VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

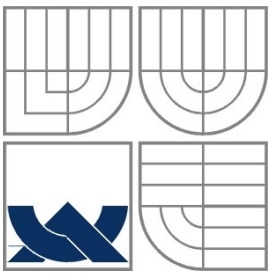
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELIGENT SYSTEMS

GENERÁTOR MÁP PRE FANTASY HRY NA HRDINOV

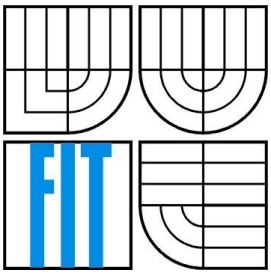
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ANTONÍN PAGÁČ
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELIGENT SYSTEMS

GENERÁTOR MAP PRO FANTASY HRY NA HRDINY

MAP GENERATOR FOR ROLE-PLAYING GAMES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ANTONÍN PAGÁČ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ALEŠ SMRČKA, Ph.D.

BRNO 2012

Abstrakt

Cílem této práce je vytvořit generátor map pro fantasy hry na hrdiny. V práci je popsána problematika tvoření map v reálném světě, a jsou rozebrány různé přístupy ke tvorbě map jak v počítačových hrách, tak v myšlenkových a stolových hrách na hrdiny. Dále je navržen program, který je založen na generování Voroného diagramu a slouží k vytváření map. Generátor umožňuje umísťovat objekty na mapu náhodně pomocí uživatelských příkazů, nebo na konkrétní pozici pomocí grafického uživatelského rozhraní. Generátor je implementován v jazyce Javascript, který umožňuje spouštění programu v internetovém prohlížeči na různých zařízeních.

Abstract

The aim of this work is to develop a generator of maps used in fantasy role-playing games. The issues of map processing and map creation in real world are described. The work also discuss a number of approaches to map generation in computer games, mind games or board games. Next, a program based on generation of Voronoi diagram is presented, which serves as map creation software. The generator has the means to place objects on map randomly, using user commands, or place objects to specific place using graphical user interface. Generator is implemented in Javascript, which enables the program to be run in web browser on many different devices.

Klíčová slova

mapa, generátor, náhoda, hra, rpg, fantasy, Voroného diagram, Lloydova relaxace

Keywords

map, generator, random, game, rpg, fantasy, Voronoi diagram, Lloyd relaxation

Citace

Antonín Pagáč: Generátor máp pre fantasy hry na hrdinov, bakalářská práce, Brno, FIT VUT v Brně, 2012

Generátor máp pre fantasy hry na hrdinov

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doktora Aleše Smrčky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Antonín Pagáč
23. dubna 2012

Poděkování

Zde bych chtěl poděkovat doktoru Aleši Smrčkovi za odborné vedení práce a za poskytnutí cenných rad a užitečných informací.

Obsah

1 Úvod

Táto práca sa zaoberá náhodným generovaním vymyslených máp. Zameriava sa na vytváranie máp využívaných pri fantasy hrách na hrdinov (tzv. RPG - role playing games) a dáva dôraz hlavne na ich použiteľnosť, ale aj realistickosť.

Ak sa človek pohybuje v rozľahlejšom prostredí, vzniká prirodzene potreba grafického stvárnenia okolia tak, aby bolo jasné, kde sa práve nachádza, ktorým smerom leží jeho ciel a ako sa k tomuto cieli dostane. Inak tomu nie je ani pri hrách, kde sa v prostredí nepohybuje človek fyzicky, ale vytvára si na to postavu, nereálny charakter, s ktorým sa v priebehu hry stotožňuje. Pri myšlienkových hrách, ako napríklad Dračí Doupě, môže byť nepoužívanie mapy prostredia príčinou rozporu medzi hráčmi, ktorí nemajú jednotnú predstavu, kde sa ich postavy vo svete nachádzajú. Použitie mapy pri takýchto hrách teda prispieva k ich hrateľnosti a k celkovému umocneniu zážitku z hry. Mapa teda slúži ku komunikácii a k lepšiemu vzájomnému porozumeniu komunikujúcich strán. Poskytuje grafický materiál na podporu hovoreného slova.

Podla [1] znie definícia mapy takto: Mapa je reprezentácia vybraných materiálnych alebo abstraktných znakov území, ktoré sa nachádzajú na povrchu Zeme alebo sa k zemskému povrchu vzťahujú, zobrazuje povrch Zeme obvykle v mierke a na plochom médiu.

Definícií pojmu mapa existuje, okrem uvedenej, ešte niekoľko, pre účely tejto práce však nie je nutné ich uvádzať. V tejto práci sa bude mapou rozumieť aj reprezentácia území a objektov, ktoré sa nenachádzajú na zemskom povrchu, a ktoré boli vytvorené na ilustráciu prostredia, v ktorom sa pohybujú hrdinovia fantasy hier.

Vývoj moderných technológií umožňuje aj do stolných hier typu Dračí Doupě integrovať výpočtovú techniku a vytváranie máp tým zjednodušiť, zrýchliť a spríjemniť. Ak bolo v minulosti potrebné použiť papiere, písacie potreby a gumu, dnes je možné využiť tablet s vhodným softvérovým vybavením.

Generátor je obecne program, ktorý na základe vstupných údajov tvorí dokumenty, alebo iné programy. V praxi sa generátory používajú k zvýšeniu produktivity práce a k zníženiu chybovosti. Výhodou je tiež znovupoužitie generátoru, ku ktorému dochádza pri zmene požiadavkov na výslednú, generovanú štruktúru - upraví sa buď generátor, alebo vstupný súbor, a celá štruktúra sa vygeneruje znova.

V [2] sa uvádza delenie generátorov na pasívne a aktívne. Pasívne generátory generujú v určitom prostredí a na základe určitých vstupných údajov výsledok a skončia. Ak je potrebné vykonať nejakú zmenu vo výsledku, napr. zmeniť riadok v texte, je nutné celý proces spustiť znova. Aktívne generátory sú na rozdiel od pasívnych zodpovedné nielen za vygenerovanie výslednej štruktúry, ale tiež za jej údržbu, takže pri nutnosti zmeny stačí túto zadať do generátoru a nie je nutné celý proces spúšťať od začiatku.

Oproti manuálnemu zadávaniu každého údaju do mapy má teda generovanie výhodu šetrenia času a zjednodušuje tiež úpravu výslednej mapy. Za nevýhodu sa dá považovať nutnosť softvérového vybavenia, teda samotného generátoru, a hardvérového vybavenia umožňujúceho činnosť generátoru.

V práci je popísaná problematika tvorby máp, súčasný stav programov pre vytváranie máp a hlavné problémy, ktoré sú v takýchto programoch riešené. Je uvedené zhodnotenie programov spolu s ich výhodami a nevýhodami. Po teoretickom úvode nasleduje návrh programového riešenia tvorby máp a popis jeho častí. Vo štvrtej kapitole je priblížená činnosť generátora s dôrazom na algoritmické riešenie konkrétnych problémov. Nasleduje overenie jazyka, použitého v programe na zadávanie požiadavkov na mapu, a práca končí záverom, v ktorom je zhodnotený prínos práce a navrhnuté možné postupy, ktorými by sa práca mohla ďalej rozširovať.

2 Tvorba máp

Táto kapitola sa venuje vytváraniu máp. Sú tu popísané programy, ktoré sa na vytváranie používajú v rôznych kontextoch - počítačové hry, stolové hry. Mapy sa z hľadiska použiteľnosti dajú rozdeliť na nasledujúce skupiny:

Mapy v reálnom svete - takéto mapy zobrazujú zemský povrch alebo plochu oblohy a sú použiteľné napríklad pri turistike, vyučovaní a všade tam, kde je potrebné presne zobraziť existujúce prostredie. Táto kapitola je zameraná na využívanie počítačových technológií a systémov pri tvorbe a spracovávaní máp.

Mapy v hrách zobrazujú neexistujúcu, fiktívnu krajinu alebo prostredie. Pri vytváraní takýchto máp hrá hlavnú úlohu predstavivosť človeka, ktorý mapu vytvára. V tejto kapitole je popísané rozdelenie máp podla podložia, na ktorom sú vykreslené, a podla miery manuálnej činnosti človeka, ktorý ich vytvára. Sú uvedené príklady máp a programov, ktoré vytvárajú, alebo pomáhajú mapy vytvárať.

2.1 Mapy v reálnom svete

Vytváraním máp a náukou o mapách sa zaoberá vedný obor – kartografia. Cieľom kartografie je objektívne zobrazenie skutočnosti pomocou mapy. K dosiahnutiu tohto cieľa sa používajú rôzne zobrazenia, z ktorých má každé svoje špecifické vlastnosti a každé sa používa na zobrazenie plochy iného tvaru a veľkosti. Pre účely tejto práce však nie je potrebné ďalej sa zobrazeniami zaoberať, keďže v prezentovanom programovom riešení sa nevyužívajú vlastnosti žiadneho zo zobrazení.

Pretože sa však jedná o počítačový program, je nutné vybrať vhodnú reprezentáciu zobrazovaných dát, a rozvrhnúť prístup k dátam. Tu je možné sa inšpirovať reálnymi systémami na spracovávanie a manipuláciu s mapami.

V tejto kapitole sú v krátkosti priblížené princípy spracovávania máp za použitia informačných technológií, výhody a nevýhody jednotlivých prístupov a dôraz je kladený na postupy, ktoré sú použité v implementovanom generátore.

2.1.1 Reprezentácia dát

V [3] sa uvádza, že pri spracovávaní máp existujú dva spôsoby reprezentácie grafických dát: vektorová reprezentácia a rastrová reprezentácia. Tieto dve reprezentácie sú od seba značne odlišné, a každá pracuje s inými pojmami. V tejto kapitole je uvedený popis oboch možností.

2.1.1.1 Vektorová reprezentácia dát

Jedná sa o prístup použitý aj v programovom riešení prezentovanom v tejto práci. Táto reprezentácia nepracuje s grafickou informáciou priamo, ale pracuje s dátovým modelom uloženým v pamäti formou dátových štruktúr. Základné prvky, s ktorými sa pri tejto reprezentácii manipuluje, sú bod, hrana a polygón.

Bod je základný geometrický element vektorovej prezentácie. Je určený v 2D alebo 3D kartézskom priestore, alebo geografickými súradnicami na povrchu referenčného elipsoidu či gule. Bod označuje izolovaný objekt (prameň, vrcholová kóta, pomník a pod.) alebo je prierezom elementu geometrickej reprezentácie hrany. V tejto práci je bod určený v 2D kartézskom priestore svojimi súradnicami.

Hrana je usporiadaná množina koncových a priebežných bodov, medzi ktorými sa predpokladajú priamkové spojenia. Hrana je geometrickým modelom čiastočného, alebo celkového obrazu líniového prvku alebo tvorí časť ohraničeného polygónu. Z topografického hľadiska predstavuje hrana prepojenie svojich koncových bodov nazývaných uzly (node). Pretože sú hrany orientované, je možné vymedziť plochu ležiacu od hrany napravo a naľavo. Hrany sa nemôžu v obraze prekrižovať, ak sa tak stane, tak je nutné bod kríženia zvoliť ako uzlový bod a vykonať dekompozíciu pretínajúcich sa líniových segmentov na dielčie hrany pôvodného obrazu.

Polygón je ucelená kompozícia vzájomne jednoznačne spojených hrán definujúca v priestore uzatvorenú oblasť, areál. Polygón je obrazom plošného prvku - pôdorysy veľkej stavby, súvislé vodné plochy, chránené krajinné oblasti a iné. V tejto práci sa myšlienka polygónu využíva nie len ako reprezentácia objektu na mape, ale aj ako základná, najmenšia a nedeliteľná jednotka podložia mapy.

Pri práci s vektorovou reprezentáciou dát a mapou je nutné udržovať medzi jednotlivými časťami (bod, hrana, polygón) v každom momente jednoznačné vzťahy tak, aby bolo vždy jasné, z akých častí sa jednotlivé objekty skladajú. V programe je potom možné takýmto spôsobom manipulovať s mapou aj bez samotného vykreslenia mapy, čo umožňuje oddeliť vykresľovanie do samostatného modulu.

2.1.1.2 Rastrová reprezentácia dát

V tejto práci nie je rastrová reprezentácia dát použitá, a táto stručná kapitola opisuje dôvod, prečo to tak je.

Rastrová reprezentácia dát je priamo viazaná na grafické zobrazenie daných dát. V programe by teda nebolo možné oddeliť samotnú manipuláciu s objektami od vykresľovania objektov. Vykresľovacie plátno je rozdelené pravidelnou mriežkou na rovnaké elementárne častice, z ktorých sa skladá obraz mapy. V programe by tak nebolo možné využiť myšlienku náhodne generovaného podložia, ktoré sa skladá z nepravidelných a rôzne veľkých polygónov. Vymedzovanie objektov v rastrovom obraze spočíva v nájdení a ohraničení súvislého zhluku buniek, ktoré majú napr. rovnakú intenzitu alebo textúru šedi a určenie typu objektu (napr. listnatý les) porovnaním tejto intenzity s predlohou. Tento prístup k určovaniu objektov sa zdá zbytočne zložitý, pri porovnaní s vektorovým prístupom, kedy objekt tvorí presne určený zhluk elementárnych polygónov.

2.1.2 Prístup k dátam

V [4] sa píše o dvoch možných prístupoch k dátam - vrstvovom a objektovom. Opäť je každý špecifický a každý vychádza z niečoho iného. Táto kapitola v krátkosti zhŕňa dôležité vlastnosti jednotlivých prístupov a určuje prístup použitý vo vytvorenom programe.

2.1.2.1 Objektový prístup k dátam

Prístup k dátam, použitý v programovom riešení prezentovanom v tejto práci, je založený na princípoch objektového prístupu k dátam. Tento prístup používa myšlienku objektového programovania: každý objekt na mape (les, jazero) je reprezentovaný objektom v programe, má svoje atribúty a metódy a pod. V prezentovanom programe však nie je každý objekt na mape reprezentovaný objektom, ale štruktúrou, znamená to teda že má vlastné atribúty, ale nemá vlastné metódy.

2.1.2.2 Vrstvový prístup k dátam

Vrstvový prístup k dátam pracuje tak, že združuje dáta do tematických vrstiev. Tento spôsob má svoj základ v používanom spôsobe vytvárania máp v kartografii. Ak by bol v programe použitý tento prí-

stup, vznikla by otázka manipulácie s objektami v rámci jednej vrstvy, prípadne taká zmena objektu, ktorá si vyžaduje zmenu údajov v inej vrstve.

