze využít. Jeden z těchto přístupů je dále vybrán a realizován. Je zde popsán postup tvorby 3D modelu bloku budov a vizualizace tohoto bloku v aplikaci Google Earth. Dále je popsáno rozšíření 3D modelu pro celé město Terezín a řešení dynamické vizualizace pro takto rozsáhlý model.

Klíčová slova

Trimble SketchUp, Google Earth, KML, dynamická, vizualizace, rozsáhlý 3D model

Abstract

This thesis is focused on solving the dynamical visualization of extensive 3D model. It provides the view of some options, that are possible to use. One of these options is then realized. The technique for creating the 3D model of one block of buildings and visualization in Google Earth application is described. The enlargement of this 3D model for the whole town Terezín and the solution of dynamical visualization for that extensive model is described further.

Key words

Trimble SketchUp, Google Earth, KML, dynamic, visualization, extensive 3D model

OBSAH

Seznam použitých zkratek

LOD	Level Of Detail (úroveň detailu)
TIN	Triangular Irregular Network (nepravidelná trojúhelníková síť)
PDF	Portable Document Format
DMT	Digitální model terénu
DMR	Digitální model reliéfu
XML	Extensible Markup Language
GML3	Geography Markup Language 3
KML	Keyhole Markup Language
GML	Geography Markup Language
COLLADA	Collaborative Design Activity
VRML	Virtual Reality Modeling Language
X3D	Extensible 3D
OGC	Open Geospatial Consortium
CityGML	City Geography Markup Languague
CAD	Computer Aided Design
DWG	Drawing
DXF	Drawing Exchange Format
DGN	Design
CSG	Constructive Solid Geometry (konstruktivní geometrie těles)
WFS	Web Feature Service
WMS	Web Map Service
WGS84	World Geodetic System 1984
EGM96	Earth Gravitational Model 1996

Seznam obrázků a tabulek

1 ÚVOD

Digitální 3D modely se v posledních letech těší velké oblíbenosti. Prostorové vyjádření přináší mnohem intuitivnější informaci, než rovinné plánky a nákresy. Takovéto trojrozměrné modely jsou často barevné, uživatel si je může prohlížet, pohybovat s nimi a otáčet a tak přitahují pozornost. Proto také vznikají interaktivní prohlížeče 3D dat a proto se čím dál více firem specializuje na tvorbu 3D výstupů. Speciální kategorii tvoří také 3D modely měst, částí měst, krajiny a dalších geografických objektů, hradů, zámků, zřícenin, atd. Tyto objekty jsou často zobrazovány v kontextu s okolím, takže uživatel vidí vymodelovaný terén a na něm „posazené“ budovy, stromy, keře, … 3D vizualizace je oblíbená jak u laické veřejnosti, tak u profesionálů, protože představuje pro uživatele nenáročnou a zajímavou cestu, jak přijmout nějakou informaci.

Rozšiřující se použití 3D modelů s sebou samozřejmě přináší problémy a jejich řešení. Jak konstruovat model tak, aby co nejlépe odpovídal našim požadavkům? Jaké jsou dostupné možnosti využití softwarů, které se tvorbou 3D zabývají? Jak svůj model vizualizovat co nejefektivněji? Jak jej zpřístupnit co nejširšímu spektru uživatelů? Na tyto a ještě další otázky by měla tato práce odpovědět.

V rámci rešerše literatury získá čtenář nejprve přehled o nejčastěji používaných formátech pro reprezentaci 3D geografických dat, dále o různých způsobech vyjádření těchto dat a v neposlední řadě také o softwarech, které lze využít pro jejich tvorbu a vizualizaci. V další části se dozví podrobnější informace o jazyce KML a jeho využití pro dynamickou vizualizaci. V praktické části práce je popsáno řešení dynamické vizualizace v aplikaci Google Earth, a to na konkrétních datech zobrazujících město Terezín.

Cílem práce je tedy prozkoumat možnosti dynamického vizualizování rozsáhlých dat. To znamená vyřešit postupné načítání a zobrazování objektů (modelů budov) v aplikaci Google Earth za použití jazyka KML a testovat tento přístup na rozsáhlých datech. Dalším cílem je navrhnout metodu, jak vybírat úrovně detailu objektů v závislosti na vzdálenosti pozorovatele (v tomto případě virtuální kamery). Celý postup by měl být využit pro vizualizaci 3D modelu pevnosti Terezín.

Jak již bylo předesláno, tato práce je úzce spojena s realizací projektu „Krajina paměti. Drážďany a Terezín jako místa vzpomínek na ŠOA“. Na tomto projektu se podílí Technická

univerzita Drážďany, Památník Terezín a Západočeská univerzita v Plzni. Výstupy a poznatky, plynoucí z této práce, budou v projektu využity.

Název práce obsahuje pojmy, které jsou zásadní pro obsah této práce a zaslouží si proto podrobnější vysvětlení. Prvním velmi důležitým pojmem je ***vizualizace***. Pro tento termín existuje nespočet definic, z nichž některé zde uvedu. Vizualizace tedy znamená:

- dle Terminologického slovníku zeměměřičství a katastru nemovitostí [31] :

„Způsob viditelné grafické reprezentace numerických dat nebo způsob velmi přesvědčivé grafické reprezentace méně srozumitelných grafických dat“

- dle diplomové práce Ing. Milana Krejného [21] :

„Jakýkoliv postup, při kterém vyjadřujeme nějaké informace, data nebo hodnoty pomocí obrazu. Jedná se o zobrazení obsahu pomocí velmi názorné formy“

- dle Wikipedie [35] :

„Jakákoli technika pro vytváření obrázků, diagramů a animací pro předávání informace“

Jde tedy o zobrazování určitých informací formou různých modelů. V rámci této práce bude řešena především digitální 3D vizualizace. Jak příklad lze uvést 3D modely krajiny, budov nebo měst, které si uživatelé prohlížejí ve 3D scéně.

Dalším důležitým pojmem je slovo ***dynamická.*** Co znamená, když je vizualizace dynamická? Jde o způsob zobrazování dat, který umožňuje interaktivní prohlížení scény. Znamená to, že se obraz hýbe, otáčí, přibližuje, atd. Ve 3D vizualizacích to znamená možnost k modelům přistupovat pomocí klasických nástrojů virtuálních 3D prohlížeček (rotace, posun, zoom, ...). Dynamickou vizualizací však může být i video (průletové animace či měnící se sekvence snímků). V rámci práce je pod pojmem *dynamická* myšleno, že jednotlivé modely budov se ve 3D scéně načítají a zobrazují postupně a také se postupně mění jejich úroveň detailu v souvislosti s přibližováním/oddalováním.

Poslední pojem, vyskytující se v názvu práce, je ***rozsáhlý 3D model.*** Nutno podotknout, že se jedná o digitální modely. V praxi neexistuje přesně definovaná hranice, která by nám říkala, kdy je model rozsáhlý. Záleží na tom, kolik detailů chceme zobrazovat. Pokud například řešíme detailně jednotlivé objekty v domácnosti (nábytek, zařízení, interiéry, ...), pak model celého domu i s těmito objekty můžeme považovat za rozsáhlý. Pokud ale modelujeme dům jako takový a jde nám pouze o detailní zobrazení exteriéru, pak

model celého města nebo části města složen z takovýchto domů je rozsáhlým. V rámci práce je pod pojmem *rozsáhlý* myšlen model města Terezín.

Při dynamické vizualizaci rozsáhlých modelů jsou častým problémem velké objemy dat, spojené s pomalým vykreslováním scény. Jsou tak kladeny velké nároky na výpočetní techniku. Celý proces lze však různými cestami optimalizovat a ovlivňovat jak velikost, tak způsob ukládání a načítání dat. Této problematice bude věnována část práce.

2 3D MODELOVÁNÍ

2.1 Formáty reprezentace 3D (geo)dat

V současné době se pro ukládání 3D modelů používá celá řada formátů. Některé z nich jsou standardizovány (např. KML, CityGML, …), jiné jsou jen proprietárními formáty producentů software (např. DWG, DXF, Multipatch, …). Řada formátů je založena na značkovacím jazyce XML. Tato kapitola uvádí přehled těch nejznámějších a často používaných formátů, používaných pro reprezentaci 3D geografických dat.

Jedním z 3D formátů založených na XML je **CityGML** *(City Geography Markup Language)* **–** otevřený datový model sloužící pro reprezentaci, ukládání a výměnu virtuálních 3D objektů, týkajících se města. Poskytuje 3D potenciál pro vizualizaci v různých aplikacích. Jedná se o standard OGC, je otevřený a zadarmo. Je implementován jako aplikační schéma pro GML3 (což je modelovací jazyk pro geografické informace, mezinárodní OGC standard pro výměnu prostorových dat). CityGML dodává mechanismus pro popis 3D objektů s ohledem na jejich geometrii, topologii a sémantiku. Nezaměřuje se pouze na grafický vzhled modelů, ale také na tématické vlastnosti a vztahy mezi prvky. Tematický model je tvořen základními třídami, přičemž nejdetailněji propracovaná je třída budov. CityGML soubory mohou obsahovat reprezentaci objektů v několika úrovních detailu (LoD, Level of Detail) od LoD0 až po LoD4, ve které jeden model může obsahovat současně prvky různých úrovní. Geometrie 3D objektů je zde založena na reprezentaci hranic (viz kapitola 2.2.2).
Cílem vývoje CityGML je dosáhnout definice základních entit, atributů a vztahů pro 3D model města. [7, 23, 27]

CityGML definuje následujících 5 úrovní detailu (LoD):

- **LoD0**

– tato nejhrubší úroveň detailu je 2D mapou nebo 2.5D modelem terénu, přes který lze přetáhnout například letecký snímek
– budovy jsou reprezentovány půdorysem (FootPrint) nebo nebo promítnutím střechy na vodorovnou rovinu umístěnou ve výšce paty střechy (RoofEdge).
– měřítkově: regionální, krajina

Obr. 1 : Vysvětlení FootPrint a RoofEdge polygonů

zdroj [23]

- **LoD1**

– různé části budov jsou spojeny do jednoduchého bloku a nejsou zobrazeny detaily, takže tento známý blokový model obsahuje pouze prosté kvádry

– jedná se o hranové budovy s jednotnou strukturou střechy

– měřítkově: město, region

- **LoD2**

– úroveň detailu je zde vyšší, bloky jsou doplněny o tvary střech s rozlišenou strukturou střechy

– dále jsou přidávány menší části budovy jako např. vikýře, komíny, balkony, venkovní schodiště, atd. Tyto přidané prvky jsou však zobrazeny velmi jednoduše, schématicky, bez detailů

– měřítkově: městské čtvrti

- **LoD3**

– v této úrovni jsou kladeny větší nároky na geometrickou přesnost, jsou zobrazeny menší prvky budovy (antény, ...) a také se detailně modelují všechny charakteristické prvky budovy, zobrazené v LoD2 (vikýře, balkony, komíny, ...)

– tato úroveň je velmi složitá a časově náročná na zpracování, ovšem nejlépe odpovídá skutečnosti, celý exteriér budovy je detailně zpracován

– je zde zobrazena struktura střechy i fasáda včetně vymodelovaných oken a dveří

– měřítkově: architektonické modely (exteriéry), pamětihodnosti

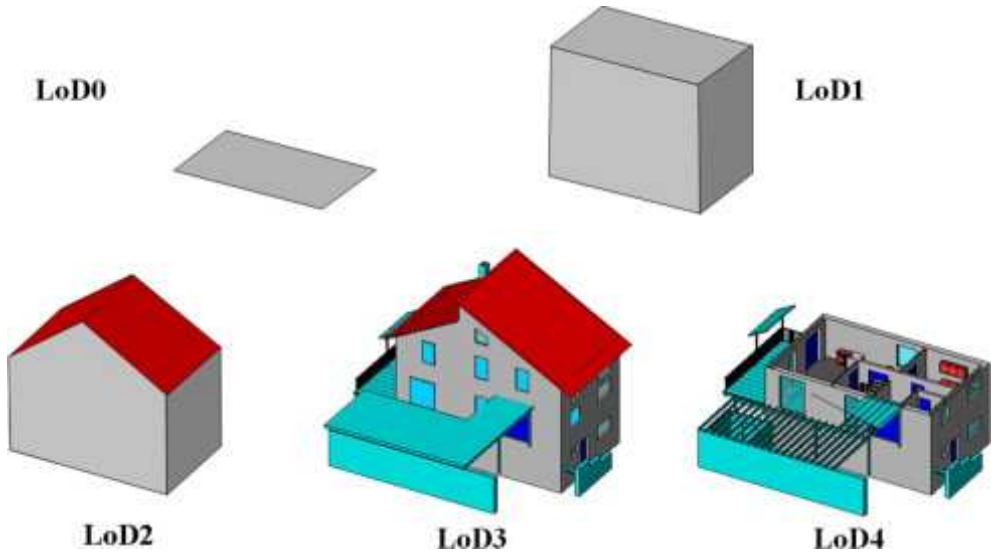
- **LoD4**

– tato úroveň je bezpochyby nejdetailnější, dovršuje LoD3 tím, že do něj přidává vnitřní strukturu pro budovy

– co se týká exteriéru budovy, ten je totožný jako v LoD3, protože hlavní rozdíl mezi těmito úrovněmi je právě v interiéru

– budovy v této úrovni detailu mají pokoje, vnitřní dveře, schody, nábytek, …
– měřítkově: architektonické modely (interiéry)

Ve všech úrovních lze do modelu přidat vzhledovou informaci např. přidáním textur na povrch objektu. Lze také definovat i materiál a světelné chování objektu. [7, 8, 17, 23, 27]



Obr. 2 : Jednotlivé úrovně detailu dle CityGML specifikace
zdroj [23]

KML (*Keyhole Markup Languague*) je značkovacím jazykem pro vizualizaci geodat. Byl vyvinut firmou Keyhole, Inc. Od roku 2008 je tento formát otevřeným standardem OGC, vychází se syntaxe XML a je používán v aplikaci Google Maps či Google Earth. Vedle nativního formátu KML existuje ještě rozšířená verze KMZ, která je ve skutečnosti přejmenovanou příponou archívu ZIP. Obsahuje kořenový soubor *doc.kml* a vedle něho další dodatečné soubory (např. pro textury). [19, 22, 27]

GML *(Geography Markup Language)* je značkovací jazyk pro geografické prvky. Je založen rovněž na bázi XML, takže se dá snadno číst i editovat v libovolném textovém editoru. Jedná se o standard OGC a splňuje také normu ISO (ISO 19136:2007). [9, 22]

Formát **COLLADA** *(COLLAborative Design Activity)* je spojen s počítačovou grafikou a používá se pro ukládání 3D objektů a animací. Je dalším z řady formátů, založených na otevřeném XML schématu. COLLADA soubory mají příponu DAE.

