VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GENERÁTOR HERNÍ MAPY GALAXIE

GALAXY GENERATOR FOR GAMES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. KAREL BŘEZINA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ STARKA**
SUPERVISOR

BRNO 2018

Abstrakt

Tato práce se zabývá návrhem a implementací procedurálního generátoru galaxie pro využití v počítačových hrách nebo simulacích. Součástí je také implementace demonstrační aplikace, která ukazuje možný způsob využití generátoru. Generátor je schopný vytvářet galaxie na základě předvolených typů, ale i uživatelsky vytvořených map.

Abstract

This thesis is focused on design and implementation of procedural generator of galaxy for games or simulations. Second goal is implementation of demonstration application which is showing possible usage of generator. Generator is able to create galaxy by predefined types or by user created map.

Klíčová slova

galaxie, hra, mapa, generátor, procedurální, slunce, qt quick, opengl, editor, koule, spirála, mřížka, shluk, heat mapa.

Keywords

galaxy, game, map, generator, procedural, sun, qt quick, opengl, editor, sphere, spiral, grid, cluster, heatmap.

Citace

BŘEZINA, Karel. *Generátor herní mapy galaxie.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Starka

Generátor herní mapy galaxie

Prohlášení

Prohlašuji, že jsem tuto diplomouvou práci vypracoval samostatně pod vedením pana Ing. Tomáše Starky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Karel Březina
17. května 2018

Poděkování

Rád bych poděkoval svému vedoucímu za jeho čas, který si dokázal udělat, kdykoliv jsem potřeboval, a jeho četným konstruktivním radám, které mi ušetřily spoustu času spojeného s touto prací. Dále bych také rád poděkoval panu Ing. Tomáši Miletovi za jeho odbornou konzultaci.

Obsah

Kapitola 1

Úvod

Procedurální generování se objevuje v oblasti vývoje počítačových her déle než 30 let. Pro počítačové hry se jednalo zpočátku o technologii rozšiřující možnosti tehdejších počítačů, které nedisponovaly dostatečnými paměťovými úložišti, umožňující použít větší množství multimediálních dat (jako jsou textury, zvuky nebo hudba). Procedurální generování umožňovalo v tomto případě vytvoření her nových žánrů, jako například `Roguelike` nebo `Space flight simulator`.

V průběhu několika dalších desítek let se počítačový hardware neustále zdokonaloval a s ním se rozšiřovaly i možnosti procedurálního generování v počítačových hrách. Dříve se generovala především multimediální data, jako například zvuky a textury. Ty jsou dnes vytvářeny ve vyšší kvalitě umělci různých odvětví. Generátory našly uplatnění na poli generování herního obsahu. Nejčastěji herní lokace (labyrinty, cesty mezi statickými oblastmi), kombinace vlastností získaných předmětů nebo dynamicky generované události a příběh. Tyto vlastnosti bychom mohli najít u počítačových her ze série Diablo, Borderlands, The Elder Scrolls II: Daggerfall, Minecraft nebo Left 4 Dead. Na obrázku 1.1 lze vidět využití procedurálního generování spirálovité galaxie obsahující sluneční soustavy, mlhoviny a jiná vesmírná tělesa.

Obrázek 1.1: Vygenerovaná galaxie tvaru spirála z počítačové hry Stellaris.

V této práci se zaměřuji dále na zmíněný žánr `Space flight simulator` a využití technologie procedurálního generování k vytvoření galaxie použitelné jako herní mapa. Podobně jako v již zmíněné počítačové hře `Stellaris`, budou generované galaxie složené ze slunečních soustav, mající své vlastnosti a specifikace. Galaxie budou generované podle sady definovaných vlastností generátoru. Výstup bude možné snadno použít ve vlastní aplikaci díky možnosti exportu vygenerovaných dat. Další fází projektu je vytvoření ukázkové aplikace, která bude demonstrovat použití vytvořené knihovny generátoru galaxie v praxi. Zároveň bude aplikace sloužit jako inspirace, jakým způsobem lze vytvořit vlastní řešení.

V kapitole 2 popisuji algoritmy použité pro procedurální generování, jejich využití v různých žánrech počítačových her a praktické ukázky z konkrétních herních titulů. V kapitole 3 je popsán návrh generátoru galaxie jako aplikační knihovny, návrh ukázkové aplikace a její propojení s knihovnou generátoru. Implementační detaily knihovny i aplikace se nachází v kapitole 4. V závěrečné kapitole 5 je zhodnocení výsledné práce a možnosti jejího rozšíření.

Kapitola 2

Procedurální generování

V této kapitole popisuji metody procedurálního generování, použité v různých aspektech vývoje počítačových her. Tyto metody nejsou striktně omezené pouze na popisovaný druh generovaného obsahu, a lze je tak snadno modifikovat i pro zcela odlišné úlohy. Procedurální generování není založené pouze na stoprocentní náhodnosti, ale zahrnuje často interakci návrháře v podobě definování pravidel, vlastností a omezení generátoru. Tato kombinace vytváření herního obsahu přináší pozitivní vlastnosti z obou světů, kdy na jedné straně stojí vždy nově vygenerovaný obsah generátoru a na druhé směr a rozsah, jakým je tento obsah generován. Takový obsah může v kladném případě přispět k podpoře znovuhratelnosti, zábavnosti a dalších důležitých vlastností počítačových her. Generátory jsou v počítačových hrách často zpřístupňovány samotným hráčům, kteří si tak mohou do jisté míry sami přizpůsobit generovaný obsah. Jednotlivé herní žánry využívají procedurální generování na specifické úlohy spojené s daným žánrem. Následující popsané metody procedurálního generování jsou zařazeny k žánrům počítačových her, které využívají běžně procedurálního generování obsahu. Generátory se však neobejdou bez využití (pseudo)náhodných čísel (které tvoří z velké části jejich základ), případně dalších matematických algoritmů.

2.1 Poissonova distribuce

Toto rozdělení můžeme popsat jako náhodnou hodnotu, která je definována procentuálním výskytem daného jevu ve zvoleném intervalu jedné z veličin, jako je například čas nebo vzdálenost[12]. Distribuce je pojmenovaná po francouzském matematikovi Siméonovi Denisovi Poissonovi[1] a často se používá při simulacích typu kolikrát za čas nastane událost. To, zda jev nastane nebo nenastane, není nijak závislé na tom, zda stejný jev nastal v jiný čas nebo na jiném místě. Zároveň nemůže nastat případ, že by nastaly dva jevy přesně ve stejný okamžik nebo na stejném místě. Hodnota průměrného počtu výskytů je označována s délkou 1. Výpočet hodnoty je znázorněn rovnicí 2.1, kde x je posloupnost čísel (0,1,2,..) náhodné veličiny X a $\lambda > 0$ parametr.

$$P(X = x) = \frac{\lambda^x}{x!} e^{-\lambda} \tag{2.1}$$

[1] Originál jeho práce z roku 1837 byl digitalizován a lze si jej prohlédnout na https://books.google.cz/books?id=uovoFE3gt2EC&pg=PA206

2.2 Generátory pseudonáhodných čísel

Nedílnou součástí všech procedurálních generátorů jsou generátory pseudonáhodných čísel, zajišťující odlišnost generovaného obsahu při každém volání. Generátory čísel se vyznačují několika charakteristikami, které definují jejich celkovou kvalitu. Do těchto charakteristik můžeme zahrnout výkonnost (jak dlouho trvá výpočet dalšího čísla) nebo kvalitu rozložení generovaných čísel (za jak dlouho se začne sada vygenerovaných čísel opakovat). Většinu těchto generátorů lze modifikovat nastavením vnitřních proměnných nebo nastavením tzv. počátečního čísla (v angl. `seed`). Pomocí těchto proměnných se pak mění sada generovaných čísel a při pozdějším použití stejných proměnných dostaneme ta stejná čísla[2]. Vedle těchto generátorů existují i generátory náhodných čísel. Tyto generátory jsou však z důvodu své slabé výkonnosti používané pouze při operacích, které nutně vyžadují jejich náhodnost (generování bezpečnostních hesel nebo náhodný výběr objektu pro detekci bezpečnostních hrozeb). V následujících sekcích jsou popsané vybrané generátory pseudonáhodných čísel.

2.2.1 Generátory s posuvným registrem

Tento druh generátorů je označován jako `Linear Feedback Shift Register` (LSFR) a je snadno implementovatelný v hardware. Samotná činnost probíhá výběrem jedné z bitových operací (například OR nebo XOR). Vybraná operace je aplikována na zvolené bity čísla vygenerovaného v předchozím kroku generátoru a výsledek je zapsán zpět do posuvného registru. Generátory tohoto druhu jsou snadno použitelné s dobrou statistikou výstupních čísel.

2.2.2 Lineární kongruentní generátory

Anglickým pojmem `Linear Congruential Generator` (LCG) je označován nejjednodušší druh generátorů pseudonáhodných čísel, představený americkým matematikem D. H. Lehmerem v roce 1949[9]. Princip generování čísel je založen na využití posledního čísla ve vygenerované posloupnosti, přičemž první číslo posloupnosti je `seed` generátoru. Vygenerované číslo je ořezáno rozsahem generátoru, který je nejčastěji udáván mocninou 2, kde mocnina je rovna délce použitého datového typu. Výhoda těchto generátorů je jejich výkonnost a nízká náročnost na paměť. Nevýhodou je kvalita generovaných čísel, které se po určité době začnou opakovat. Generátory tohoto druhu nejsou vhodné na komplexnější procedury, mezi které může patřit i procedurální generování obsahu. Naopak se mohou hodit ke generování `seedu` pro jiný, komplexnější generátor pseudonáhodných čísel.

2.2.3 Mersenne Twister

Asi nejpoužívanější z generátorů dostal své jméno podle pojmu "`Mersenne prime`", který znázorňuje prvočíslo, které vzniká odečtením čísla jedna od jakékoliv mocniny dvou. Generátor byl vytvořen japonskou dvojicí Makoto Matsumoto a Takuji Nishimura v roce 1997[10]. Oproti všem předchozím generátorům nabízí vysokou kvalitu generovaných čísel při zachování vysoké výkonnosti. Často se tak používá v oblasti simulací. Standardní implementace generátoru se nazývá `MT19937` a používá ve svém výpočtu délku 32-bitová slova. Existuje však i 64-bitová varianta, která generuje jinou posloupnost čísel. Generátor splňuje velkou část statistických testů. Stejně jako ostatní zmíněné generátory se tento nehodí pro využití v oblasti zabezpečení (například kryptografie). Nevýhodou tohoto algoritmu může

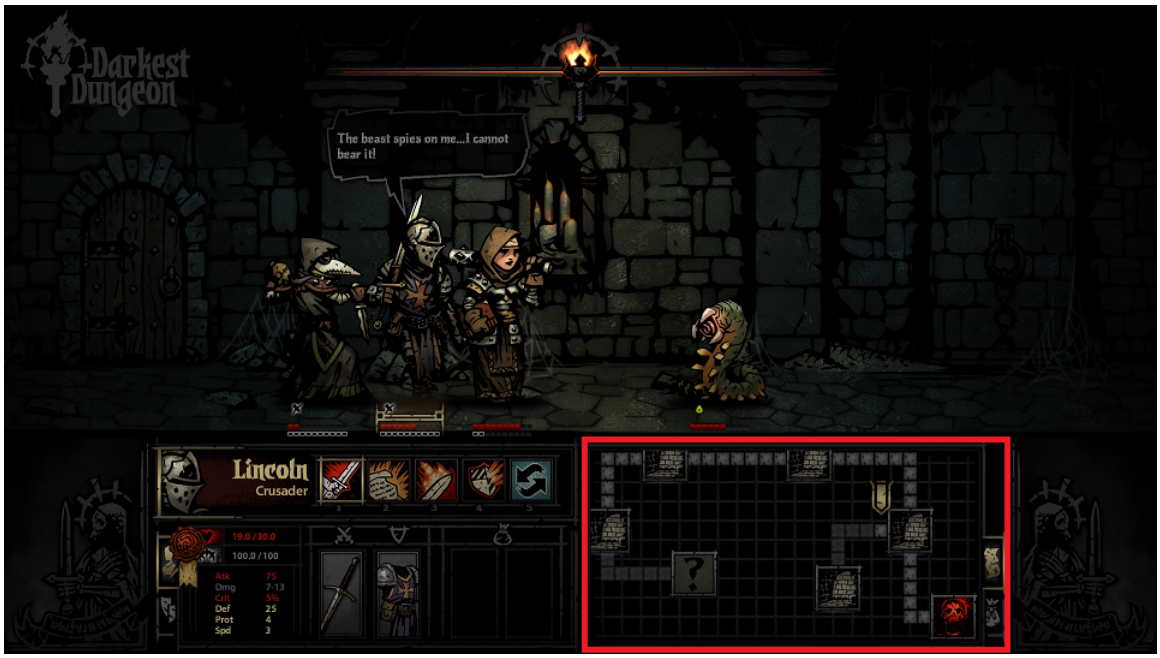
[2] Proto označení generátorů generující pseudonáhodné čísla.

být delší čas ustálení výstupních hodnot generátoru, které nemusí splňovat statistické testy. Podrobnější informace lze nalézt na oficiální webové stránce generátoru[4].

2.3 Metody generování úrovní – Dungeons

Pod označením žánru `dungeon` si lze představit herní úroveň složenou z množiny spletitých chodeb a místností, kterými se hráč snaží procházet ve snaze nalézt cestu ven z úrovně, zneškodnit jinou herní postavu nebo spustit událost nacházející se v jedné z vygenerovaných místností. Tento žánr byl oblíbený převážně v 80. a 90. letech minulého století a v posledních letech se tento žánr opět objevuje v modernějším pojetí herních prvků. Typ generovaných úrovní by se dal připodobnit k dobře známým bludištím (resp. labyrintům), ve kterých je důležitá dobrá orientace v prostoru. Zatímco dříve bylo nutné vygenerované mapy zakreslovat na papír, dnes je průchod hráče úrovní automaticky zaznamenáván. Při pohybu mezi jednotlivými úrovněmi vygenerovaného světa naráží hráč na (ne)přátelské postavy nebo předměty, ulehčující hráči průchod lokacemi. Všechny zmíněné aspekty žánru lze zahrnout do návrhu generátoru, jehož výstupem je plně funkční a hratelná úroveň. V pravé dolní části obrázku 2.1 lze vidět mapu znázorňující část vygenerované úrovně.