2.2 Mapy v hrách

Táto kapitola pojednáva o mapách v hrách počítačových, doskových aj myšlienkových. K hraniu hier počítačových je nutný počítač (alebo zariadenie fungujúce na podobnom princípe), k hraniu hier stolových je potrebný hrací plán a hracie kamene, a k hraniu hier myšlienkových je nutná len ľudská predstavivosť.

Mapy v hrách sa všeobecne dajú rozdeliť podla kritéria tvorby - mapy vytvárané manuálne, ktoré kreslí alebo iným spôsobom vytvára človek bez automatizovanej pomoci, a mapy vytvárané generovaním, ktoré vytvára generátor za pomoci dopredu definovaných častí a s minimálnou korekciou vykonávanou človekom. Mapy sa ďalej dajú rozdeliť na mapy, používané v myšlienkových hrách typu Dračí Doupě, a mapy vytvárané za účelom použitia v počítačových hrách. Iné kritérium rozdelenia máp je podla podložia, na ktorom sú vykreslené. Takéto podložia môžu byť štvorcové, hexagonálne, alebo je mapa nakreslená na ploche bez podložia.

V tejto kapitole sú postupne popísané jednotlivé rozdelenia, doplnené o príklady programov, ktorými sa dané typy máp dajú vytvárať, a o ilustračné obrázky.

2.2.1 Mapy podla podložia

Podložím mapy sa rozumie podklad, vzor na plátne, na ktorom je mapa vykreslená. Mapy sa podla podložia dajú rozdeliť na mapy so štvorcovým podložím, mapy s hexagonálnym podložím a mapy bez viditeľného podložia.

2.2.1.1 Mapy na štvorcovom podloží

Vytváranie máp na štvorcovom podloží je asi najrozšírenejšia a najjednoduchšia technika, ktorá sa používa hlavne pri ručnom kreslení mapy na papier. Štvorcové podložie poskytuje základnú oporu pre zobrazenie hraníc jednoduchých objektov, vzájomných vzdialeností objektov na mape a veľkosti objektov. Príklad mapy na tomto podloží je na obr. 2.1.

Vhodné na:

- interiéry budov,
- rozmiestnenie budov v mestách,
- chodby a miestnosti v jaskyniach.

Nevhodné na:

- zobrazovanie nepravidelne zakrivených línií (nie sú presne vymedzené hranice objektu),
- zobrazovanie skutočnej polohy hrdinov a interakcie medzi nimi (napr. komplikovaný diagonálny posun postáv).

2.2.1.2 Mapy na hexagonálnom podloží

Hexagonálne podložie je rozšírením štvorcového podložia - namiesto štvorcov sa podložie skladá z elementárnych šesťuholníkov. Takéto podložie poskytuje väčšiu voľnosť v znázorňovaní tvarov, kedy človek, vytvárajúci mapu, nie je obmedzený len na prísne štvorcové tvary. Vzdialenosti sa aj diagonálne pomerne jednoducho spočítajú. Na obr. 2.3 je zobrazená ukážka hexagonálneho podložia.

Vhodné na:

- zobrazenie polohy a pohybu postáv v rámci interakcie postáv.

Nevhodné na:

- kreslenie rovných hrán (nutnosť deliť šesťuholníky na menšie polia).

2.2.1.3 Mapy bez viditeľného podložia

Takéto mapy poskytujú pri tvorbe neobmedzené množstvo možností, ako zakresliť jednotlivé objekty a kam ich na mape umiestniť. Vytváranie mapy dáva priestor fantázii tvorcu. Tento typ máp sa najviac približuje reálnym mapám, a preto dodáva hre určitú realistickosť a vylepšuje tak zážitok z hry. Príklad takejto mapy je možné vidieť na obr. 2.2.

Vhodné na:

- zobrazenie širokého okolia postáv,
- zobrazenie nepravidelných tvarov objektov na mape.

Nevhodné na:

- znázorňovanie presného pohybu, prípadne interakcie postáv.

2.2.2 Mapy podla miery manuálnej tvorby

Mierou manuálnej tvorby sa v tejto kapitole rozumie manuálna činnosť človeka, vytvárajúceho mapu, vzhľadom ku činnosti počítača ako pomocného prostriedku. Mapy sa podľa tohto kritéria dajú rozdeliť na mapy vytvárané manuálne a mapy vytvárané generovaním.
Všetky programy, prezentované v častiach o doskových hrách, sú voľne dostupné a je možné ich používať zdarma. Programy uvádzané v častiach o počítačových hrách sú väčšinou dostupné so zakúpením príslušnej hry.

2.2.2.1 Mapy vytvárané manuálne

Patria sem mapy, ktoré človek kreslí na papier, alebo vytvára za pomoci počítačového programu. Takýto software však poskytuje iba prostriedky pre vybranie a umiestnenie konkrétneho objektu na vybrané miesto na mape, prípadne poskytuje možnosť daný objekt pomocou základných zložiek vytvoriť.

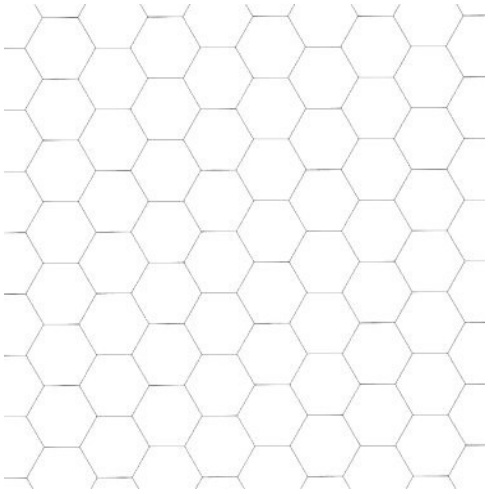
Mapy v doskových hrách

V [5] je definícia doskovej hry uvedená takto: Doskovou hrou sa rozumie taká hra, ktorá je realizovaná na doske stolu, na ktorej je plán hry, a pri hre sa používajú kamene alebo figúry. Tieto sú na dosku umiestňované a sú z nej odstraňované podla predom daných pravidiel. U klasických doskových hier je pevne daný hrací plán, pri moderných doskových hrách sa používa aj variabilný herný plán.

Doskovú hru je teda možné hrať bez akéhokoľvek technického vybavenia (v zmysle počítačových technológií), aj keď prvky počítačov sa postupne do týchto hier prepracovávajú. V dnešnej dobe je reálne, že hráči nebudú mať k dispozícii hracie kocky, alebo iné príslušenstvo, potrebné k hraniu doskovej hry, ale budú mať pri sebe niekoľko inteligentných mobilných telefónov alebo podobne zameraných elektronických zariadení. Existuje množstvo aplikácií, ktoré umožnia využitie takýchto prístrojov pri hre a nahradenie konvenčného príslušenstva takýchto hier, napr. elektronická hracia kocka, alebo špecifické tabuľkové procesory na spracovávanie údajov či akokoľvek prispôsobené generátory náhodných čísel.

Obrázok 2.1: Mapa na štvorcovom podloží

Obrázok 2.2: Mapa bez podložia



Obrázok 2.3: Ukážka hexagonálneho podložia

The Creator System

Jedná sa o jednoduchý editor, v ktorom je možné na štvorcovom podloží vytvárať mapy jaskýň a interiérov budov. Mapy sa vytvárajú jednoduchým umiestňovaním preddefinovaných objektov na podložie. Editor neposkytuje žiadnu možnosť úpravy vstavaných elementárnych objektov, je však možné si stiahnuť a importovať nový set objektov a ten potom použiť na vytvorenie mapy. Na obr. 2.4 je znázornené rozhranie tohoto editoru, spolu s ukážkou vytvorenej mapy.

Vhodné na:

- vytváranie interiérov budov,
- vytváranie chodieb a miestností v jaskyniach.

Nevhodné na:

- vytváranie máp exteriérov.

Hex - World Creator

Tento editor poskytuje možnosť vytvárať mapy na hexagonálnom podloží. Užívateľ má na výber množstvo preddefinovaných objektov, ktoré môže umiestňovať do jednotlivých šesťuholníkov. K dispozícii sú objekty typu usadlosti, vegetácia, divočina, voda, vrchy a špeciálne značky. Rozhranie editoru je na obr. 2.5.

Vhodné na:

- vytváranie širokého okolia - kontinenty, svety.

Nevhodné na:

- zobrazovanie detailov vybranej oblasti,
- vytváranie interiérov budov alebo jaskýň.

Pyromancers Dungeon Painter

Ide o rozsiahly on-line editor máp, ktorý v mape kombinuje vlastnosti máp vytváraných na štvorcovom podloží a máp vytváraných bez podložia. Užívateľ má na výber z veľkého množstva rôznych textúr na tvorbu otvoreného sveta, ale aj interiérov budov a jaskýň. Vybrané objekty je možné v tomto editore umiestňovať mimo štvorcovú mriežku a vznikajú tak nové možnosti tvorby mapy, a zvyšuje sa aj realistickosť mapy. Editor poskytuje možnosť uložiť rozpracovanú mapu, exportovať mapu do grafického formátu či vytvoriť z máp galériu. Rozhranie spolu s výslednou mapou je možné vidieť na obr. 2.6.

Vhodné na:

- zobrazovanie interiérov budov,
- vytváranie chodieb a miestností v jaskyniach,
- vytváranie rozsiahlejších oblastí.

Nevhodné na:

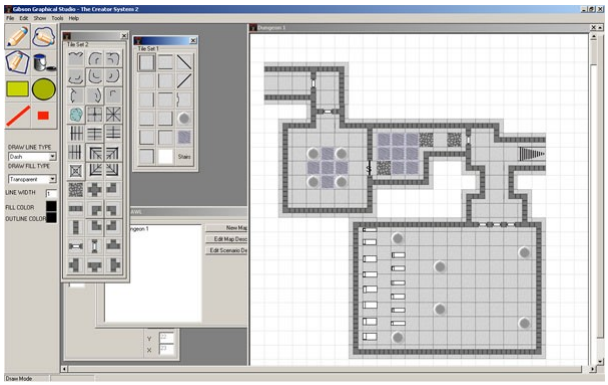
- vytváranie kontinentov, svetov.

Mapy v počítačových hrách

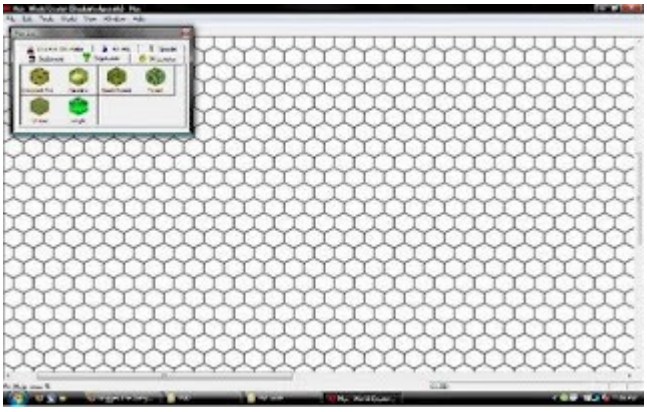
Jedná sa o mapy, ktoré sú zobrazované na obrazovke monitora a ktoré sa používajú pri hraní počítačových hier. Takéto mapy sa vytvárajú pomocou špecializovaných nástrojov - editorov. Jednotlivé editory sú väčšinou použiteľné pre jeden herný titul, obecné riešenie sa neobjavuje. Z tohoto dôvodu je ťažké presne poukázať na výhody a nevýhody jednotlivých editorov, preto sú takéto programy predstavené len s obecným popisom, kde sú uvedené ich vlastnosti a prístupy, ktoré pri vytváraní máp používajú.

Obecne sa dá povedať, že editory herných máp sa začali hojne používať až s príchodom 3D herných prostredí, ktoré umožňuje matematické vyjadrenie povrchu hernej mapy. U starších tituloch niektorých hier sa síce editor dodával, ale jeho možnosti boli značne obmedzené. Bolo to hlavne preto, že šlo v podstate o skladanie jednotlivých zložiek mapy interpretovaných ako obrázky vedľa seba. Pre účely tejto práce je uvedených niekoľko príkladov editorov, používaných pri vytváraní máp v počítačových hrách.

Údaje o počítačových hrách sú prevzaté z [9].



Obrázok 2.4: The Creator System



Obrázok 2.5: Hex World Creator

Obrázok 2.6: Pyromancers Dungeon Painter

UnrealEd

Je to editor používaný na vytváranie úrovní pre hru Unreal. Funguje ako program na editáciu trojrozmerných objektov, ktorý umožňuje skriptovanie chovania a funkcií jednotlivých objektov. Výstupom programu sú mapy malej veľkosti, ktoré sú vhodné práve pre akčné hry typu Unreal. Herná mapa nie je rozdelená na žiadne elementárne políčka, takže tvorca mapy má voľnosť vo vytváraní akýchkoľvek tvarov rôznych objektov. Rozhranie tohoto editoru je na obr. 2.7.

Bioware Aurora Neverwinter Nights

Editor, ktorý umožňuje vytvárať nové lokácie do hry Neverwinter Nights. Táto hra sa odohráva v trojrozmernom prostredí, ktoré je zložené zo štvorcových, na seba nadväzujúcich políčok. Pri vytváraní mapy je najprv nutné zvoliť typ prostredia, podla toho program zvolí najvhodnejšiu sadu polí, z ktorých sa bude skladať základný ráz krajiny. Po vytvorení takejto sady je možné na mapu umiestňovať ďalšie objekty. Napriek tomu, že sa tento systém tvorby máp môže zdať ako dostatočný, dá sa po chvíli používania programu natrafiť na opakujúce sa miesta. Tento problém sa dá odstrániť zväčšením množstva políčok, ale aj napriek tomu sa jedná o značne zväzujúci systém. Na druhú stranu je tento spôsob užívateľsky príjemný, a umožňuje rýchlu tvorbu herných máp. Rozhranie editoru spolu s ukážkou rozpracovaného objektu je na obr. 2.8.