Je zapouzdřena ve formátu KMZ a představuje tedy výchozí formát pro popis 3D modelů v aplikaci Google Earth. [17]

Multipatch je datový formát od firmy ESRI, který pro geometrii 3D objektů využívá hraniční reprezentaci založenou na trojúhelnících (viz kapitola 2.2.2). Dá se konstruovat, editovat a ukládat v softwaru ArcGIS od téže firmy. Je kompatibilní s formátem DAE a SKP. Tento formát představuje možnost konstrukce plné 3D reprezentace objektů, neomezené pravidly 2D (např. svislé stěny). Dovoluje uživateli ukládat textury, barvy i průhlednosti povrchů. Používá se pro modelování jak jednoduchých objektů, jako jsou sféra nebo krychle, tak i komplexnějších objektů, jako například stromů, budov nebo izoploch. [17]

Jazyk **VRML** *(Virtual Reality Modeling Language)* slouží k popisu virtuální reality jako způsob textového zápisu virtuálních světů, které vytváří z tzv. CSG primitiv (jednoduchých geometrických objektů – viz kapitola 2.2.5). Popisuje tedy 3D scény a objekty a umožňuje je publikovat na webu, a to pomocí statických i dynamických jevů (využívá aktivní a pasivní objekty pro popis reality). Pro povrchy objektů lze nastavovat různé barvy, textury, odrazivost nebo průhlednost. K popisu scény využívá hierarchickou stromovou strukturu, kde objekty jsou definovány v uzlech a listech stromu. Obsah VRML je uložen v souboru WRL. K prohlížení v prostředí Internetu slouží VRML prohlížeče, které existují buď jako plug-in standardního prohlížeče, nebo jsou implementovány přímo. Poslední verze VRML obsahuje i nástavbu pro geodata – GeoVRML. Ta umožňuje lokalizovat modely v různých souřadnicových systémech. V současné době je formát VRML nahrazen novým formátem X3D. [9, 11, 27]

X3D *(eXtensible 3D)* je formát pro ukládání 3D scén a objektů. Koncepčně vychází z VRML, důležitou změnou oproti VRML však je především jeho XML syntaxe. X3D dokument tak může být mnohem jednodušeji zpracováván s využitím různých knihoven pro práci s XML. Tento formát byl přijat jako ISO standard. [16, 36]

Mezi další známé a velmi často používané formáty pro reprezentaci 3D dat patří ty, které vycházejí z CAD technologií. Jedná se o formát DWG a DXF, které jsou nativní pro software AutoCAD od firmy Autodesk. **DWG** *(DraWinG)* je neveřejný souborový formát pro 2D a 3D výkresy. Dnes je velmi široce rozšířen a pro jeho prohlížení slouží bezplatná aplikace *Autodesk DWG TrueView*. Tuto aplikaci lze využít i pro převod DWG souborů

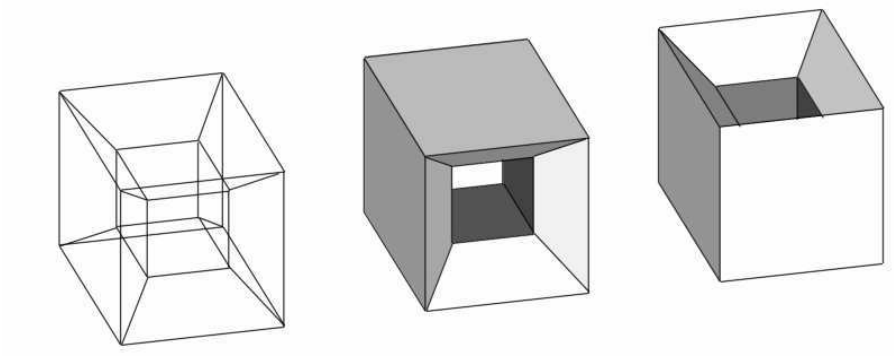
do starší verze formátu. **DXF** *(Drawing eXchange Format)* zajišťuje komunikaci a přenos dat mezi CAD aplikacemi. Dalším formátem je **DGN** *(DesiGN)*, který používá software Microstation od firmy Bentley. [6, 9]

2.2 Typy 3D modelů

Postup vytváření obrazu v trojrozměrném prostoru je poměrně složitou úlohou. Existuje k ní několik řešení, které vedou k různým metodám vyjádření 3D objektů. Následující kapitola uvádí používané způsoby 3D reprezentací těles.

2.2.1 Drátový model

Nejstarší a nejjednodušší metodou, jak popsat povrch tělesa, je tzv. drátový model (*wire-frame*). Jedná se o **hranovou reprezentaci**. Jde o graf, jehož vrcholy jsou body v prostoru a mezi nimi jsou hrany. Při implementaci této struktury je tvořen jeden seznam vrcholů a jeden seznam hran, přičemž každá hrana obsahuje dva ukazatele na seznam vrcholů. Prostým zobrazením hran a vrcholů však nepopíšeme topologické vztahy mezi prostorovými objekty, a proto tuto strukturu nelze jednoznačně interpretovat. Jeden model tak může být chápán jako několik různých těles. Další nevýhodou této metody je, že se nehodí pro popis objektů s křivočarými povrchy. Naopak výhodou zůstává možnost pozorovat vnitřek tělesa, který by jinak zakryly hraniční plochy. [26, 27, 38]



Obr. 3 : Nejednoznačnost drátového modelu

zdroj [26]

2.2.2 Povrchový model

Při aproximaci křivočarých povrchů se používá náhrada mnohostěnem. Tento přístup spadá do kategorie **hraniční reprezentace** (*Boundary Representation*, zkráceně *B-Rep*)

a přidává k vrcholům a hranám ještě plochy. Tyto plochy, často trojúhelníky či čtyřúhelníky nebo obecně polygony, jsou uspořádány do sítě. Implementačně jde o množinu vrcholů, hran a polygonů, přičemž hrana je tvořena dvěma vrcholy a polygon je tvořen uzavřenou sekvencí hran. Hrana je sdílena dvěma polygony a vrchol je sdílen alespoň třemi hranami. Tato reprezentace pomocí hraničních ploch nám dává o tělese více informací, než předchozí drátový model. Jednotlivé plochy přitom nemusí nutně být rovinné – tam, kde mnohostěnový model nestačí, používá se model složený např. z kvadratických ploch (části paraboloidu, elipsoidu, hyperboloidu, ...)

Pro reprezentaci geografických dat jsou povrchové modely velmi často používané a členěné do škály různých druhů a typů. Nejzákladnějšími zástupci jsou TIN – nepravidelná trojúhelníková síť pro vektorová data a GRID – pravidelná čtvercová síť pro rastrová data. Oba modely se používají pro reprezentaci povrchů, jako např. digitální model terénu (DMT) a reliéfu (DMR). TIN je síť trojúhelníků, které jsou definovány třemi body umístěnými kdekoliv v prostoru. Tyto trojúhelníky bývají konstruovány tak, že splňují tzv. Delaunay kritérium, kdy v kružnici opsané danému trojúhelníku nesmí ležet žádný další vrchol. Toto kritérium ovšem nemusí nutně být splněno (např. když jsou v modelu tzv. povinné hrany). GRID je tvořen pravidelnou maticí čtvercových buněk, kdy každá čtvercová ploška nese informaci o výšce (výška je konstantní a vztahuje se ke středu buňky). [5, 27, 28, 32, 38]

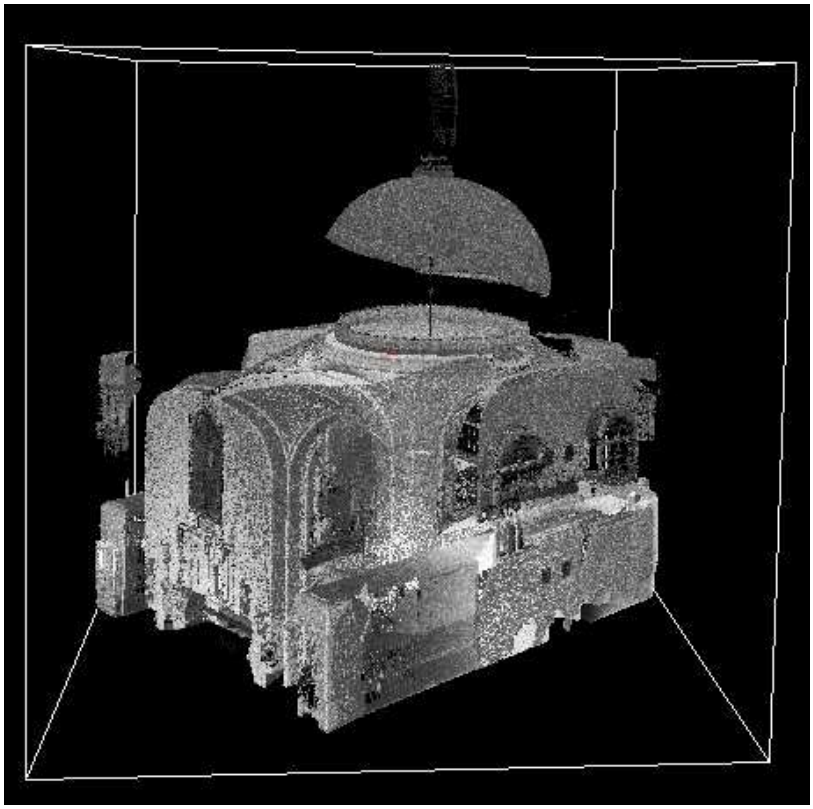
Obr. 4 : Povrchový model konvičky z polygonů

zdroj [29]

2.2.3 Bodová reprezentace

Tento speciální případ hraniční reprezentace je založen na myšlence, že povrch tělesa lze vyjádřit velmi hustým shlukem bodů. Tyto body tvoří tzv. *mračno bodů* a jsou obvykle získávány snímáním reálných objektů. Mračno bodů je typickým výstupem pozemního

i leteckého laserového skenování. Každý bod kromě informace o své poloze může obsahovat i další informace (o normálovém vektoru, barvě, ...). Tato reprezentace je velmi náročná na paměť. [15, 26, 27]



Obr. 5 : Ukázka mračna bodů při pozemním laserovém skenování budovy
zdroj [15]

2.2.4 Voxel

Dalším způsobem, jak popsat 3D těleso je **objemová reprezentace**. Objekty jsou děleny na malé elementy objemu – voxely. Voxel (z angl. *volume element*) je analogie dvourozměrného pixelu (nazývaný někdy *3D pixel*). Buňky jsou uspořádané v pevné pravidelné mřížce. Voxel je krychle (nejmenší element prostoru), která má na celém svém objemu konstantní hodnotu dané veličiny, vztahující se k jejímu středu. Výsledný objekt je tedy „poskládán“ z těchto jednotek. Lze tak snadno reprezentovat skutečné 3D objekty. Praktické využití nacházejí voxely např. při reprezentaci objemových dat v medicíně (data z tomografu, ...). Nevýhodou je však velká náročnost na výkon i paměť počítače a proto je tento přístup pro geografická data využíván jen zřídka. [26, 27, 32, 38]

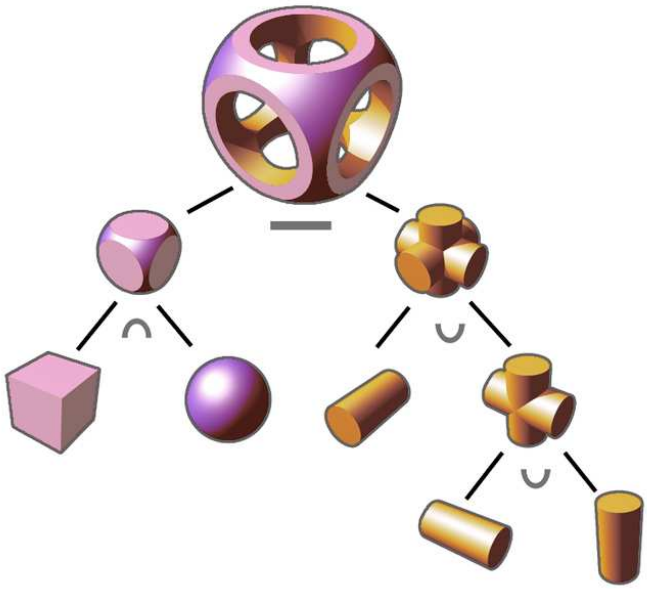
Obr. 6 : Princip objemové reprezentace pomocí voxelů

(vlevo: voxelová konvička , vpravo: polygonální konvička)

zdroj [18]

2.2.5 Konstruktivní geometrie těles

Tato metoda, známá spíše pod anglickou zkratkou *CSG – Constructive Solid Geometry*, používá jako základ množinu jednoduchých 3D objektů (kvádr, válec, koule, kužel, hranol, ...). Tyto objekty, zvané CSG primitiva, se kombinují v určité pozici, zvětšení a orientaci pomocí booleovských operací (průnik, sjednocení a rozdíl). Výsledkem je 3D těleso nebo prázdná množina. Model je reprezentován stromovou strukturou, ve které listy odpovídají jednotlivým elementárním tělesům a vyšší uzly představují booleovské operace. Stromová struktura uchovává historii dílčích konstrukčních kroků a mohou do ní vstupovat i celé CSG stromy (již zkonstruované modely). Tato metoda je výpočetně velmi náročná. Používá se ve fázi vytváření tělesa, pro jeho vykreslování však již není vhodná. A tak přestože existují metody, které nám umožní zobrazit přímo CSG těleso, většinou se pro zobrazení přejde k jiné reprezentaci, a to objemové nebo hraniční (nejčastěji jako síť trojúhelníků). [20, 26, 27, 38]



Obr. 7 : Popis tělesa CSG stromem

zdroj [37]

2.2.6 Zobecněné válce

Tato metoda je někdy řazena do skupiny **tahových reprezentací**. Používá se pro popis objektů s výraznými symetriemi. Zobecněný válec je definován pomocí křivky v prostoru zvané *páteř* a plochy průřezu objektu v každém bodě páteře. Výsledný objekt je tedy tvořen tažením rovinného objektu po určité trajektorii dané páteří. Obvykle je každý průřez kolmý na páteř. Průřez lze reprezentovat parametricky, kde parametr je posun podél páteře. Páteř nemusí být v průřezu obsažena. [38]

2.3 Přehled aplikací

V této kapitole budou na základě zdrojové literatury popsány různé aplikace, které se používají pro tvorbu a vizualizaci 3D dat. Speciálně je zaměřeno na ty, které lze využít pro 3D modely budov, bloků budov nebo celých měst. Kapitola uvádí základní přehled o těch nejznámějších a nejvíce používaných způsobech vizualizace 3D geografických dat.

Vizualizace 3D objektů lze samozřejmě uživatelům zpřístupnit ve formě různých 2D pohledů, obrázků modelu, „screenshotů“ modelu provedených z různých úhlů pohledu. Další možnost, jak uživateli model více přiblížit, je tvorba videí, ve kterých je model přibližován/oddalován, natáčen atd. U rozsáhlejších modelů (například část města) jsou oblíbené tzv. průletové animace, ve kterých je sledována určitá trasa (sleduje silnice, cesty mezi domy, ...). Výhodou prezentování modelů touto cestou je nenáročnost na software,

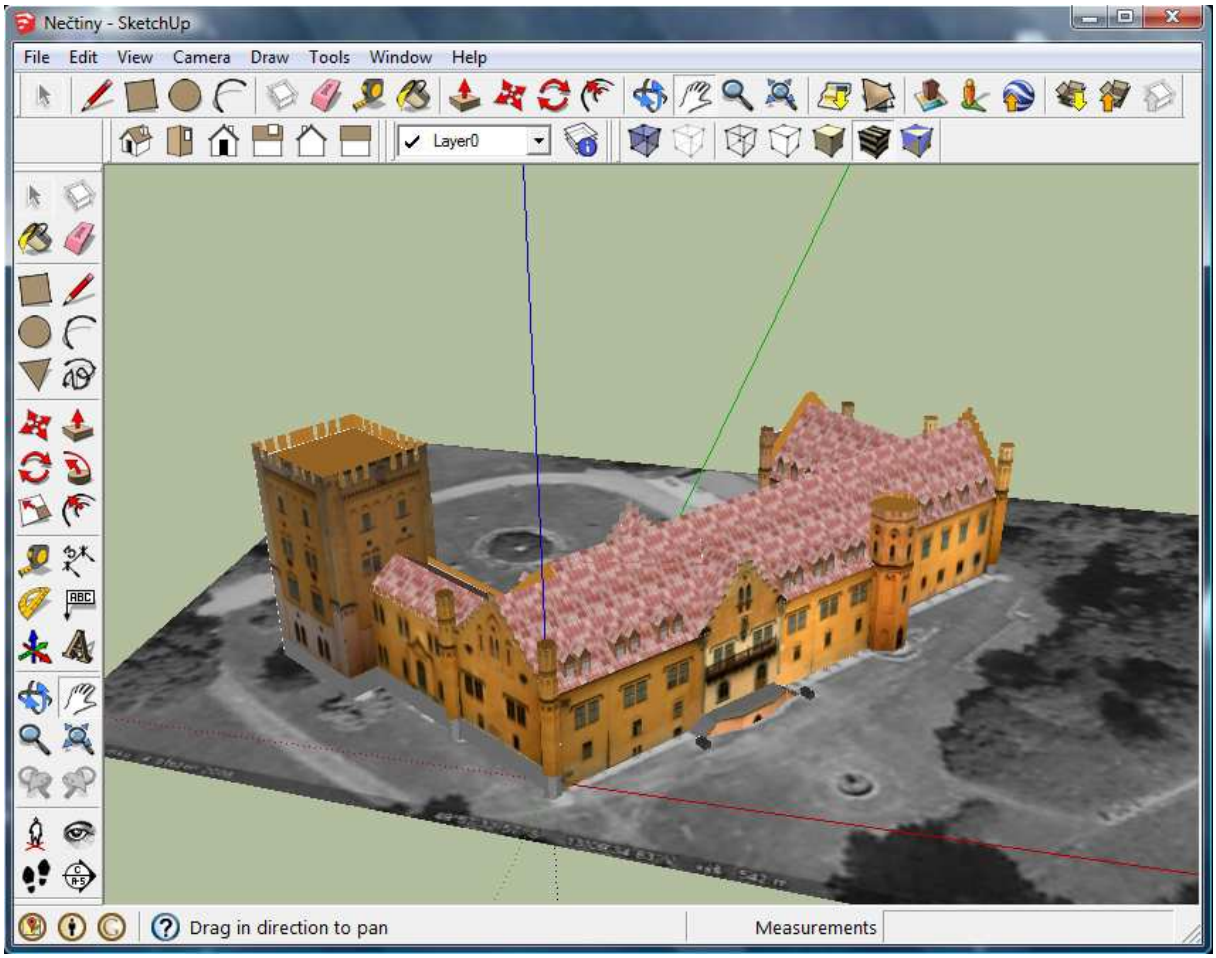
uživatel nepotřebuje žádný speciální nástroj, nezajímá ho velikost modelu ani rychlost vykreslování textur, pouze si prohlédne obrázek nebo spustí video. Tento způsob ovšem nevede k přístupu k plné 3D reprezentaci modelu, proto se jím tato práce nebude zabývat.