Prvním krokem generátoru je definice rozsáhlosti a způsobu vytváření úrovní. Například, zda je možné vygenerovat všechny lokace na počátku spouštění nové instance hry a mít všechna tato data uložená v paměti, nebo pokud je nutné dynamicky reagovat na poslední akce hráče, či jiné události a přizpůsobit tomu generovanou úroveň. Následující popsané metody jsou upravitelné pro oba zmíněné scénáře přístupu.



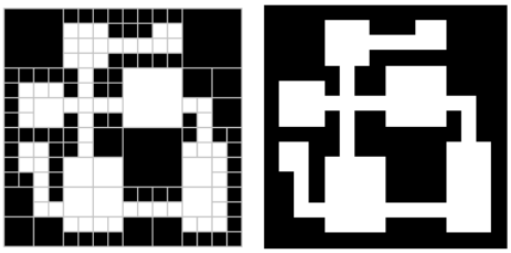
Obrázek 2.1: Darkest Dungeon - počítačová hra žánrů role-playing game (RPG) a dungeon crawl z roku 2016. Místnosti zobrazené na mapě mohou ukrývat nepřátelské postavy, cenné předměty nebo jiné důležité objekty.

2.3.1 Dělení prostoru

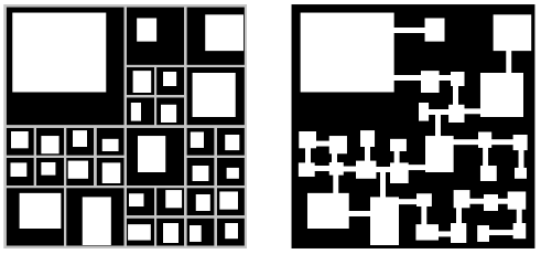
Metoda dělení prostoru je dobře známá z prostředí akceleračních datových struktur, které pomáhají výrazně zvyšovat výkon při zpracování časově náročných operací, jakými jsou například metody globálního osvětlení (`raytracing`, `pathtracing` ad.) nebo detekce kolizí[6]. Dělení prostoru může probíhat ve 2D nebo 3D prostoru a mezi její algoritmy patří například struktury `quadtree` resp. `octree`. V případě generování `dungeonů` se soustředíme na `quadtree` a způsob jeho využití pro definování herních lokací.

Úroveň můžeme popsat jako množinu místností, které jsou navzájem propojeny chodbami. Místnosti mohou být generované zarovnaně vzhledem k mřížce děleného prostoru `quadtree` nebo náhodně tak, že v jedné buňce mřížky může být pouze jedna místnost, jejíž velikost ale nemusí odpovídat velikosti buňky. Oba přístupy můžeme vidět na obrázcích 2.2 a 2.3, kde černá barva znázorňuje zeď (nedostupný prostor) a bílá barva prostor, ve kterém se může hráč pohybovat.

Dělení prostoru úrovně probíhá rekurzivně až do chvíle, kdy velikost dělené buňky dosahuje minimální definované velikosti. Celý proces je popsán pomocí následujícího pseudokódu.



Obrázek 2.2: Vygenerovaná úroveň s místnostmi zarovnanými na mřížku děleného prostoru[13].



Obrázek 2.3: Vygenerovaná úroveň s místnostmi s náhodnou velikostí v mřížce děleného prostoru[13].

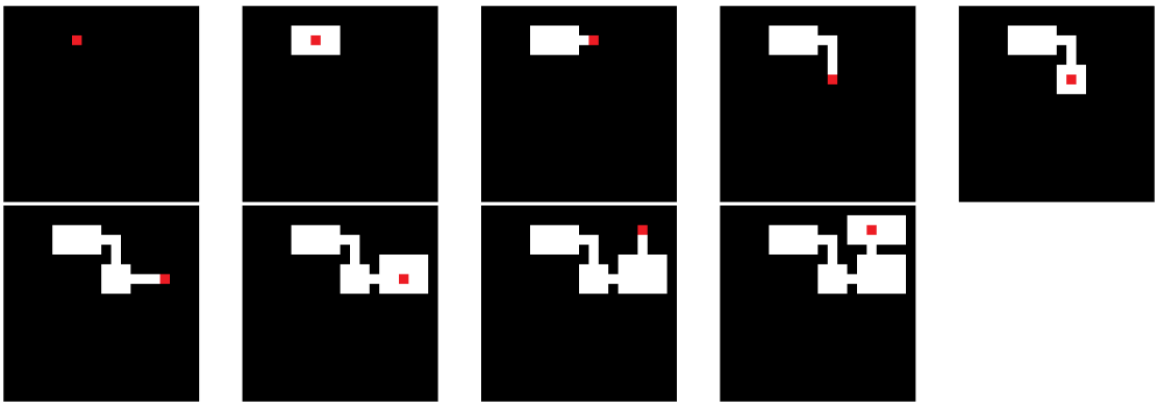
Data: Celý prostor úrovně
Result: Úroveň složená z místností propojených chodbami
1 Rozdělíme prostor horizontálně nebo vertikálně v jejím středu;
2 Vybereme jeden ze dvou nově vzniklých prostor;
3 **if** *Pokud je vybraný prostor větší než minimální požadovaná velikost* **then**
4 | Přejdi na krok 1 s vybraným prostorem
5 **else**
6 | Vyber druhý ze vzniklých prostor a přejdi na krok 3
7 **end**
8 Pro každý prostor vytvoř místnost zvolením levého horního a pravého dolního bodu definující její rozsah v prostoru;
9 Začni vytvářet chodby mezi místnostmi, které pocházejí ze stejného rodičovského prostoru;

Algoritmus 1: Generování úrovně metodou dělení prostoru.

2.3.2 Postupné generování agentem

Oproti metodě dělení prostoru, kde byla úroveň vygenerována pro celý prostor, je metoda agenta více zaměřená na strategii postupného generování. Agent je spuštěn na zadané pozici v úrovni a svým pohybem do stran vytváří nové části chodeb. Jednou ze strategii je tzv. slepé generování, při kterém se agent po každém svém pohybu pokusí změnit náhodně směr a vytvořit místnost. S těmito pokusy jsou pevně spjaty jejich pravděpodobnosti. V případě neúspěšného vygenerování se zvyšuje tato pravděpodobnost pro další pokus agenta v následujícím kroku. Pokud je pokus úspěšný, pravděpodobnost akce je snížena na nulu. Tato strategie je jednoduchá na implementaci, ale zároveň trpí mnoha neduhy, mezi které patří překrývání vytvořených místností nebo zablokování agenta v některém z rohů vymezeného prostoru úrovně.

Druhou strategií je predikované generování, které se od slepého generování liší kontrolou dostatečného prostoru pro generování nové místnosti a zamezení tak vzájemného průniku obou místností. Pokud by nastala tato situace, agent se přesune dostatečně daleko na to, aby šlo novou místnost vygenerovat. V případě, že by agent nenašel prostor dostatečný pro požadovanou místnost, bude se pokoušet o zmenšení generované místnosti na hranici minimální velikosti. Agent končí svoji činnost poté, co zjistí, že již nemůže přidat žádnou další místnost nebo chodbu (viz. obrázek 2.4). I v případě této strategie může být agent zablokován v některém z rohů úrovně. Mým navrhovaným řešením by mohl být backtracking agenta do předchozí místnosti a změna směru do zatím neprozkoumaných částí úrovně.



Obrázek 2.4: Úroveň generovaná agentem se strategií predikovaného generování. Každý snímek zobrazuje samostatnou akci agenta. Poslední snímek zobrazuje stav, kdy agent není schopen dále pokračovat ve své činnosti a ukončí generování úrovně[13].

2.3.3 Celulární automaty

Celulární automaty využité pro generování herních lokací disponují oproti ostatním metodám vyšší mírou „přirozenosti“ generovaného prostoru. Tato metoda[8] generování probíhá v reálném čase a vytvořené lokace připomínají spíše než místnosti jeskyně. Bez větších problémů lze tak generovat prostory dynamicky rozšiřitelné ve všech směrech úrovně. Metoda používá čtyři základní parametry, kterými je generování řízeno.

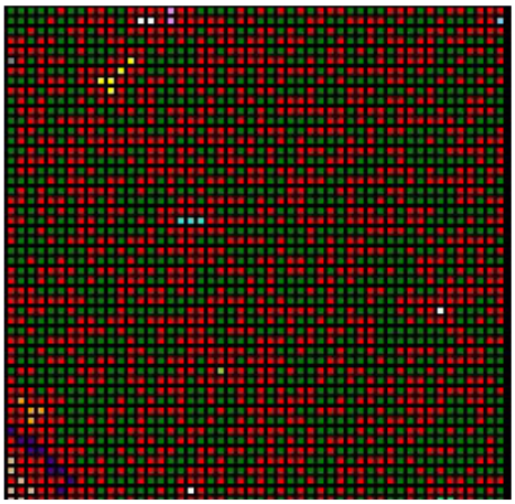
- Procento nepřístupných prostor
- Počet generací celulárních automatů

- Hraniční hodnota počtu sousedů definujících nepřístupný prostor
- Počet sousedních buněk

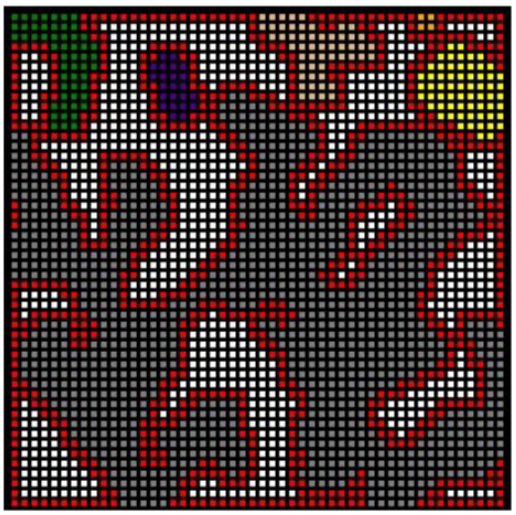
Úroveň je v následujícím příkladu definována mřížkou o velikosti 50x50, ve které může každá buňka být v jednom ze stavů: prázdný nebo nepřístupný. Při inicializaci mřížky je každá její buňka ve stavu prázdný. Generování prostoru v mřížce pak probíhá následovně.

- Na začátku se s definovanou pravděpodobností určí, zda bude buňka ve stavu nepřístupná nebo zůstane ve stavu prázdná.
- Spustí se celulární automat na mřížku pro n kroků. Pokud má buňka v mřížce nejméně T počet sousedů ve stavu nepřístupný, pak i tato buňka bude ve stavu nepřístupný pro další krok automatu. V opačném případě bude buňka ve stavu prázdný.
- Nepřístupný prostor je na svých okrajích označen jako zeď. Ta má stejnou vlastnost jako nepřístupný prostor, pouze vypadá jinak.

Na obrázcích 2.5 a 2.6 můžeme vidět vlevo náhodně vygenerovanou mapu a vpravo mapu vygenerovanou celulárním automatem s parametry počtu kroků $n = 4$ a počet sousedů $T = 5$. Nepřístupný prostor je znázorněn bílou barvou a zeď červenou. Ostatní barvy znázorňují navzájem izolované prostory. Tato metoda je velice rychlá a intuitivní, ale zároveň je obtížné určit míru ovlivnění vlastností vygenerované mapy změnou jednoho či více z uvedených parametrů. Mnou navrhovanou optimalizací této metody by mohla být analýza přístupných prostor, oddělených tenkou[3] zdí a jejich následné propojení odstraněním této zdi mezi oběma místnostmi.



Obrázek 2.5: Náhodně generovaná mapa s definovaným poměrem nepřístupného a prázdného prostoru 1:1[8].



Obrázek 2.6: Vygenerovaná mapa s použitím celulárního automatu s definovaným počtem kroků 4 a počtem sousedů 5[8].

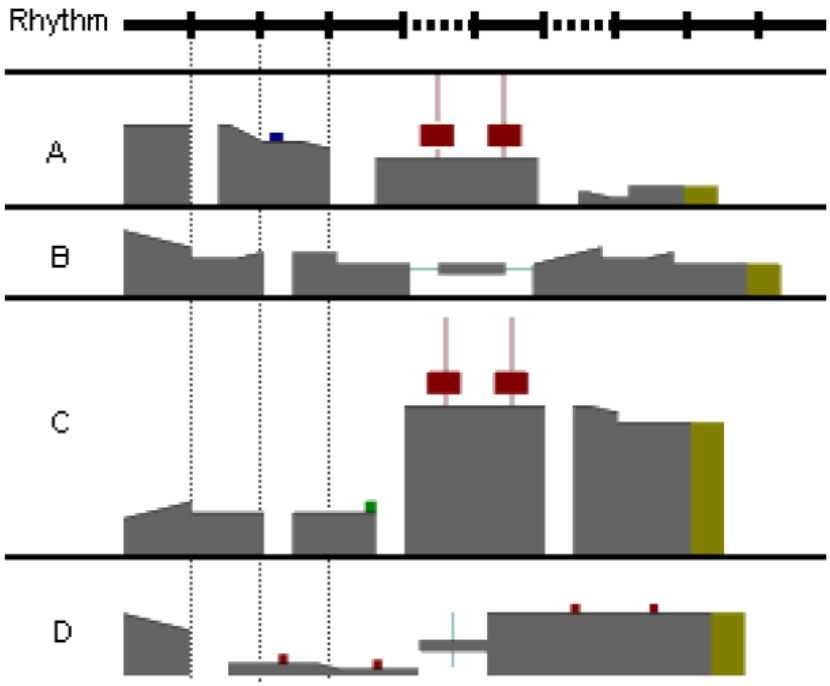
[3] Hodnota definovaná generátorem.

