3D Vizualizácia mapy

Projekt na predmet Tvorba informačných systémov
Analýza technológií, dekompozícia a dátový model

Vedúci projektu:
- Peter Náther

Členovia vývojárskeho tímu:
- Timotej Jurášek
- Martin Miklis
- Jakub Motýľ

Verzia dokumentu 1.1

1. Analýza technológií

Pojmy:
omap - jedná sa o súbory podobné xml. Slúžia teda na ukladanie štruktúrovaných dát, v našom prípade údajov o vrstevniciach v mape
parsovanie - spracovanie dát z textových súborov do dátových štruktúr
mesh - kostra grafického modelu

Pri tvorbe aplikácie je potrebné zvoliť technológie, ktoré podporujú koncepty použité v návrhu tejto aplikácie.

Robiť aplikáciu od úplného základu sme považovali za stratu času, preto sme sa rozhodli použiť niektoré z existujúcich prostredí, ktoré podporujú prácu s modelmi a umožňujú rýchlo vytvoriť príjemné a praktické GUI.

Po úvahách a diskusiách sme sa zhodli, že najvhodnejším prostredím pre tvorbu našej aplikácie bude Unity3D. Unity3D má viacero výhod, od jednoduchého budovania modelov (použitím meshov), ktoré bude jadrom našej aplikácie, cez možnosť používať knižnice C#, ktoré nám dovolia jednoducho parsovať omap súbory (použitím štandardnej triedy XmlReader), po jednoduchú distribúciu aplikácie na ľubovolnú platformu. Taktiež nám vyhovuje možnosť v scéne pracovať s kamerou, čo nám umožní sústrediť sa na funkcionalitu, miesto toho, aby sme investovali čas do prepočtov rotácii modelov, pannovania a zoomovania. Unity3D sa teda javí ako najvhodnejšie vzhľadom na naše požiadavky.

Ďalej sme sa zaoberali použitím prostredia Panda3D. Avšak narazili sme na viaceré problémy. Jedným z problémov boli samotné programovacie jazyky podporované v prostredí. Python sme pre túto rozsiahlejšiu aplikáciu nevybrali kvôli menej vhodným nástrojom na manipuláciu s kódom.
Pri C++ nám nevyhovovalo, že štandardné knižnice nepodporujú spracovanie xml súborov.

2. Model vstupného súboru Omap

Súbor sa skladá z hlavičky súboru a jadra.