2.3.1 Trimble SketchUp

Trimble SketchUp je software pro modelování ve 3D. Původně byl produktem firmy Last Software, v roce 2006 byl odkoupen společností Google a následně v roce 2012 jej odkoupila společnost Trimble. Tento software je v současné době velmi oblíben u široké veřejnosti. Jeho ovládání je intuitivní a uživatelsky příjemné. Nejnovější je verze 8 této aplikace. Trimble SketchUp je volně stažitelný software a umožňuje jak tvorbu vlastních 3D modelů, tak sdílení modelů ve veřejné knihovně 3D objektů – ***3D Warehouse***. V této knihovně uživatelé dávají k dispozici své modely, a zároveň si mohou stáhnout a použít modely ostatních. Jsou zde k dispozici nejen modely celých budov, ale i části budov, interiéry, jednotlivé předměty (nábytek, auta, mosty, stromy, dopravní značky, lidé, ...). Nativní formát této aplikace je SKP. Model však lze vyexportovat do formátu KMZ, nebo DAE. [33, 34]

Existuje také komerční verze této aplikace, Trimble SketchUp Pro, která nabízí navíc některé nástroje a také je zde více kompatibilních formátů. [20, 34]

Trimble SketchUp je také určen pro tvorbu 3D modelů pro Google Earth (viz dále). Aby mohl model být přidán do prostředí Google Earth, musí být lokalizován a musí být přijat a schválen. Proces schválení probíhá tak, že uživatel svůj model nahraje do 3D Warehouse a zaškrtne přitom možnost „*Google Earth ready*“. Během několika dní je informován o tom, zda byl model akceptován a tak zpřístupněn „celému světu“. [34]



Obr. 8 : Ukázka prostředí Trimble SketchUp

Výhodou aplikace Trimble SketchUp je jistě jeho volně stažitelná verze, ve které si uživatelé mohou vytvářet, sdílet a editovat 3D modely, využívat knihovnu 3D Warehouse, mohou také lokalizovat svůj model a pokusit se jej vizualizovat v Google Earth.

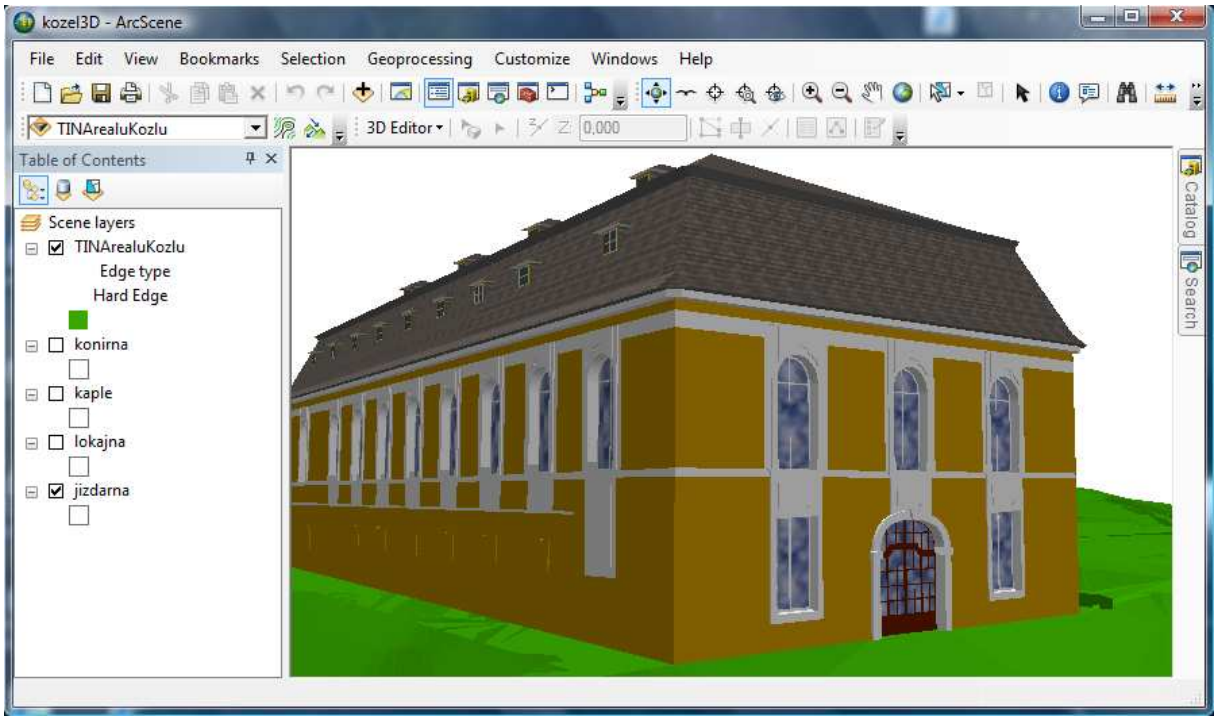
2.3.2 ESRI ArcScene

ArcScene je 3D vizualizační aplikace platformy ArcGIS od společnosti ESRI. Je vhodná pro vytváření perspektivních scén, které umožňují pracovat s 3D vektorovými nebo rastrovými daty. Co se týče digitálního vyjádření povrchu, ArcScene podporuje tvorbu a zobrazování kontinuální reprezentace povrchu jako TIN. Všechna data jsou načtena do paměti, což umožňuje poměrně rychlé vykreslení scény při používání funkcí jako např. posun, zoom, „otáčení s celou scénou“ atd. Vektorová data jdou vykreslována jako vektor, rastrová data jsou buď převzorkována nebo konfigurována do nastaveného počtu řádků/sloupců.

Aplikace je doporučována pro vizualizaci menšího objemu dat. Oproti aplikaci ArcGlobe (viz níže) je zde výhodou možnost tvorby průletových animací. [2]

Většina autorů literatury, ze které je čerpáno toto téma, využívá aplikaci pro tvorbu a ukázku svých vlastních modelů. Modely jsou buď konstruovány přímo v ArcScene nebo jsou vytvářeny v jiných aplikacích. Častý přístup je tvorba modelu v softwaru Trimble SketchUp a import výsledného modelu ve formátu SKP do ArcScene. Jinou možností je export SketchUp modelu do jiného formátu (např. DAE), který lze načíst do ArcScene. [3, 20] Další aplikací, využívanou pro tvorbu 3D modelů, je Bentley Microstation. Zde je výsledný model ve formátu DGN a lze jej případně vyexportovat do několika formátů, které lze načíst do ArcScene (např. DAE). [20]

Jak již bylo řečeno, aplikace ArcScene nabízí možnost vizualizace modelu formou animací. V ovládacím panelu *Animation* jsou k dispozici potřebné nástroje – funkce *Fly* umožňuje „létání“ a *Record* nahrává daný průlet. Cestu letu, podél níž se pohybuje kamera, lze vytvořit i graficky (*3D Graphics – New Line*). V nástroji *Animation Manager* můžeme jednotlivá videa kontrolovat a editovat. Dále můžeme sledovat jednotlivé snímky (pohledy), střídat je a z toho tvořit animace. Tento postup lze dobře uplatnit například při tvorbě změnových animací, kdy se střídají snímky jedné oblasti, které mapují dané období. Po ukončení tvorby animací je pomocí funkce *Export Animation* uložíme do formátu AVI. [20]

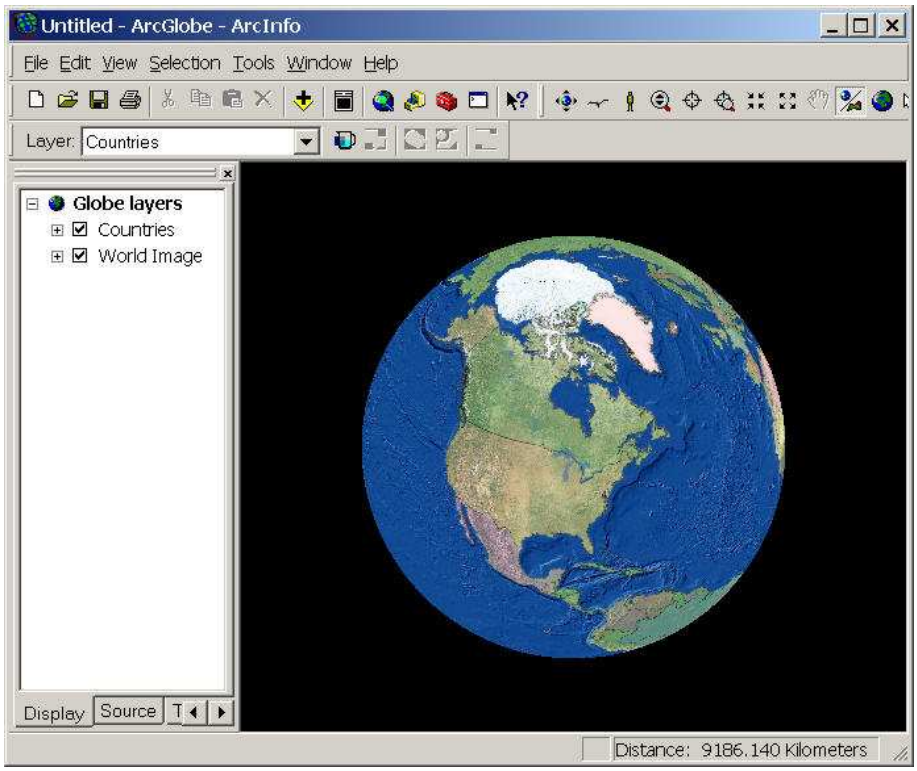


Obr. 9 : Ukázka prostředí ESRI ArcScene

2.3.3 ESRI ArcGlobe

ArcGlobe je 3D vizualizační aplikace od společnosti ESRI. Umožňuje prohlížení GIS dat na povrchu glóbu. Narozdíl od aplikace ArcScene, která není vhodná pro velké objemy dat, je tato aplikace firmou ESRI doporučována právě pro velké objemy dat. Umožňuje bezešvou vizualizaci rastrových i vektorových dat. Data jsou zobrazována v různých úrovních detailu (LoD), které jsou organizovány do dlaždic. Vektorové objekty jsou zpravidla rasterizovány a zobrazovány podle odpovídající úrovně detailu, což vede k rychlému vykreslování scény.

Aplikace pracuje s daty odlišným způsobem, než ArcScene. S tím souvisí naříklad možnost vykreslování v závislosti na měřítku, které v aplikaci ArcScene není podporováno. [2]



Obr. 10 : Ukázka prostředí ESRI ArcGlobe

Nevýhodou obou aplikací – jak ArcScene tak ArcGlobe – je samozřejmě skutečnost, že se jedná o komerční software. Tudíž model v těchto prostředích si prohlédnou pouze ti uživatelé, kteří mají licenci a používají software ArcGIS. Nehodí se proto v případech, kdy chceme model zpřístupnit pro široké spektrum uživatelů, ale spíše v případech, kdy chceme zpřístupnit model úzké skupině odborníků nebo kdy chceme, aby model byl dosažitelný pouze z lokálního disku konkrétního počítače (kiosku).

2.3.4 Google Earth

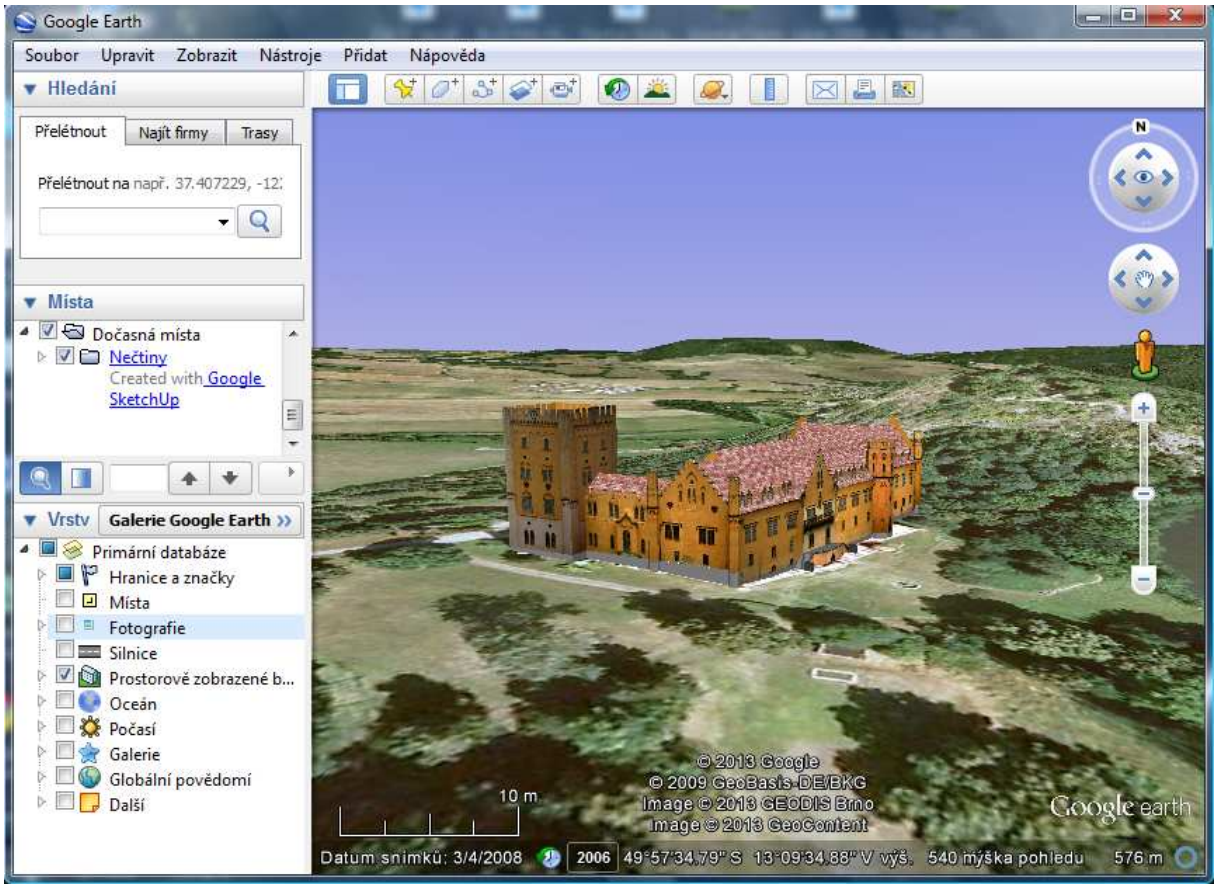
Google Earth je interaktivní aplikace umožňující uživatelům pohled na virtuální glóbus. Původně je to produkt firmy Keyhole, v roce 2004 byl zakoupen společností Google. Povrch glóbu je pokryt satelitními snímky s možností vypínat a zapínat další překryvné vrstvy, zobrazující např. silnice, popisky, 3D budovy, hranice a další geografické prvky. Kromě zobrazení planety Země jsou ještě k dispozici další tělesa, jako Mars nebo Měsíc. V režimu Obloha mohou uživatelé zkoumat vesmírné objekty.