2.4 Metody generování úrovní – Plošinovky

Následující metody popisují způsoby generování herních úrovní pro žánr počítačových her zvaných plošinovky (v angl. `Platform game`). Tento žánr je obvykle situován ve 2D prostoru a generování herních úrovní obvykle podléhá herní logice, ve které se postava pohybuje v horizontálním směru pro pohyb vlevo, vpravo a vertikálním směru akcemi skok nebo pád. Plošinovky často na herní postavu aplikují vlastnost gravitace a na jednotlivé části herní lokace se tedy často nedá dostat z jakékoliv pozice. Generované úrovně musí s tímto omezením počítat, aby nedocházelo ke generování úrovní, ve kterých by byly místnosti dostupné z pohledu spojitých místností, ale prakticky nedostupné z pohledu hratelné postavy. Opět jsou i tyto metody generování použitelné v modifikované podobě v jiných žánrech počítačových her.

2.4.1 Rytmické skupiny

První metodou je tzv. rytmické generování[14], které se inspiruje v notaci rytmů, do kterého patří například časování nebo opakování hráčových akcí ve hře. Na počátku hry se vytvoří menší část úrovně, která je označovaná jako rytmická skupina. Tato struktura pracuje na bázi dvouvrstvé gramatiky, kde první vrstva zachycuje množinu možných hráčových akcí ve hře, která je převedena na tzv. korespondující geometrii. Druhá vrstva obsahuje dostupný herní prostor. Části úrovně pak následně vznikají propojením rytmických skupin a množinou ručně implementovaných úrovní.



Obrázek 2.7: Čtyři možné varianty vygenerované geometrie podle rytmu. Malé červené čtverce znázorňují nepřátelé, které musí hráč eliminovat. Zelené čtverce pružiny umožňující hráči vysoký skok a modré čtverce nepřátele, které musí hráč přeskočit. Žluto-zelené obdélníky na konci znázorňují spojovací pás s další rytmickou skupinou. Délka časové osy znázorňuje délku rytmu a čárky místa, ve kterých bude zahájena akce.

Návrhářům úrovní je tak umožněno lepšího ovládání generování částí úrovně, jakými jsou například začátek, konec a klíčové body úrovně, frekvence generování geometrických útvarů, způsob, jakým jsou umístěny v úrovni užitečné předměty (mince, životy, munice ad.). Další parametry se zaměřují přímo na definici rytmické skupiny, u které lze řídit její délku, hustotu nebo typ a vzor rytmu, se kterou generátor pracuje.

2.4.2 Regulovaná obsazenost rozšíření

V anglickém znění `Occupancy-Regulated Extension`. Metoda[11] je vhodná pro generování obsahu do předem připravených částí úrovní. Tyto části tvoří základ a jsou vybírány generátorem z knihovny ručně vytvořených částí úrovní. Každá část úrovně odkazuje na množinu prvků, podle kterých si může generátor lépe vybrat. Algoritmus generování úrovně vypadá následovně.

- Vybere se náhodně pozice, kde by se mohl hráč v blízké budoucnosti vyskytnout, pro umístění části úrovně.
- Vybere se nejvhodnější část úrovně z knihovny podle specifikovaných vlastností sousední části úrovně a požadovaných specifik.
- Vybraná část úrovně se začlení do stávající geometrie úrovně. Tento proces pokračuje, dokud v úrovni nezůstává jediná část, kterou by bylo možné obsadit. Po ukončení této fáze dojde k umísťování objektů do úrovně.

Obrázek 2.8: Spelunky - 2D plošinovka z roku 2008 patří mezi open source indie počítačovou hru, která byla vydána pod freeware licencí. Hráči si mohou vytvářet vlastní generátory map. Na adrese[5]je k dispozici návod a online nástroj, jak vytvořit vlastní generátor.

[5] http://tinysubversions.com/spelunkyGen/

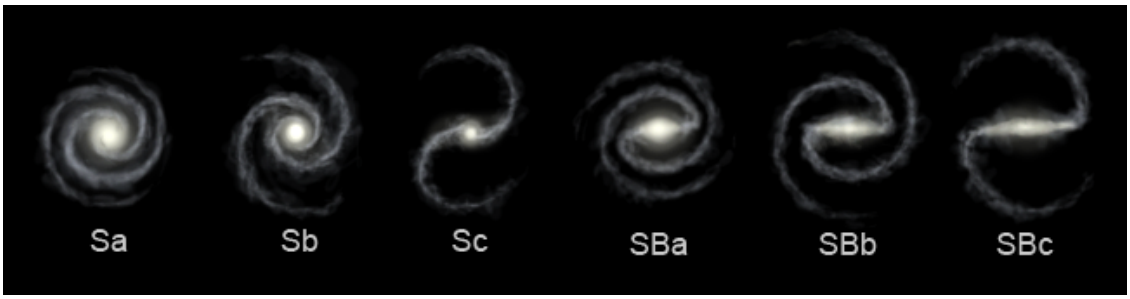
Návrháři mohou u této metody ovlivnit pravděpodobnost, s jakou jsou dané části úrovně vybírány generátorem, nebo budoucí geometrii úrovně a možné akce hráče. Tato metoda se řadí svou kombinací procedurálního generování a manuálně vytvořeného obsahu mezi lépe ovladatelné z pohledu návrhu hratelnosti. Opakem je samozřejmě menší míra účasti generátoru na vytváření herní lokace a tedy i větší časová náročnost celkového řešení. Jednou z počítačových her, které implementují tuto metodu v praxi, je hra Spelunky (viz. obrázek 2.8).

2.5 Metody generování úrovní – Vesmírné strategie

Žánr vesmírných simulátorů patří do kategorie, kde je procedurální generování obsahu využíváno velmi často. Je tomu tak především z důvodu rozsahu herních úrovní, které pokrývají celé galaxie, nebo absencí nutnosti precizní návaznosti vygenerovaných prvků. Generátory vesmírných simulátorů se zaměřují často na věrné ztvárnění vesmírných útvarů, které se snaží co nejlépe kopírovat jejich reálné předlohy s co největší množinou popisovaných vlastností. Pro dosažení těchto výsledků je vhodná kombinace několika metod, při kterých je generován vždy jiný vesmírný útvar a tvoří tak celek, který je ovládán hlavní metodou generátoru. V následujícím textu jsou popsány některé z útvarů, které byly použity ve vesmírných simulátorech, a příklady jejich použití.

2.5.1 Spirálovitá galaxie

Tento tvar galaxie patří do množiny tříd Hubblovy sekvence. Galaxie se skládá obvykle z centrálního disku, který je plochý a rotuje spolu s hvězdami a hvězdným prachem. Vnitřní uskupení uprostřed disku se nazývá boule. Podstatnou část struktury galaxie tvoří spirálovitá ramena, která jsou mladší, a proto i jasnější než centrální disk. Nezřídka lze u spirálovitých ramen pozorovat strukturu podobnou mostu vycházející z boule a končící na počátku ramene. U spirálovité galaxie rozlišujeme několik jejích typů (Sa/SBa, Sb/SBb, Sc/SBc), které se liší mírou těsnosti celé galaxie, shlukováním ramen a velikostí centrální boule. Například galaxie typu Sc a SBc (B značí přítomnost spojujícího mostu ramen) mají velmi volná ramena oproti galaxiím typu Sa a SBa s těsně umístěnými rameny.

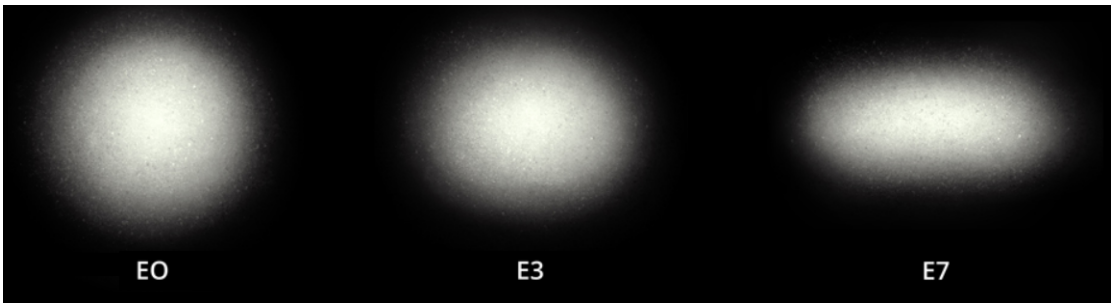


Obrázek 2.9: Jednotlivé tvary spirálovité galaxie. Písmeno B značí přítomnost spojujícího mostu ramen.

2.5.2 Eliptická galaxie

Jak napovídá název, tento typ galaxie je tvarem podobný elipse. Galaxie je oproti spirálovitému tvaru více rozprostřená v prostoru a obsahuje většinou pouze hvězdy. Hvězdy galaxie

bývají obvykle rovnoměrně rozmístěny v celém jejím prostoru. Eliptická galaxie je podle Hubble[7] jednou ze tří základních tříd galaxií (spolu se spirálovitou a čočkovitou galaxií). V tomto typu galaxie se vyskytují spíše menší hvězdy většího stáří s minimální aktivitou, které se nacházejí poblíž středu shluku galaxie. Tvar galaxie (konkrétně jak je podlouhlá) je určován hodnotami značenými En (E značí eliptický), kde n je v rozsahu hodnot 0-7 (viz. obrázek 2.10). Hodnota 0 značí galaxii nejvíce podobnou kulovitému tvaru a hodnota 7 naopak vysoce podlouhlý tvar, ve kterém je galaxie častokrát označována za nestabilní (tento stav je také označován angl. termínem `FireHose instability`[6]).



Obrázek 2.10: Rozdíly mezi různými stupni tvaru eliptické galaxie.

Hodnotu n lze vypočítat následujícím vzorcem číslo 2.2, kde b/a určuje poměr křivek světelné intenzity galaxie vedlejší (b) a hlavní (a) osy.

$$n = 10 * (1 - \frac{b}{a}) \tag{2.2}$$

2.5.3 Prstencová galaxie

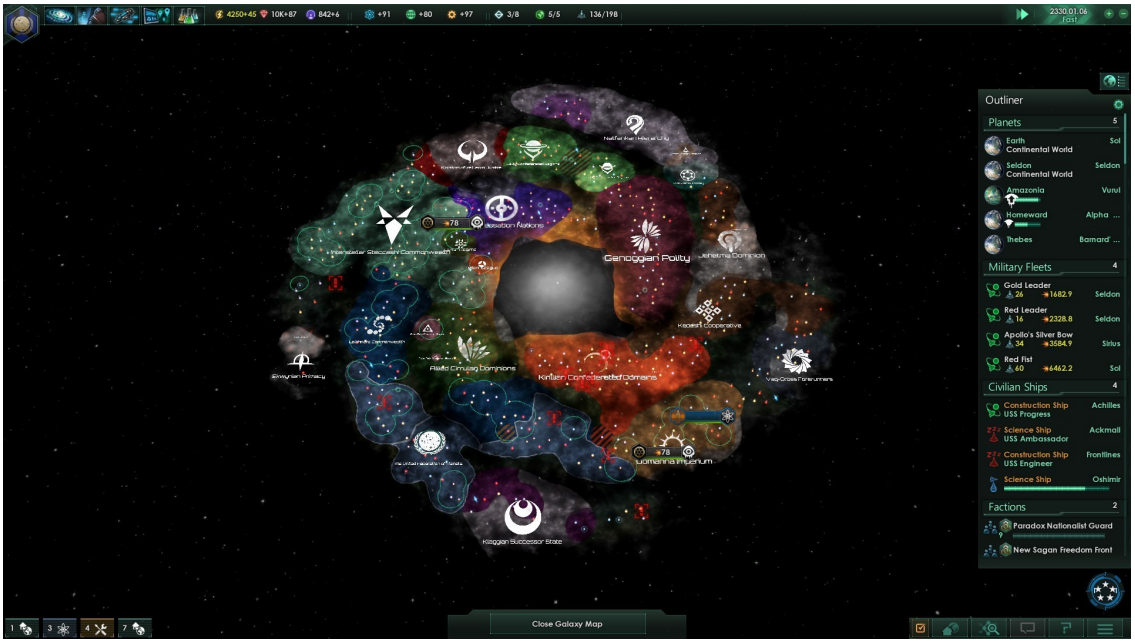
Galaxie je rozpoznatelná svým vnějším prstencem s kulovitým středem[2]. Zatímco na okraji galaxie se nachází větší množství světle modrých hvězd, které lze díky jejich jasu snadno pozorovat, uprostřed lze vidět vyšší zastoupení červenějších hvězd, které jsou pravděpodobně mnohem starší. Mezi těmito dvěma útvary je téměř dokonalá tma. Prstencová galaxie na snímku se nazývá `Hoag's Object` a byl pořízen Hubblovým teleskopem v roce 2001.

Obrázek 2.11: Hoag's object reprezentuje prstencovou galaxii[2].

[6] https://en.wikipedia.org/wiki/Firehose_instability

2.5.4 Příklad generátoru galaxie – Stellaris

Jednou z počítačových her implementujících generátor galaxie je 4X strategie Stellaris[1]. Označení 4X popisuje kategorii her zahrnující prvky `explore` (průzkum okolí), `expand` (rozšiřování civilizace), `exploit` (těžení surovin) a `exterminate` (podmanění ostatních civilizací). Hra vyšla v roce 2015 a její procedura generování galaxie vypadá následovně. Před spuštěním generátoru je nutné určit tvar galaxie a počet počítačem ovládaných oponentů. Velikost galaxie se pohybuje v rozmezí 200-1000 hvězd. Mezi předpřipravené tvary galaxie patří dříve zmíněná spirála (na obrázku 2.12), elipsa a prstenec.



Obrázek 2.12: Mapa galaxie zobrazuje prostor ovládaný různými civilizacemi. Velikost galaxie je rozdělena do pěti odlišných velikostí (Tiny, Small, Medium, Large, Huge).

Hvězdy jsou u spirály rozmístěny po délce všech jejích ramen. Mezi rameny je prázdno, a hráčova flotila se může přesouvat mezi nimi pomocí červí díry nebo `Warp` pohonu vyvinutém na požadované úrovni. Galaxie tvaru elipsy má hvězdy rozmístěné do elipsovitého tvaru, které tvoří rovnoměrné zastoupení po celé galaxii. Galaxie tvaru prstenec má hvězdy umístěné ve vnější kružnici a veškerý pohyb probíhá většinou po jejím obvodu. Všechny tyto vlastnosti galaxie může hráč upravovat.