C&C Generals: World Builder

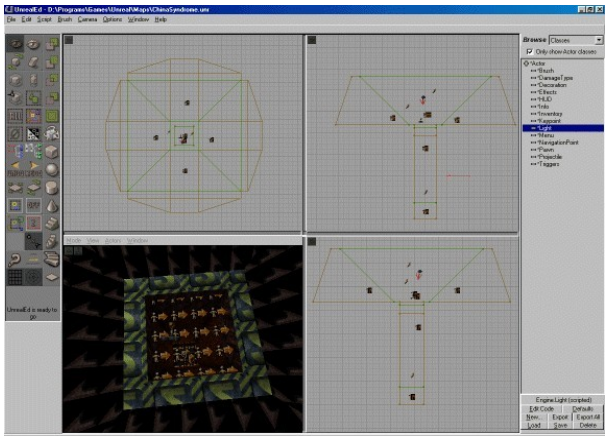
Jedná sa o editor, ktorým sa tvoria mapy do strategickej hry C&C Generals. Herná mapa je jednoliata plocha s meniteľným súradnicami vrcholov jednotlivých polygónov, čo umožňuje užívateľovi dostatočnú slobodu pri vytváraní krajiny (viď obr 2.9). Zaujímavým spôsobom je tu riešené vytváranie riek a jazier, oboje sú definované ako polygónové objekty s plnou možnosťou editácie, takže je možné vytvárať rieky s veľkým spádom. Celá mapa je rozdelená do dvoch základných vrstiev, v prvej vrstve je povrchová vrstva krajiny, a v druhej vrstve sú ostatné objekty.

2.2.2.2 Mapy vytvárané generovaním

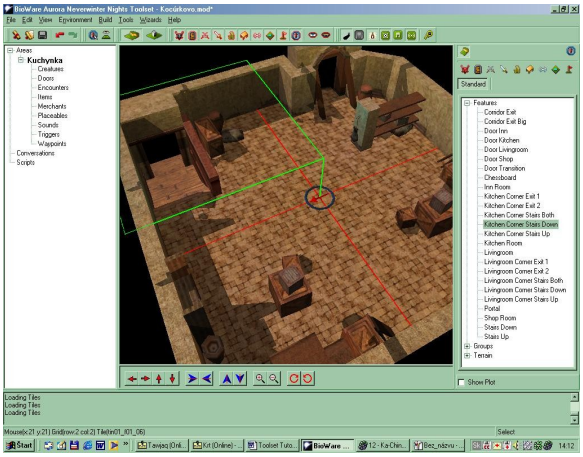
Takéto mapy sú z väčšej časti vytvárané generátorom, od človeka je vyžadovaný minimálny, až žiadny vstup. Generátor pracuje na základe predom definovaných pravidiel a umiestňuje na mapu objekty. Po dokončení jeho činnosti je mapa uložená vo výslednom výstupnom súbore. Pre účely tejto práce je opäť uvedených niekoľko príkladov.

Mapy použiteľné pri doskových hrách

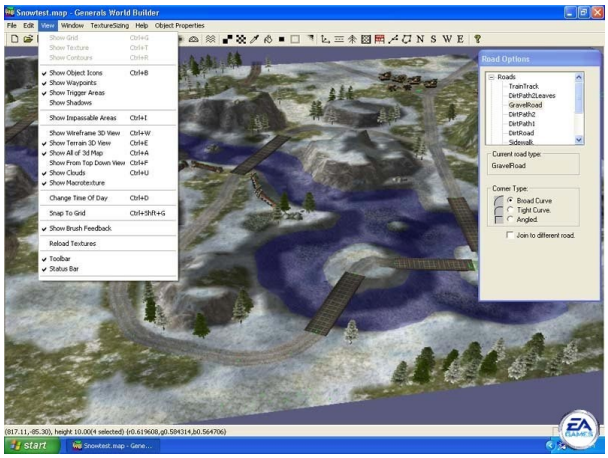
V tejto kapitole sú uvedené tri príklady programov, ktoré generujú mapy, použiteľné pri doskových hrách.



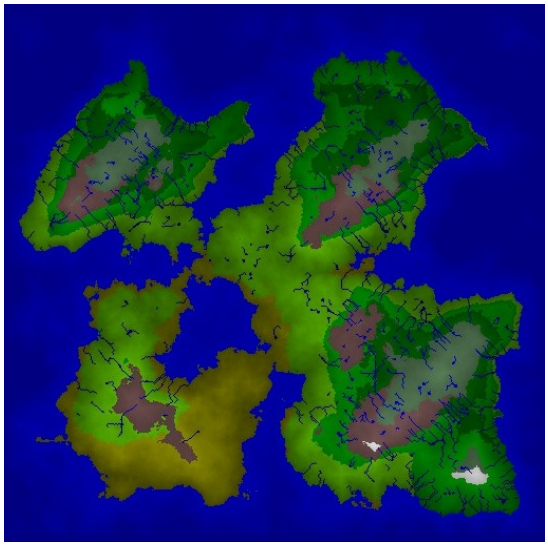
Obrázok 2.7: UnrealEd



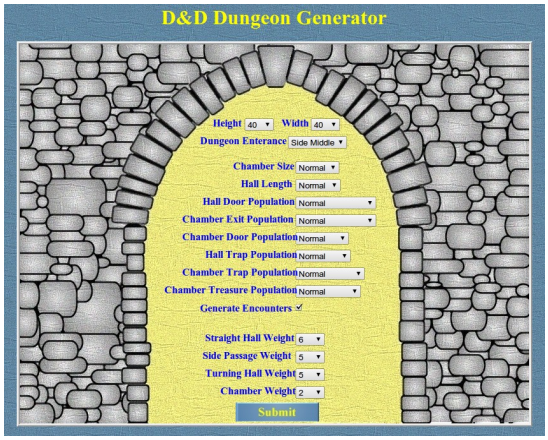
Obrázok 2.8: Bioware Aurora



Obrázok 2.9: C&C Generals: World Builder



Obrázok 2.10: World Generator, ukážka máp



Obrázok 2.11: D&D Dungeon Generator

Obrázok 2.12: Random Wilderness Map Creator, ukážka mapy

Hero Extant: World Generator

Tento generátor dokáže vytvárať veľmi realistické ostrovy či kontinenty (viď obr 2.10). Do mapy komponuje simuláciu dažďa a vetra, a obsahuje aj algoritmus pre generovanie teploty. Každé spustenie generátoru má za následok vygenerovanie novej, jedinečnej mapy. Mapa sa vykresľuje na plátno bez podložia. Vstupné parametre, ktoré sú na začiatku generovania požadované po užívateľovi, nie sú zatiaľ implementované do generátoru, takže celok funguje relatívne autonómne bez vstupu od užívateľa.

Vhodné na:

- generovanie širokého okolia postáv, kontinentov.

Nevhodné na:

- generovanie budov, jaskýň, interiérov a miest.

D&D Dungeon Generator

Online generátor jaskýň - chodieb a miestností. Užívateľ má na začiatku generovania možnosť zadať vstupné parametre, ako sú výška a šírka mapy, veľkosť miestností, dĺžka chodieb, počet dverí, pascí a pokladov a podobne. Mapa sa vykreslí ako obrázok na štvorcové podložie, a je možné ju jednoducho uložiť a vytlačiť. Rozhranie tohoto generátoru je na obr. 2.11.

Vhodné na:

- generovanie chodieb a miestností v jaskyniach.

Nevhodné na:

- generovanie širokého okolia postáv,
- generovanie interiéru budov, rozloženia budov v meste.

Random Wilderness Map Creator

Online generátor prostredia - divočiny. Tento program je príkladom využitia mapy generovanej bez podložia v menších lokáciách. Po zadaní vstupných parametrov sa vygeneruje mapa zobrazujúca časť prostredia, ktorá sa dá využiť na ilustráciu najbližšieho okolia hrdinov (viď obr. 2.12).

Vhodné na:

- generovanie časti oblasti, kde sa postavy nachádzajú.

Nevhodné na:

- generovanie rozľahlejších oblastí,
- generovanie jaskýň, domov, interiérov.

Mapy v počítačových hrách

V tejto kapitole sú uvedené tri príklady programov, ktoré generujú mapy do počítačových hier.

Diablo II

Táto hra obsahuje generátor máp, ktorý sa spúšťa vždy pri jej načítaní, čo znamená, že herná mapa je vždy iná. Generátor kombinuje náhodne vytvárané lokácie s tými, ktoré sú dané príbehom a teda preddefinované. Program vytvára dva základné druhy máp - interiéry (katakomby, väzenia) a exteriéry (pastviny, púšte). Tieto dva druhy máp od seba odlišuje celkové usporiadanie koridorov, ktoré sú v interiéri oveľa užšie, a ich hranice sú tu interpretované ako steny. V exteriéri sú koridory širšie a hranice sú reprezentované ako objekty typu plot, sústava skál, púštnych dún a podobne. Algoritmus pracuje na princípe náhodného umiestňovania políčok, ktoré sú potom prepojené. Generátor tiež musí zaisťovať, aby bola herná mapa priechodná, tzn. musí existovať cesta medzi vstupným a výstupným

bodom mapy. Po vygenerovaní základnej vrstvy sa na herný plán umiestňujú ďalšie objekty, ktorých poloha je v rámci políčka náhodná, v rámci mapy sa však vyskytujú na predom definovanom type políčok. Ukážka mapy mesta, ktorú vytvoril tento generátor, je na obr. 2.13.

Civilisation IV

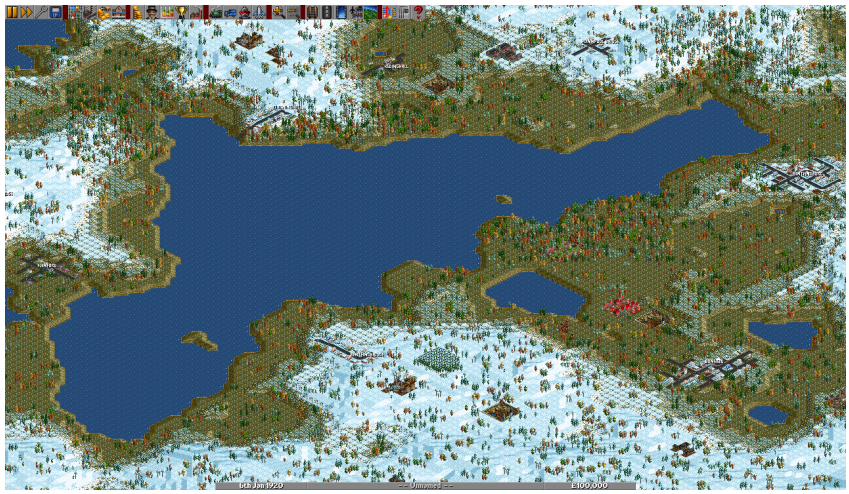
Generátor máp pre túto hru je založený na princípe náhodného výberu políčok s rôznymi vlastnosťami povrchu (kopec, pobrežie, rieka). Potom sa na takto vzniknutú mapu aplikuje vrstva kvality povrchu, ktorá určuje, či dané políčko bude reprezentovať púšť, tundru, step či iný typ krajiny. Nakoniec sa pridajú objekty ako sú stromy, lesy a obchodovateľný tovar. Generátor si pri spustení vyžiada od užívateľa niekoľko parametrov. Jedná sa hlavne o parametre typu veľkosť mapy, podiel súše a vody, priemerná teplota sveta, priemerná vlhkosť sveta, množstvo ostrovov a podobne. Táto kombinácia políčkového systému a generovania textúry podla parametrov zadaných užívateľom vykazuje veľmi dobré výsledky hlavne čo sa týka doby tvorby mapy a možnosti generovať vierohodne vyzerajúce mapy (v rámci limitu daného veľkosťou políčka). Príklad mapy je na obr. 2.14.

Open Transport Tycoon Deluxe

Jedná sa o open verziu jednej z najznámejších budovateľských stratégií. Táto verzia obsahuje integrovaný generátor máp Terra Genesis Perlin, ktorý využíva k vytvoreniu výškovej mapy prostredia Perlinovu šumovú funkciu. Keď sa povrchová mapa vygeneruje, je vo všetkých osách rozdelená na políčka. Podla parametrov sa pridávajú ďalšie objekty, napr. stromy, mestečká, továrne. Umiestnenie takýchto objektov je náhodné. Z hľadiska generovania výškovej mapy, ktorá je v hre reprezentovaná pseudo 3D mriežkou, je tento spôsob vhodný najmä vďaka jednoduchej konfigurovateľnosti algoritmu parametrami a krátkej dobe výpočtu. Ukážka takto generovanej mapy je na obr. 2.15.

Obrázok 2.13: Diablo 2, ukážka mapy

Obrázok 2.14: Civilisation IV, ukážka mapy



Obrázok 2.15: Open Transport Tycoon Deluxe, ukážka mapy

3 Návrh generátora máp

Návrh programu sa odvíja od špecifikácie požiadavkov na program, ktoré sú v tejto kapitole tiež uvedené. Dôraz je kladený na návrh jazyka, návrh interakcie s grafickým užívateľským rozhraním a návrh samotného programu.

3.1 Špecifikácia požiadavkov

Požiadavky na generátor máp sa dajú zhrnúť do viacerých základných bodov:

- vytvoriť generátor 2D máp pre fantasy hry na hrdinov,
- jednoduchosť,
- platformová nezávislosť,
- prispôsobiteľnosť,
- zotaviteľnosť.
- použiteľný pri hrách typu Dračí Doupě,
- algoritmy zahŕňajúce náhodnosť,
- umožňuje vytvoriť mapu úplne náhodne,
- umožňuje úpravy mapy užívateľom:
  - umožňuje úpravy, ktorá zahŕňajúca náhodnosť,
  - umožňuje úpravy s presne zadanými parametrami,
- zobrazuje názvy objektov,
- vytvára objekty rôznych tvarov:
  - generuje pravidelné tvary,
  - generuje nepravidelné, náhodné tvary,
- vstup od užívateľa:
  - jazyk na zadávanie príkazov:
    - podporuje skloňovanie kľúčových slov,
    - odkazuje sa na nepomenované objekty,
    - podporuje kľúčové slová vo viacerých jazykoch (čeština, angličtina, …),
    - príkazy:
      - vytvoriť nový objekt,
      - zrušiť vytvorený objekt,
      - upraviť vlastnosť objektu:
        - pomenovať objekt,
        - zmeniť veľkosť objektu,
        - zmeniť polohu objektu,
      - vytvoriť skupinu rovnakých objektov:
        - vytvoriť každý objekt náhodne,
      - príkazy zahŕňajú náhodnosť:
        - zadať údaj o polohe z určitého intervalu jediným príkazom,
        - zadať údaj o veľkosti z určitého intervalu jediným príkazom,
  - vstup pomocou GUI:
    - vytvárať objekty na mape,
    - rušiť objekty z mapy,
    - premiestňovať objekty na inú lokáciu na mape,
    - príkazy nezahŕňajú náhodnosť:
      - po kliknutí generovať stred objektu presne v bode kliku,
- umožňuje dopĺňať vlastnosti objektov náhodne:
  - ak užívateľ nezadá hodnotu, dopĺňa veľkosť, polohu,

- zobrazovanie objektov:
  - generuje objekty typu lesy, lúky,
  - generuje objekty typu línia – cesty, rieky.