Také existuje ***Galerie Google Earth*** – veřejná knihovna 3D objektů v Google Earth (těch, které jsou přijaty a schváleny). V této aplikaci je tedy možné prohlížet krom jiných dat i 3D modely, vytvořené jinými uživateli a nahrané do této knihovny. Jedná se například o budovy ve městech, hrady, zámky, a další.

V této aplikaci lze vizualizovat vlastní model, nelze jej však v Google Earth přímo vytvářet. Model tedy musí být konstruován v jiné aplikaci. Častý přístup je tvorba modelu v softwaru Trimble Sketch Up (viz výše) a jeho export do formátu KML / KMZ. [27, 33]

Při vizualizaci vlastního modelu v Google Earth máme k dispozici dva přístupy:

1) Svůj model nechceme zpřístupnit pro všechny uživatele této aplikace. Chceme jej ukázat pouze určité skupině lidí. V tomto případě nám stačí vizualizace přes KML / KMZ soubor nahraný na lokálním disku nebo publikovaný na webu. Nemusíme se tak zabývat procesem schválení modelu a lze tak vizualizovat i modely, které nejsou např. topologicky čisté.
2) Svůj model chceme zpřístupnit „celému světu“, tedy jej musíme nahrát do Galerie Google Earth. Na tento model se pak může podívat každý uživatel aplikace Google Earth. V tom případě model musí být správně lokalizován a musí projít procesem schválení (viz kapitola 2.3.1)



Obr. 11 : Ukázka prostředí Google Earth

Nespornou výhodou Google Earth je skutečnost, že aplikace není komerční, je tudíž zdarma dostupná všem. Uživatelům jsou k dispozici již zmíněné překryvné vrstvy a hlavně modely objektů, nahrané v Galerii Google Earth. Tudíž pro zpřístupnění svého modelu co nejširšímu spektru uživatelů je tato velmi známá a oblíbená aplikace ideální. Model se navíc zobrazí v kontextu s okolím – okolní terén, popř. jiné 3D objekty v blízkosti daného místa. [27]

2.3.5 Adobe Reader a 3D PDF

Formát PDF od firmy Adobe Systems je dnes již široce rozšířen. Je to přenositelný formát, ve kterém se všem uživatelům zobrazí obsah dokumentů stejně. Do dokumentu je možné vkládat i audio či video nahrávky.

Se stejným záměrem byl pro 3D objekty vytvořen i formát 3D PDF. Pro prohlížení dat v tomto formátu se používá program Adobe Reader verze 7 a vyšší. Adobe Reader je volně stažitelný a umožňuje uživatelům kromě prohlížení a tisku klasických PDF dokumentů také interaktivní prohlížení 3D objektů (uložených ve formátu 3D PDF) nezávisle na hardware či software. Umožňuje 3D modely natáčet, posouvat, měřit, měnit osvětlení a barvu pozadí,

přepínat mezi režimy vykreslování modelu, atd. Modelem lze dokonce procházet pomocí hypertextových odkazů a záložek.

Přirozeně vyvstává otázka, jak dostat svůj model do formátu 3D PDF. Z prostředí ArcScene nelze přímo vyexportovat 3D PDF, lze však provést export do VRML a tento soubor pak pomocí jiného nástroje zkonvertovat do 3D PDF. Z prostředí SketchUp lze exportovat 3D PDF pomocí nástroje *RPS 3D PDF*. Z prostředí MicroStation lze přímo exportovat model do 3D PDF. Existuje také nástroj *3D PDF Converter* od firmy Tetra 4D, který umožňuje konverzi dat z CAD systémů (jako například CATIA, Pro/ENGINEER, SolidWorks,...) právě do formátu 3D PDF. [1, 4, 27]

Výhodou programu Adobe Reader ve spojení s formátem 3D PDF je jeho dostupnost, otevřenost (je volně stažitelný) a rozšířenost. Těchto vlastností lze široce využít ve chvíli, kdy chceme uživateli předat model bez obav, zda jej uvidí a bez nutnosti instalovat speciální aplikaci.

2.3.6 TERRA software

Software TERRA je produktem americké firmy Skyline Software Systems, jejíž distributor v České republice je firma Geometra Opava. Terra technologie umožňuje vytvářet, analyzovat a vizualizovat 3D data. Pomocí tohoto softwaru bylo již zpracováno několik projektů, týkajících se 3D vizualizace různých míst (nejen) v ČR, jako například Hradec Králové, Blansko, Znojmo, Frýdlantsko, Novobystřicko, Vimperk nebo zřícenina Zubštejn. Dále také město Plzeň a Liberec. Ze zahraničních projektů může být příkladem Londýn, Paříž nebo Sydney.

TERRA software se skládá ze tří částí:

1) **TerraExplorer** je desktopová prohlížečka. Je to volně stažitelná aplikace, která nabízí interaktivní prohlížení a editaci 3D scén. Její ovládání je intuitivní a k dispozici je i česká jazyková verze. TerraExplorer je rodina produktů, které používají stejnou technologii, ale existují v různých verzích: *TerraExplorer Viewer* (základní prohlížečka), *TerraExplorer Plus* (oproti základní verzi nabízí navíc import rastrových a vektorových vrstev a pokročilou editaci), a *TerraExplorer Pro* (přidává možnost publikace dat a některé nástroje pro editaci a pro práci s vektorovými vrstvami). *TerraExplorer Professional* je určen pro tvorbu a editaci datové náplně projektů, podporuje široké spektrum formátů pro vstup dat a umožňuje napojení na databázové zdroje.

2) **TerraBuilder** je desktopová aplikace pro vytváření 3D scény. Tato scéna typicky vzniká kombinací digitálního modelu terénu a leteckými nebo satelitními snímky. Na takto vytvořený terén se přidávají další 2D nebo 3D vrstvy, například modely budov nebo turistické trasy. TerraBuilder podporuje import dat v různých formátech a souřadnicových systémech.
3) **TerraGate** je serverová aplikace pro zpřístupnění projektů na webu.

TERRA software ve svých 3D scénách zobrazuje popisná data (odkazy, texty, ikony, ..), 2D liniová a plošná data (turistické trasy, chráněná území, dopravní infrastrukturu, ...), 3D data (modely budov, vegetace, ...). K dispozici jsou různé analytické nástroje, měřicí nástroje, nástroje pro nahrávání průletů, vytváření pohledů, atd. Nejnovější je verze 6 této aplikace. [10, 30]

Pokud bychom chtěli pouze jako uživatelé prohlížet již hotové 3D modely měst, je tento software ideální řešení. Je nutné si nainstalovat prohlížecí aplikaci, ta je však volně stažitelná a nabízí intuitivní ovládání a klasické nástroje pro práci s 3D scénou. Pokud ovšem chceme vytvářet, vizualizovat a publikovat vlastní 3D modely, potřebujeme k tomu i ty části softwaru (TerraBuilder a TerraGate) , které jsou již komerční. Celá tato technologie je spíše komplexním řešením pro firmy (jako např. Geometra Opava), které se touto činností zabývají profesionálně.

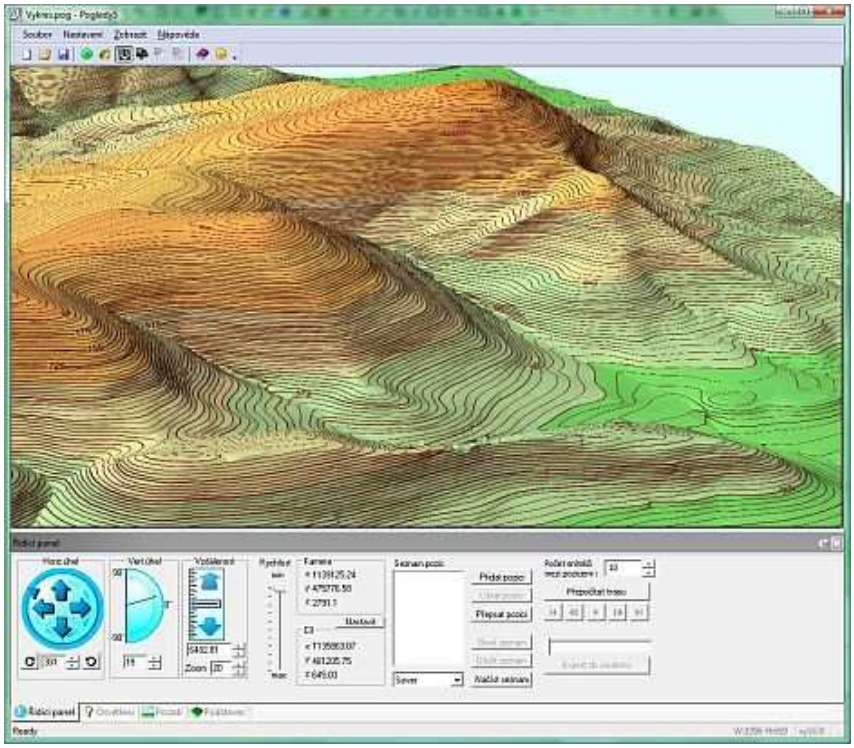
2.3.7 ATLAS DMT

Český programový systém ATLAS DMT je produktem společnosti ATLAS s.r.o. . Je využíván geodety pro digitální modelování terénu, dá se však použít i k trojrozměrným vizualizacím krajiny a měst. Jeho součástí jsou různé speciální funkce a moduly, vyvinuté pro projektování komunikací, liniových sítí a těžebních prací, k tvorbě vrstevnicových plánů a terénních profilů. Systém je určen pro práci s terénem, kde základem modelu terénu je TIN. Společnost ATLAS úzce spolupracuje se společností GEPRO a proto velmi dobře funguje propojení systémů KOKEŠ, MISYS a ATLAS DMT. Je umožněno sdílení dat i funkcí.

Jedním z modulů systému je i vizualizační modul, který slouží k perspektivnímu pohledu na model a k tvorbě vizualizací jak statických (pohledy), tak dynamických (průlety). Součástí vizualizace mohou kromě terénu být i budovy, stromy, atd.

Jedná se o komerční software, je však možné si jej zdarma zapůjčit a vyzkoušet. Tento program byl využit v diplomové práci Milana Krejného, který v něm zpracoval 3D model Malé pevnosti Terezín. Na základě kombinace DMT ve formě TINu a ortofotomapy byl

v jeho práci vytvořen model terénu, použitý pro vizualizaci. V modulu Kres proběhla tvorba modelů stavebních objektů, stromů a hradeb. V modulu Pogledy pak byla provedena kontrola celého modelu Malé pevnosti, a také byla vytvořena dynamická vizualizace ve formě animace. [21, 25]



Obr. 12 : Ukázka prostředí ATLAS

zdroj [25]

Výhodou programu je jistě možnost konstruování terénu i budov, a také vizualizace ve vlastním modulu. ATLAS ovšem zvládne pouze 2,5D reprezentaci objektů, nikoliv plnou 3D reprezentaci. To může v některých případech působit nepřekonatelné problémy, například při konstrukci budov nelze modelovat zapuštěná okna, římsy, podloubí, atd. Další nevýhodou zůstává, že se jedná o komerční software. Pokud tedy chceme zpřístupnit model širokému obecenstvu, lze tak udělat jen prostřednictvím různých pohledů a animací, které lze v tomto systému vytvářet.

3 VÝBĚR A PRAKTICKÁ REALIZACE ZVOLENÉHO ŘEŠENÍ

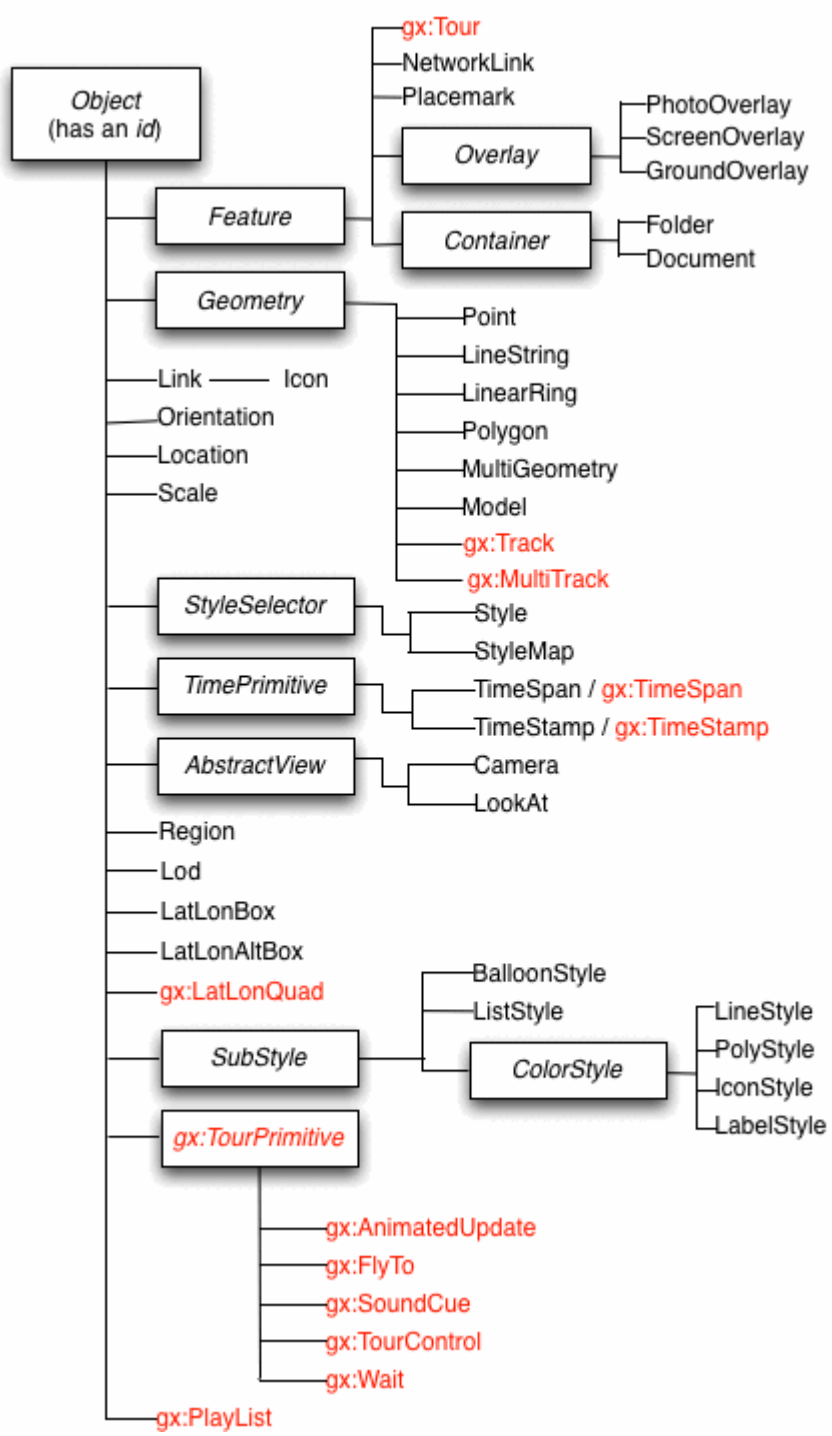
Z nastíněných možností výběru softwarů, popsaných v kapitole 2.3, byla pro praktickou část této práce vybrána aplikace Google Earth. Důvodů pro tuto volbu je hned několik. Aplikace je dostupná (jedná se o nekomerční software) a oblíbená u široké veřejnosti. Poskytuje možnost vybrat si pro vizualizaci vlastního modelu jednu z možností, popsaných v kapitole 2.3.4 (týkající se publikace modelu buď pouze z lokálního disku nebo z Galerie Google Earth). Dalším důvodem je určitá zkušenost s touto aplikací získaná v rámci předešlých prací a v neposlední řadě také skutečnost, že tato aplikace byla původně zamýšlena jako finální řešení i pro celý projekt „Krajina paměti. Drážďany a Terezín jako místa vzpomínek na ŠOA“, se kterým tato práce úzce souvisí.