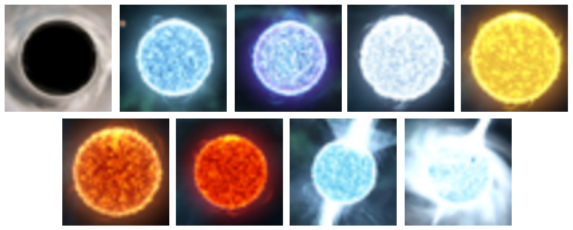
Poté, co hráč nastaví všechny potřebné vlastnosti generátoru, začne proces vytváření hvězd galaxie. Hvězdě je při svém vzniku přiřazena jedna z několika tříd[7], která určuje její specifika. Třídy jsou označené písmeny A, B, F, G, K, M. Výjimečně může být vytvořena speciální varianta hvězdného systému, jakými je například černá díra, pulzar nebo neutronová hvězda. Třídy hvězd jsou blíže popsané v tabulce 2.1 a jejími ikonami na obrázku 2.13. Každý systém, který má přiřazenou svoji třídu, disponuje množinou pravidel, podle kterých je hvězdná soustava vygenerována. Například je nepravděpodobné, že by byla objevena obydlená planeta poblíž černé díry. Opět jsou všechny tyto vlastnosti plně modifikovatelné, případně je zde možnost přidávat nebo odebírat přiřazované třídy hvězd.

[7] https://stellaris.paradoxwikis.com/Map

Mimo hvězdy generátor vytváří i jiné vesmírné útvary, mezi které patří například nebuly[8]. Ty jsou viditelné na herní mapě a často sdružují skupinu hvězdných systémů, na kterých se mohou vyskytovat vzácnější druhy zdrojů. Po skončení práce generátoru přichází na řadu ostatní generátory (například generátor civilizací), které vytváří ostatní prvky vesmírného simulátoru pro zachování vyvážené hratelnosti.

Černá díra	Útvar vzniklý kolapsem obrovské hvězdy na konci jejího života. Díky silné gravitaci přitahuje ostatní vesmírné objekty do svého nitra, odkud nic nevyjde zpět.
A	Relativně mladá hvězda bílé nebo modro-bílé barvy. Hvězda je velká a otáčí se velice rychle, ale postupem času se může vyvinout v pomalejšího, chladnějšího červeného obra.
B	Velmi jasná hvězda modré barvy. Dobře viditelná pouhým okem.
F	Docela velká hvězda, která je přirovnávaná ke žluto-bílému trpaslíkovi. Přesto, že vyzařuje značné množství UV radiace, povrch okolních planet je vhodný pro vznik života.
G	Označována jako žlutý trpaslík. Disponuje dobrou zásobou vodíku a podmínky okolních planet jsou vhodné pro život.
K	Také zvaná jako oranžový trpaslík. Stabilní a její cyklus života je dostatečný pro rozvoj evoluce života.
M	Nejčastější hvězda ve vesmíru, označovaná jako červený trpaslík. Méně jasná a tak obtížněji objevitelná. Absence UV záření znemožňuje vývoj většiny forem života.
Neutronová hvězda	Vysoce hustá množina vesmírných pozůstatků, která vznikla rychlým kolapsem a explozí obrovské hvězdy v supernovu.
Pulzar	Vysoce magnetická neutronová hvězda vyzařuje paprsky elektromagnetické radiace. Paprsek lze vidět pouze tehdy, kdy směřuje přímo k pozorovateli a lze jej využít pro velice přesné měření času.

Tabulka 2.1: Charakteristika hvězdných tříd[9].



Obrázek 2.13: Shora z levé strany třídy černá díra, A, B, F, G, K, M, neutronová hvězda a pulzar.

[8] Mezihvězdné mračno prachových částic a plynů.
[9] Bližší informace lze najít na https://stellaris.paradoxwikis.com/Map

2.6 Příklady generátorů map – Strategie

Žánr strategických počítačových her byl oblíbený především v 90. letech a na přelomu tisíciletí. Tyto hry disponují často kampaněmi i sadou předem vytvořených map, které jsou navrhnuté s precizní vyvážeností herních prvků. Některé z těchto her podpořily svou znovuhratelnost zavedením generátoru herních map jako nedílnou součást hry. V následujících sekcích jsou uvedeny příklady generátorů map z několika počítačových her žánru strategie (resp. real-time strategie).

2.6.1 Heroes of Might and Magic 3

Počítačová hra z roku 1999 vyvinutá studiem New World Computing patří mezi tahové strategie, u kterých je herní mapa nejdůležitější prvek celé hratelnosti. Herní mapa je definovaná ve 2D a umísťuje hráče na zadané místo (náhodné v případě generátoru) a ten objevuje neprozkoumané části mapy, sbírá suroviny, zlepšuje dovednosti svých hrdinů a snaží si podmanit ostatní frakce. Kromě předem vytvořených map lze využít vestavěný generátor map. Tento generátor funguje na principu šablon[3], které přizpůsobují jeho činnost. Po spuštění generátoru se vybere jedna z dostupných šablon a podle ní se pak generuje. Hráč může tuto šablonu ovlivňovat přímo ze hry (jak lze vidět na obrázku 2.14) nastavením velikosti mapy (S,M,L,XL), obtížností nepřátel, množstvím vodní plochy nebo počtu oponentů a týmů. I tyto vlastnosti lze nastavit na náhodný výběr.

Obrázek 2.14: Nastavení vlastností generátoru map v Heroes of Might and Magic 3.

Šablony se nacházejí zabalené v balíku šablon, kterou hra používá, a pro kterou lze nastavit parametry uplatňující se na všechny šablony v ní obsažené. Mezi tyto vlastnosti patří povolené frakce a hrdinové, kteří se mohou ve hře vyskytnout. Poté už následuje nastavení jednotlivých šablon. Ty definují klíčové vlastnosti generované mapy a vytváří vyvážené mapy pro všechny zúčastněné hráče (včetně těch ovládaných umělou inteligencí). Mapa je rozdělena na zóny (odlišující se terénem), které jsou navzájem propojovány, aby mohli hráči cestovat napříč celou mapou. Vzhled zón, ve kterých byl vygenerován hráč, koresponduje se startovací frakcí, kterou hráč zastupuje. Propojení může probíhat nejenom kontaktem obou zón, ale také přístupem skrz podzemí[10] nebo jednosměrných/obousměrných portálů. Jednotlivé zóny mohou být označené příznakem "odpuzované", který požaduje po generátoru co největší odloučení zóny od všech ostatních zón označené stejným příznakem. Generátor se vždy řídí podle priorit jednotlivých nastavení šablony.

Mimo generování zón se na mapě umísťují suroviny, artefakty nebo budovy. Všechny tyto objekty mají definovanou hodnotu (jak hodnotný je objekt pro hráče), frekvenci (pravděpodobnost výběru objektu vzhledem k ostatním objektům se stejnou hodnotou), maximální počet objektu na mapě, maximální počet objektu v zóně a výčet terénů, ve kterých se může objekt vygenerovat.

2.6.2 Age of Empires 2

Real-time strategie z roku 1999 od studia Ensemble Studios je zasazena do období středověku. Hráč si může vybírat z většího počtu civilizací, které se liší svým vývojem v každém ze čtyř období. Civilizace začíná v období temna a získáváním surovin a zkoumáním nových technologií se dostává následně do období feudalismu, období hradů (pokročilý středověk) až po období imperialismu (srovnatelné s renesancí). Generování map probíhá na základě skriptů[5], které obsahují sadu instrukcí a parametrů umožňující vytvářet komplexní variace map. Skript je v textové podobě a je rozdělen na sedm sekcí, kde se každá sekce zaměřuje na jednu kategorii vlastností generované mapy. Tyto sekce jsou nastavení hráče, generování země, kopců, hor, terénu, mostů/cest, jednotek, budov a použitelných zdrojů. Některé z uvedených sekcí není nutné zahrnovat do skriptu.

Syntaxe skriptovacího jazyka zahrnuje podmínky, konstanty, funkce náhodnosti a komentáře. Zbytek jsou klíčová slova příslušná pro dané sekce kódu. V tabulce 2.3 jsou popsány všechny zmíněné sekce generování a některé z jejich možností.

Velikost map je odstupňována šesti typy, které se vztahují k referenční velikosti mapy 100x100 políček. V tabulce 2.2 lze vidět rozdíly mezi jednotlivými typy. Podobný systém generování map je také přítomný v počítačové hře Age of Mythology od stejného tvůrce.

Typ	Rozměry mapy	Celkový počet políček	Poměr velikosti k referenční mapě
Tiny	120x120	14400	1.4
Small	144x144	20736	2.1
Medium	168x168	28224	2.8
Large	200x200	40000	4.0
Huge	220x220	48400	4.8
Gigantic	240x240	57600	5.8

Tabulka 2.2: Typy velikostí generovaných map v Age of Empires 2.

[10] Sekundární mapa, jejíž vzhled připomíná systém jeskyní

```
<LAND_GENERATION>
base_terrain WATER
create_player_lands
{
        terrain_type GROUND  /* grass or snow, as defined above */
        land_percent 60  /* 60% is quite high, but will be divided among players */
        base_size 10  /* player lands are wide enough */
        zone 1  /* all player lands are zone 1, they can touch each other */
        other_zone_avoidance_distance 5  /* but stay 5 tiles away from zone 2 */
}
create_land  /* the "relic isle" :) */
{
        terrain_type GROUND
        number_of_tiles 1000  /* fixed size */
        left_border 30  right_border 30  /* keep this isle near the center of the map */
        top_border 30  bottom_border 30
        clumping_factor 15  /* roundish */
        zone 2
        other_zone_avoidance_distance 5
        land_id 111
}
```

Obrázek 2.15: Část skriptu definující způsob generování země. Příkaz create_player_lands definuje, jak bude vypadat generovaná země pro každého hráče na mapě.

Nastavení hráče	Specifikuje umístění hráče, seskupování do týmů a vzdálenost mezi členy týmu.
Generování země	Zde se generuje území (především ve startující oblasti hráčů). Každý hráč může začínat s jedním nebo více startovními městy. Dále lze například specifikovat, jaký typ terénu a kolik procent zabírá na mapě.
Generování kopců	Kopce jsou útvary dostatečně nízké pro jejich průchod jednotkami hráče. Zde lze specifikovat celkový počet vygenerovaných kopců a vzdálenost mezi nimi.
Generování hor	Hory jsou neprůchozí a vytváří na základě specifikace obecné statistky jejich zastoupení na mapě. Hodnoty zahrnují minimální/-maximální počet a délku hor.
Generování terénu	Vhodné pro případ generování rozmanitějšího terénu. Zde záleží na pořadí definovaných ploch terénu. Plochy mohou mít specifikovaný výškový limit, procentuální zastoupení při generování země.
Generování propojení	Stará se o propojení jednotlivých zemí a zajišťuje jejich dostupnost pro pozemní přístup hráčských jednotek. Specifikace může obsahovat, jakým způsobem bude upravován typ terénu pro zajištění spojení mezi zeměmi.
Generování objektů	Do této kategorie patří vytváření jednotek, budov, zdrojů a dekorací mapy. Každému definovanému objektu lze definovat jeho počet, příslušnost k hráči, minimální/maximální vzdálenost od hráče nebo umístění do konkrétní země.

Tabulka 2.3: Popis jednotlivých sekcí generátoru map v Age of Empires 2.

Kapitola 3

Návrh implementace

V této kapitole je blíže specifikován návrh generátoru galaxie, jeho vlastností a způsob, jakým lze rozšířit. Dále pak zahrnutí těchto vlastností v návrhu implementace ukázkové aplikace.

3.1 Generátor galaxie

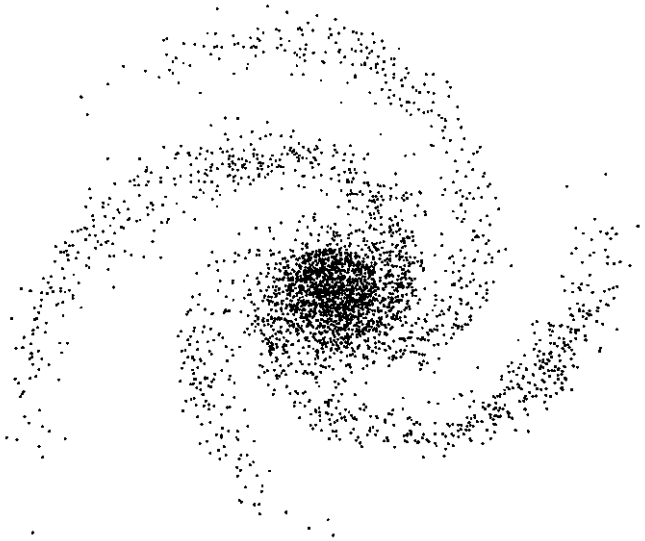
Generátor je realizován v podobě `C/C++` knihovny. Po importování knihovny do projektu se uživateli zpřístupní aplikační rozhraní, které umožňuje použití vestavěného generátoru galaxie. Tento generátor pokrývá všechny základní tvary galaxií a generování s využitím `heat mapy` (bude vysvětleno později).

Knihovna je postavená na návrhovém vzoru `factory`, a proto se uživatel nemusí omezovat pouze na možnosti vestavěného generátoru. V případě odlišných požadavků na výstup generátoru lze vytvořit vlastní generátor. Ten musí dědit ze základní třídy generátoru a implementovat jeho rozhraní. Zde lze provést několik možností, jakým vlastní generátor vystavět. Prvním případem je využití vestavěného generátoru na varianty generování, jejichž výstup uživateli vyhovuje, a vytvoření vlastních metod generátoru pro ostatní případy. V druhém případě jde o druhotné zpracování výstupu vestavěného generátoru, kdy mohou být přidány další informace o generovaných slunečních soustavách.

Do předem definovaných tvarů implementovaných vestavěným generátorem patří koule, spirála, mřížka a shluk. Generování těchto tvarů je ovládáno skrz jednotlivá nastavení, které jsou mezi sebou odlišné a lze je přenastavit. V případě rozšiřování funkčnosti generátoru lze jednotlivá nastavení rozšiřovat o vlastní data.

