Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Procedurální generování 3D scény v grafických enginech

Bakalářská práce

Autor: Tomáš Ondřej

Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Bruno Ježek, Ph.D.

Hradec Králové Červenec 2018

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne Tomáš Ondřej

Poděkování:

Děkuji Ing. Brunovi Ježkovi, Ph.D. za odborné vedení mé bakalářské práce, cenné rady, připomínky a velkou trpělivost. Poděkování patří i mé rodině za velkou morální podporu a trpělivost.

Anotace

Bakalářská práce se zabývá procedurálním modelováním 3D scény. V práci jsou představeny principy a metody, které umožňují vznik scény a její další úpravy. V rámci procedurálního modelování jsou využity prostředky, pomocí kterých lze vytvářet scénu s možností parametrizace jejích částí. Implementace takové scény je provedena pro možnost interakce ve virtuální realitě.

Annotation

Title: Procedural generation of 3D scene in graphics engines

Bachelor Thesis deals with procedural modeling of 3D scene. The work presents the principles and methods that allow the scene to be created and further edited. Procedural modeling uses resources by which a scene can be created with the possibility of parameterization of its parts. Implementation of such scene is done for the possibility of interaction in virtual reality.

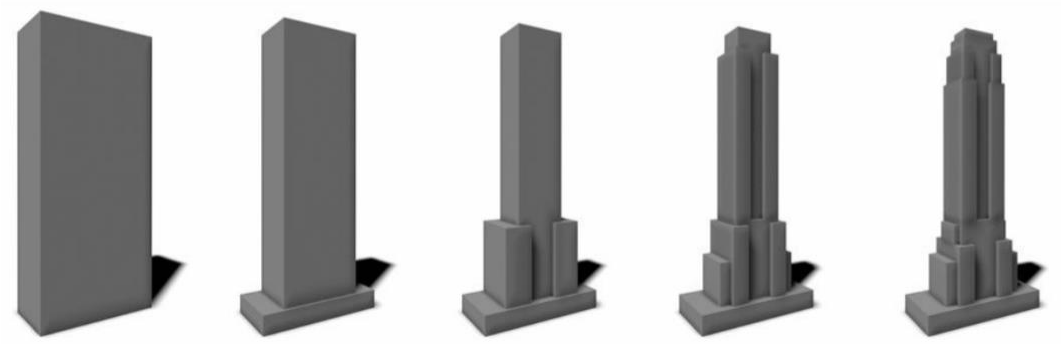
Obsah

1 Úvod

Rozvoj počítačových technologií a zvyšování výkonu umožnil v počítačové grafice širší využití přístupů pro generování zobrazované scény. Zobrazovaná scéna tak nemusí být pouze pevnou strukturou definovanou grafickým návrhářem, ale může být průběžně aktualizována podle požadavků aplikace či daného zobrazení. Jednou z metod je procedurální generování scény s parametrizací. Za využití parametrizace scény lze dosáhnout dynamické struktury zobrazované scény a možnost určovat její vzhled.

V počítačové grafice se lze setkat s použitím procedurálního generování pro mnoho účelů. Často jsou procedurálně modelovány objekty, jejich tvar a vlastnosti jsou popsatelné funkčním předpisem a není tedy nutné jednotlivé detaily ukládat v datových strukturách. Příkladem může být procedurální generování jednotlivých rostlin, povrchu terénu nebo celé krajiny včetně rozmístění budov a jiných objektů (19). Speciální kategorií je generování objektů, které nemají pevný tvar a mění se v čase jako je zobrazení tekutin, plynů, explozí nebo ohně. Do procedurálně generovaného obsahu lze zahrnout procedurálně generované textury, které zajišťují možnost aplikace opakujících se vzorů, např. cihlová zeď, plot apod.

Při využití algoritmů pro procedurální generování je možné dosáhnout výsledků založených na předem definovaných pravidlech, pomocí kterých je budována celá scéna. Výhodou algoritmů pro generování je možnost modifikace vstupních parametrů, která při každé změně zajistí změny ve tvořeném modelu scény (17).



Obr. č. 1 – Procedurálně vygenerovaný model budovy (10).

Hlavní výhodou procedurálního generování je především schopnost generovat obsah na základě znalosti jen určitých informací, např. půdorysu budovy, který poslouží k vygenerování modelu budovy. Procedurální generování nachází své využití i v řadě počítačových her, u kterých se vzrůstajícími nároky na herní zážitek je nutné využívat automatizovaných metod pro tvorbu obsahu (19). Pro ověření metod procedurálního

generování je vytvořena aplikace, na které jsou jednotlivé metody aplikovány a následně hodnoceny.

Práce je členěna do sedmi kapitol, které se skládají z teoretické části, testování principů, implementace a zhodnocení výsledků. V teoretické části jsou předvedeny různé metody procedurálního modelování, které lze aplikovat na 3D scénu. Testování principů procedurálního modelování je popsáno v kapitole Návrh řešení a konkrétní implementace v kapitole Implementace v Unreal Engine 4. Zhodnocení výsledků jsou především výstupy aplikace a jejich popsání.