3.1 KML

Google Earth je neodmyslitelně spjat s jazykem KML (viz kapitola 2.1). Tato aplikace zpracovává KML soubory podobně jako webové prohlížeče zpracovávají HTML nebo XML soubory. Jazyk KML má totiž podobnou strukturu jako HTML, založenou na značkách s názvy a atributy, které specifikují vizualizační požadavky. Takže Google Earth se chová jako „prohlížeč“ KML dat. [14]

Jazyk KML přináší možnost, jak reprezentovat geografická data v prostředí webových mapových aplikací nebo virtuálních prohlížečů Země. Specifikuje, jaká data a jak se mají zobrazit a doplňuje tak skupinu nejvíce používaných OGC standardů, jakými jsou například GML *(Geography Markup Language)*, WFS *(Web Feature Service)* nebo WMS *(Web Map Service)*. KML je podporován širokou škálou aplikací. [9] Výchozím souřadnicovým systémem je WGS84, výškový systém je definován na základě geoidu WGS84 EGM96. [24] Struktura jazyka je zobrazena na Obr. 13.

Souřadnice zeměpisné délky jsou definovány od –180° do +180°, přičemž hodnota 0° reprezentuje nultý poledník (Greenwichský). Směrem na východ od nultého poledníku nabývá zeměpisná délka kladných hodnot, směrem na západ záporných. Souřadnice zeměpisné šířky jsou definovány od –90° do +90°, přičemž hodnota 0° reprezentuje rovník. Směrem na sever od rovníku nabývá zeměpisná šířka kladných hodnot, směrem na jih od rovníku záporných. [12, 13]



Obr. 13 : Struktura jazyka KML

zdroj [13]

3.1.1 Region

Pro vizualizaci rozsáhlých dat v prostředí Google Earth jsou silným nástrojem *Regiony*. Poskytují možnost načítání a vykreslování dat pouze v případě, kdy tato data jsou v zorném poli a v dostatečném přiblížení. Umožňují pracovat právě s různými úrovněmi detailu dat, takže podrobnější detaily jsou zobrazeny až v případě většího přiblížení. Region

obsahuje tzv. *bounding box*, což je kvádr který ohraničuje data. Podobně jako minimální ohraničující obdélník pro 2D data, tento *bounding box* ohraničuje 3D data. Definuje jej element <*LatLonAltBox*>. Ten je tvořen potomky: [12, 13]

- <*north*> – zeměpisná šířka severní hrany *bounding boxu*
- <*south*> – zeměpisná šířka jižní hrany *bounding boxu*
- <*east*> – zeměpisná délka východní hrany *bounding boxu*
- <*west*> – zeměpisná délka západní hrany *bounding boxu*
- <*minAltitude*> – minimální výška v metrech
- <*maxAltitude*> – maximální výška v metrech

Údaje o výškách jsou v KML doprovázeny elementem <*altitudeMode*>, který specifikuje, k čemu se výška vztahuje. Tento element může být určen následujícími elementy: [12, 13]

- <*clampToGround*> – toto implicitní nastavení v podstatě ignoruje výšky a umisťuje objekt přímo na povrch Země.
- <*relativeToGround*> – nastavuje výšku relativně k povrchu Země, tzn. výšku objektu přičte k aktuální výšce povrchu Země v daném místě.
- <*absolute*> – výška je vztažena k vertikálnímu datu (výše zmíněnému geoidu) bez ohledu na aktuální výšku zemského povrchu. V případě, že objekt zadaný absolutní výškou není viditelný, mohlo se stát, že se nachází pod povrchem Země.

Region obsahuje také element <*Lod*> (*Level of Detail*). Ten poskytuje velmi efektivní možnost specifikovat, kdy bude Region definovaný *bounding boxem* zobrazen. Můžeme tak nastavit, aby se větší objem dat (spojený se zobrazením objektu, který obsahuje mnoho detailů) načetl až ve chvíli, kdy bude tento objekt viditelný v dostatečném přiblížení a detaily se zobrazí adekvátně. Element <*Lod*> je určen těmito potomky:

- <*minLodPixels*> – minimální počet pixelů
- <*maxLodPixels*> – maximální počet pixelů
- <*minFadeExtent*> – počet pixelů, na kterém se objekt zcela ztrácí
- <*maxFadeExtent*> – počet pixelů, na kterém se objekt stává zcela viditelným (průhlednost je nulová)

První dva elementy specifikují, jakou plochu (v pixelech) na obrazovce musí Region obsazovat, aby byl zobrazen. To znamená, že daný objekt bude viditelný pouze v případě, že Region, kterým je dán, zabírá na obrazovce plochu větší než je hodnota elementu

<*minLodPixels*> a menší než je hodnota elementu <*maxLodPixels*>. Poslední dva elementy specifikují rozsah pixelů, který je použit pro postupné blednutí (pozvolný přechod od viditelného na průhledný, postupné „ztrácení se“) objektu a naopak. Hodnota elementu <*minFadeExtent*> nemůže být menší než hodnota <*minLodPixels*> pro daný objekt, stejně tak hodnota elementu <*maxFadeExtent*> nemůže být větší než hodnota <*maxLodPixels*>. Použití těchto dvou elementů je dobrovolné, zatímco první dva elementy jsou vyžadovány. [12, 13]

Region je tedy označen jako aktivní v případě, že jsou splněny právě dvě podmínky. Za prvé musí být *bounding box* v zorném poli a za druhé musí jeho velikost zobrazená na obrazovce počítače odpovídat nastavenému počtu pixelů.

3.1.2 NetworkLink

Element <*NetworkLink*> v kombinaci s <*Region*> poskytuje velmi efektivní cestu, jak se vypořádat s velkými objemy dat. Řídí načítání dat spojených s Regionem, takže pokud je Region neaktivní, žádná data se nenačítají. Odkazuje na KML soubor nebo KMZ archiv umístěný na lokálním disku nebo na síťi. Obsahuje element <*Link*>, který specifikuje, kde je KML soubor umístěn. Tento soubor je pak systematicky načítán a obnovován v závislosti na nastavení dalších parametrů elementu <*Link*>. Těmito parametry jsou [13] :

- <*href*> – (*Hypertext Reference*), odkaz na KML soubor, který má být načten. Může to být HTTP adresa, nebo specifikace souboru na lokálním disku.
- <*refreshMode*> – aktualizace souboru založená na čase, může nabývat hodnot:
  - **onChange** – provede aktualizaci tehdy, když je soubor načten a dále kdykoli se změní parametry elementu <*Link*>
  - **onInterval** – provede aktualizaci každých *n* sekund (specifikovaných v elementu <*refreshInterval*>)
  - **onExpire** – provede aktualizaci ve chvíli, kdy dosáhne času expirace
- <*refreshInterval*> – nastavuje dobu, po jejímž uplynutí se aktualizace provádí
- <*viewRefreshMode*> – aktualizace souboru založená na pohybu kamery, může nabývat hodnot:
  - **never** – ignoruje změny v pohledu na scénu
  - **onStop** – provede aktualizaci *n* sekund poté, co se kamera přestane pohybovat (*n* je specifikováno v elementu <*viewRefreshTime*>)
  - **onRequest** – provede aktualizaci pouze v případě, kdy si uživatel o ni zažádá (např. kliknutím na tlačítko *Refresh*)

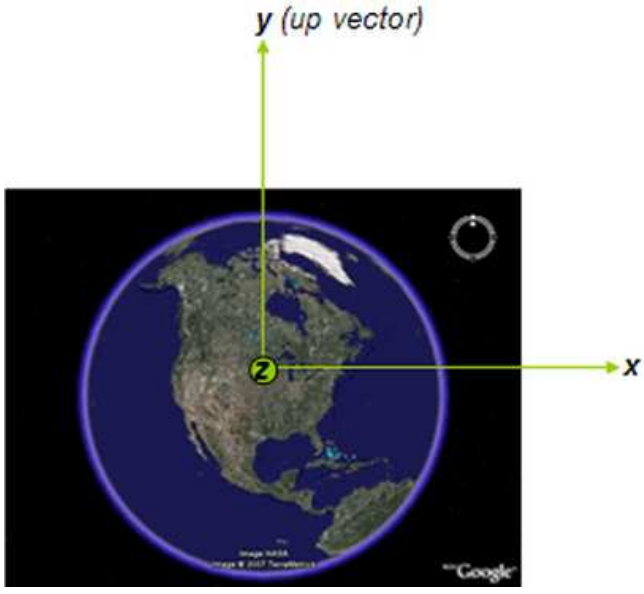
- **onRegion** – provede aktualizaci ve chvíli, kdy se Region stane aktivní (viz kapitola 3.1.1)

- *<viewRefreshTime>* – když se kamera přestane pohybovat, specifikuje počet sekund, během kterých se čeká, pak spustí aktualizaci

3.1.3 Camera a LookAt

Nastavení výchozí pozice kamery lze s výhodou využít, pokud chceme uživateli „předpřipravit“ nejvhodnější pohled na danou scénu. S použitím elementů *<Camera>* nebo *<LookAt>* můžeme nastavit nejen pozici kamery, ale i další parametry. Funkce těchto dvou elementů je velmi podobná, oba umožňují zvolit umístění virtuální kamery. Rozdíl je ovšem v tom, že *<LookAt>* nastavuje úhel pohledu v závislosti na objektu, který má být prohlížen. Oproti tomu *<Camera>* specifikuje pohled na základě pozice a orientace kamery. [12, 13]

Element *<Camera>* nabízí možnost umístit „oko pozorovatele“ kamkoliv do prostoru a natočit pohled požadovaným směrem. K tomu je využit systém tří na sebe kolmých os, pomocí kterých definujeme natočení virtuální kamery. Počátkem tohoto systému je kamera (nejedná se tedy o geocentrický systém), proto se nejprve umístí kamera do prostoru (pomocí specifikace zeměpisné šířky, délky a výšky kamery – viz níže) a poté se provádí natáčení kamery za pomoci těchto os. Osa Y má směr „nahoru“ relativně k obrazovce, osa X míří doprava od kamery a osa Z míří kolmo na obrazovku k oku uživatele. Systém os je znázorněn na Obr. 14.

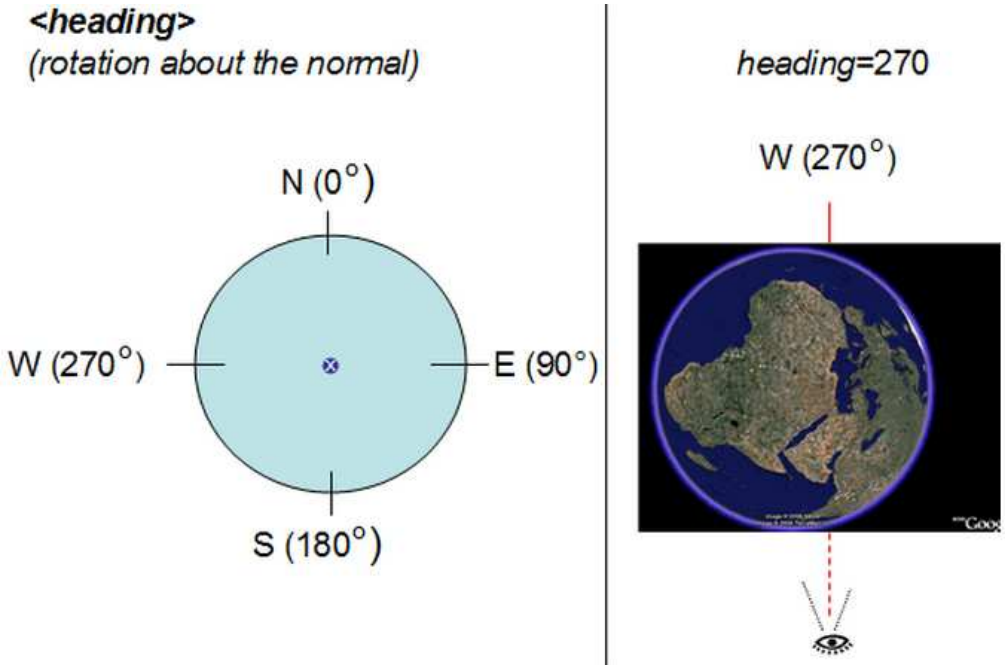


Obr. 14 : Systém os spojený s virtuální kamerou

zdroj [12]

Element <*Camera*> je definován těmito potomky [12, 13] :

- <*latitude*> – zeměpisná šířka pozice kamery
- <*longitude*> – zeměpisná délka pozice kamery
- <*altitude*> – vzdálenost kamery od povrchu Země v metrech. Tato hodnota je ovlivněna elementem <*altitudeMode*>. (více viz kapitola 3.1.1)
- <*heading*> – rotace kamery okolo osy Z ve stupních. Může nabývat hodnot od 0° po 360°, přičemž hodnota 0° vyjadřuje směr na sever. (viz Obr. 15)
- <*tilt*> – rotace kamery okolo osy X ve stupních, nabývá hodnot 0° až 180°. Při hodnotě 0° směřuje kamera kolmo na zemský povrch (toto nastavení je implicitní), při hodnotě 90° směřuje podél horizontu a při hodnotách větších než 90° směřuje kamera na oblohu
- <*roll*> – rotace kamery okolo osy Z ve stupních (podruhé). Může nabývat hodnot od –180° do +180°.

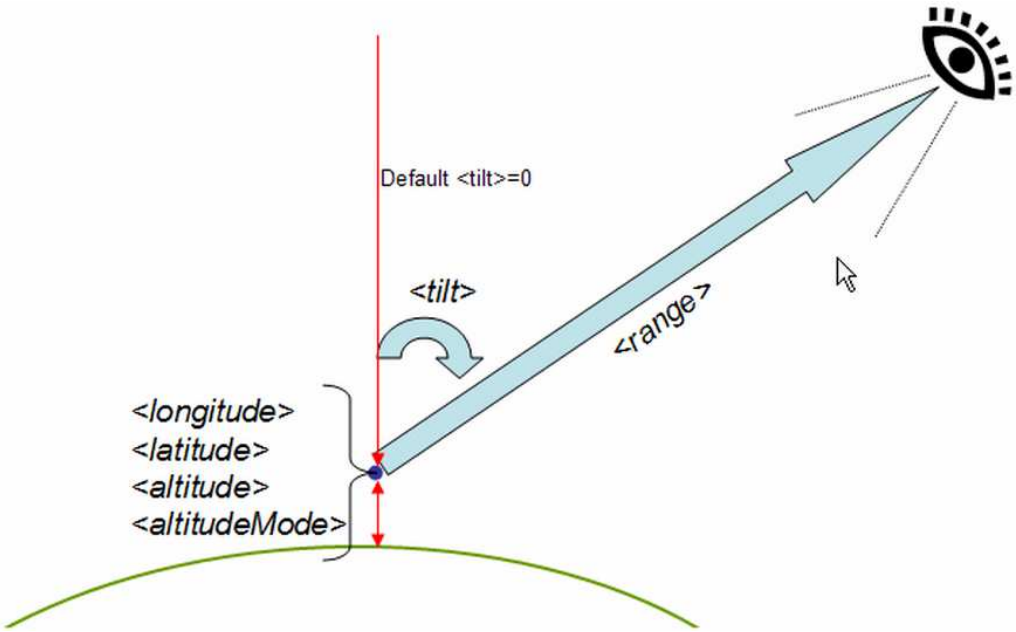


Obr. 15 : Vysvětlení elementu <heading>

zdroj [12]

Element <*LookAt*> definuje bod na Zemi, který je prohlížen, dále vzdálenost kamery od tohoto bodu a úhel pohledu (viz Obr. 16). Tento element je definován následujícími potomky: [12, 13]

- <*longitude*> – zeměpisná délka bodu, na který chceme pohled nastavit
- <*latitude*> – zeměpisná šířka bodu, na který chceme pohled nastavit
- <*altitude*> – vzdálenost zmíněného bodu od zemského povrchu v metrech. Tato hodnota je ovlivněna elementem <*altitudeMode*> (více viz kapitola 3.1.1)
- <*heading*> – směr pohledu (ve smyslu sever, jih, východ nebo západ) ve stupních. Může nabývat hodnot 0° až 360°, přičemž hodnota 0° znamená sever, hodnota 90° znamená východ, atd. (viz Obr. 15)
- <*tilt*> – úhel, který svírá normála procházející zmíněným bodem a spojnice požadované polohy kamery s tímto bodem. Může nabývat hodnot od 0° do 90°, přičemž hodnota 0° znamená pozorování daného bodu z pozice přímo nad bodem (ze směru normály), hodnota 90° znamená pozorování podél horizontu.
- <*range*> – vzdálenost kamery od zmíněného bodu, definovaného elementy <*longitude*>, <*latitude*> a <*altitude*>



Obr. 16 : Vysvětlení parametrů, definujících element <LookAt>

zdroj [12]

3.2 Tvorba 3D modelu

Informace, popsané výše bylo potřeba prakticky vyzkoušet na konkrétních datech. Pro tyto účely byly zhotoveny modely tří bloků budov ve městě Terezín. Tyto modely slouží výhradně pro testovací účely. Vytvořeny byly v aplikaci Trimble SketchUp (viz kapitola 2.3.1) a respektují hierarchické členění do jednotlivých úrovní detailu (LoD), definovaných jazykem CityGML (viz kapitola 2.1). Dále bylo řešeno rozšíření dat tak, aby rozsahem odpovídala celému městu.