3.2 Návrh jazyka pre zadávanie požiadavkov

Jazyk pre zadávanie požiadavkov na mapu bol navrhnutý s ohľadom na maximálnu jednoduchosť používania aj pre ľudí, ktorí sa s programovacími jazykmi bežne nestretávajú. V jazyku je možné zadať jednoduchým spôsobom požiadavok na generovanie nového objektu, na zmenu vlastností existujúceho objektu a na zmazanie objektu. V nasledujúcich podkapitolách sa text zameriava na návrh jednotlivých požiadavkov na jazyk tak, ako boli definované v kapitole 3.1.

3.2.1 Podpora skloňovania

Aby sa použitý jazyk čo najviac približoval hovorenej reči, je navrhnutá možnosť skloňovania. Spracovávaním hovorenej reči sa zaoberá obor Natural language processing (spracovávanie prirodzeného jazyka, NLP), pre účely tejto práce sú však princípy tohoto oboru príliš komplikované a je navrhnutá jednoduchšia alternatíva skloňovania. Použiteľnosť takto vybaveného jazyka je veľká aj pre užívateľov, ktorí sa v bežnom živote so žiadnym podobným princípom zadávania príkazov nestretli.
Skloňovanie je riešené pomocou takej štruktúry, ktorá má pre každý záznam aspoň dva parametre: presné znenie skloňovaného slova a význam tohto slova v programe. Prvý parameter predstavuje jeden z tvarov kľúčového slova, ktoré môže zadať užívateľ, a druhý parameter je jednotná hodnota pre všetky skloňované varianty jedného kľúčového slova a predstavuje jeho reprezentáciu v programe. Pomocou takejto štruktúry sa vykoná preklad skloňovaného slova na jednoznačne definované kľúčové slovo, s ktorým bude program ďalej pracovať.
Je dôležité zmieniť, že kľúčové slová sa delia do skupín podla významu, čo následne uľahčuje syntaktickú analýzu. Existujú 4 skupiny kľúčových slov a každá skupina je označená príslušným číslom. Toto číslo predstavuje tretí parameter prvku v štruktúre kľúčových slov, ktorá teda vyzerá takto:

```
keywords = {keyword, keyword, keyword, ...}
```

pričom každé kľúčové slovo `keyword` pozostáva z nasledujúcich prvkov:

```
keyword = {
      index: declensed_keyword,
      value: resolved_keyword,
      type: number
      }
```

kde `declensed_keyword` predstavuje skloňovaný tvar kľúčového slova, `resolved_keyword` je jednotná varianta pre všetky skloňované hodnoty daného slova a `number` označuje príslušnosť daného kľúčového slova do skupiny. Skupiny kľúčových slov sú rozdelené takto:

0. skupina - kľúčové slová označujúce slovo na začiatku príkazu (chci, nechci, budiž),
1. skupina - kľúčové slová označujúce objekty (les, more, potok a podobne),
2. skupina - kľúčové slová označujúce veľkosti objektov (malé, stredné, veľké),
3. skupina - kľúčové slová označujúce lokácie (na severe, na juhu a podobne).

Napríklad pre kľúčové slovo "les" vyzerá celý záznam v štruktúre `keywords` takto:

```
keywords = {
      {index: 'les', value: 'forest', type: 1},
      {index: 'lesy', value: 'forest', type: 1},
      {index: 'lesu', value: 'forest', type: 1},
      {index: 'lesa', value: 'forest', type: 1},
      {index: 'lese', value: 'forest', type: 1}
      }
```

3.2.2 Odkazovanie na nepomenované objekty

V rámci jazyka je implementácia odkazovania na nepomenované objekty nutná. Ak chce užívateľ vytvoriť objekt a potom mu zadať názov, zmeniť niektorý jeho atribút, či ho úplne z mapy zmazať musí mať možnosť sa na daný objekt odkázať.

Každý objekt, ktorý sa bude na mape vyskytovať, teda musí obsahovať jednoznačný vnútorný identifikátor. Za takúto hodnotu bolo zvolené poradové číslo vytvorenia objektu daného typu na mape. V jazyku sa potom dá odkázať na akýkoľvek objekt za použitia jeho poradového čísla za názvom typu objektu. Poradové čísla sa počítajú od nuly. Príkaz, ktorý sa odkáže na nepomenovaný objekt, vyzerá takto:

```
<command> ::= <first_word> <object> <number> <rest_of_command>
```

kde:

```
<first_word> ::= 'chci' | 'nechci' | 'budiz'
<object> ::= 'les' | 'jezero' | 'poust' | 'more' | 'bazina' |
'mesto' | 'cesta' | 'reka' | 'potok'
```

a `number` je nezáporné číslo v desiatkovej sústave, ktoré udáva poradové číslo vytvorenia daného objektu, a `rest_of_command` predstavuje postupnosť kľúčových slov dokončujúcich príkaz. Takýchto postupností je viac (podrobne v kap. 5), ale pre účely odkazovania sa na nepomenovaný objekt ich nie je potrebné rozvádzať. Konkrétny príklad odkázania sa na tretí vytvorený les a jeho pomenovanie sa zadá nasledujúcou postupnosťou:

```
budiz les 2 "Černý les"
```

3.2.3 Podpora kľúčových slov vo viacerých jazykoch

Ak užívateľ neovláda hovorený jazyk, v ktorom sú kľúčové slová zadávané, alebo si chce len sprí-jemniť a zjednodušiť zadávanie príkazov, je tu preňho možnosť preložiť kľúčové slová do akéhokoľvek jazyka, ktorý preferuje. Prispieva to k použiteľnosti jazyka pre zadávanie požiadavkov na mapu a aj k použiteľnosti celej aplikácie.

Samotný preklad spočíva v preložení hodnoty parametra `index` vo vyššie uvedenej štruktúre `keywords`, ktorá obsahuje všetky kľúčové slová. Hodnota `value` zostáva aj po preklade nezmenená, čo umožňuje programu pracovať rovnakým spôsobom s kľúčovými slovami v akomkoľvek hovorenom jazyku. Pri preklade je nutné dbať na to, aby v sa parametri index nachádzalo len jedno slovo označujúce kľúčové slovo. Ako príklad nech slúži preklad niekoľkých kľúčových slov v českom jazyku do anglického jazyka a ich použitie:

Štruktúra `keywords` obsahuje:

```
keyword = {index: 'les', value: 'forest', type: 1}
keyword = {index: 'chci', value: 'iwant', type: 0}
```

Príkaz, ktorý zadá užívateľ, vyzerá takto:

```
chci les
```

Po preklade štruktúra `keywords` obsahuje:

```
keyword = {index: 'forest', value: 'forest', type: 1}
keyword = {index: 'iwant', value: 'iwant', type: 0}
```

Príkaz, ktorý zadá užívateľ, vyzerá takto:

```
iwant forest
```

3.2.4 Príkazy pre manipuláciu s objektami na mape

Príkazy jazyka na zadávanie požiadavkov sú navrhnuté s ohľadom na duálnosť ovládania užívateľského rozhrania. Jednotlivé príkazy obsahujú rôznu mieru náhodnosti. Ak chce teda užívateľ vytvoriť objekt niekde na severe od iného objektu, použije na to príkaz jazyka. Ak naopak má celkom presnú predstavu, kde by sa daný objekt mal nachádzať, použije k označeniu takéhoto miesta grafické užívateľské rozhranie.

Príkazy jazyka sa delia do tematických skupín podla ich sémantického významu: príkazy na vytváranie objektov na mape, príkazy na mazanie objektov z mapy a príkazy na zmenu atribútov objektu.

3.2.4.1 Vytváranie nových objektov na mape

Vytvoriť na mape nový objekt je možné pomocou nasledujúcich štyroch príkazov:

```
1. <command> ::= 'chci' <object>
2. <command> ::= 'chci' <size> <object>
3. <command> ::= 'chci' <number> <object>
4. <command> ::= 'chci' <number> <size> <object>
<object> ::= 'les' | 'jezero' | 'poust' | 'more' | 'bazina' |
'mesto' | 'cesta' | 'reka' | 'potok'
<size> ::= 'velky' | 'stredny' | 'maly'
```

kde `number` je nezáporné číslo v desiatkovej sústave, ktoré značí počet objektov, ktoré sa majú vytvoriť. Jednotlivé príkazy vykonávajú tieto činnosti:

1 - vytvorí zadaný objekt a umiestni ho na náhodné miesto na mape,
2 - vytvorí objekt zadanej veľkosti a umiestni ho na náhodné miesto na mape,
3 - vytvorí zadaný počet objektov, a každý umiestni na náhodné miesto na mape,
4 - vytvorí zadaný počet objektov uvedenej veľkosti, a každý umiestni na náhodné miesto na mape.

Ak chce užívateľ vytvoriť väčší súvislý objekt, má možnosť pospájať jednotlivé menšie objekty do väčšieho celku. Toto je možné pomocou manipulácie s GUI, konkrétne napr. použitím myšky a presúvaním objektov na zvolené pozície.

3.2.4.2 Rušenie objektov z mapy

Objekty sa z mapy rušia pomocou dvoch príkazov:

```
1. <command> ::= 'nechci' <object>
2. <command> ::= 'nechci' <object> <name>
<object> ::= 'les' | 'jezero' | 'poust' | 'more' | 'bazina' |
'mesto' | 'cesta' | 'reka' | 'potok'
```

kde `name` je textový reťazec pozostávajúci z alfanumerických znakov, ktorý predstavuje názov objektu, ktorý sa má zrušiť. Príkazy vykonajú toto:

1 - zmaže z mapy posledný vytvorený objekt daného typu,
2 - zmaže z mapy objekt s daným názvom.

3.2.4.3 Zmena parametrov objektu

Príkaz zmeny polohy objektu môže nadobúdať týchto štyroch tvarov:

```
1. <command> ::= 'chci' <name1> <location> <name2>
2. <command> ::= 'chci' <object> <name1> <location> <object>
   <name2>
3. <command> ::= 'chci' <name1> <location> <object> <name2>
4. <command> ::= 'chci' <object> <name1> <location> <name2>
<location> ::= 'na sever' | 'juh' | 'východ' | 'západ' | 'sv' |
'sz' | 'jv' | 'jz' | 'na kraji'
<object> ::= 'les' | 'jezero' | 'poust' | 'more' | 'bazina' |
'mesto' | 'cesta' | 'reka' | 'potok'
```

kde `name1` (resp. `name2`) je textový reťazec pozostávajúci z alfanumerických znakov, ktorý udáva názov objektu, ktorý sa bude presúvať (resp. názov objektu, vzhľadom na ktorý sa bude prvý objekt presúvať). Príkaz teda umiestni objekt s názvom `name1` na novú polohu vzhľadom na objekt s názvom `name2`.
Po zadaní príkazu na zmenu polohy sa vypočíta nový štartovný bod objektu, a objekt sa vygeneruje na novej pozícii odznova. Objekt si zachováva zadané parametre, napriek tomu je možné, že sa bude vizuálne odlišovať od svojho pôvodného tvaru.
Ďalší dôležitý parameter objektu, ktorý je možné zmeniť, je jeho názov. Zmena názvu objektu sa vykoná jedným z nasledujúcich príkazov:

```
5. <command> ::= 'budiz' <object> <name>
6. <command> ::= 'budiz' <object> <number> <name>
<object> ::= 'les' | 'jezero' | 'poust' | 'more' | 'bazina' |
'mesto' | 'cesta' | 'reka' | 'potok'
```

kde `name` je textový reťazec pozostávajúci z alfanumerických znakov, ktorý bude použitý ako nový názov daného objektu, a `number` je je nezáporné číslo v desiatkovej sústave, ktoré predstavuje poradové číslo daného objektu na mape. Príkazy teda vykonajú:

5 - priradí danému objektu názov v podobe zadaného identifikátora,
6 - priradí objektu s daným poradovým číslom názov v podobe zadaného identifikátoru.

3.2.4.4 Príkazy zahŕňajúce náhodnosť

Ako bolo už uvedené, príkazy jazyka sa zameriavajú na určitú náhodnosť v akciách, ktoré reprezentujú. Táto náhodnosť sa v programe pretransformuje na konkrétne číslo, s ktorým sa bude pracovať. Každá hodnota z množiny prípustných veľkostí udáva rozsah, z ktorého sa náhodne vyberie číslo, ktoré sa použije.

Iný princíp sa uplatňuje pri náhodnom umiestňovaní objektu na mapu. Tam sa vygeneruje náhodné číslo udávajúce index polygónu, ktorý bude na mape predstavovať štartovný bod objektu.

3.3 Návrh interakcie s GUI

Použitie GUI (konkrétne zadávanie príkazov pomocou myši) predstavuje z hľadiska interakcie s programom opačný prístup ako príkazy jazyka. Umožňuje jednoznačne a presne určiť polohu objektu na mape bez použitia náhodnosti. Tento postup je vhodný, ak má užívateľ presnú predstavu, kam chce objekt umiestniť, a zvyšuje celkovú použiteľnosť aplikácie.

Pomocou GUI je možné vykonať rovnaké elementárne operácie, ako pomocou príkazov jazyka. Je možné vytvoriť objekt na mape, zmazať objekt z mapy a premiestniť objekt na iné miesto na mape. Po kliknutí myšky na plochu mapy je vybraný polygón, ktorý bude tvoriť štartovný polygón nového objektu. Ak sa myškou klikne na už vytvorený objekt, je tento umiestnený do prechodnej pamäte, a po kliknutí na prázdne miesto na mape sa zobrazí na tomto mieste. Prechodná pamäť má kapacitu presne jedného objektu, takže sa týmto spôsobom dajú objekty z mapy aj odstraňovať.

3.4 Návrh generátora máp

Aplikácia musí spĺňať tieto požiadavky:

- jednoduchosť,
- platformová nezávislosť,
- prispôsobiteľnosť,
- zotaviteľnosť.

Jednoduchosťou sa v tomto prípade rozumie proces získavania a inštalovania aplikácie. Tento termín súhrnne označuje všetky kroky, ktoré musí užívateľ vykonať, než bude aplikácia pripravená na používanie, teda napr. sťahovanie a inštalácia aplikácie.