Nastavení koule definuje její velikost, hustotu vygenerovaných slunečních soustav určenou střední hodnotou a její odchylkou a zvolením hodnot možných odchylek pro výslednou pozici soustavy na všech třech osách. Shluk je tvořen množinou jiného tvaru (například koule), kde každý její prvek může být odchýlen na všech třech osách a počet prvků množiny je definován střední hodnotou a její odchylkou. Dalším tvarem je mřížka, která je definovaná svou velikostí a rozmezím slunečních soustav. Posledním tvarem je spirála. Ta kombinuje předchozí tvary jako je shluk a koule. Spirála se skládá ze dvou částí doplněných vygenerovanými hvězdami v pozadí. První část spirály se nazývá jádro nebo také střed a je vytvářeno tvarem shluk, jehož základem je koule. Zde platí pro shluk stejné atributy, které byly popsány výše. Druhou částí jsou ramena spirály. Ramena jsou tvořena množinou shluků, kde každá vygenerovaná sluneční soustava je posunuta od středu spirály a zatočena okolo jádra. Úhel zatočení se odvíjí podle počtu vygenerovaných ramen. Vygenerovanou

spirálu se čtyřmi rameny lze vidět na obrázku 3.1. Pro lepší viditelnost je vygenerovaná galaxie vykreslena v debug módu.



Obrázek 3.1: Vygenerovaná galaxie tvaru spirála. Černé tečky označují vygenerované sluneční soustavy.

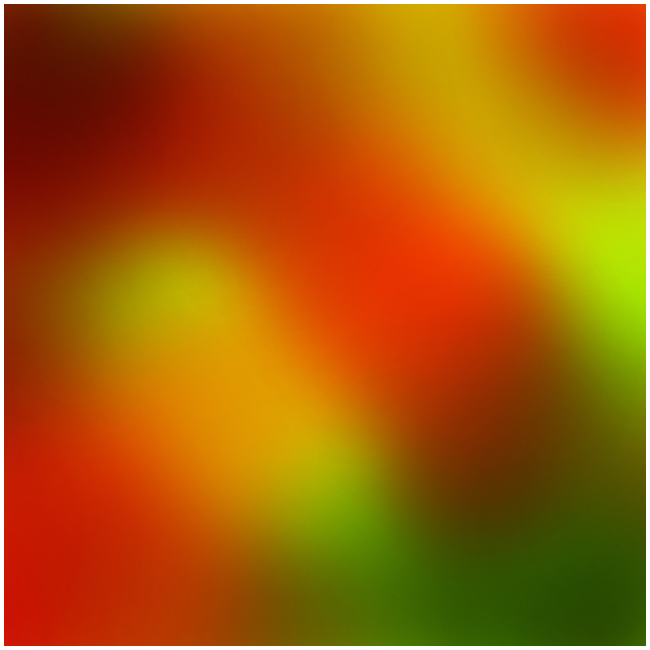
Další možností generátoru je generování podle tzv. `heat mapy`, což je 2D textura definující vlastnosti generování slunečních soustav v galaxii. Každý pixel mapy určuje několik na sobě nezávislých informací. Textura musí obsahovat všechny kanály ve formátu RGBA, kde každý kanál definuje jednu vlastnost generátoru. Hlavním parametrem mapy je míra pravděpodobnosti vygenerování sluneční soustavy v daném místě (pixelu heat mapy). Tuto metriku pokrývá kanál zelené barvy a její rozsah je definovaný minimální hodnotou 0 (nejnižší pravděpodobnost) a maximální hodnotou 255 (nejvyšší pravděpodobnost).

Kanál červené barvy definuje střední hodnotu generované teploty sluneční soustavy. Hodnota 0 reprezentuje velmi nízké teploty (znázorněné modrou barvou sluneční soustavy) a hodnota 255 velmi vysoké teploty (znázorněné červenou barvou sluneční soustavy). Teplota sluneční soustavy může ovlivňovat větší množství doprovodných efektů, jakými je již zmíněná vnější barva sluneční soustavy, vhodnost pro vznik života na planetách okolo sluneční soustavy nebo složení okolních vesmírných entit.

Kanály modré barvy a alfy jsou v této práci nevyužity a jsou zde ponechány jako prostor pro budoucí rozšíření k již existujícím vlastnostem. Pro bližší znázornění lze vidět na obrázku 3.2 ukázkovou heat mapu se zobrazenými všemi kanály a s použitím filtru červeného a zeleného kanálu na obrázcích 3.3 a 3.4.

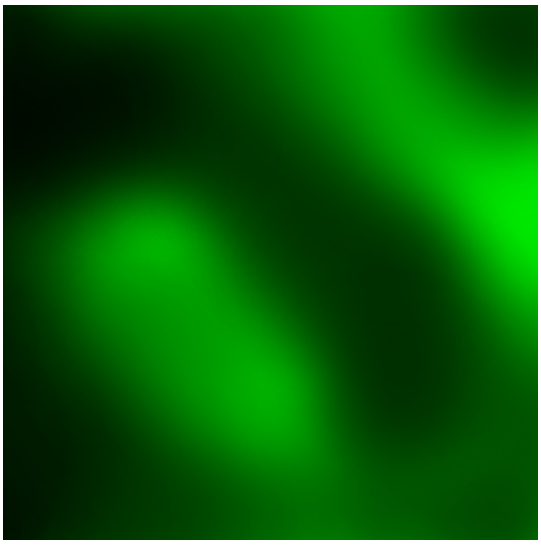
Po dokončení generování galaxie dostává uživatel na výstupu vygenerovanou množinu slunečních soustav se svými přiřazenými vlastnostmi. Data jsou naformátovaná pro přímé využití v grafických aplikacích. Mimo vygenerovaná data jsou uživateli k dispozici doplňující informace, získané při generování galaxie. Mezi tyto informace patří vypočtené rozměry galaxie ve 3D prostoru nebo například počet vygenerovaných množin při generování tvaru shluk.

Možné rozšíření knihovny by mohlo být použití několika heat map pro významné rozšíření generovaných vlastností, případně vlastnosti generované složením informací z většího počtu barevných kanálů.



Obrázek 3.2: Heat mapa obsahující červený a zelený kanál.

Obrázek 3.3: Červený kanál specifikuje míru pravděpodobnosti teploty sluneční soustavy.



Obrázek 3.4: Zelený kanál určuje míru pravděpodobnosti vygenerování sluneční soustavy.

3.2 Ukázková aplikace

Aplikace interaktivně zobrazuje možnosti knihovny generátoru galaxie. Uživatel si tak může vyzkoušet všechny dostupné varianty generování v grafickém uživatelském rozhraní bez nutnosti jakéhokoliv zásahu do kódu aplikace. Toto rozhraní je vytvořené pomocí frameworku `Qt` a jeho modulu `Quick`.

Rozhraní v Qt Quick je definované v jazyce QML (Qt Meta Language), který vychází ze syntaxe jazyka JSON. Grafické prvky jsou vkládány do stromové hierarchie a lze tak jednoduše vytvářet a skládat sofistikovanější komponenty. Qt Quick poskytuje širokou knihovnu standardních komponent, které lze snadno upravovat a rozšiřovat. Aby bylo možné dosáhnout interaktivnosti jednotlivých komponent, QML podporuje syntaxi skriptovacího jazyka Javascript. Kód v tomto jazyku by měl provádět pouze jednodušší operace, aby nedocházelo k zamrzání grafického rozhraní. Každá komponenta může být označena atributem id, pomocí kterého lze přistupovat k vnitřním vlastnostem (tzv. properties) komponenty.

Důležitou vlastností Qt Quick je možnost propojení kódu grafického uživatelského rozhraní v QML s kódem napsaným v jazyce C/C++ aplikace, ve kterém je toto rozhraní spouštěno. Pro toto propojení lze vytvořit QML komponentu v jazyce C++. Tato komponenta se vytváří stejně jako běžná třída C++. Pro její integraci v systému Qt je nutné, aby tato třída dědila z třídy QObject a definovala vevnitř makro Q_OBJECT. Pro komunikaci mezi oběma jazyky lze využít několik možností. Qt framework je známý svým systémem slotů a signálů, které fungují na principu návrhového vzoru observer, kde slot funguje jako ten, kdo vyvolává aktualizace a signál místem přijímaní těchto aktualizací. Tento systém se používá také při definici vlastností komponenty, která po aktualizaci stávající hodnoty vyvolá příslušný signál o změně property. Všechny zmíněné vlastnosti jsou použity v aplikaci.

Aplikace dále využívá frameworku GPUEngine zapouzdřující funkce knihovny OpenGL pro vykreslování 3D grafiky. Rozhraní OpenGL bylo vybráno z důvodu snadného propojení s QML a tak možností propojit renderovanou grafiku OpenGL s grafickým uživatelským rozhraním Qt Quick. Framework nabízí funkce, které v mnoha případech zjednodušují použití standardního rozhraní OpenGL (například zapouzdření shader programů nebo obsluhu bufferů) a ponechává všechny dosavadní. Propojení obou frameworků probíhá na základě již zmíněných signálů a slotů, díky kterým lze definovat, v jakém pořadí bude aplikace vykreslovat jednotlivé vrstvy grafických prvků.

Aplikace po spuštění zobrazí uživateli hlavní menu, ve kterém si může zvolit preferovanou metodu generování galaxie. Nastavení poté koresponduje s aktuálně nastavenou metodou a její variantou. Mezi metodami jsou k dispozici obě varianty generátoru, generování přednastavených tvarů a generování řízené heat mapou. Výběr metody přednastaveného tvaru doprovází nabídka volby tvaru v doprovodném výběru. Nastavení každého tvaru má přednastavené hodnoty, se kterými lze okamžitě vygenerovat ukázkový vzorek galaxie. V případě, že by uživatel nechtěl přijít o své nastavení generátoru nebo naopak chtěl své nastavení obnovit z předchozí relace, jsou uživateli k dispozici možnosti uložení (resp. načtení) nastavení s využitím lokálního úložiště. Uživatel má také možnost se vrátit zpět k přednastaveným hodnotám nastavení generátoru. Při volbě přednastaveného tvaru se uživateli pro lepší představu v náhledu zobrazí snímek demonstrující příklad vygenerované galaxie. Tento snímek je statický a nezohledňuje aktuální nastavení generátoru.

Pokud uživatel zvolí metodu generování podle heat mapy, zpřístupní se u náhledu volba pro načtení heat mapy z úložiště. Po jejím načtení se heat mapa zobrazí v náhledu a zpřístupní se i druhá volba editace heat mapy. Pro snadnou úpravu heat mapy je k dispozici jednoduchý editor, který poskytuje nástroje k filtraci obrazu na požadované kanály a následně i úpravu hodnot heat mapy. Náhled zobrazuje celou mapu s možností přiblížení a oddálení pro přesnější editaci jednotlivých pixelů mapy. Pro snažší úpravu hodnot lze měnit mezi dvěma druhy štětců (čtverec a kruh) a velikost štětce. Pro dokončení změn si musí uživatel zvolit z možností uložení provedených změn nebo jejich zrušení.

Stejně jako u předdefinovaných tvarů, také u metody generování podle heat mapy je k dispozici nastavení generátoru. Mezi vlastnosti patří velikost generované galaxie, kterou

bude `heat mapa` pokrývat. Čím větší rozměry galaxie jsou, tím vzniká větší odstup mezi jednotlivými slunečními soustavami. Dále pak výškovou odchylku generovaných soustav. Při nastavení hodnoty 0 bude vygenerovaná galaxie placatá a tento stav je tak vhodný pro analýzu kvality vygenerované galaxie se zvoleným nastavením a použitou `heat mapou`. Dalšími vlastnostmi je minimální vzdálenost generovaných slunečních soustav, teplotní rozsah galaxie a korespondující teplotní odchylka slunečních soustav vzhledem k definované hodnotě v `heat mapě`.

Po nastavení všech nezbytných vlastností v hlavním menu aplikace může uživatel spustit generování galaxie. Po dokončení generování se přepne aplikace do 3D režimu v pozici, kdy kamera vidí celou vygenerovanou galaxii. Grafické uživatelské rozhraní v tomto případě nabízí informace o vygenerované galaxii, jako je v případě metody generování podle přednastaveného tvaru její název, počet vygenerovaných slunečních soustav a pozice kamery v galaxii. Ve 3D pohledu se může uživatel volně pohybovat v prostoru vygenerované galaxie z pohledu první osoby, podobně jako v případě `fly manipulátoru`. Na galaxii se dá tak dívat pohledem z ptačí perspektivy, i bližším zkoumáním vygenerovaných slunečních soustav. Pokud by chtěl uživatel právě vygenerovanou galaxii použít ve své aplikaci, je možné jej uložit z menu nabídky. Mimo tuto možnost lze také načíst dříve vytvořenou galaxii zpět do aplikace. Dále je možný návrat do 3D pohledu nebo návrat do hlavního menu aplikace. Galaxii lze také přímo načíst z hlavního menu aplikace bez nutnosti přechodu do 3D pohledu spuštěním generování.