3.2.1 Konstrukce modelu bloku budov

Nejprve byl vybrán vhodný blok budov ve městě Terezín. Jedná se o blok, který se nachází vedle parku a jeho roh je zároveň rohem náměstí. Domy v tomto bloku jsou různorodé, nejsou všechny stejně vysoké a fasády vypadají u každého jinak. Takto heterogenní blok je vhodný pro testování exportu modelů jednotlivých budov jako komponent v aplikaci Trimble SketchUp a jejich následného zobrazování v aplikaci Google Earth.

Následně byly vytvořeny půdorysy domů. Do aplikace Trimble SketchUp byla nahrána dlaždice s texturou Google Earth, zobrazující daný blok budov. To bylo provedeno pomocí ikony se žlutou šipkou (*Add Location*). Dále proběhlo nastavení os (*Tools – Axes*) tak, aby počátek byl v rohu bloku, červená osa vedla podél bloku (v podstatě kopírovala vnější půdorys), zelená osa byla kolmá na červenou osu a vedla podél sousední strany bloku a modrá osa doplňovala pravoúhlý systém. V takto upraveném prostředí byly nad dlaždicí zvektorizovány půdorysy jednotlivých domů. Dlaždice byla poté vymazána a lokalizovaný půdorys každého domu byl uložen jako samostatná komponenta (objekt je třeba označit a zvolit nástroj *Make Component*). Model v této fázi odpovídá úrovni detailu 0 (**LoD0**).

Dále byl pomocí nástroje *Push/Pull* každý půdorys vytažen do výšky, takže vznikl 3D blokový model odpovídající úrovni detailu 1 (**LoD1**). V tomto případě jsou výšky budov vztaženy k patě střechy (úroveň okapů).

U každé budovy byl poté zkonstruován tvar střechy a opatřen texturou. To odpovídá úrovni detailu 2 (**LoD2**). Nutno podotknout, že u každého LoD byl zachován a uložen dokument SKP, který zobrazuje celý model bloku budov vždy jen v dané úrovni detailu.

Při tvorbě **LoD3** byly do modelu doplněny detaily na fasádách, jakými jsou okna, dveře nebo římsy. Při konstruování oken byl vytvořen jeden prototyp okna. Ten byl uložen jako samostatná komponenta a ta byla nakopírována na stěny domu. To znamená, že všechna okna vypadají naprosto stejně. Výhodou tohoto přístupu je snadná možnost editace. Ve chvíli, kdy jakkoli změníme jednu tuto komponentu, změna se automaticky promítne na všechny

od ní odvozené komponenty. Tak se dají rychle měnit například barvy nebo tvar a velikost všech oken. Z důvodu různé velikosti jednotlivých domů v bloku nebylo vhodné použít pouze jeden prototyp okna pro celý blok. Proto byly řešeny a použity dva různé prototypy, které se liší tvarem, velikostí i barvou. Dveře i římsy byly konstruovány pro každý dům zvlášť. Úroveň detailu 4 (**LoD4**) nebyla v rámci práce řešena.

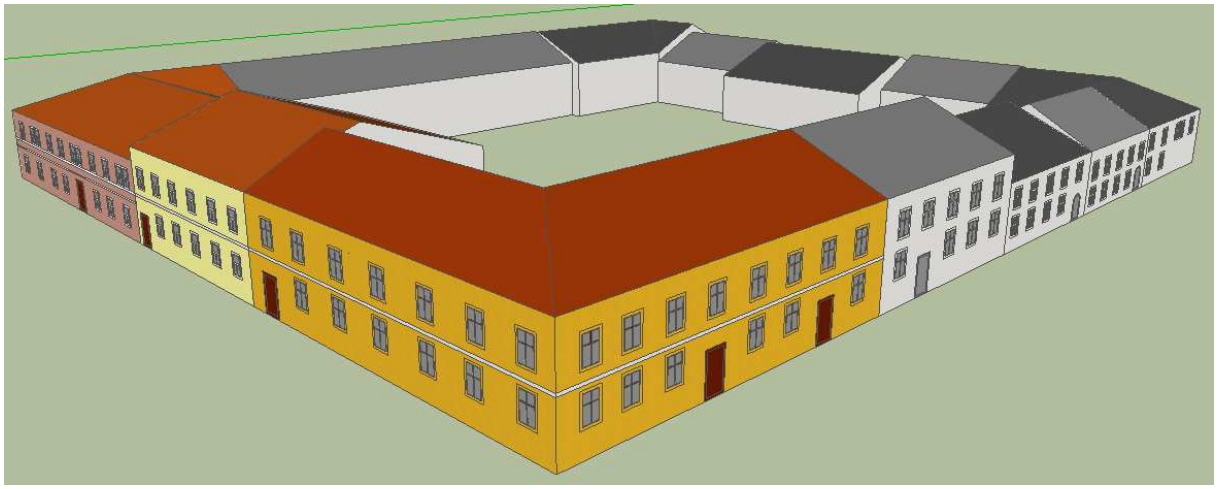
3.2.2 Texturování

V rámci konstrukce modelu pro úroveň LoD3 bylo řešeno přidání vlastní textury do palety barev a materiálů aplikace SketchUp. Po spuštění nástroje *Paint Bucket* se zobrazí dialogové okno. V horní pravé části tohoto okna se nachází nástroj *Create Material.* Po spuštění se zobrazí další dialogové okno, které nám umožní vlastní texturu vytvořit. Nejprve ji pojmenujeme, v sekci *Color* si můžeme nastavit požadovanou barvu. K tomu slouží několik barvových modelů. V sekci *Texture* lze nahrát jakoukoli texturu ve formátu JPG, PNG, PSD, TIF, TGA nebo BMP. Lze také měnit velikost dlaždice s texturou. V sekci *Opacity* můžeme nastavovat průhlednost textury. Po kliknutí na tlačítko OK se textura uloží do sekce *In Model*, která je součástí dialogového okna nástroje *Paint Bucket*. Texturu lze i nadále editovat, stačí ji zvolit a přepnout ze záložky *Select* na záložku *Edit*. Existuje také možnost exportovat texturu, upravit si ji v libovolném externím editoru (například v aplikaci GIMP) a poté opět nahrát zpátky. K tomu slouží nástroj *Edit texture image in external editor* v záložce *Edit*. Při nahrávání vlastních textur je důležité dbát na to, aby textura byla speciálně upravena nebo konstruována tak, aby kresba na protějších stěnách dlaždice na sebe navazovala. Pokud toto není splněno, jsou při použití textury viditelná rozhraní jednotlivých dlaždic.

3.2.3 Rozdílná vizualizace zájmových a ostatních budov

Hotové modely budov v bloku byly dále vizuálně rozděleny do dvou skupin – na zájmové a ostatní budovy. Toto dělení vycházelo z požadavků Památníku Terezín, aby při finální vizualizaci celého města bylo uživateli na první pohled patrné, na které budovy zaměřit pozornost. Skupina **ostatních budov** je vizualizována v odstínech šedi. U LoD2 jsou v tomto případě střechy obarveny šedě tak, aby se u sousedních domů nesetkaly dva stejné odstíny šedi. Je tak i z pohledu shora velmi dobře patrné, kolik budov se v bloku nachází a kde jsou předěly mezi nimi. V aplikaci SketchUp jsou barvy fasád ponechány jako standardní materiál a pro zobrazení oken a dveří v LoD3 bylo vytvořeno několik možností. Tyto prvky mají být pouze naznačeny a proto jsou zobrazeny jen okenní tabulky, u dveří

je viditelný obrys a zárubeň. Pro dveře bylo vytvořeno několik variant obarvení do různých stupňů šedi. Pro okenní tabulky také existuje několik variant ve stupních šedi, dále je vytvořena stínovaná varianta. Z těchto možností byla vybrána a použita nejvhodnější varianta vizualizace. Skupina **zájmových budov** je provedena barevně. V LoD1 jsou horní plochy kvádrů barevně otexturovány, stejně jako střechy budov v LoD2. V LoD3 je barevná textura střech samozřejmě zachována a přibývá barevné provedení fasád a objektů na nich (okna, dveře, římsy). Pro fasády byl proveden postup přidání vlastní textury (viz kapitola 3.2.2). Byla vybrána textura splňující vlastnosti zde uvedené, ta byla načtena a uložena v různých barevných provedeních a velikostech dlaždic a nanesena na příslušné budovy. Výsledek je skutečně bezešvý.



Obr. 17 : Zájmové a ostatní budovy

3.2.4 Export jednotlivých komponent

Pro potřeby výsledné vizualizace bylo nutné provést export každé jednotlivé budovy zvlášť a uložit jej jako samostatný KMZ soubor. V aplikaci SketchUp lze přes záložku *File – Export – 3D Model* ukládat své modely jako KMZ nebo DAE soubory. Provede se však export celého modelu, který je zobrazen. Model bloku je složen z jednotlivých budov, přičemž každá budova je uložena jako komponenta. Při exportu provedeném tímto způsobem se však uloží do KMZ souboru celý blok budov, nikoli jediná budova. Tento problém byl nakonec vyřešen pomocí skrývání komponent. Skryjeme všechny komponenty kromě jedné, která zůstane viditelná (pomocí nástroje *Select* vybereme a dále zvolíme *Edit – Hide* pro skrytí). Následně provedeme export. Výsledek odpovídá požadavkům. Ve výsledném KMZ souboru je již uložena pouze jedna komponenta, představující jednu budovu. Tento postup byl aplikován pro všechny budovy v bloku a také pro všechny úrovně detailu. To znamená, že pro

každou budovu existují celkem čtyři KMZ soubory (každý pro jednu úroveň detailu). Tímto postupem jsou exportovány budovy správně lokalizované.

3.2.5 Konstrukce modelu celého města

V další části práce bylo řešeno rozšíření dat pro celé město Terezín. Bohužel v současné době ještě není k dispozici kompletní 3D model celého města. Tvorba jednotlivých modelů budov teprve probíhá. Pro potřeby této práce však bylo nutné otestovat dynamickou vizualizaci pro data odpovídající rozsahu celého města. Proto byly vytvořeny kopie bloku budov (popsaného v kapitole 3.2.1) a byly rozmístěny tak, aby pokryly plochu města a aby rozsahem odpovídaly reálné situaci. Rozsah dat je znázorněn na Obr. 18.

Následující odstavce popisují řešení rozšíření dat pro celé město. Hlavním důvodem pro použití tohoto postupu byla správná lokalizace všech budov a také příprava dat pro další práci v souvislosti s automatickým generováním KML kódu (viz kapitola 3.3.1). Byl využit software Trimble SketchUp, Google Earth a ArcGIS Desktop.

V aplikaci Trimble SketchUp byla načtena dlaždice s texturou Google Earth zobrazující část města. Podle této dlaždice byla zvektorizována místa, kam se nakopírují bloky budov (u každého bloku jedna budova na rohu). Dále byla načtena dlaždice, zobrazující další část města, nad kterou pokračovala vektorizace. Dlaždice pak byly smazány a vznikl tak referenční soubor (jakási předloha) pro kopírování komponent. Tento soubor byl uložen jako SketchUp model a také byl vyexportován do formátu KMZ. Podle této předlohy byl blok budov kopírován a umisťován, a to celkem 27-krát. Při každém umístění bloku proběhl export a rozdílné pojmenování všech budov ve čtyřech úrovních detailu do KMZ souborů.

Následně byly vytvořeny modely dvou konkrétních bloků budov. Jedná se o bloky, které jsou tvořeny celé jedinou budovou. Postup tvorby těchto bloků odpovídá postupu, popsanému v kapitole 3.2.1. Rozměry těchto budov jsou diametrálně odlišné od všech ostatních budov, které byly pro potřeby práce vytvořeny. Využití těchto dvou modelů je nejen ve výsledné vizualizaci, ale také pro testování metod výběru úrovně detailu (viz kapitola 3.3.2). Tyto modely byly také umístěny a vyexportovány ve všech úrovních detailu. Výsledkem je přes 1 300 souborů představujících budovy v Terezíně ve čtyřech úrovních detailu.

Další práce probíhala v softwaru ArcGIS Desktop 10.1. Po nastavení souřadnicového systému proběhl import výchozího bloku budov v úrovni LoD0 (tedy pouze půdorysy budov). Pro import byl využit nástroj *KML To Layer*, dostupný z ArcToolboxu v sekci *Conversion*

Tools – From KML. Načtené půdorysy se zobrazily ve formátu Multipatch. Byla vytvořena souborová geodatabáze *Terezin.gdb* a v ní polygonová třída prvků s názvem *Pudorysy*. Do této třídy byly zvektorizovány půdorysy všech budov v bloku nad zobrazenými daty. Do atributové tabulky byly uloženy informace o výšce každé budovy, o souřadnicích rohů budov a také název budovy. Poté byla opět pomocí nástroje *KML To Layer* načtena výše zmíněná předloha pro kopírování komponent. Dále proběhlo kopírování a umisťování bloku budov na místa, daná předlohou. Při umístění každého bloku byly s pomocí nástroje *Field Calculator* upraveny informace v atributové tabulce, a to především názvy budov a souřadnice rohů. Byly také přidány půdorysy dvou zvlášť vytvořených bloků. Výsledkem jsou půdorysy budov v Terezíně, ke kterým jsou uložena potřebná data v atributové tabulce. Vše je uloženo jako MXD dokument aplikace ArcMap 10.1.

Obr. 18 : Rozsah dat – celé město zobrazené v LoD1

3.3 Dynamická vizualizace v Google Earth

K řešení dynamické vizualizace v této aplikaci lze přistupovat třemi různými cestami. Tyto cesty se týkají způsobu, jaký vede k vytvoření KML kódu, který by zahrnoval data z celého města a zajišťoval, aby vizualizace dat (vytvořených v rámci kapitoly 3.2) byla skutečně dynamická. Takovýto kód lze vytvořit :

- ručně předem

- automaticky předem na základě dat uložených v databázi
- automaticky „on the fly“ na základě dat uložených v databázi, podle aktuálního pohledu

Uživatel by neměl poznat rozdíl mezi jednotlivými přístupy. První přístup je technicky nejjednodušší, ovšem je vhodný pouze pro malé modely (např. jedna budova). Druhý přístup na základě dat uložených v databázi vytvoří KML kód pro celé město předem, aplikace Google Earth pak tento kód využívá k vizualizaci. Komunikace je tedy pouze jednosměrná. Třetí přístup je technicky nejnáročnější, protože vyžaduje obousměrnou komunikaci mezi aplikací Google Earth a databází. Zde je na základě aktuální pozice a orientace kamery vytvořen KML kód, který obsahuje pouze budovy ve viditelném výřezu a v odpovídající úrovni detailu. Tento kód je poslán do aplikace Google Earth. Při manipulaci s modelem v Google Earth je zaslána nová informace o pozici a orientaci kamery, je vygenerován nový KML kód a ten je opět zobrazen. KML kód je tedy generován dynamicky v závislosti na čase nebo pohybu kamery.