Platformová nezávislosť umožňuje rozšíriť a používať aplikáciu na rôznych typoch operačných systémov, bez toho, aby bol užívateľ nútený zmeniť používanie či inštaláciu programu.

Prispôsobiteľnosťou sa rozumie schopnosť aplikácie správne fungovať na rôznych typoch zariadení , napr. pc, inteligentný telefón, tablet. Rôzne zariadenia majú odlišné veľkosti obrazoviek a hodnoty rozlíšenia, ktorým sa musí aplikácia prispôsobiť.

Zotaviteľnosť znamená, že užívateľ nepríde o dáta, ktoré si uložil. Tento pojem zahŕňa aj vytvorenie mapy v jednom zariadení a jej následná úprava v zariadení inom.

3.4.1 Voľba programovacieho jazyka

Po zohľadnení uvedených podmienok sa voľba zúžila na programovací jazyk Java (prenositeľná off-line aplikácia) alebo súbor jazykov pod pojmom webové technológie (php, html, javascript, ajax) (online aplikácia). Rozhodnutie nakoniec padlo na webové technológie, a porovnanie tohoto riešenia oproti Jave bude ilustrované postupne na všetkých bodoch uvedených v kapitole 3.4:

- jednoduchosť - pri použití webových technológií je požadovaný webový prehliadač a pripojenie k internetu, a užívateľ musí pred samotným použitím aplikácie zadať do webového prehliadača adresu, prípadne kliknúť na odkaz na webových stránkach programu. Ak by bola aplikácia v napísaná jave, požadovala by sa po zariadení podpora javy, a užívateľ by musel aplikáciu pred použitím inštalovať.
- platformová nezávislosť - pri použití webových technológií sa aplikácia zobrazuje vo webovom prehliadači, ktorý môže byť špecifický pre každý systém, ale princíp zobrazovania sa nemení. Pri použití javy je po zariadení opäť vyžadovaná jej podpora a inštalácia.
- prispôsobiteľnosť - táto vlastnosť sa dá účinne dosiahnuť pri programoch písaných v jave, aj pri programoch vytvorených pomocou webových technológií.
- zotaviteľnosť - pri použití webových technológií sa dáta ukladajú na externý server, ktorý funguje nezávisle na zariadení, na ktorom pracuje aplikácia. Tento princíp umožňuje perzistenciu dát v prípade poruchy v užívateľskom zariadení, a je tiež možné jednoducho zobraziť mapu v inom zariadení, než bola pôvodne vytvorená, a tam ju ďalej upravovať. V prípade použitia Javy by zotaviteľnosť pri poruche zariadenia nebola možná, a úprava mapy v inom zariadení by bola možná až po manuálnom prenose súboru s údajmi o mape.

3.4.2 Voľba podložia

Aplikácia bude zobrazovať objekty typu lesy, jazerá, moria, mestá, cesty medzi mestami - bude mapovať relatívne veľkú oblasť. Pre takýto typ máp by bolo najvýhodnejšie vykresľovať objekty na plochu bez podložia. Takáto voľba by však prispela k horšej algoritmickej riešiteľnosti niektorých problémov, napr. presun objektu, alebo prekreslenie mapy. Štvorcové podložie by zase nebolo vhodné z dôvodu malej realistickosti výslednej mapy. Rozhodol som sa teda vykresľovať mapu na náhodne generované podložie, ktoré sa pri každej novej mape vždy znova vygeneruje. Týmto sa dosiahne rôznorodosť tvarov na mape, na žiadnych dvoch vytvorených mapách by sa nemal vyskytovať ten istý tvar. Realistickosť sa dosiahne dostatočným zmenšením jednotlivých generovaných polygónov, a ich relatívne veľký počet vzhľadom na rozlíšenie mapy. Generovanie takéhoto podložia bolo už úspešne použité napríklad v [8].

Konkrétne sa jedná o generovanie tzv. Voroného diagramu. V [6] sa uvádza táto definícia:

Voronoi (Voroného) diagram $V(P)$ predstavuje rozklad množiny bodov P na n uzavretých či otvorených oblastí $V(p) = \{V(p1), V(p2), ..., V(pn)\}$ takých, že každý bod $q \in V(pi)$ je bližšie k bodu pi než k akémukoľvek bodu $pj \in P$.

Diagram sa vytvára tak, že sa na plochu najprv náhodne rozmiestni určitý počet bodov P, z ktorých sa vygenerujú Voroného bunky $V(p)$. Sústava takýchto útvarov potom predstavuje podložie mapy.

Voroného bunky vytvárajú náhodné útvary okolo jednotlivých bodov P. Tieto útvary sa pred použitím ešte upravia pomocou tzv. Lloydovho algoritmu [7] pre relaxáciu Voroného diagramu. Táto úprava slúži k vyhladeniu ostrých uhlov v jednotlivých polygónoch, a prispieva k výslednej realistickosti mapy (viz obr. 4.3)

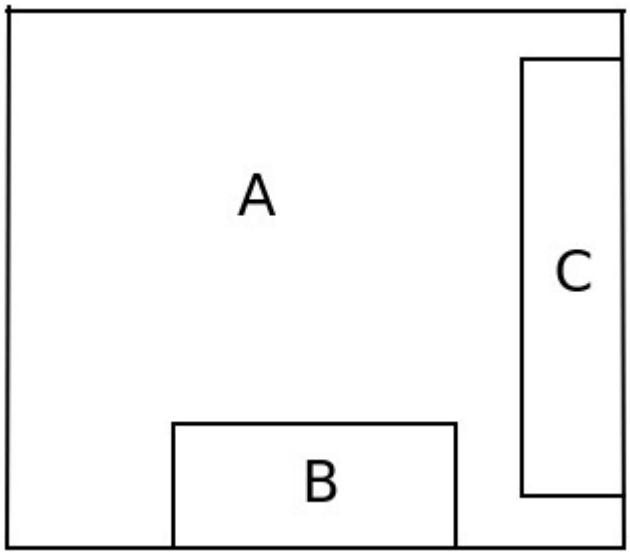
3.4.3 Návrh grafického užívateľského rozhrania

Užívateľské rozhranie sa skladá z plochy pre vykresľovanie a interakciu s mapou - plátna, a z dvoch vysúvacích panelov obsahujúcich ovládacie prvky. Prvý panel obsahuje prvky použiteľné myškou a druhý panel predstavuje konzolu pre zadávanie príkazov jazyka.

Panely sa pri deaktivácii zasunú mimo plátno tak, aby čo najmenej prekážali v manipulácii s mapou. Aktivujú sa presunutím kurzoru myšky nad plochu panelu.

Na obr. 3.1 je znázornené GUI generátora máp. Oblasť je rozdelená na tri časti:

A - plátno, na ktorom sa manipuluje s mapou,
B - panel, ktorý obsahuje konzolu pre zadávanie príkazov,
C - panel, ktorý obsahuje ovládacie prvky použiteľné pri manipuláciou s GUI.



Obrázok 3.1: Grafické užívateľské rozhranie

4 Implementácia generátora máp

V tejto kapitole je popísaná implementácia jednotlivých modulov, ktoré sú použité v programe, a jadra, ktoré moduly spája. Pre lepšie znázornenie použitých princípov sú uvedené najdôležitejšie algoritmy vo forme pseudokódu. Na obr. 4.1 je znázornená celková štruktúra programu z pohľadu modularity. Samotná interakcia medzi modulmi je vysvetlená v podkapitole 4.13. Kvôli rozsahu práce nie sú v tejto kapitole uvedené metódy, ktoré z každého modulu môže volať jadro. Tieto metódy je však možné nájsť na začiatku zdrojového kódu každého modulu v poznámke.

Časti programu, ktoré sú v programovom riešení tejto práce použité, ale neboli vytvorené autorom práce, sú: jquery[1], rhill voronoi core[2], canvas2image[3], base64[4].

4.1 Jadro

Jadro programu v sebe spája funkcionalitu všetkých modulov. Inštancuje poskytované triedy, definuje požadované konštanty a umožňuje komunikáciu s modulmi. Obsahuje funkcie potrebné pre inicializáciu prostriedkov pre manipuláciu s GUI (napr. myšky) a ďalších potrebných komponent a tiež pre prekreslenie celej mapy. Nasleduje stručný popis dôležitých funkcií, ktoré jadro obsahuje:

- `preInit, init` - funkcie zaisťujú počiatočnú inicializáciu premenných , ako napr. rozmery plátna, na ktoré sa bude mapa vykresľovať, a ďalej zaisťujú volanie prípravných funkcií, ktorých účel je po zobrazení stránky nachystať generátor na používanie.
- `clickedOn` - funkcia, ktorá zaobstaráva a vyhodnocuje prácu s GUI, teda konkrétne kliknutia myšky. Určuje, ktorý objekt sa z mapy má odobrať alebo na mapu pridať.
- `save, load` - funkcie, ktoré zabezpečujú odoslanie dát na server, a príjem dát zo serveru v prípade uloženia mapy, respektíve v prípade načítania uloženej mapy.
- `randomSites` - funkcia generuje a na plátno rozmiestni náhodné body (pseudo-stredy Voroného buniek), z ktorých sa generuje Voroného diagram.
- `lloydRelax` - funkcia vykonáva algoritmus lloydovej relaxácie (viď algoritmus 4.1).
- `positionThis` - funkcia obsahuje podmienky a výpočty bodov pre presun objektov na mape pomocou príkazov jazyka na zadávanie požiadavkov.
- `main1, main2. main3, main4` - funkcie, ktoré prebiehajú po načítaní stránky s generátorom a po inicializácii. Majú za úlohu pripraviť program na používanie. Generujú náhodné body a následne Voroného diagram ako podložie, spúšťajú lloydovu relaxáciu, inicializujú kľúčové slová a modul jazyka a farbia podložie mapy na základnú farbu. Funkcie reprezentujú ucelené časti príkazov, takže celá príprava sa deje v štyroch krokoch.

Algoritmus 4.1
Vstupy: `sites` - množina pseudo-stredov Voroného diagramu, `cells` - množina buniek voroného diagramu
Výstupy: `sites2` - množina upravených pseudo-stredov (tj. dvojice x, y) Voroného diagramu

```
foreach site in sites do
  find cell in cells with cell.site == site
  foreach edge in cell do
    x += edge.x
```

1 http://jquery.com/
2 https://github.com/gorhill/Javascript-Voronoi
3 http://www.nihilogic.dk/labs/canvas2image/canvas2image.js
4 http://www.nihilogic.dk/labs/canvas2image/base64.js

```
    y += edge.y
    divide++
  end for
  insert (x/divide, y/divide) into sites2
end for
```

4.2 Modul Voroného diagramu

Generovať Voroného diagram znamená vytvárať podložie mapy, na ktorom sa budú vykresľovať objekty mapy. Diagram sa vytvorí z náhodne vygenerovaných bodov rozmiestnených po plátne. Tieto body potom tvoria pseudo-stredy Voroného buniek. Na obr. 4.2 je vidieť, že takto vytvorené bunky majú pre účely vytvárania realistických objektov na mape príliš náhodný tvar (jednotlivé bunky sa od seba príliš odlišujú). Určitá náhodnosť v tvaroch buniek je vhodná (a požadovaná), avšak pri vytváraní objektov na takomto podloží by vznikali nerealistické uhly a zakončenia objektov, prípadne nedostatočne vyhladené hrany objektov. Preto sa takto vygenerovaný Voroného diagram ešte upravuje Lloydovov relaxáciou. Viditeľné zlepšenie sa dosiahne už po dvoch iteráciách tohoto algoritmu (viď obr. 4.3). Bunky sú rôzneho tvaru a veľkosti, a pritom je dodržaná určitá pravidelnosť. Takýto diagram sa používa ako podklad pre všetky objekty na mape. Štruktúra, ktorá uchováva vygenerovaný diagram, sa skladá z týchto častí:

Objekt `Voronoi diagram` uchováva všetky vygenerované hrany - `Edges`, a všetky vygenerované bunky - `Cells`. Každá hrana obsahuje referenciu na pseudo-stred bunky ležiacej vľavo – `lSite`, a vpravo - `rSite`. Každá hrana začína v bode `va` a končí v bode `vb` (obr. 4.4).

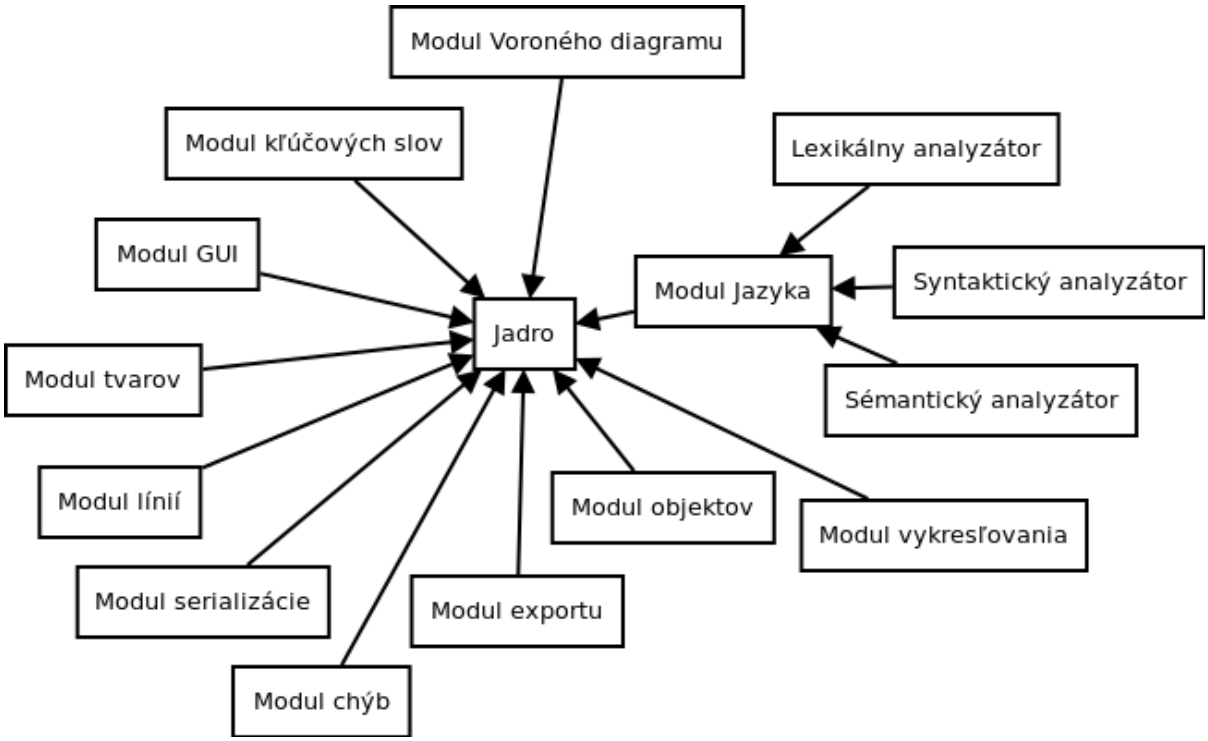
Každá bunka diagramu obsahuje množinu hrán, ktoré ju ohraničujú - `Halfedges`. Tieto hrany sú nadstavbou nad hranami celého diagramu - `Edges`, a v každej z nich je referencia na pseudo-stred `Site`, ktorý hrany ohraničujú, a s ktorým tvoria bunku (obr. 4.5).