Kapitola 4

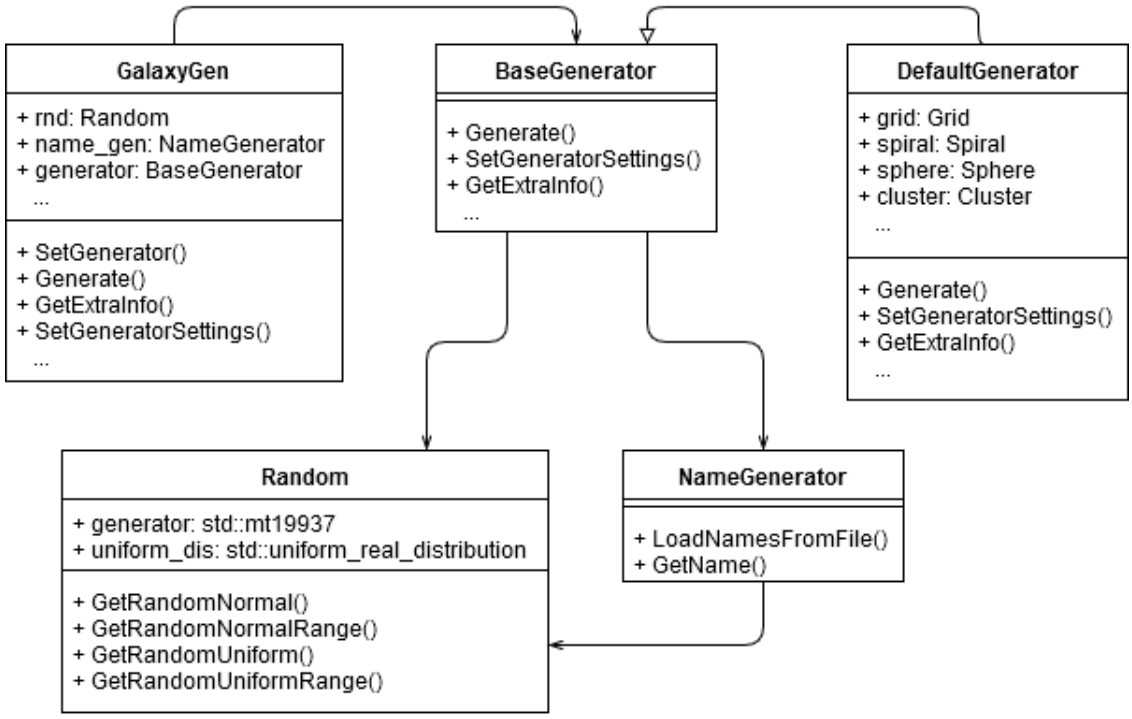
Implementace

Kapitola popisuje, jakým způsobem byla naprogramována knihovna generátoru galaxie a ukázková aplikace včetně jejího propojení s jednotlivými knihovnami. Oba projekty používají systém `CMake` pro vytvoření a instalaci projektu. Při konfiguraci knihovny generátoru galaxie je nutné mít v systému nainstalovanou knihovnu `GLM`[1], která poskytuje vysokou kompatibilitu s rozhraním `OpenGL` a funkcionalitou jazyka `GLSL`. `GLM` lze také použít pro práci s maticemi, kvaterniony a dalšími matematickými strukturami často používanými v počítačové grafice.Konfigurace ukázkové aplikace pak vyžaduje nastavení cest ke knihovnám `Qt`, `GPUEngine` a `GalagyGen`. Po úspěšné konfiguraci a vytvoření projektu je možné kompilovat oba projekty v režimu `Debug` nebo `Release`.

4.1 Generátor galaxie

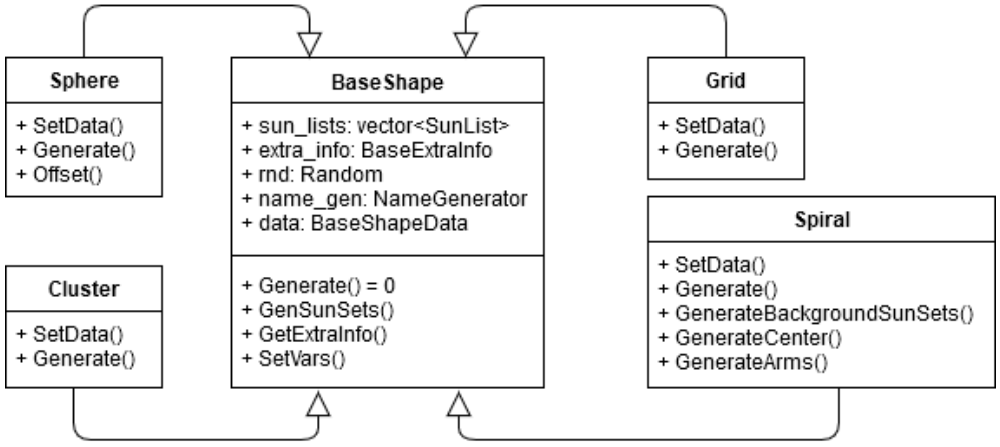
Jak již bylo zmíněno v kapitole 3.1, knihovna je implementována v jazyce `C/C++`. Vstupním bodem generátoru je třída `GalaxyGen`, pomocí které se spouští generování galaxie. Třída poskytuje dále metody pro získání extra informací (`GetExtraInfo`) z procesu generování a nastavení jednotlivých částí generátoru. Při volání metody `Generate` lze nahradit vestavěný generátor vlastním generátorem, který implementuje rozhraní abstraktní třídy `BaseGenerator`. Vestavěný generátor je definovaný třídou `DefaultGenerator`, která také dědí ze základní třídy generátoru. Před spuštěním metody `Generate` je nutné provést načtení vlastností předdefinovaných tvarů galaxie metodou `SetGeneratorSettings` a vložením specifické datové struktury dědící ze základní datové struktury `BaseData`. Datová struktura je detekována podle typu a nastaví tak danou část generátoru. Pokud by uživatel rozšiřoval sadu předdefinovaných tvarů galaxie, je nutné vytvořit novou datovou strukturu dědící ze struktury `BaseData`. Zároveň je potřeba provést rozšíření hodnot enum typu `galaxy_type`.

[1] https://glm.g-truc.net/



Obrázek 4.1: Diagram tříd knihovny generátoru galaxie.

Každý předdefinovaný tvar galaxie implementuje samostatná třída, implementující metody z rodičovské abstraktní třídy `BaseShape`. Tato třída slouží jako standardní rozhraní definující metody jako `Generate` nebo `GetSunList`. Třída si uchovává seznam vygenerovaných slunečních soustav reprezentované třídou `SunList` a v případě exportování pak v datové struktuře `SunListData`. Všechny předdefinované tvary galaxii jsou implementovány třídami `Sphere`, `Grid`, `Cluster` a `Spiral`.



Obrázek 4.2: Diagram tříd předdefinovaných tvarů galaxie dědící ze základní třídy tvaru BaseShape.

Při generování podle heat mapy je nutné využít datovou strukturu `HeatmapGenData`, která zapouzdřuje nejenom nastavení generátoru, ale také strukturu heat mapy `Heatmap`. Tato struktura poskytuje rozhraní pro snadné načtení vstupních dat a jejich přímé využití při generování.

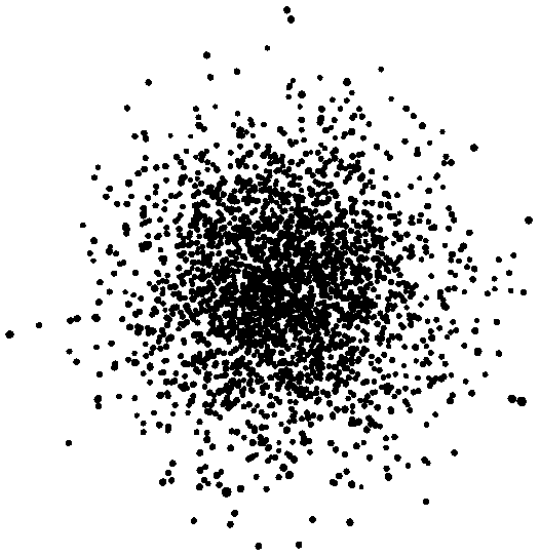
Následující podsekce popisují způsob implementace generování galaxie pomocí jednotlivých předdefinovaných tvarů a metody generování podle heat mapy. U všech variant je ke každé vygenerované sluneční soustavě přiřazeno její jméno z generátoru jmen (třída `NameGenerator`) a počítány rozměry ohraničující krychle, které mohou sloužit k přesnému umístění kamery. Nedílnou součástí procedurálních generátorů jsou funkce náhodnosti. Třída `Random` poskytuje základní sadu funkcí náhodnosti implementující uniformní a normální rozložení. Funkce normálního rozložení využívají standardní třídy `std::mt19937`, která implementuje algoritmus Mersenne Twister (popsaný v sekci 2.2.3).

4.1.1 Koule

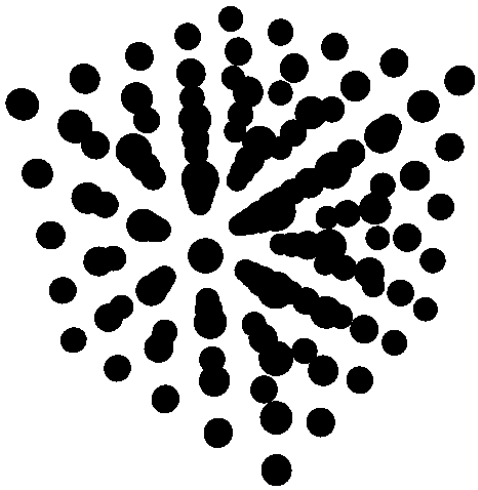
Jednodušší z předdefinovaných tvarů. Nejdříve se určí normální distribucí hustota a maximální počet slunečních soustav v galaxii. Poté se náhodně vybere počet generovaných slunečních soustav mezi nulou a vypočteným maximálním počtem. Pro každou vygenerovanou sluneční soustavu se vygeneruje její pozice definovaná odchylkou na všech třech osách od počáteční nulové pozice.

4.1.2 Mřížka

Tento tvar patří mezi nejjednodušší ze všech uvedených. Generování probíhá v rovnoměrné mřížce, definované velikostí galaxie a rozestupy mezi jednotlivými slunečními soustavami. Zde se neaplikuje žádný prvek náhodnosti, a proto je vygenerovaná galaxie se stejnými parametry vždy stejná.



Obrázek 4.3: Příklad generované galaxie tvaru koule.



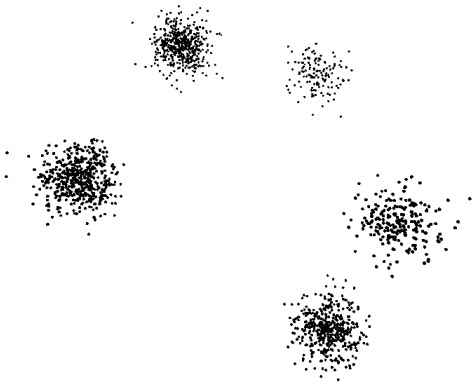
Obrázek 4.4: Příklad generované galaxie tvaru mřížka.

4.1.3 Shluk

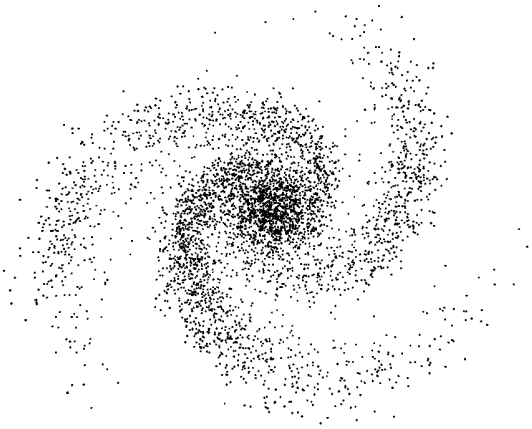
Shluk je implementován jako iterace generování nastaveného základního tvaru, kterým mohou být všechny předdefinované tvary. Definovanou střední hodnotou a odchylkou se pomocí normální distribuce určí počet generovaných tvarů. V každé iteraci proběhne vygenerování slunečních soustav základním tvarem a následně jejich posunutí do prostoru od nulové pozice galaxie s definovanou odchylkou na všech osách.

4.1.4 Spirála

Nejsložitějším tvarem je spirála. Generování zde probíhá ve třech fázích. První je generování jádra spirály, kde se využívá tvar shluku s nastaveným základním tvarem koule. Velikost koule lze škálovat vzhledem k celkové velikosti spirály. Stejně tak hustotu celého shluku. Výstupní sluneční soustavy jsou otáčeny kolem středu galaxie, kde míra zatočení odpovídá vzdálenosti sluneční soustavy od středu galaxie. Dalším krokem je generování ramen spirály, jejichž počet se určí náhodně v definovaném rozsahu. Zatočení ramen pak odpovídá poměru rovnoměrného rozložení jejich celkového počtu na jednotkové kružnici. Jednotlivá ramena jsou generována jako množiny shluků s nastaveným základním tvarem koule. Počet shluků ramene je definován normálním rozložením se závislostí na maximálním počtu shluků galaxie. Koule shluku jsou po vygenerování posunuty od středu shluku a zatočeny stejně jako v případě jádra spirály. V poslední fázi se generují okolní sluneční soustavy, které dotváří pozadí spirály. Toto pozadí je generováno tvarem koule, který pokrývá celou velikost prostoru spirály a výstupní sluneční soustavy se nezapočítávají do rozměru ohraničující krychle.



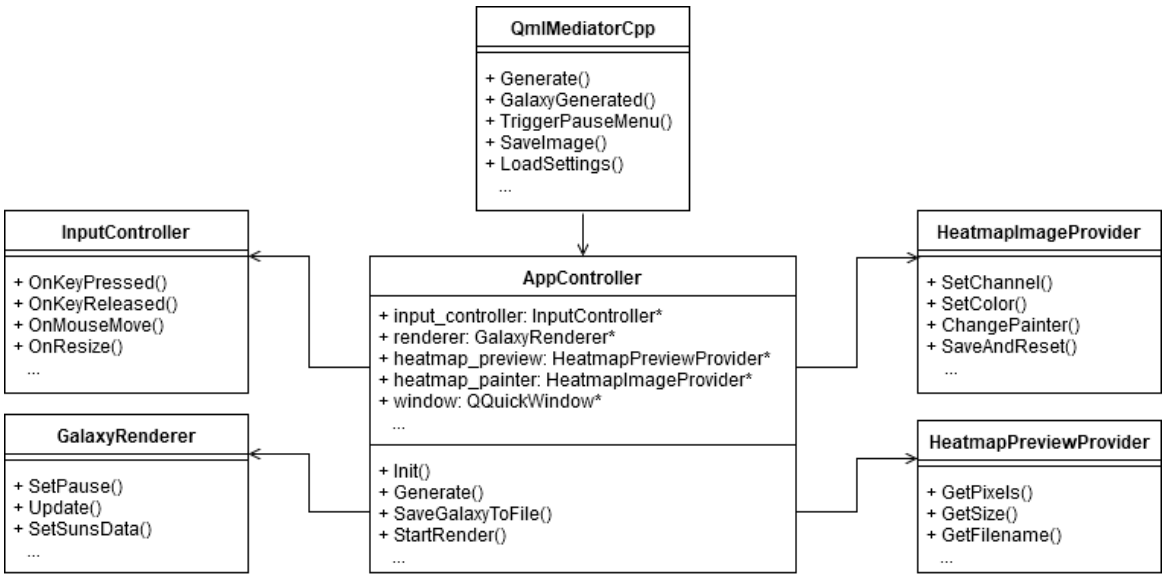
Obrázek 4.5: Příklad generované galaxie tvaru shluk se základem tvaru koule.