V rámci práce byl řešen druhý přístup. Původní myšlenkou bylo zahrnout do práce také třetí přístup. Zásadnější však bylo řešení problémů, které se vyskytly při testování druhého přístupu.

3.3.1 KML kód

Na vytvořený model města, jehož tvorba je popsána v kapitole 3.2, byl aplikován KML kód, který řeší funkcionalitu zobrazování budov a přepínání úrovní detailu během přibližování. Ukázka tohoto kódu je v příloze práce.

Pro definování výchozí pozice a orientace kamery je použit element <*LookAt*>. V rámci tohoto elementu je jako výchozí bod nastaven střed náměstí v Terezíně. Dále je využito elementů <*NetworkLink*> a <*Region*>. V elementu <*Region*> je definován *bounding box* ohraničující danou budovu. Pro údaje o výškách je zde využito nastavení *relativeToGround* elementu <*altitudeMode*>. Dále je také s využitím elementu <*Lod*> pro tuto budovu a příslušnou úroveň detailu definováno, kdy budou související data zobrazena. Hodnoty *bounding boxu* jsou tedy pro jednu budovu stále stejné pro všechny čtyři její úrovně detailu, mění se pouze nastavení elementu <*Lod*>. Pomocí elementu <*Link*> je odkazováno na KMZ soubor, který je uložen na lokálním disku a představuje 3D model budovy. Pro načítání tohoto souboru je zvoleno nastavení *onRegion* elementu <*viewRefreshMode*>. Aby mohl zmíněný kód správně fungovat, musí být umístěn v adresářové struktuře ve stejné

složce, jaká obsahuje KMZ soubory pro objekty, na které je z kódu odkazováno. Vysvětlení a podrobný popis všech zde zmíněných elementů jazyka KML lze nalézt v kapitole 3.1.

Element <*NetworkLink*> se poté opakuje pro všechny budovy a pro všechny úrovně detailu. Výsledný kód je tedy velmi dlouhý, obsahující opakující se části. Tento kód byl nejprve testován pouze na jednom bloku budov. Výsledek odpovídal očekáváním a proto byl rozšířen na data z celého města. To lze provést několika způsoby, jak již bylo zmíněno v úvodu kapitoly 3.3. První možnost, ruční editace kódu, je velmi neefektivní a proto nebyla použita. Druhá možnost, generování kódu automaticky, byla v rámci práce řešena.

Pro zautomatizování procesu vytváření KML kódu byl použit programovací jazyk Python a jeho implementace pro Windows – PythonWin. Výsledný skript je v příloze práce.

Aby mohl být kolem každé budovy automaticky zkonstruován *bounding box*, byla použita data uložená v geodatabázi *Terezin.gdb*. Souřadnice rohů budov byly použity jako hodnoty elementů <*north*>, <*south*>, <*east*> a <*west*>. Výška budovy byla použita jako hodnota elementu <*maxAltitude*>. Na základě velikosti *bounding boxu* jsou počítány hodnoty elementů <*minLodPixels*> a <*maxLodPixels*>, které řeší zobrazování jednotlivých úrovní detailu (viz kapitola 3.3.2). Názvy budov, uložené v atributové tabulce třídy *Pudorysy*, jsou zde použity pro element <*href*>, který odkazuje na jednotlivé KMZ soubory.

Ve skriptu jsou dva vnořené for cykly. První z nich pomocí kurzoru projde všechny řádky v atributové tabulce (odpovídající jednotlivým budovám), druhý cyklus pak pro každou budovu projde 4 úrovně detailu a provede přepočet hodnot <*minLodPixels*>, <*maxLodPixels*> a <*href*>.

Tento skript:

- nastaví výchozí pozici a orientaci kamery
- vygeneruje KML kód pro všechny budovy ve městě ve všech úrovních detailu
- pro každou budovu nastaví *bounding box*
- podle zvolené metody výběru úrovně detailu (viz kapitola 3.3.2) provede výpočet hodnot elementu <*Lod*>
- nastaví odkaz na příslušný KMZ soubor

Výsledkem je KML kód, který pro celý Terezín čítá přes 28 000 řádků (445 kB), což potvrzuje neefektivnost ruční editace. Po spuštění kódu se aktivuje aplikace Google Earth, kamera „nalétne“ do požadované pozice a začnou se načítat data odpovídající této pozici.

3.3.2 Metody výběru úrovně detailu

Důležitým aspektem je návrh metody výběru úrovně detailu objektů v závislosti na vzdálenosti pozorovatele, v tomto případě virtuální kamery. Tato metoda by tedy měla určit, jak vybrat úroveň detailu, která má být zobrazena pro dané přiblížení kamery.

K tomu je využito vlastností elementu <*Lod*> (viz kapitola 3.1.1). Ten umožňuje pro každou úroveň detailu daného objektu specifikovat, kdy se má zobrazit. Samozřejmě je žádoucí nastavit zobrazování tak, aby se jednotlivé úrovně střídaly při přibližování, respektive oddalování virtuální kamery. To znamená, že ve chvíli kdy jedna úroveň detailu zmizí, ihned se zobrazí další.

Výpočet hodnot elementu <*Lod*> je založen na velikosti *bounding boxu*. Ze známých hodnot souřadnic rohů budov jsou spočteny délky stran půdorysu *bounding boxu*, dále jsou využity tři různé přístupy:

- maximum ze dvou délek stran
- úhlopříčka
- odmocnina plochy půdorysu

Tato čísla slouží jako základní nastavení a jsou použita pro hodnoty počtu pixelů, které specifikují přechod od LoD0 k LoD1. Pro vyšší úrovně detailu je pak tento základ postupně násoben dvěma experimentálně zjištěnými konstantami. Tyto konstanty jsou optimalizovány tak, aby metody dobře fungovaly i pro diametrálně odlišná data. Zohledňují jak malé, tak velké rozdíly ve velikostech budov. Malé rozdíly se projeví u jednotlivých budov v rámci jednoho bloku. Velké rozdíly jsou patrné u bloků, které jsou tvořeny celé jedinou budovou. Při testování toho, jak zmíněné metody fungují pro takto různorodá data, byly využity dva samostatně modelované bloky budov (tvořené jednou budovou) v kombinaci s kopiemi původního bloku (tvořeného 12 budovami). Nastavení hodnoty <*minLodPixels*> pro zobrazení úrovně LoD0 je 1 pixel. Pro hodnotu elementu <*maxLodPixels*> pro zobrazení úrovně LoD3 je použita hodnota –1, což podle [12] znamená, že data jsou aktivní do nekonečného přiblížení.

Jednotlivé metody vykazují drobné rozdíly, všechny jsou však přijatelné a fungují velmi dobře. Ve chvíli, kdy je zobrazeno celé město, jsou načteny pouze půdorysy budov (LoD0). Při postupném přibližování se brzy objeví LoD1. Při dalším přibližování, kdy již není vidět celé město najednou, ale pouze část města, je zobrazen LoD2. LoD3 se načte až ve chvíli, kdy je kamera velmi blízko danému objektu. Je tak dosaženo načítání nejdetailnějších dat až ve chvíli, kdy jsou tyto detaily dobře viditelné (protože jsou zobrazeny

zblízka). Není tedy nutné načítat tento obrovský objem dat pro celé město najednou, ale postupně, vždy pro několik málo bloků, které jsou v požadovaném přiblížení. Zobrazení různých úrovní detailu v závislosti na vzdálenosti kamery ukazuje Obr. 19.

Všechny tři metody jsou implementovány ve výše zmíněném skriptu. Posouzení vhodnosti zobrazování daných úrovní detailu při tom či onom přiblížení je individuální, ovšem nastavení tohoto zobrazování lze díky skriptu velmi pohodlně a rychle měnit.

Obr. 19 : Ukázka zobrazení různých úrovní detailu

3.3.3 Problémy při praktickém nasazení

Při testování dynamické vizualizace celého města v aplikaci Google Earth se vyskytlo několik problémů. Při prvním spuštění aplikace se data načítají velmi pomalu, a to i ve chvíli, kdy se jedná o velmi jednoduché modely budov reprezentující nízké úrovně detailu. Například při počátečním zobrazení úrovně LoD0 pro všechny budovy ve městě je nutné počkat řádově desítky sekund, než načtení proběhne. Při přibližování je také načítání dalších úrovní detailu poměrně zdlouhavé. Tento problém je nejspíše způsoben velkým množstvím odkazů, vedoucích na KMZ soubory s modely budov, které se musí zpracovávat. Dalším problémem je, že v průběhu postupného načítání dalších dat v souvislosti s přibližováním a posouváním modelu nastane v určité chvíli okamžik, od kterého se žádná další data nenačtou. Zůstanou tak viditelná všechna data, která se načetla do té doby, podrobnější data se již nezobrazí. Jakmile se však určitá data načtou, zůstanou uložena v paměti a při další

manipulaci s modelem se zobrazují okamžitě. Pokud ale v určité chvíli nenecháme proběhnout načítání a provedeme přiblížení příliš rychle (tj. chceme načíst vyšší úrovně detailu ve chvíli, kdy ještě není dokončeno načítání nižších úrovní nebo momentálně viditelných dat z vyšší úrovně detailu jiného místa), systém se zablokuje a žádná další data nenačte. Tato chyba je pravděpodobně na straně programátorů aplikace Google Earth.

Celkově tedy dynamická vizualizace města řešená pro každou budovu zvlášť funguje, zatěžuje však aplikaci Google Earth více, než je schopná efektivně zvládnout, a to i přes to, že je využito zobrazení v několika úrovních detailu. Interaktivní prohlížení modelu je tak na průměrném počítači přinejmenším nepohodlné.

Pro řešení zmíněných problémů byly vyzkoušeny dva přístupy:

- vypustit úroveň LoD0
- generovat data po celých blocích budov

V prvním případě byla data přepracována tak, že se místo LoD0 zobrazoval rovnou LoD1. Složitost LoD1 je totiž jen mírně větší, než složitost LoD0. Byl upraven skript, generující KML kód pro celé město. Úroveň detailu 0 byla vypuštěna, takže zbyly pouze úrovně LoD1, LoD2 a LoD3. Element *<Lod>* pro úroveň LoD1 byl upraven tak, aby se tato data zobrazovala ve chvíli, kdy byla v předchozím řešení zobrazena úroveň LoD0. Hodnota *<minLodPixels>* je tedy nastavena na 1 pixel. Všechna ostatní nastavení zůstala zachována. Pomocí tohoto skriptu byl vygenerován nový KML soubor, obsahující menší počet řádků (přes 21 000). To přispělo k redukování počtu odkazů na KMZ soubory (je jich 978, což je o čtvrtinu méně, než v případě modelu s úrovní LoD0). Vzhledem k tomu, že se od začátku načítají budovy rovnou v úrovni LoD1, jsou narozdíl od původních půdorysů mnohem lépe viditelné a výsledek tedy vypadá mnohem lépe. Budovy v LoD1 pro celé město se navíc nenačítají o nic pomaleji, než původní LoD0. Výsledek je subjektivně o trochu rychlejší a působivější, ovšem setrvávají problémy na straně aplikace, související s ukončením načítání podrobnějších dat ve chvíli, kdy uživatel pohybuje s modelem příliš rychle a vyžaduje tak načtení příliš mnoha dat najednou.

Ve druhém případě bylo upuštěno od zobrazování dat po jednotlivých budovách a místo toho byla data načítána rovnou po celých blocích. K tomuto postupu vedla úvaha o možných důvodech pomalého načítání a zobrazování jednotlivých úrovní detailu. Pokud by důvodem bylo velké množství souborů a souvisejících elementů *<NetworkLink>*, mělo by vést zobrazování po blocích budov k výraznému zrychlení prohlížení. Navíc byla po dobré zkušenosti z předchozího řešení opět vynechána úroveň LoD0. Data pro celé město byla

přepracována. V aplikaci Trimble SketchUp byl celý blok budov (popsaný v kapitole 3.2.1) uložen jako jediná komponenta. Dále byl podle postupu, popsaném v kapitole 3.2.5, znovu nakopírován na místa daná předlohou. Při každém umístění bloku proběhl export do formátu KMZ a to pro tři úrovně detailu celého bloku (LoD1, LoD2, LoD3). V softwaru ArcGIS byla do již existující geodatabáze *Terezin.gdb* založena nová třída prvků s názvem *Pudorysy_bloky*. Do této třídy byla nakopírována třída *Pudorysy* (ve které jsou uloženy půdorysy budov v celém městě). Dále byla data upravena tak, aby uložené půdorysy odpovídaly celým blokům a ne jednotlivým budován. K tomu byla v rámci editace použita funkce *Merge*. Data v atributové tabulce byla rovněž upravena. Každý blok byl pojmenován a byla uložena jeho výška, dále byly nastaveny hodnoty souřadnic rohů bloku. V aplikaci PythonWin byl upraven skript, generující KML kód, tak, aby načítal data z této nové třídy prvků *Pudorysy_bloky*. Dále byly upraveny odkazy na KMZ soubory (nyní obsahující 3D modely bloků a nesoucí tak jiný název i místo uložení v adresářové struktuře). Výsledný kód byl uložen také pod jiným názvem. Celý zbytek skriptu včetně metody výpočtu úrovně detailu zůstal nezměněn. Pouze byla změněna hodnota <*minLodPixels*> pro úroveň LoD1 na 1 pixel (stejně jako v prvním řešení). Výsledný kód je výrazně kratší (přes 1 900 řádků, 41 kB) a tedy obsahuje výrazně méně odkazů (přesně 87) na další KML soubory. Vizualizace je o poznání lepší, data se načítají rychle. Problémy na straně aplikace bohužel zůstávají, nastávají však velmi výjimečně.

Součástí práce jsou obě řešení popsaná výše. V rámci každého z nich jsou implementovány tři metody výpočtu úrovně detailu. Pro každé z těchto různých řešení je vytvořen a uložen skript, který vygeneruje KML kód pro celé město. Na příslušných místech v adresářové struktuře je také uloženo všech 6 variant výsledku tohoto skriptu. Každé ze dvou nastíněných řešení má své výhody i nevýhody. V prvním případě se budovy načítají pomaleji, ale zato výsledek je působivější. Postupné načítání detailů řešené zvlášť pro každou budovu dává představu více budov ve městě. Navíc v rámci jednoho bloku lze vizualizovat vzdálené části bloku v nižší úrovni detailu, než ty blízké. Ve druhém případě se bloky zobrazují rychle, ovšem není zde již možnost diferencovat data v rámci jednoho bloku.

Provedené experimenty objasnily zásadní vlivy na rychlost prohlížení a vzhled modelu při použití různých přístupů a jednoznačně ukázaly, jakou cestou se vydat při tvorbě modelu města. Mají proto zásadní vliv na způsob, jakým bude realizován 3D model města Terezín v projektu „Krajina paměti. Drážďany a Terezín jako místa vzpomínek na ŠOA“. Skripty

vytvořené pro generování KML odkazující na modely s různými úrovněmi detailu budou (možná s drobnými úpravami) také využity pro účely projektu.

4 ZÁVĚR

Řešení tématu práce vycházelo z teoretického pozadí nastíněného v teoretické části práce. Pro testovací účely byly s využitím aplikace Trimble SketchUp vytvořeny modely tří fiktivních bloků budov ve městě Terezín. Modely byly zkonstruovány tak, aby respektovaly hierarchické členění do několika úrovní detailu, převzaté ze CityGML specifikace. Modely posloužily pro realizaci několika způsobů vizualizace, z nichž nakonec byla vybrána nejvhodnější varianta.

Pro další testování byl model výrazně rozšířen tak, aby jeho složitost a rozlehlost odpovídala celému městu Terezín. Po tomto rozšíření již nebylo možné model rozumně spravovat ručně a proto byla v software AcrGIS Desktop vytvořena databáze obsahující mj. informace o půdorysech budov. Databáze spolu se skriptem v jazyce Python byla použita pro řešení dynamické vizualizace modelu. Výsledkem zpracování skriptu je KML kód, který spojuje dohromady jednotlivé budovy a pro každou budovu zajišťuje přepínání jednotlivých úrovní detailu.