Každý pseudo-stred `Site` sa skladá z `x` a `y` súradnice jeho polohy na mape, a z čísla `voronoiId`, ktoré ho jednoznačne identifikuje (obr 4.6).

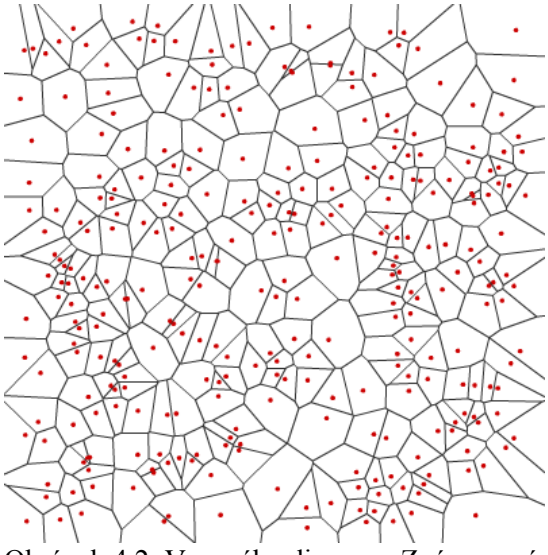
4.3 Modul vykresľovania

Tento modul obsahuje tri základné vykresľovacie funkcie, a to: funkciu pre vyfarbenie Voroného bunky - polygónu, funkciu pre nakreslenie línie (tzn. cesty, rieky alebo potoka), a funkciu pre vyfarbenie celého binómu - skupiny polygónov. Funkcie, ktoré vyfarbujú polygóny, pracujú zároveň aj s textúrami, ktoré sú pre jednotlivé objekty nastavené. Textúry je možné ľubovoľne meniť bez toho, aby to malo vplyv na vykresľovacie funkcie. Takto si užívateľ môže prispôsobiť mapu svojim potrebám. V prípade, že nie sú zadané žiadne textúry, mapa sa vykreslí len pomocou farieb dostupných v prehliadači.

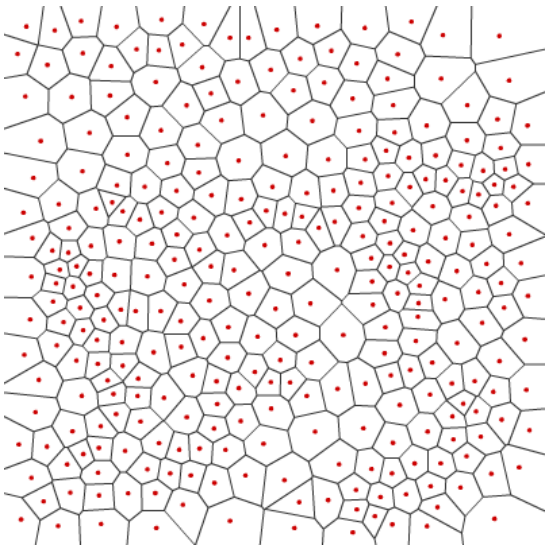
Vykreslovanie objektov na mape môže byť v niektorých prípadoch zdĺhavé, pretože niektoré akcie, vykonávané na mape, vyžadujú prekreslenie celej mapy, tzn. zmazanie všetkých objektov a ich opätovné nakreslenie.



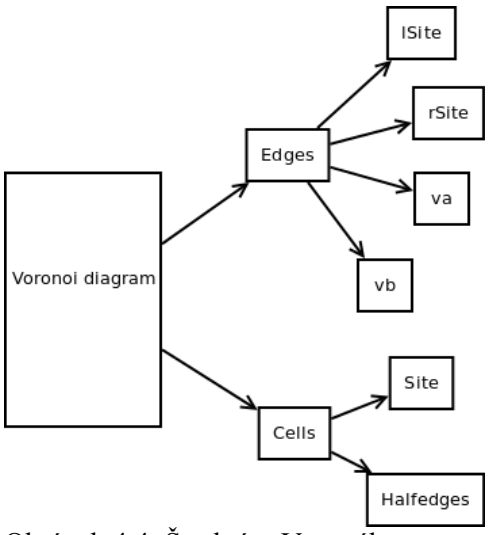
Obrázok 4.1: Hierarchia modulov



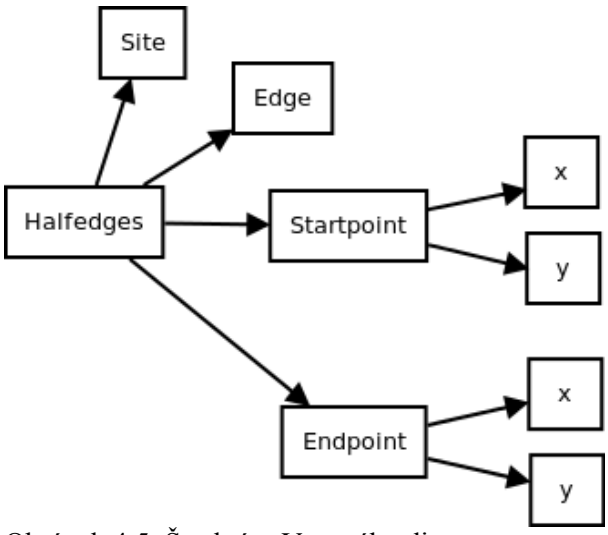
Obrázok 4.2: Voroného diagram. Zvýraznené sú body, z ktorých sa diagram generoval, teda pseudo-stredy.



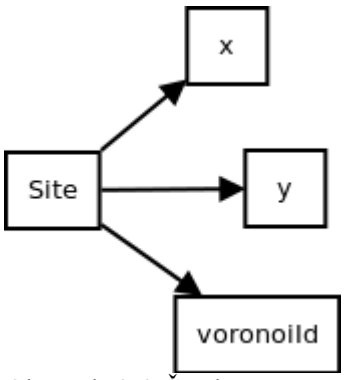
Obrázok 4.3: Voroného diagram po aplikovaní dvoch cyklov Lloydovej relaxácie.



Obrázok 4.4: Štruktúra Voronéhodiagramu. Objekt `Voronoi diagram` uchováva všetky vygenerované hrany - `Edges`, a všetky vygenerované bunky - `Cells`. Každá hrana obsahuje referenciu na pseudo-stred bunky ležiacej vľavo - `lSite`, a vpravo - `rSite`. Každá hrana začína v bode `va` a končí v bode `vb`.



Obrázok 4.5: Štruktúra Voronéhodiagramu (`halfedges`). Každá bunka diagramu obsahuje množinu hrán, ktoré ju ohraničujú - `Halfedges`. Tieto hrany sú nadstavbou nad hranami celého diagramu - `Edges`, a v každej z nich je referencia na pseudo-stred `Site`, ktorý hrany ohraničujú, a s ktorým tvoria bunku.



Obrázok 4.6: Štruktúra Voronéhodiagramu (`Site`). Každý pseudo-stred `Site` sa skladá z `x` a `y` súradnice jeho polohy na mape, a z čísla `voronoiId`, ktoré ho jednoznačne identifikuje.

4.4 Modul jazyka

Implementácia tohoto modulu sa je rozdelená na lexikálnu, syntaktickú a sémantickú analýzu. Takýto prístup je v prekladačoch bežný a rozdeľuje spracovávanie jazyka do samostatných, nadväzujúcich častí. Umožňuje tiež dôkladnú kontrolu chýb v jednotlivých analýzach.

Všetky analýzy v tomto module sú implementované priamo, teda nie pomocou konečného automatu, ale pomocou sústavy podmienok. Takúto implementáciu umožňuje relatívna jednoduchosť navrhnutého jazyka, ktoré majú analýzy spracovávať. Taktiež je táto možnosť výkonnostnejšie výhodnejšia, vzhľadom na fakt, že nie je potrebné zanárať do seba veľké množstvo malých funkcií.

Lexikálny analyzátor

Na vstupe požaduje všetky kľúčové slová v poli `keywords`, a výstup ukladá do pola `tokens`. Táto analýza vezme riadok príkazu, zadaného užívateľom do konzole programu (príkazy sa teda oddeľujú koncom riadku) a rozdeleného do pola, kde každý prvok pola je jedno slovo, ktoré užívateľ zadal, a podľa priradí každému slovu jeho programový význam. Teda ak natrafí na slovo, ktoré reprezentuje kľúčové slovo, priradí mu tento význam a uloží ho do pola `tokens`. Rovnaké je to s číslami a identifikátormi. Algoritmus analýzy je jednoduchý:

Algoritmus 4.2
Vstupy: pole slov, ktoré zadal užívateľ - `array`, a pole kľúčových slov - `keywords`
Výstup: pole spracovaných tokenov - `tokens`, kde `token` je dvojica (hodnota, identifikátor), pričom ak je rozoznané kľúčové slovo, tak sa k tejto dvojici pridá parameter typ kľúčového slova

```
foreach word in array do
  if word is in keywords then
    insert ('keyword', word.value, word.id) into tokens
  else
    if word is number then
      insert ('number', word) to tokens
    else
      if word is identificator then
        insert ('identificator', word) into tokens
      end if
    end if
  end if
end for
```

Syntaktický analyzátor

Syntaktická analýza kontroluje postupnosť tokenov, ktoré získala z výstupu lexikálnej analýzy, proti pravidlám jazyka pre zadávanie požiadavkov na mapu. Tento analyzátor negeneruje žiadny špeciálny vnútorný kód, ako je zvykom pri prekladačoch, z dôvodu jednoduchosti jazyka. Takýto kód by predstavoval zbytočné navýšenie zložitosti analýzy a programu. Algoritmus vyzerá takto:

Algoritmus 4.3
Vstupy: pole tokenov - `tokens`
Výstupy: pole syntaktických konštruktov - `constructs`

```
if first_token is in validFirstWords then
```

```
  if second_token is in validSecondWords then
    ...
    if all_tokens_valid then
      constructs.ruleNum = numberOfRule
      constructs.tokens = tokens
    end if
  else
    return error_at_second_token
  end if
else
  return error_at_first_token
end if
```

Sémantický analyzátor

Sémantický analyzátor má za úlohu zistiť a skontrolovať význam zadaného príkazu pre program. Táto analýza je spojená so samotným vykonávaním akcií na mape, pokiaľ je sémantika príkazu valídna. Tento spôsob implementácie umožňuje zjednodušiť manipuláciu s celým modulom jazyka a združuje všetky funkcie jazyka na jednom mieste – v module. Pretože tento analyzátor teda spája v sebe dve funkcie (analýza a vykonávanie príkazov), nazýva sa v programe `logic`. Obsahuje funkcie pre generovanie náhodného bodu začiatku objektu a funkcie pre vyčíslenie veľkostí a tvarov objektov. Volá metódy, ktoré umožňujú vytvárať na mape binómy, ktoré boli uvedené v návrhu v podkapitole príkazy návrhu jazyka v množine prípustných objektov. Dokáže tiež objekty premiestňovať a pomenovávať, teda meniť atribúty objektov.

Ako bolo uvedené v popise syntaktickej analýzy, nepracuje sa so žiadnym vnútorným kódom. Namiesto toho sa používa konštrukcia, ktorú má na výstupe syntaktický analyzátor, a ktorá obsahuje valídne tokeny a číslo, ktoré udáva, pod ktoré pravidlo zadaný príkaz spadá. Všetky údaje, ktoré potrebuje sémantický analyzátor, sa z takejto konštrukcie dajú získať a bez problémov použiť.

Pretože je aj táto analýza riešená sústavou podmienok, uvedený algoritmus je podstatne skrátený. Pre ilustráciu ale úplne stačí:

Algoritmus 4.4
Vstup: konštrukty vytvorené syntaktickým analyzátorom - `constructs`
Výstup: zmena na mape alebo chybový kód

```
if P_iwant_object then
  makeObject(type = constructs.tokens.second_token.value, startPoint,
  silliness, size)
  paintMap()
else if P_iwant_parameter_object then
  makeObject(type = constructs.tokens.third_token.value, startPoint,
  silliness, size = constructs.tokens.second_token.value)
  paintMap()
else if ...
else if P_iwant_number_object then
  foreach i in number do
    makeObject(type = constructs.tokens.second_token.value, startPoint,
    silliness, size)
  end for
  paintMap()
else if ...
else if P_dontwant_object then
  colorBinome(object, 'base')
  destroyBinome(object)
  deleteObject(object, index)
```

```
else if ...
else if P_letbe_object_name then
  name  = getName(constructs.tokens)
  object1 = objGet(constructs.tokens.second_token.value)
  object2 = object1
  object2.name = name
  objectAlter(object1, object2)
end if
```

Legenda k algoritmu 4.3:
`P_iwant_object`: reprezentuje prepis pravidla č.1 z podkap. 3.2.4.1
`P_iwant_parameter_object`: reprezentuje prepis pravidla č.2 z podkap. 3.2.4.1
`P_iwant_number_object`: reprezentuje prepis pravidla č.3 z podkap. 3.2.4.1
`P_dontwant_object`: reprezentuje prepis pravidla č.1 z podkap. 3.2.4.2
`P_letbe_object`: reprezentuje prepis pravidla č.5 z podkap. 3.2.4.3

4.5 Modul GUI

Obsahuje handler volaný pri manipulácii s GUI (napr. pri kliknutí myšky), funkciu pre výpočet lokálnych súradníc a funkciu, ktorá vráti index polygónu, na ktorý sa kliklo. Prepočítavanie súradníc je nutné, pretože v udalosti event, ktorá sa vytvára pri kliknutí myši, sa uvádzajú súradnice kliknutia globálne vzhľadom na celé okno webového prehliadača, kdežto pri spracovávaní kliknutia sú nutné súradnice lokálne v rámci kresliaceho plátna.