Obrázek 4.6: Příklad generované galaxie tvaru spirála s třemi rameny.

4.2 Ukázková aplikace

Aplikace je napsaná v jazyce `C/C++` v kombinaci s frameworkem `Qt5`. Pro spuštění aplikace je požadovaná minimální verze OpenGL ve verzi 4.3. Pro zobrazení příkazové řádky aplikace je nutné vložit parametr spuštění "-showcmd". Hlavní třídou aplikace je `AppController`, která se stará o přijímání a zasílání signálů grafickému uživatelskému rozhraní. Při své inicializaci je získán hlavní objekt okna definovaného v `qml` (třída `QQuickWindow`). Tento objekt je napojen na komponentu vykreslování (třída `GalaxyRenderer`), která registruje její signály pro vlastní inicializaci a vykreslování grafiky ve správném pořadí.



Obrázek 4.7: Diagram tříd demonstrační aplikace a jejich vzájemné propojení.

Dále se na `QQuickWindow` registruje komponenta `InputController` pro odchytávání vstupních událostí v okně aplikace. `Appcontroller` používá `QQuickWindow` pro ovládání pozice a viditelnosti kurzoru myši. To je potřebné v situaci, kdy uživatel přejde do 3D pohledu aplikace (podrobně popsaný v následujících sekcích) a je nutné udržet kurzor uvnitř okna aplikace.

`InputController` zpracovává převážně signály spojené s vstupními signály klávesnice a myši. Zpracovaná data pak rozesílá skrz vlastní signály, na které se `AppController` napojuje a některé dále přeposílá ke zpracování komponentě `GalaxyRenderer`, která ovládá pohyb kamery ve 3D pohledu aplikace. V tabulce 4.1 jsou rozepsány funkce jednotlivých kláves.

Klávesa	Funkce
W	Pohyb dopředu
A	Pohyb doleva
S	Pohyb dozadu
D	Pohyb doprava
Mezerník	Pohyb dolů
C	Pohyb nahoru
R	Resetování pozice kamera na (0,0,0)
M	Přepínání mezi debug a bežným módem
B	Zvětšení modelu slunce
N	Zmenšení modelu slunce
Levý shift	Zrychlení pohybu
Levý control	Odemknutí/Zobrazení kurzoru myši
Escape	Zobrazení/Skrytí nabídky menu
/	Zrychlení pohybu kamery
*	Zpomalení pohybu kamery
+	Zvýšení senzitivity rotace kamery
-	Snížení senzitivity rotace kamery

Tabulka 4.1: Funkcionalita kláves v 3D pohledu aplikace.

4.2.1 Grafické uživatelské rozhraní

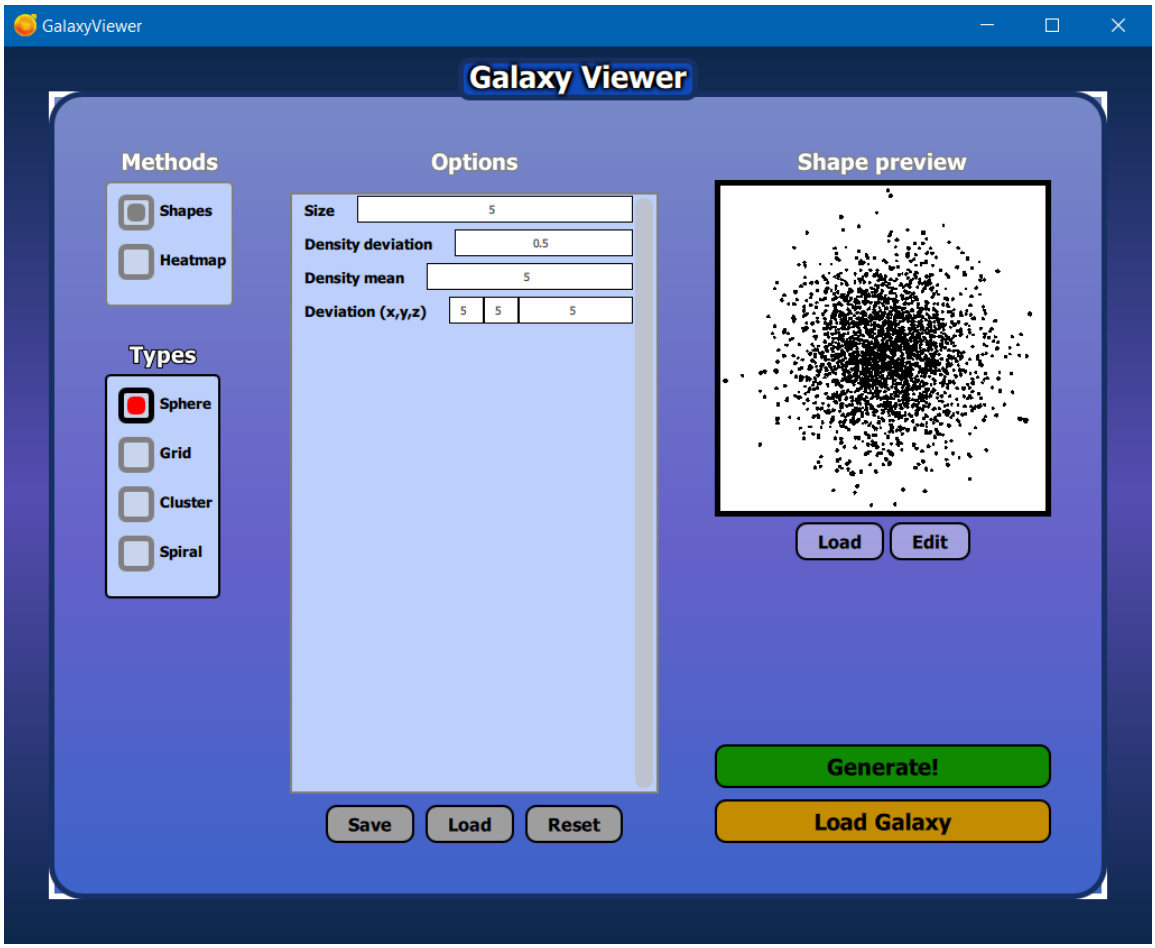
Po spuštění aplikace se nejdříve provádí inicializace grafického uživatelského rozhraní. K tomu je zapotřebí registrace třídy `QmlMediatorCpp` jako komponentu použitelnou v `qml`. Jak již bylo zmíněno v kapitole 3.2, tato komponenta slouží jako prostředník mezi grafickým uživatelským prostředím a aplikací. Komunikace grafického uživatelského rozhraní a aplikace s touto komponentou pak probíhá skrz napojení jejich signálů na sloty komponenty. Opačným směrem komunikuje `QmlMediatorCpp` přes volání funkcí `AppControlleru` a hlavního objektu okna `QQuickWindow`. Dynamické změny v grafickém uživatelském rozhraní jsou dosaženy napojením `property` v `qml` obrazovkách na `property mapy` obsažené v `QmlMediatorCpp` a tyto mapy slouží jako datové kontejnery v obousměrné komunikaci.

`QQuickWindow` je získán při načtení hlavní komponenty `MainWindow`, která funguje jako kontejner pro všechny obrazovky grafického uživatelského rozhraní. Komponenta definuje velikost zobrazeného okna a funkce spojené s přepínáním jednotlivých obrazovek aplikace.

Po spuštění aplikace se jako první zobrazí hlavní menu. Tato obrazovka je implementována komponentou `MainMenu`. Menu spravuje nastavení generátoru galaxie a slouží jako výchozí bod aplikace, ze kterého je možné se dostat na obrazovku editoru `heat mapy` nebo 3D pohledu s vygenerovanou galaxií. Po zobrazení menu si musí uživatel zvolit variantu generování a její proměnné. Při výběru předdefinovaného tvaru se zobrazuje v náhledu ukázka příkladu generované galaxie (zajišťuje třída `HeatmapPreviewProvider`). Hodnoty proměnných jsou načítané při spuštění aplikace ze souboru `DefaultSettings.galaxysettings` nacházející se ve složce `resources/configs`. Konfigurační soubor je ve formátu `JSON` a lze jej snadno upravovat. Tlačítka `Save` a `Load` provádí uložení (resp. načtení) hodnot proměnných využívající lokálního úložiště. Tlačítko `Reset` pak provádí načtení hodnot proměnných z již uvedeného konfiguračního souboru `DefaultSettings.galaxysettings`. Při výběru varianty generování podle `heat mapy` se v menu zpřístupní tlačítko jejího načtení. Po jejím

načtení se zobrazí mapa v náhledu a zpřístupní se i druhé tlačítko pro její editaci a přepnutí se na patřičnou obrazovku aplikace.

Obrazovku editace heat mapy implementuje komponenta `HeatmapEditor`. Na levé straně se nachází panel nástrojů, který obsahuje minipanely jednotlivých kanálů `heat mapy`, velikost štětce, tvar štětce a tlačítka pro uložení nebo zrušení provedených změn na `heat mapě`. Na pravé straně je zobrazená načtená `heat mapa`, která má pro maximální přesnost kreslení vypnuté filtrování obrazu a nedochází tak ke vzájemné interpolaci barev sousedních pixelů (lze vidět na obrázku 4.9, kde je načtená `heat mapa` s nízkým rozlišením 10x10).



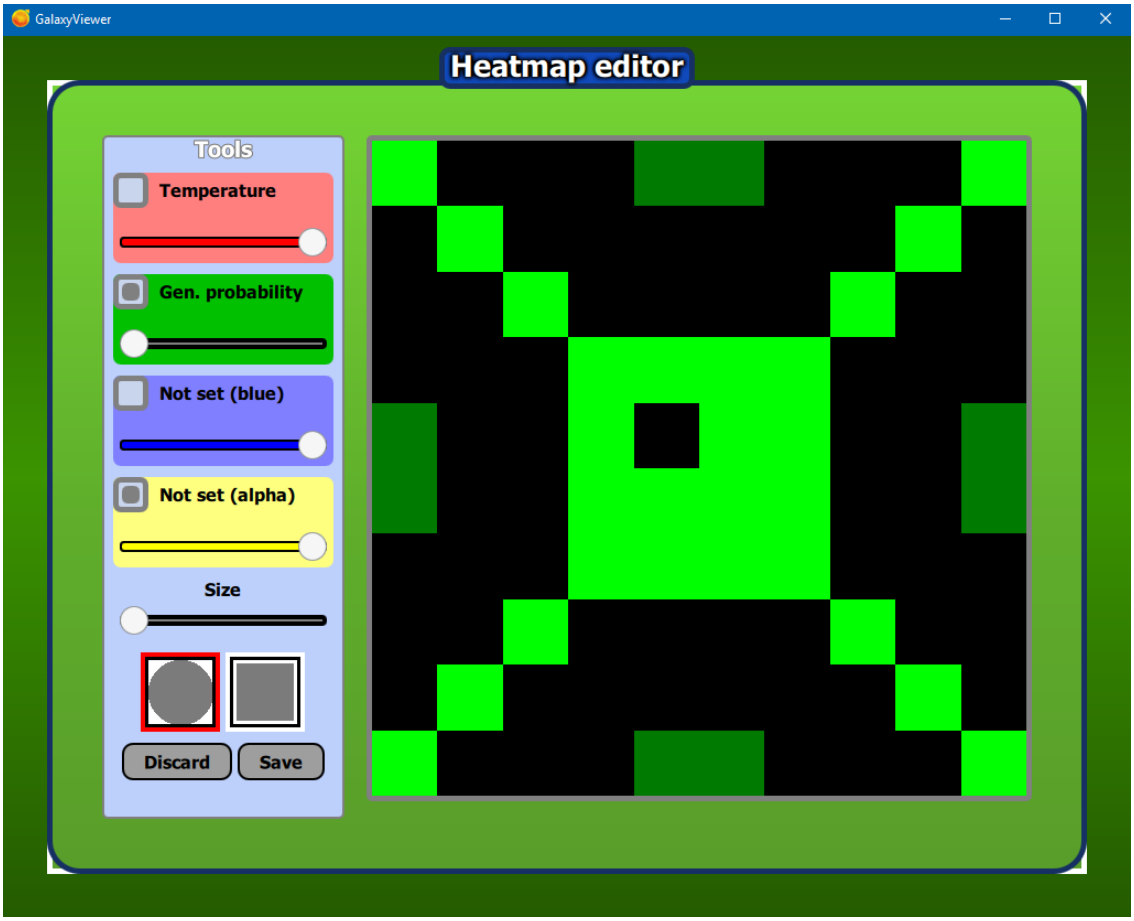
Obrázek 4.8: Hlavním menu aplikace. Před zahájením generování je nutné zvolit variantu požadované galaxie.

Kreslení do zobrazené `heat mapy` probíhá skrze signály ve třídě `HeatmapImageProvider`, která dědí z třídy `QQuickImageProvider` a používá se primárně pro načítání obrázků v systému `qml`. V případě této aplikace se tato třída také stará o implementaci ostatních funkcí editoru, jako je zapínání/vypínání jednotlivých kanálů `heat mapy`, kreslení náhledu štětce nebo uložení změn `heat mapy` na lokální úložiště. Po každé akci spojené s editací `heat mapy` se vyvolá aktualizace náhledu. Zobrazené varianty štětců implementují definované rozhraní základní třídy `PainterBase`. Dostupné štětce obsahují čtverec implementovaný třídou `PainterRect` a kruh implementovaný třídou `PainterCircle`.

Náhled `heat mapy` je vystavěný na `qml` komponentě `Flickable`, která zpřístupňuje vlastnosti prohlížení map známé z webových stránek. Kolečkem myši lze tak snadno přibližovat, oddalovat a v případě, kdy je mapa přiblížená, také posouvat zobrazenou pozici pohybem myši se stlačeným levým tlačítkem. Kreslení do mapy pak probíhá obdobně (stlačením pravého tlačítka namísto levého).