Další část práce se věnuje řešení problémů, které se vyskytly při vizualizaci celého modelu města a pokouší se optimalizovat způsob vizualizace v aplikaci Google Earth. Prohlížení původní verze modelu bylo totiž jen stěží možné označit jako interaktivní. Prvním krokem bylo vypuštění nejnižší úrovně detailu (LoD0). Výsledkem pak byl kromě výrazného zjednodušení KML kódu a zrychlení vizualizace také subjektivně lepší vzhled modelu. Další, dokonce ještě výraznější vliv na plynulost vizualizace pak přineslo sloučení modelů budov do celých bloků.

Výsledky práce ukázaly, jaký bude vzhled modelu města Terezín, jaké aspekty mají vliv na rychlost načítání a zobrazování modelu a jak je lze ovlivnit. Tyto poznatky budou uplatněny v projektu „Krajina paměti. Drážďany a Terezín jako místa vzpomínek na ŠOA“ a budou mít zásadní vliv na strukturu modelu města. Přesto, že téma skrývá řadu dalších námětů pro další bádání, je možné konstatovat, že cíle práce byly naplněny.

Použitá literatura a zdroje

[1] ADOBE. *3D PDF sample files* [online]. [cit. 2013-3-17]. Dostupné z: < http://www.adobe.com/manufacturing/3dpdfsamples/3dsolutions/ >

[2] ArcGIS Resource Center. *Desktop Help 10.0 - Working with ArcGlobe and ArcScene* [online]. Aktualizace: 6.11.2012 [cit. 2013-3-17]. Dostupné z: < http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Working_with_ArcGlobe_and_ArcScene/00q8000000sv000000/ >

[3] BELAI, Elena. *3D model památkově chráněného areálu na základě kombinace jeho geodetické a architektonické dokumentace*. Plzeň, 2012. 57s. Bakalářská práce na Fakultě aplikovaných věd Západočeské univerzity v Plzni na katedře matematiky. Vedoucí diplomové práce Ing. Karel Jedlička, Ph. D.

[4] BENTLEY. *Interactive 3D Within Adobe PDF* [online]. [cit. 2013-3-17]. Dostupné z: < http://www.bentley.com/en-US/Products/MicroStation/Interactive+3D+PDF.htm >

[5] BŘEHOVSKÝ, M., JEDLIČKA, K., ŠÍMA, J. *Úvod do geografických informačních systémů. Přednáškové texty* [online]. [cit. 2013-4-9]. Dostupné z: < http://gis.zcu.cz/studium/ugi/e-skripta/ugi.pdf >

[6] CAD wiki. *Nejpoužívanější CAD formáty* [online]. [cit. 2013-4-14]. Dostupné z: < http://www.cadwiki.cz/Nejpouzivanejsi-CAD-formaty.ashx >

[7] *CityGML - Exchange and Storage of Virtual 3D City Models* [online]. Aktualizace: 24.4.2012 [cit. 2013-3-17]. Dostupné z: < http://www.citygml.org/ >

[8] *CityGML - Wikipedia, The Free Encyclopedia* [online]. Aktualizace: 2.3.2013 [cit. 2013-3-17]. Dostupné z: < http://en.wikipedia.org/wiki/CityGML >

[9] FIKEJZ, Jan. *Možnosti technologie Google Earth pro 3D vizualizaci geografických dat*. Plzeň, 2009. 71s. Diplomová práce na Fakultě aplikovaných věd Západočeské univerzity v Plzni na katedře matematiky. Vedoucí diplomové práce Ing. Radek Fiala.

[10] GEOMETRA OPAVA. *Technologie TerraSuite* [online]. [cit. 2013-3-17]. Dostupné z: < http://www.geometra-opava.com/cz/v_terra.php >

[11] *GeoVRML.org. 3D CONSORTIUM* [online]. [cit. 2013-4-14]. Dostupné z: < http://www.ai.sri.com/geovrml/ >

[12] Google Developers. *Keyhole Markup Language. Developer´s Guide* [online]. Aktualizace: 24.2.2012 [cit. 2013-4-22]. Dostupné z: < https://developers.google.com/kml/documentation/topicsinkml >

[13] Google Developers. *Keyhole Markup Language. KML Reference* [online]. Aktualizace: 2.1.2013 [cit. 2013-4-22]. Dostupné z: < https://developers.google.com/kml/documentation/kmlreference >

[14] Google Earth Help. *About KML* [online]. [cit. 2013-4-22]. Dostupné z: < http://support.google.com/earth/bin/answer.py?hl=en&answer=148118 >

[15] HRÁDKOVÁ, Monika. *Studie možností sběru a zpracování podrobných 3D dat pro účely památkové péče*. Plzeň, 2009. 102s. Diplomová práce na Fakultě aplikovaných věd Západočeské univerzity v Plzni na katedře matematiky. Vedoucí diplomové práce Ing. Radek Fiala.

[16] *ISO Standards* [online]. [cit. 2013-4-14]. Dostupné z: < http://www.iso.org/iso/home/standards.htm >

[17] JEDLIČKA,Karel. *3D data v GIS III.* Přednáškové texty z předmětu KMA/AGI. [cit. 2013-3-17]

[18] KALÁBOVI. *Transformace, reprezentace a zobrazení 3D objektů* [online]. Aktualizace: 3.7.2012 [cit. 2013-4-14]. Dostupné z: < http://kalabovi.org/pitel:isz:3d >

[19] *Keyhole Markup Language - Wikipedia, The Free Encyclopedia* [online]. Aktualizace: 7.4.2013 [cit. 2013-4-14]. Dostupné z: < http://en.wikipedia.org/wiki/Keyhole_Markup_Language >

[20] KOUCKÁ, Lucie. *3D rekonstrukce zaniklých částí města Dobříš*. 2011. 49 s. Bakalářská práce na Přírodovědecké fakultě Univerzity Karlovy v Praze na katedře aplikované geoinformatiky a kartografie. Vedoucí bakalářské práce RNDr. Přemysl Štych, Ph.D.

[21] KREJNÝ, Milan. *Prostorová vizualizace geodetických dat* [online]. 2004. Diplomová práce na Fakultě stavební Českého vysokého učení technického v Praze na katedře mapování a kartografie. Vedoucí diplomové práce Ing. Petr Soukup. [cit. 2013-3-17]. Dostupné z: < http://geo2.fsv.cvut.cz/~soukup/dip/krejny/index.htm >

[22] *OGC Standards* [online]. [cit. 2013-4-9]. Dostupné z: < http://www.opengeospatial.org/standards/is >

[23] Open Geospatial Consortium. *OGC City Geography Markup Language (CityGML) Encoding Standard*. 2012. 326 s.

[24] Open Geospatial Consortium Inc. *OGC KML* [online]. 2008. 233 s. Dostupné z: < http://portal.opengeospatial.org/files/?artifact_id=27810 >

[25] PENČEV, Filip. *ATLAS DMT – projektování a geodetické práce* [online]. [cit. 2013-3-17]. Dostupné z: < http://www.cad.cz/gis/80-gis/2254-atlas-dmt-projektovani-a-geodeticke-prace.html >

[26] POPELKA, Stanislav. *Google a ArcGIS. Nové možnosti v 3D vizualizaci*. 2008. 60 s. Bakalářská práce na Přírodovědecké fakultě Univerzity Palackého v Olomouci na katedře geoinformatiky. Vedoucí bakalářské práce Mgr. Kamil Vykopal.

[27] RUSSNÁK, Jan. *3D model areálu Přírodovědecké fakulty Masarykovy univerzity*. 2012. 85 s. Diplomová práce na Přírodovědecké fakultě Masarykovy univerzity v Brně. Vedoucí diplomové práce RNDr. Tomáš Řezník, Ph.D.

[28] ŘEHÁK, Tomáš. *Analytické možnosti GIS nad rastrovými daty*. 2008. 76 s. Diplomová práce na Fakultě aplikovaných věd Západočeské univerzity v Plzni na katedře matematiky. Vedoucí diplomové práce Ing. Karel Jedlička.

[29] SAMEC, Václav. *Modelování pomocí lokálních deformací*. 2005. 41 s. Diplomová práce na Fakultě informatiky Masarykovy univerzity. Vedoucí diplomové práce Ing. Jiří Sochor, CSc.

[30] SKYLINE. *Products Overview* [online]. [cit. 2013-3-17]. Dostupné z: < http://www.skylinesoft.com/skylineglobe/corporate/products/overview.aspx >

[31] *Slovník VÚGTK* [online]. [cit. 2013-3-17]. Dostupné z: < http://www.vugtk.cz/slovnik/ >

[32] STRACHOTA, Pavel. *Modelování pevných těles* [online]. Aktualizace: 22.3.2013 [cit. 2013-4-9]. Dostupné z: < http://saint-paul.fjfi.cvut.cz/base/public-filesystem/admin-upload/POGR/POGR2/05.modelovani_teles.pdf >

[33] STREJCOVÁ, Jana. *Digitální 3D model zámku Nečtiny*. 2010. 42 s. Bakalářská práce na Fakultě aplikovaných věd Západočeské univerzity v Plzni na katedře matematiky. Vedoucí bakalářské práce Ing. Radek Fiala.

[34] TRIMBLE. *SketchUp* [online]. [cit. 2013-3-17]. Dostupné z: < http://www.sketchup.com/ >

[35] *Visualization (computer graphics) - Wikipedia, The Free Encyclopedia* [online]. Aktualizace: 11.3.2013 [cit. 2013-3-17]. Dostupné z: < http://en.wikipedia.org/wiki/Visualization_(computer_graphics) >

[36] *Web 3D CONSORTIUM: Open Standards for Real-Time 3D Communication. What is X3D?* [online]. [cit. 2013-4-14]. Dostupné z: < http://www.web3d.org/realtime-3d/x3d/what-x3d/ >

[37] WIKIMEDIA COMMONS. *File:CSG Tree 2.png* [online]. Aktualizace: 6.1.2012 [cit. 2013-4-17]. Dostupné z: < http://commons.wikimedia.org/wiki/File:CSG_Tree_2.png >

[38] ŽELEZNÝ, Miloš. *Zpracování digitalizovaného obrazu. Katerda kybernetiky.* Přednáškové texty z předmětu KKY/ZDO. [cit. 2013-4-9]

Obsah přiloženého CD

Přiložený disk CD obsahuje následující složky:

- **ArcGIS**

– hlavním obsahem této složky jsou dva MXD dokumenty *terezin_bloky.mxd* a *terezin_budovy.mxd*. Ostatní obsah je zde uložen proto, aby správně fungovaly všechny vrstvy, uložené v dokumentech.

- **Google Earth bloky**

– složka obsahuje KMZ soubory představující 3D modely bloků budov. Dále jsou zde tři KML soubory *Python_bloky_maximum.kml* , *Python_bloky_odmocnina.kml* a *Python_bloky_uhlopricka.kml*. Tyto soubory představují výslednou vizualizaci pro každou ze třech zmíněných metod.

- **Google Earth budovy**

– složka obsahuje KMZ soubory představující 3D modely budov. Dále jsou zde tři KML soubory *Python_budovy_maximum.kml* , *Python_budovy_odmocnina.kml* a *Python_budovy_uhlopricka.kml*. Tyto soubory představují výslednou vizualizaci pro každou ze třech zmíněných metod.

- **Předloha pro kopírování**

– složka obsahuje předlohy ve formátu KMZ a SKP, použité pro kopírování a umisťování komponent

- **Python**

– složka obsahuje skripty vytvořené pro všech 6 variant vizualizace, a dále textové soubory, které jsou výstupy těchto skriptů. Pro spuštění skriptů je nutné nejprve nastavit cestu do adresáře, obsahujícího všechny zde zmíněné složky (tzn. změnit třetí řádku skriptu).

- **SketchUp**

  – složka obsahuje modely tří bloků budov, každý ve čtyřech úrovních detailu

- **Text práce**

- **Varianty vizualizace bloku**

  – složka obsahuje několik KMZ souborů, představujících různé varianty vizualizace bloku budov.

Přílohy

Příloha 1: Ukázka části KML kódu

```
<?xml version="1.0" encoding="windows-1250"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:gx="http://www.google.com/kml/ext/2.2">
<Document>

    <LookAt>
      <longitude>14.149736</longitude>
      <latitude>50.510691</latitude>
      <altitude>0</altitude>
      <range>1500</range>
      <tilt>70</tilt>
      <heading>0</heading>
      <altitudeMode>relativeToGround</altitudeMode>
    </LookAt>

  <NetworkLink>
    <Region>
      <LatLonAltBox>
        <north>50.511643</north>
        <south>50.511372</south>
        <east>14.147408</east>
        <west>14.147071</west>
        <minAltitude>0</minAltitude>
        <maxAltitude>9</maxAltitude>
        <altitudeMode>relativeToGround</altitudeMode>
      </LatLonAltBox>
      <Lod>
        <minLodPixels>1</minLodPixels>
        <maxLodPixels>27.0</maxLodPixels>
      </Lod>
    </Region>
    <Link>
      <viewRefreshMode>onRegion</viewRefreshMode>
      <href>FIII_1_LOD0.kmz</href>
    </Link>
  </NetworkLink>
```

Příloha 2: Python skript

```
import arcpy
import math
cesta = "D:\Skola\Diplomova_prace\CD"

fw = open ( cesta + "\Python\Python_budovy_maximum.txt","w")
fw.write('<?xml version="1.0" encoding="windows-1250"?>\n')
fw.write('<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:gx="http://www.google.com/kml/ext/2.2">\n')
fw.write('<Document>\n\n')

fw.write('<LookAt>\n')
fw.write('<longitude>14.149736</longitude>\n')
fw.write('<latitude>50.510691</latitude>\n')
fw.write('<altitude>0</altitude>\n')
fw.write('<range>1500</range>\n')
fw.write('<tilt>70</tilt>\n')
fw.write('<heading>0</heading>\n')
fw.write('<altitudeMode>relativeToGround</altitudeMode>\n')
fw.write('</LookAt>\n\n')

rows = arcpy.SearchCursor(cesta + "\ArcGIS\Terezin.gdb\pudorysy")
for row in rows:
    sever = row.sever
    jih = row.jih
    vychod = row.vychod
    zapad = row.zapad
    vyska = str(row.vyska)
    nazev = str(row.nazev)
    sirkaBBoxu = (sever - jih) * 3600 * 30
    delkaBBoxu = (vychod - zapad) * 3600 * 20
    pixelyZaklad = round(max(sirkaBBoxu, delkaBBoxu))

    for i in range(0,4):
        fw.write('<NetworkLink>\n')
        fw.write('<Region>\n')
        fw.write('<LatLonAltBox>\n')
        fw.write('<north>' + str(sever) + '</north>\n')
        fw.write('<south>' + str(jih) + '</south>\n')
        fw.write('<east>' + str(vychod) + '</east>\n')
        fw.write('<west>' + str(zapad) + '</west>\n')
        fw.write('<minAltitude>0</minAltitude>\n')
        fw.write('<maxAltitude>' + vyska + '</maxAltitude>\n')
        fw.write('<altitudeMode>relativeToGround</altitudeMode>\n')
        fw.write('</LatLonAltBox>\n')
```

```
            fw.write('<Lod>\n')
            if i==0:
                fw.write('<minLodPixels>1</minLodPixels>\n')
                fw.write('<maxLodPixels>' + str(pixelyZaklad) + '</maxLodPixels>\n')
            elif i==1:
                fw.write('<minLodPixels>' + str(pixelyZaklad) + '</minLodPixels>\n')
                fw.write('<maxLodPixels>' + str(2 * pixelyZaklad) + '</maxLodPixels>\n')
            elif i==2:
                fw.write('<minLodPixels>' + str(2 * pixelyZaklad) + '</minLodPixels>\n')
                fw.write('<maxLodPixels>' + str(8 * pixelyZaklad) + '</maxLodPixels>\n')
            else:
                fw.write('<minLodPixels>' + str(8 * pixelyZaklad) + '</minLodPixels>\n')
                fw.write('<maxLodPixels>-1</maxLodPixels>\n')

            fw.write('</Lod>\n')
            fw.write('</Region>\n')
            fw.write('<Link>\n')
            fw.write('<viewRefreshMode>onRegion</viewRefreshMode>\n')
            fw.write('<href>' + nazev + '_LOD' + str(i) + '.kmz' + '</href>\n')
            fw.write('</Link>\n')
            fw.write('</NetworkLink>\n\n')

fw.write('</Document>\n')
fw.write('</kml>\n')
fw.close()
```