Pri spracovávaní kliknutia na konkrétny polygón je zaujímavá konštanta `dispersion`. Jej účel bude lepšie viditeľný z pseudokódu:

Algoritmus 4.5
Vstupy: lokálne súradnice kliknutia - `coords`, množina všetkých polygónov - `cells` a kontext canvasu - `ctx`
Výstup: index polygónu, na ktorý bolo kliknuté

```
foreach cell in cells do
  if absoluteValue(cell.x - coords.x) < dispersion then
    if absoluteValue(cell.y - coords.y) < dispersion then
      ctx.makePath(cell)
      if ctx.isPointInPath(coords.x, coords.y) then
        return cell.index
      end if
    end if
  end if
end for
```

Konštanta `dispersion` teda udáva rozsah, v ktorom algoritmus zisťuje, či bolo kliknuté na konkrétny polygón. Táto technika predstavuje výkonnostné vylepšenie oproti verzii, kedy sa kontroloval každý polygón z množiny `cells`.

4.6 Modul tvarov na mape

Obsahuje implementovaný algoritmus pre vytváranie akýchkoľvek prípustných tvarov na mape. Algoritmus pracuje smerom od daného štartovacieho bodu - polygónu - k jeho susediacim polygónom

a postupne ich označuje. Mieru takýchto cyklov udáva parameter expansion level - `expLvl`. Za level sa považuje množina polygónov susediacich so štartovacím polygónom, alebo obecne množina polygónov, ktorú generuje množina štartovacích polygónov.

Algoritmus sám rieši kolízie objektov tým, že každý objekt vytvára iba z polygónov, na ktorých sa žiadny objekt nenachádza. Generovanie štartovacieho bodu je riešené náhodným výberom polygónu z množiny všetkých polygónov a následnou kontrolou, či sa na tomto polygóne nenachádza žiadny objekt. V prípade, že je vybraný polygón obsadený, spustí sa výber znova. Počet takýchto krokov je obmedzený a je pevne daný.

Zaujímavosťou je parameter `silliness`, ktorý umožňuje náhodné vyberanie susedných polygónov, ktoré sa budú ďalej rozgenerovávať. Na mape potom vznikajú okrem pravidelnejších kruhových tvarov aj tvary náhodnejšieho charakteru. Takéto tvary môžu obsahovať (a spravidla obsahujú) artefakty - tvar nevypĺňa celý vymedzený priestor, ale obsahuje v sebe časti pôvodného podložia. V mape sa s touto vlastnosťou počíta a nie je to považované za nedostatok. Takéto útvary potom predstavujú napr. čistinku v lese, lesopark v meste a podobne. Algoritmus je nasledovný:

Algoritmus 4.6
Vstupy: údaje o type polygónov - `polygonType`, o novom type polygónov - `setType`, o miere generovania tvaru - `expLvl`, o štartovnom bode - `startPoinIndex`, ďalej vstupuje parameter `silliness` a množina všetkých polygónov na mape - `cells`
Výstup: množina polygónov určených pre daný objekt - `retCells`, prípadne chyba.

```
foreach cell in startPointNeighbours do
  insert (cell.index) into retCells
end for
nxtLvlCells = cells
foreach i in expLvl do
  repeat = getRandomInteger(resolveSilliness(silliness),
  nxtLvlCells.length)
  foreach j in repeat do
    random = getRandomInteger(resolveSilliness(silliness),
     --nxtLvlCells.length)
    insert (getUnusedNeighboursOf(nxtLvlCells[random])) into tempCells
    insert (tempCells) into retCells
  end for
  nxtLvlCells = tempCells
  tempCells = []
end for
```

Výsledný tvar je uložený v premennej `retCells`. Táto je potom vstupom do vykresľovacej funkcie, ktorá postupne vykreslí na každý polygón textúru podla typu objektu.

4.7 Modul línií na mape

Líniou sa rozumie objekt typu cesta, rieka alebo potok. Princíp kreslenia línií je nájsť na mape cestu medzi štartovným a konečným bodom. Bolo vyskúšaných niekoľko štandardných algoritmov, ktoré sa zaoberajú touto problematikou (BFS - Breadth First Search, UCS - Uniform Cost Search, Backtracking), nakoniec bola implementovaná verzia Dijkstrovho algoritmu hľadania najkratšej cesty.

Algoritmus v tomto module pracuje tak, že rozgenerováva všetkých následníkov štartovného polygónu, a potom všetkých ďalších, až kým niektorý z rozgenerovaných polygónov nie je konečný polygón. Pri rozgenerovávaní priradí každému polygónu hodnotu - číslo, reprezentujúce jeho vzdialenosť od štartovného polygónu. Po skončení tohoto cyklu sa spustí cyklus ďalší, ktorý rozgeneruje nasledovníkov konečného polygónu, a spomedzi nich vyberie jeden s najmenším ohodnotením.

Takýto polygón uloží, a následne rozgeneruje. Tento cyklus sa skončí, keď sa pri rozgenerovávaní narazí na počiatočný polygón. Množina uložených polygónov potom predstavuje najkratšiu cestu zo štartu do cieľa.

4.8 Modul objektov na mape

Predstavuje objekt, kam sa ukladajú údaje o všetkých objektoch (binómoch, mestách a líniách). Tieto informácie sú uložené v štruktúre reprezentovanej polom objektov. Takýto objekt vyzerá takto:

```
object = {
  'name': name,
  'type': type,
  'startPoint': startPoint,
  'expLvl': expLvl,
  'silliness': silliness,
  'points': points
}
```

kde `name` predstavuje názov objektu na mape, `type` je typ objektu na mape (viď podkap. 3.2.1), `startPoint` je počiatočný bod generovania objektu, `expLvl` je expansion level (údaj maximálnom rozšírení objektu pri jeho generovaní) objektu, `silliness` je parameter, ktorý určuje mieru náhody pri generovaní tvaru objektu a `points` je pole obsahujúce informácie o polygónoch, z ktorých sa objekt skladá.

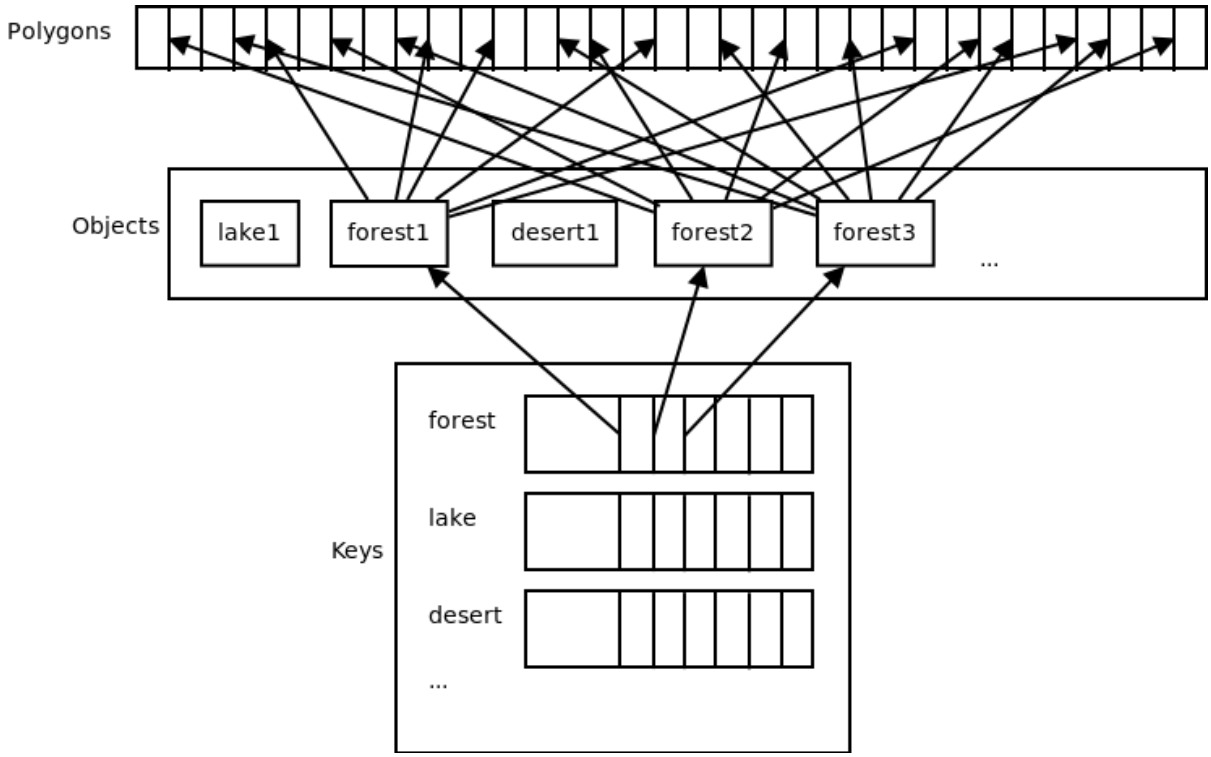
Nad týmto polom sa nachádza prístupová vrstva. Je to asociatívne pole, kde kľúč tvorí typ objektu a hodnotu pole indexov do pola objektov. Pole indexov reprezentuje pozície objektu daného typu v poli objektov. Táto vrstva sa používa na jednoduchšiu a rýchlejšiu manipuláciu s objektami na mape. Prepojenie prístupovej vrstvy, objektov a polygónov je znázornené na obr. 4.7.

4.9 Modul kľúčových slov

Obsahuje všetky prípustné kľúčové slová použitého jazyka. Ako samostatný modul je táto konštrukcia reprezentovaná preto, aby si mohol užívateľ pridať kľúčové slová vo svojom obľúbenom jazyku. Množina kľúčových slov je reprezentovaná polom objektov, ktoré vyzerajú nasledovne:

```
keyword = {
      index: 'les',
      value: 'forest',
      type: 1
      }
```

`Index` je textový reťazec, ktorý v príkaze zadávanom z klávesnice do konzoly predstavuje kľúčové slovo (v tomto prípade `les`). `Value` je hodnota, ktorá toto kľúčové slovo reprezentuje v programe (tu `forest`), a type je číselné označenie tematického zamerania kľúčového slova.
Ak chce užívateľ pridať kľúčové slová vo vlastnom jazyku, stačí nahradiť textový reťazec reprezentovaný indexom, ostatné hodnoty zostávajú nezmenené. Skloňovanie sa rieši veľmi jednoducho:



Obrázok 4.7: Prepojenie štruktúr uchovávajúcich objekty. Asociatívne pole `Keys` predstavuje prístupovú vrstvu, ktorá zjednodušuje prístup k objektom `Objects`. V poli `Objects` sú uložené samotné objekty a ich parametre, včetne údajov o tom, z akých polygónov z `Polygons` sa objekty na mape skladajú. Prepojenie je znázornené na príklade objektu `les` - `forest`, ostatné objekty sú samozrejme prepojené rovnako.

```
keywords[n]({index: 'lesy', value: 'forest', type: 1});
keywords[n+1]({index: 'lesu', value: 'forest', type: 1});
keywords[n+2]({index: 'lesa', value: 'forest', type: 1});
keywords[n+3]({index: 'lese', value: 'forest', type: 1});
```

Každý index sa berie ako nové kľúčové slovo, ktorému však `value` stanovuje ten istý význam – `les`.

4.10 Modul chýb

Chybové hlásenia sa nachádzajú v poli. V tomto poli predstavuje index návratovú hodnotu programu, a príslušné chybové hlásenie sa nachádza na tomto indexe. Užívateľ si môže jednoducho napísať chybové hlásenia vo svojom jazyku tak, že prepíše chybové hlásenia v tomto poli.

4.11 Modul exportu

Export mapy do výsledného obrázku formátu png je vyriešený pomocou zakódovania celého plátna s mapou do uvedeného formátu vrámci vstavanej funkcie `toDataURL`, ktorá vráti zakódovaný reťazec pomocou URI. Tieto dáta sa ponúknu užívateľovi na stiahnutie vo formáte `image/png`.

Uvedená funkcia však generuje chybu, pokiaľ sú na plátne použité obrázky z domény rôznej od domény, na ktorej práve beží generátor. Všetky textúry sú preto umiestnené na rovnakej doméne.

4.12 Modul serializácie

Serializácia a deserializácia dát sa používa pri prenose údajov na server, a pri ich sťahovaní zo servera. Dáta sa umiestnia do jedného dlhého textového reťazca, ktorý je uložený na serveri v súbore. Medzi jednotlivé údaje sa do reťazca vkladajú oddeľovacie znaky, aby deserializér vedel, kde daný údaj začína a kde končí. Ako oddelovače boli pôvodne použité xml tagy, ale takéto riešenie sa ukázalo ako zbytočne zdĺhavé. Po zavedení čiarok a nových riadkov ako oddelovačov a ponechaní len najnutnejších tagov sa podarilo zmenšiť veľkosť výsledného súboru na tretinu. Príkladom budiž serializácia stredov jednotlivých polygónov. Každý stred sa serializuje takto:

Algoritmus 4.7
Vstupy: množina všetkých stredov polygónov - `sites`
Výstupy: textový reťazec obsahujúci serializované dáta - `string`

```
for site in sites
  string += cell.voronoiId + ','
  string += cell.x + ','
  string += cell.y + ','
  string += '\n'
end for
return string
```

Serializované stredy polygónov teda budú od seba oddelené koncom riadka a celok bude vyzerať takto:

```
'2356,30,155'
```

Deserializér je zložitejší, pretože jeho vstupom je textový reťazec, z ktorého musí extrahovať rôzne informácie a previesť ich na správny typ (napr. ak je serializované číslo, je prenášané ako textový reťazec, a deserializér teda musí tento text previesť naspäť na číslo). Algoritmus deserializéru pre údaje o stredoch polygónov je nasledovný:

Algoritmus 4.8
Vstup: text, v ktorom sú serializované údaje o polygónoch - `string`
Výstup: pole objektov, ktoré reprezentujú stredy polygónov (trojica voronoiId, x, y)- `result`

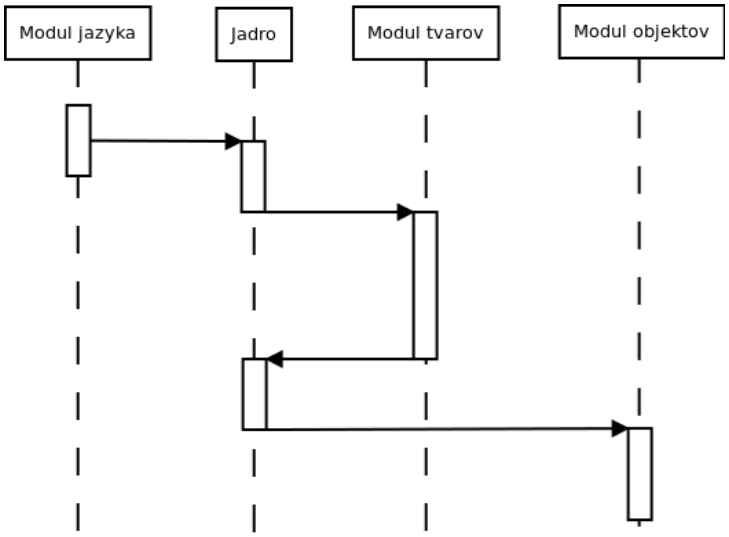
```
sites = string.split('\n')
for site in sites do
  splitted = site.split(',')
  result.push(parseInt(splitted[0]), parseInt(splitted[1]),
    parseInt(splitted[2]))
end for
return result
```

Funkcia `parseInt` vytvorí zo zadaného reťazcu hodnotu typu `integer`, alebo `NaN` ak textový reťazec nereprezentuje číslo. Funkcia `split` rozdelí textový reťazec na pole častí, ktoré sa nachádzajú pred a za daným identifikátorom. V poli `sites` sa teda nachádzajú údaje takéhoto tvaru:

```
site[0] = '2356,30,155'
site[1] = '2385,251,2'
…
```

4.13 Interakcia medzi modulami

Štruktúra modulov v programe je znázornená na obr. 4.1. Ak nejaký modul potrebuje vykonať operáciu, ktorá je implementovaná v inom module, najprv zavolá jadro programu. Jadro potom zavolá metódu príslušného modulu, a operácia sa vykoná. Celý proces je znázornený pomocou sekvenčného diagramu na obr. 4.8 na príklade vytvorenia nového objektu na mape.