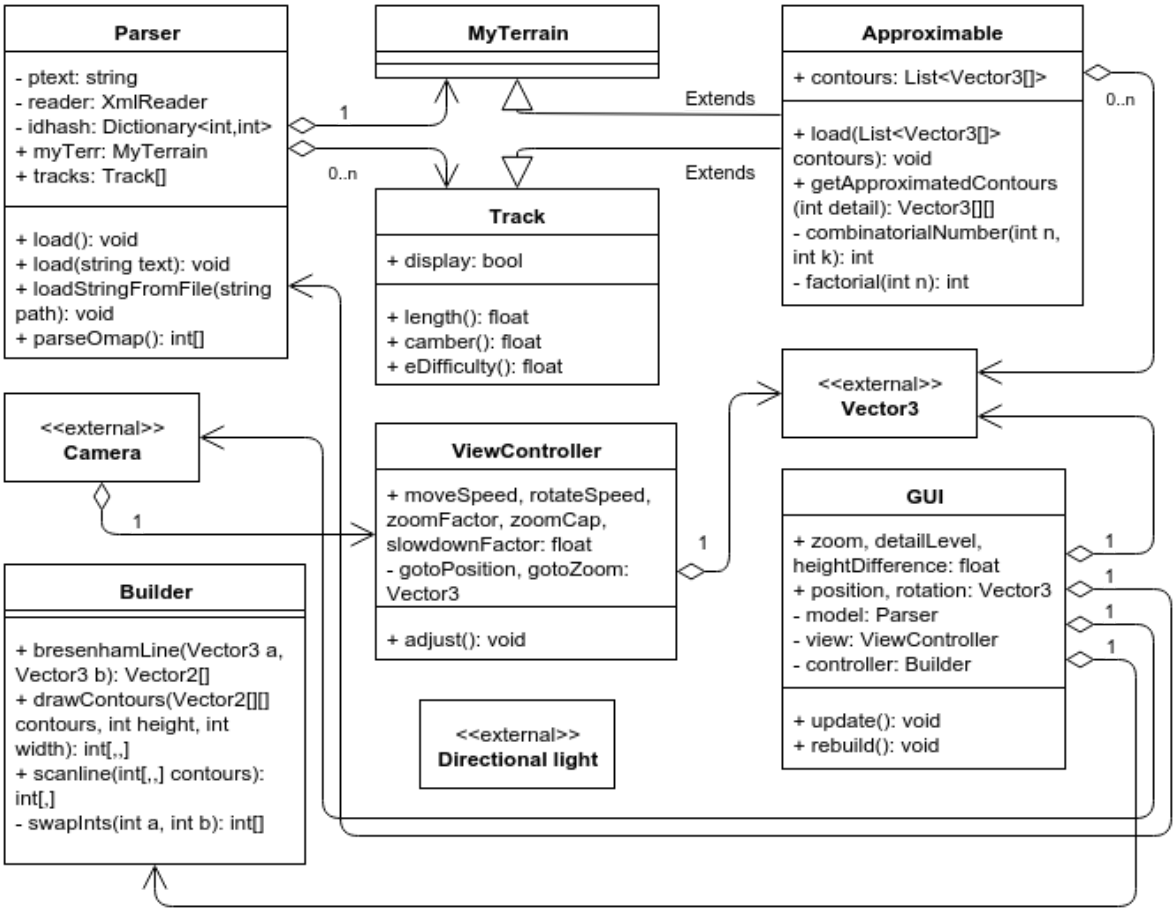
Hlavička obsahuje definície a popisy jednotlivých typov terénu, ktoré sa nachádzajú v súboroch omap. Hlavička je pre nás kľúčom, podľa, ktorého vieme zistiť aké objekty sa nachádzajú v jadre súboru.

Jadro súboru má vlastnú hlavičku, v ktorej sú definované iba terénne objekty použité v konkrétnej mape. Ich ID sú mapované cez hlavičku súboru na konkrétny typ terénu. Nás budú v základnej verzii zaujímať len vrstevnice a trasy. Jedná sa konkrétne o objekty s kódom 101,102,103, 104 a 105. ID sú pridelené v hlavičke jadra súboru, na objekty sa súbor neskôr odvoláva len použitím týchto ID.

Nasleduje samotný výčet všetkých objektov so súradnicami bodov, ktoré tieto terény definujú a upresnením, či sa jedná o hmotný bod objektu alebo len bod používaný na presnejšie dokreslenie kriviek. Toto upresnenie je vyjadrené pomocou flagu. Z tohto

zoznamu objektov vyberieme tie, ktoré majú nami požadované ID a zapamätáme si všetky body jednotlivých objektov v objektoch jednotlivých vrstevníc a trás.

3. Dátový model



4. Podrobná špecifikácia komponentov

Parser - komponent načítavajúci a držiaci údaje potrebné pre chod aplikácie

load([string text]) - načíta zo vstupného súboru oMap potrebné dáta o vrstevniciach. Metóda na načítavanie používa XmlReader a počas načítavania naplní pole vrstevníc v objekte MyTerrain. Okrem toho naplní aj objekty Track údajmi o trasách.

Approximable - má údaje o kontúrach (vrstevnice alebo cesta trasy) z objektu parser pomocou metódy load.

getApproximatedContours(int detail) - vráti kontúry aproximované Bernsteinovým polynómom.