Každý minipanel definuje, zda je kanál aktuálně viditelný v náhledu, jakou vlastnost definuje a jaká hodnota se bude zapisovat při kreslení do `heat mapy`. Pokud je daný kanál vypnutý, dochází tak i k vypnutí kresleného kanálu. Posuvník definuje minimální hodnotu vlastnosti na jeho levé straně (hodnota 0) a maximální hodnotu na straně pravé (hodnota 1). Hodnoty posuvníku velikosti štětce se pohybují v rozsahu 1 až 100 pixelů v průměru.

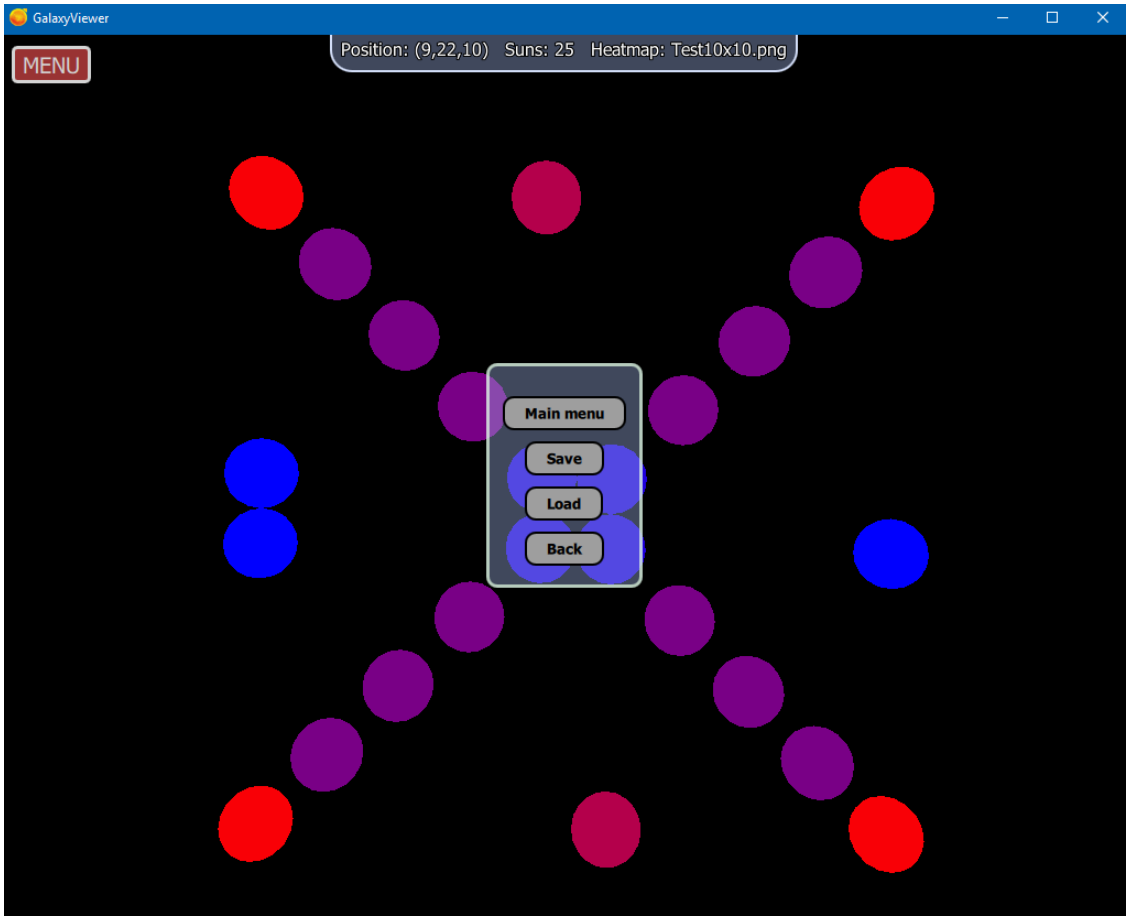
Po najetí kurzoru myši na mapu se aktivuje náhled nastaveného štětce, který zobrazuje výslednou barvu štětce zapnutých kanálů.



Obrázek 4.9: Editor heat mapy. Vlevo je panel nástrojů, ve kterém jsou vypnuty kanály červené a modré barvy znázorňující generovanou teplotu slunečních soustav a nespecifikované vlastnosti generátoru. Zobrazená heat mapa je v rozlišení 10x10.

Po spuštění generování galaxie nebo jejího načtení se zobrazí obrazovka 3D pohledu, ve kterém může uživatel pohybovat kamerou. Ve scéně se zobrazuje v horní části obrazovky panel a tlačítko `menu` aplikace. Obrazovku implementuje komponenta `SpaceView`. Horní panel zobrazuje informace o právě zobrazované galaxii. Jak již bylo dříve zmíněno, data jsou získávána s `property mapy` komponenty `QmlMediatorCpp`, která si udržuje veškerá data

o nastavení generátoru, informací o proběhlém generování nebo například informace o pozici a natočení kamery. Při zobrazení menu se pozastaví možnost pohybu kamery ve scéně a odblokuje se kurzor pro zvolení některé z následujících možností. Mezi nimi jsou uložení (resp. načtení) galaxie s využitím lokálního úložiště. Galaxie jsou ukládány ve formátu `JSON` a soubor obsahuje informace o vlastnostech vygenerovaných slunečních soustav (v aktuálním stavu informace o pravděpodobnosti generování sluneční soustavy a její teplotě) a dodatečné informace pro aplikaci ve 3D pohledu. Tato data lze jednoduše použít v kterékoliv jiné aplikaci. Ukázku této obrazovky lze vidět na obrázku 4.10.



Obrázek 4.10: 3D pohled aplikace. V horní části se nachází panel se základními informacemi o galaxii měnící se v závislosti na vybrané variantě galaxie. Uprostřed je zobrazené menu aplikace a na pozadí galaxie vygenerovaná z heat mapy z obrázku 4.9.

4.2.2 Vykreslování a pohyb kamery

Vykreslování galaxie ve 3D je řízeno třídou `GalaxyRenderer`, která dědí z třídy `QObject`, aby bylo možné provést propojení signálů spojené s vykreslováním hlavního okna `MainWindow` s vykreslováním této třídy. Jedním ze signálů je `openglContextCreated`, po jehož vyvolání dochází k aktivaci `OpenGL` kontextu poskytnuté oknem aplikace `QQuickWindow` a inicializace modulu `geGL` knihovny `GPUEngine`. Po inicializaci se zpřístupní funkce `OpenGL` a načítají se shadery, vytváří model koule znázorňující sluneční soustavu, nastavuje kamera pro volný pohyb a perspektiva zobrazení (třídy `FreeLookCamera` a `PerspectiveCamera` z mo-

dulu geUtil knihovny GPUEngine) a vytváří se OpenGL objekty (vertex buffer object a vertex array object) pro vykreslování.

Vykreslování probíhá po vyvolání signálu beforeRendering, který dává prostor pro vykreslování OpenGL před kreslením grafického uživatelského rozhraní Qt Quick. Aplikace vykresluje pouze tehdy, pokud se uživatel nachází ve 3D pohledu. V obsluze signálu se nachází aktualizace grafických dat (pouze po vygenerování nebo načtení některé z předchozích galaxií), aktualizace pozice a natočení kamery nebo nastavení OpenGL stavových proměnných. Kreslení probíhá v indirect režimu, ve kterém se příkaz kreslení nahraje na grafickou kartu a volání metody draw pak probíhá bez účasti procesoru. Další optimalizací vykreslování je instancování vytvořeného modelu koule namísto kreslení samostatných koulí pro každou sluneční soustavu. Do vertex shaderu jsou v rámci aktualizace dat načteny veškeré potřebné informace pro správné pozicování. To funguje tak, že každá instance je identifikována vestavěnou proměnnou gl_InstanceID, která slouží jako index do vektoru. Počet vygenerovaných slunečních soustav není omezený a může tak dosahovat i nad sto tisíc soustav.

4.3 CMake konfigurace

Jak již bylo zmíněno v úvodu kapitoly, knihovna i ukázková aplikace používají systém CMake (s požadovanou minimální verzí 3.8.0). Pro konfiguraci projektu lze použít libovolný z dostupných generátorů CMake, mezi které patří různé verze vývojového prostředí Visual studio, CodeBlocks nebo Sublime. Konfigurace obou částí projektu byla testována s generátorem Visual Studio 15 2017 Win64. Překlad lze pak provádět v konfiguracích Debug a Release.

4.3.1 Konfigurace knihovny

Konfigurace knihovny vyžaduje nastavení cesty k CMake modulu knihovny GLM. Dále pak musí uživatel nastavit cestu k externím souborům, které knihovna využívá (složka resources). Pokud se předpokládá využití knihovny v aplikaci, která bude přenositelná mezi systémy, pak je nutné zatrhnout možnost IS_STANDALONE. Po zvolení této možnosti se očekává přítomnost složky resources v místě spustitelného souboru aplikace.

V průběhu konfigurace projektu se vytváří také balíček, pomocí kterého lze snadno nastavit cestu ke knihovně v CMake konfiguraci jiného projektu. Generované soubory knihovny (.dll, .lib pro Windows a .so, .a pro Linux) jsou v Debug konfiguraci odlišeny příponou 'd'.

4.3.2 Konfigurace aplikace

Konfigurace aplikace vyžaduje nastavení cest k CMake modulům knihoven GLM, Qt5, GPUEngine a GalaxyGen. Podle vybraného generátoru projektu se navíc CMake pokusí nalézt složku s nezbytnými knihovnami pro zajištění přenositelnosti aplikace mezi systémy. Stejně jako v případě knihovny, si uživatel nastavuje cestu k externím souborům složky resources. I zde se zobrazuje možnost IS_STANDALONE, definující předpokládanou přítomnost složky resources v místě spustitelného souboru aplikace.

Konfigurace CMake nastavuje automatický překlad Qt objektů (definovaných makrem Q_OBJECT), automatické zpracování UI prvků napsaných v qml a automatické zpracování resources (specifikované souborem .qrc).

Kapitola 5

Závěr

V této práci bylo ukázáno, jakým způsobem byla navržena a implementována knihovna generátoru galaxie. Dále pak návrh a implementace ukázkové aplikace a její propojení s knihovnou generátoru galaxie. Knihovna je předurčena pro budoucí použití v navazujících nezávislých projektech nebo v projektech rozšiřujících její funkcionalitu. Praktické využití knihovny je možné sledovat v ukázkové aplikaci, která slouží jako interaktivní ukázka klíčových vlastností knihovny galaxie generátoru a zároveň také jako inspirace pro navazující projekty, které by chtěly implementovat aplikaci s využitím knihoven `Qt` a `GPUEngine`.

Knihovna je navržená tak, aby byla snadno modifikovatelná a rozšiřitelná s možným nahrazením jakýchkoliv částí implementace vlastním řešením. Do budoucna lze knihovnu rozšířit o zvýšení počtu předdefinovaných tvarů galaxie nebo začlenění nových vlastností do zbývajících dvou kanálů `heat mapy` (případně rozšíření o větší počet `heat map`). Dále pak rozšíření vlastností generovaných slunečních soustav nebo vytvoření nových metod generování galaxií. Mimo generování slunečních soustav by mohla knihovna zahrnout do procesu také generování dalších vesmírných těles, jakými jsou například černé díry, pulzary nebo planety nacházejících se uvnitř slunečních soustav.

Ukázková aplikace by mohla dostat vylepšení v podobě propracovanější grafické reprezentace slunečních soustav. Aplikaci by bylo možné rozšířit do podoby počítačové hry, vesmírného simulátoru nebo výukové aplikace. Možnost živého náhledu v hlavním menu na možnou variantu vygenerované galaxie podle právě zvolených parametrů. Uživatel by mohl ve 3D pohledu označit jakoukoliv sluneční soustavu a upravovat její vlastnosti, jakými je pozice v galaxii, jméno nebo její teplota.

Literatura

[1] Stellaris Dev Diary #3 - Galaxy Generation. [Online; navštíveno 18.11.2017].
URL https://forum.paradoxplaza.com/forum/index.php?threads/stellaris-dev-diary-3-galaxy-generation.885267/

[2] A Strange Ring Galaxy. [Online; navštíveno 26.11.2017].
URL https://www.nasa.gov/multimedia/imagegallery/image_feature_1747.html

[3] Template Editor. [Online; navštíveno 4.4.2018].
URL http://heroes.thelazy.net/wiki/Template_Editor

[4] What & how is MT? [Online; navštíveno 6.3.2018].
URL http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/ewhat-is-mt.html

[5] Random Map Scripting Guide. [Online; navštíveno 6.4.2018].
URL http://aok.heavengames.com/blacksmith/showfile.php?fileid=12178

[6] Ericson, C.: *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology)*. Morgan Kaufmann Publishers Inc., 2004, ISBN 1558607323.

[7] Hubble, E.: *The Realm of the Nebulae*. Mrs. Hepsa Ely Silliman memorial lectures, Yale University Press, 1936, ISBN 9780300025002, 124-151 s.
URL https://books.google.cz/books?id=kgiXdDGLpFUC

[8] Johnson, L.; Yannakakis, G. N.; Togelius, J.: Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 2010.

[9] Knuth, D. E.: *Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd Edition)*. Addison-Wesley Professional, 1997, ISBN 0201896842, 10-15 s.

[10] Matsumoto, M.; Nishimura, T.: Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator. *ACM Trans. Model. Comput. Simul.*, ročník 8, č. 1, Leden 1998: s. 3–30, ISSN 1049-3301.
URL http://doi.acm.org/10.1145/272991.272995

[11] Mawhorte, P.; Mateas, M.: Procedural level generation using occupancy-regulated extension. In *IEEE Symposium on Computational Intelligence and Games (CIG)*, 2010, s. 351–358.

[12] Meredith, W. M.: The poisson distribution and poisson process in psychometric theory. *ETS Research Bulletin Series*, 1968: s. 12–15.

URL
https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2333-8504.1968.tb00565.x

[13] Shaker, N.; Togelius, J.; Nelson, M. J.: *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016.

[14] Smith, G.; Treanor, M.; Whitehead, J.; aj.: Rhythm-based level generation for 2D platformers. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, 2009, s. 175–182.