Obrázok 4.8: Diagram komunikácie medzi modulami.

5 Testovanie jazyka

Jazyk sa testuje pomocou na syntaxi založenom testovaní. Pomocou sady testovacích príkazov sa overí správnosť každého pravidla jazyka. Tieto pravidlá sú uvedené v rozvinutej Backus-Naurovej forme (EBNF):

```
    <command-line> ::= <command> <obj-id-1> [<obj-id-1> [<obj-id-2>
[<obj-id-2> [<identificator>]]]]
    <command> ::= 'chci' | 'nechci' | 'budiz'
    <obj-id-1> ::= <identificator> | <keyword> | <number>
    <obj-id-2> ::= <identificator> | <keyword>
    <identificator> ::= '"' <text> '"'
    <keyword> ::= <k-object> | <k-size> | <k-location>
    <k-object> ::= 'les' | 'jezero' | 'poust' | 'more' | 'bazina' |
'mesto' | 'cesta' | 'reka' | 'potok'
    <k-size> ::= 'male' | 'stredne' | 'velke'
    <k-location> ::= 'na sever' | 'na juh' | 'na vychod' | 'na zapad'
| 'na sv' | 'na sz' | 'na jv' | 'na jz' | 'na kraji'
```

Pričom `text` je postupnosť ľubovoľných alfanumerických znakov a `number` je nezáporné číslo v desiatkovej sústave. K testovaniu bolo použité pole `synTest`, kde každý index tohoto pola reprezentuje jedno pravidlo a hodnota na tomto indexe značí počet použití daného pravidla. Pravidlá sú implementované takto:

```
synTest[0]: 'chci' <k-object>
synTest[1]: 'chci' <k-object> <identificator> <k-location>
<k-object> <identificator>
synTest[2]: 'chci' <k-object> <identificator> <k-location>
<identificator>
synTest[3]: 'chci' <k-size> <k-object>
synTest[4]: 'chci' <number> <k-object>
synTest[5]: 'chci' <number> <k-size> <k-object>
synTest[6]: 'chci' <identificator> <k-location> <identificator>
synTest[7]: 'chci' <identificator> <k-location> <k-object>
<identificator>
synTest[8]: 'nechci' <k-object>
synTest[9]: 'nechci' <k-object> <identificator>
synTest[10]: 'budiz' <k-object> <identificator>
synTest[11]: 'budiz' <k-object> <number> <identificator>
```

Zvolená sada testovacích príkazov:

1. chci les
2. chci jezero
3. budiz les "Temný les"
4. budiz jezero "Hluboké jezero"
5. chci jezero "Hluboké jezero" na severu lesa "Temný les"
6. chci velikou poust
7. budiz poust "Sahara"
8. chci poust "Sahara" na kraji "Temný les"
9. chci 6 mori
10. chci 2 male baziny

11. budiz bazina 0 "Veselá bažina"
12. budiz bazina 1 "Smutná bažina"
13. chci "Veselá bažina" na jihu "Hluboké jezero"
14. chci "Smutná bažina" na zapade lesa "Temný les"
15. nechci more
16. nechci bazinu "Veselá bažina"

Po zadaní všetkých príkazov z testovacej sady je výsledná hodnota testovacieho poľa nasledovná:

```
synTest = [2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 2]
```

Znamená to, že všetky pravidlá boli úspešne otestované, a hodnota v poli `synTest` značí počet použití pravidla na danom indexe.

6 Záver

Po preštudovaní tvorby máp v reálnom svete, a možností, ktoré nám vo svete hier poskytujú rôzne editory a generátory, sa podarilo vytvoriť funkčný on-line generátor máp. Pomocou tohoto generátoru je možné vytvárať komplexné mapy krajiny a širokého okolia, v ktorom sa pohybujú hrdinovia. Program je možné ovládať pomocou vlastného jazyka na zadávanie požiadavkov, ktorý je možné si preložiť do akéhokoľvek hovoreného jazyka, alebo pomocou myšky, či iného podobného zariadenia. Duálne ovládanie programu umožňuje zahrnúť do príkazov jazyka náhodnosť. Všetky algoritmy v moduloch je možné upraviť nezávisle na hlavnom jadre programu. Vytvorený materiál tiež poskytuje možnosti program ďalej rozširovať a doplňovať do programu nové funkcionality.

Použitie Voroného diagramu ako podložia sa ukázalo ako výhodné riešenie vzhľadom na realistickosť, náhodnosť a použiteľnosť mapy. Pri dostatočnom počte vygenerovaných polygónov nie je už nutné upravovať okraje objektov, podložie sa kvôli svojej nerovnakosti a nerovnomernosti do objektu nepremieta. Užívateľ teda môže mať pocit, akoby vytváral mapu na prázdnej ploche – zvyšuje sa realistickosť mapy.

Aplikáciu je možné spustiť na desktopovom počítači, ale aj kdekoľvek inde, kde je k dispozícii prehliadač s podporou javascriptu, zároveň toto riešenie ale kladie vysoké nároky na výpočtový výkon zariadenia.

6.1 Vylepšenia a pokračovanie projektu

Tu sú uvedené niektoré myšlienky, o ktoré by sa mohol program rozšíriť, a je tu načrtnutý smer prípadného pokračovania vývoja projektu.

- Doplniť jazyk o nové príkazy a rozšíriť tak možnosti manipulácie s objektami zahrňujúcej náhodnosť, napr. príkazy na určenie vzdialeností objektov a podobne. Doplniť do jazyka ďalší príkaz, ktorý by umožňoval vygenerovať kompletnú mapu náhodne, bez ďalšieho vstupu od užívateľa.
- Uviesť pohľady. Jednotlivé pohľady by zobrazovali jednotlivé časti sveta – pohľad, ktorý zobrazuje svet (teda kontinenty, moria), ďalší pohľad, ktorý zobrazuje plochu na kontinente včetne okrajov morí (stávajúci generátor), pohľad na zobrazenie mesta (polôh budov, ulíc, námestí) a konečne pohľad na zobrazenie interiérov jednotlivých domov (jednotlivé poschodia, rozloženie nábytku).
- Integrovať do programu generátor jaskýň na štvorcovom podloží, pre dobrodružstvá hrdinov
- Pri generovaní masívnejších máp (napr. včetne pohľadov) presunúť záťaž spojenú s generovaním na server, teda mimo užívateľské zariadenie.
- Spájať väčšie celky, zložené z viacerých objektov, a reprezentovať ich ako jediný objekt.
- Uviesť do programu parameter objektu nadmorská výška. Prispôsobiť generovanie vodných tokov, prípadne ciest. Takáto úprava by viedla k zvýšeniu realistickosti.
- Generovať názvy objektov náhodne, napr. výberom z množiny názvov.
- Doplniť program o možnosť zvolenia rozlíšenia mapy.

Literatúra

[1] Mapa - kritický rozbor definice podle ICA [online]. Február 2006 [cit. 2012-05-08]. Dostupný z WWW: http://www.fi.muni.cz/usr/richter/lekce/u02rozbordefinicemapy.pdf

[2] Automatic Code Generation [online]. Apríl 2004 [cit 2012-04-20]. Dostupný z WWW: http://dar.site.cas.cz/download.php?bd=313

[3] Metoda Digitální kartografie [online]. Júl 2011 [cit 2012-05-08]. Dostupný z WWW: http://www.la-ma.cz/?p=86

[4] Úvod do Geografických Informačních Systémů [online]. Október 2007 [cit 2012-05-08]. Dostupný z WWW: http://gis.zcu.cz/studium/ugi/e-skripta/ugi.pdf

[5] Moderní deskové a společenské hry a jejich postavení ve volném čase dnešní populace [online]. Apríl 2009 [cit 2012-05-08]. Dostupný z WWW: http://is.muni.cz/th/84899/pedf_m/Tomaskova_Zaneta.txt

[6] Voronoi diagram. Vlastnosti, použití, konstrukce. Zobecněné Voronoi diagramy [online]. November 2010 [cit 2012-05-08]. Dostupný z WWW: http://web.natur.cuni.cz/~bayertom/Adk/adk6.pdf

[7] Over-relaxation Lloyd Method for Computing Centroidal Voronoi Tessellations [online]. Február 2010 [cit 2012-05-08]. Dostupný z WWW: http://people.sc.fsu.edu/~jburkardt/vt2/miniconference_2010/xiao.pdf

[8] Polygonal Map Generation for Games [online]. September 2010 [cit 2012-05-08]. Dostupný z WWW: http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/

[9] Generátor map pro hry (RPG, počítačové, stolní) [online]. Máj 2007 [cit 2012-05-08]. Dostupný z WWW: http://www.fit.vutbr.cz/study/DP/rpfile.php?id=3921

Príloha 1: Návod na ovládanie programu

Tu sú popísané niektoré dôležité funkcie programu a ich ovládanie.

Uloženie mapy

Pre správnu funkciu ukladania a načítania mapy je nutné, aby bola aplikácia nahraná na serveri, ktorý je schopný spracovávať príkazy v jazyku PHP. Toto samozrejme nie je jediná možnosť, je možné spustiť aplikáciu aj na lokálnom počítači v off-line móde, avšak požiadavok spracovávania príkazov v jazyku PHP stále zostáva nutnosťou.

Ukladanie mapy sa spustí pomocou tlačítka `Save` v hornej časti panela vpravo. Ak uloženie prebehlo bez chyby, zobrazí sa v paneli číslo, pod ktorým sa mapa uložila. Toto číslo je jednoznačné ID pre uloženú mapu, a pri každom uložení sa vygeneruje znova. Takto je možné vrátiť sa k ľubovolnej uloženej mape aj v priebehu vývoja tej istej mapy.

Načítanie mapy sa spustí pomocou tlačítka `Load`. Pred spustením načítania mapy je nutné zadať do textového pola pod tlačítkami `Load` a `Save` ID požadovanej mapy. Pokiaľ načítanie prebehlo bez chyby, zobrazí sa v programe načítaná mapa a je možné s ňou okamžite pracovať.

Vytváranie objektov pomocou interakcie s GUI

Vytváranie objektov pomocou GUI (konkrétne pomocou tlačítka myši) prebieha pomocou kliknutia na požadovaný objekt v paneli vpravo. Toto kliknutie je programovo rovnocenné s vytvorením objektu pomocou príslušného príkazu jazyka.

Pomocou GUI je ďalej možné zadať počiatočný a konečný bod líniového objektu (cesty, rieky či potoka). Po kliknutí na tlačítko `lineStart` je možné kdekoľvek na mape zadať počiatočný bod línie pomocou ďalšieho kliknutia. Podobne sa zadáva aj konečný bod línie, pričom sa ale musí kliknúť na tlačítko `lineEnd`.

V pravom paneli je tiež možné označiť parameter `Silliness`. Pokiaľ je táto voľba označená, program bude generovať nepravidelné objekty s väčšou mierou náhodnosti. Ak užívateľ túto voľbu neoznačí, program bude generovať objekty nepravidelne kruhového tvaru. Táto voľba sa nevzťahuje na objekty líniového typu.

Konzola pre zadávanie príkazov

Konzola sa nachádza v dolnej časti obrazovky. Do spodnej časti konzole je možné zadávať príkazy pomocou jazyka na zadávanie požiadavkov na mapu. V hornej časti konzole sa nachádza oznamovacia oblasť, kde sa zobrazuje niekoľko posledných zadaných príkazov a v prípade nesprávne zadaného alebo nesplniteľného príkazu tiež chybové hlásenia. Každý príkaz je nutné potvrdiť klávesou enter, to znamená že posledný znak každého príkazu musí byť nový riadok.

Posledný príkaz zadaný do konzole je možné zobraziť pomocou šípky hore, a zadávacie okno konzoly sa vymaže pomocou šípky dole.

Export mapy do obrázkového formátu

Export mapy do obrázku vo formáte PNG je možné spustiť pomocou tlačítka `Save as PNG` v pravom paneli. Pre správnu funkčnosť exportu je nutné, aby obrázky (textúry objektov) boli umiestnené na rovnakej doméne, ako samotný program. Ak sú všetky časti programu umiestnené na rovnakom externom serveri, funguje export správne, zatiaľčo pri spúšťaní z lokálneho počítača nie je správna funkčnosť exportu zaručená.

Príloha 2: Obsah CD

CD priložené k tejto práci obsahuje adresáre s nasledujúcim obsahom:

- odt - text tejto práce vo formáte ODT,
- src - všetky súbory potrebné pre spustenie generátora máp.