Track - Drží v sebe údaje o trase.

length() - vráti celkovú priestorovú dĺžku trasy

camber() - vráti celkové prevýšenie trasy

eDifficulty() - vráti energetickú náročnosť trasy

Builder - Enkapsuluje metódy pre prácu s terénom a trasami.

bresenhamLine(Vector3 a, Vector3 b) - vráti súradnice úsečky medzi bodmi *a* a *b* rasterizovanej Bresenhamovým algoritmom.

drawContours(Vector2[][] contours, int height, int width) - zakreslí a vráti pole z rasterizovaných úsečiek na mape veľkosti *height* x *width*

scanline(int[,,] contours) - preskenuje vodorovne a zvisle vykreslené pole úsečiek a vráti mapu reprezentujúcu výšku na jednotlivých poliach.

ViewController - Združuje ovládacie funkcie kamery. Približovanie, rotácia a pozícia v scéne sa ovláda týmto komponentom.

adjust() - podľa užívateľského vstupu (myšou) upraví vlastnosti kamery v scéne.

GUI - Užívateľské rozhranie zobrazujúce všetky informácie dostupné užívateľovi, umožňuje aj ich editáciu.

update() - obnoví informácie o kamere v rozhraní

rebuild() - pretvorí terén a trasy v scéne na základe zmenených informácií v rozhraní

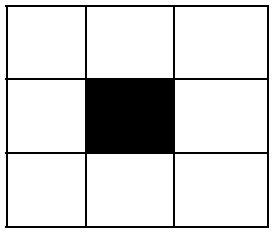
5. Testovacie scenáre

Overenie správnosti DDA algoritmu - testujeme správnosť algoritmu pomocou výsledkov na jednoduchých vstupoch, jedná sa o priamku rovnobežnú s x-ovou osou, uhlopriečku štvorca a jednu úsečku, ktorá je uhlopriečkou obdĺžnika.

Overenie správneho parsovania - Parser vypíše minimálnu a maximálnu hodnotu x-ových a y-ových súradníc. Ďaľej skontrolujeme, či sa zo súboru načítali vrstevnice a koľko ich bolo.

Overenie správnej aproximácie kriviek vrstevníc - testy overujú, či správne vytvárame body zo splinov. Pre cyklickú a acyklickú vrstevnicu. Overujeme správnosť Bernsteinových polynómov.

Vizuálne overenie správnosti heightmáp - vizuálne overíme, či naša aplikácia správne vytvorila heightmapu pre súbor.
Napríklad:



Tento vstup má vrátiť mapu, ktorá ma v strede vyvýšenú platformu a okolo rovinu.

Užívateľské testovanie zoomovania, presúvania a rotácie v scéne

Otestovanie správnosti výpočtu náročnosti trasy

Otestovanie správnosti výpočtu dĺžky trasy

6. Popis použitých algoritmov

Aplikácia načíta zo súbora údaje o vrstevniciach mapy vo formáte omap.

Tieto dáta, vo forme splinov, aproximuje použitím Bernsteinových polynómov, čím dostaneme konkrétne body ležiace na vrstevniciach.
Do dvojrozmerného poľa vyrasterizujeme pomocou Bresenhamovho algoritmu úsečky týchto vrstevníc.
Pokračujeme dvojitým použitím scanline algoritmu na štvorcovej sieti, tým dostaneme odhad toho, aká je výška jednotlivých plôch medzi vrstevnicami.
Následne dáta kvantizujeme, čím zmenšíme ich objem pre rýchlejší beh programu.
Z týchto dát vytvoríme heightmapu ofarbenú odtieňmi šedej, ktorá bude symbolizovať výšky jednotlivých políčok dvojrozmerného poľa. Túto heightmapu ďalej posunieme interface-u Unity3D, ktorý vypracuje z nej trojrozmerný objekt, ktorý užívateľovi zobrazíme.