2 Principy a metody procedurálního modelování 3D scény

Principy a metody procedurálního modelování umožňují vytvářet různorodou scénu, jejíž vzhled může podléhat určitým pravidlům. Pravidla, která určují vzhled vytvářené scény lze obvykle definovat funkčním předpisem.

Mezi jednotlivé metody patří využívání gramatik, parametrizace scény vstupním parametrem, vzorkování podle mřížky a dynamické generování scény. Jednotlivé metody popisují, jakým způsobem je vytvářena výsledná scény a jak je možné jí dále modifikovat.

2.1 Modelování podle gramatik

Jednou z metod procedurálního modelování je využívání gramatik, pomocí kterých lze iterativním způsobem generovat scénu. Modelování je založeno na gramatice definované čtyřmi parametry G (N, T, P, S) :

- N - Množina neterminálních symbolů, které jsou nahraditelné jinými symboly
- T - Množina terminálních symbolů, které nejsou nahraditelné jinými symboly
- P - Množina pravidel, které definují způsob nahrazování neterminálních symbolů
- S - Počáteční symbol

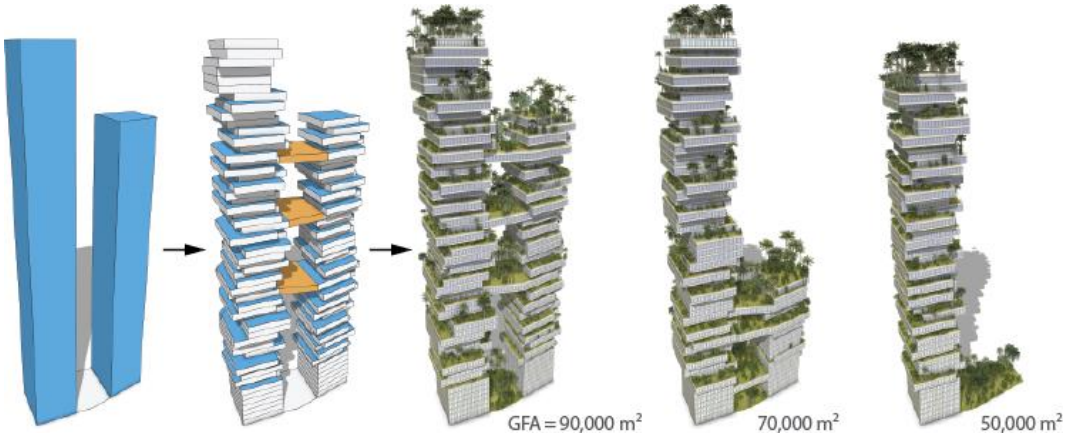
Iterativní způsob znamená, že jednotlivá pravidla jsou aplikována postupně a v každé iteraci dojde ke změně vzhledu výsledné scény. Pomocí gramatik je tedy možné generovat scénu v jednotlivých krocích, ve kterých se řídíme určitými pravidly. Tyto pravidla určují, jak mohou být neterminální symboly nahrazeny jinými (14).

Modelování podle gramatik lze uplatnit v praxi například při generování budov, u kterého jsou symboly reprezentovány konkrétními grafickými primitivy. Množinou neterminálních symbolů může být konkrétní část kvádru, která se v průběhu iterací mění na jiné grafické primitivy. Množinou terminálních symbolů může být prostor v okolí kvádru, který se již nebude nahrazovat jinými symboly, například podstava kvádru, která zůstává stejná. Pravidla mohou určovat postupné iterace, ve kterých bude z kvádrů vznikat kompozice dvou kvádru, které pomocí vhodné textury připomínají budovu s přidanými stromy na střeše a prostorách reprezentující balkón. Počáteční symbol může být reprezentován klasickým modelem dvou kvádrů umístěném v prostoru scény. Na

obrázku č. 2 je viditelné využití generování pomocí gramatik, ve kterém je pomocí čtyřech iterací dosaženo vytvoření modelu budovy (11). V první iteraci je nutné na podstavu určitého typu umístit dva kvádry. Postup je takový, že prostor, na kterém mají kvádry být, se označí jako parcela. Parcela se rozdělí na tři části, dvě jako stavební a jedna jako volná. Stavební parcely jsou poté podstoupeny funkci pro distribuci kvádrů. Pravidlo, které tento postup zajišťuje lze zapsat způsobem uvedením níže (16).

Parcel --> split("x") { ~1: BldArea | ~1: GreenSpace | ~1: BldArea }

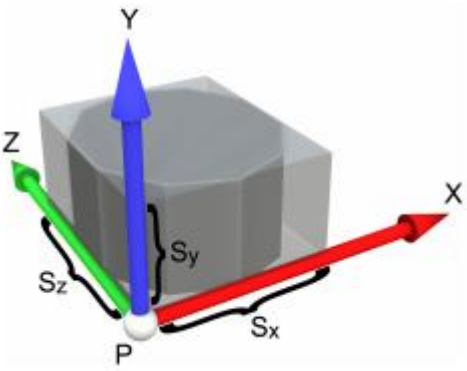
BldArea --> event(DistributeGFA) extrude(get("nFloors") * floorH) Tower



Obr. č. 2 – Generování budov na základě gramatik (16).

2.2 Počítačem generovaná architektura

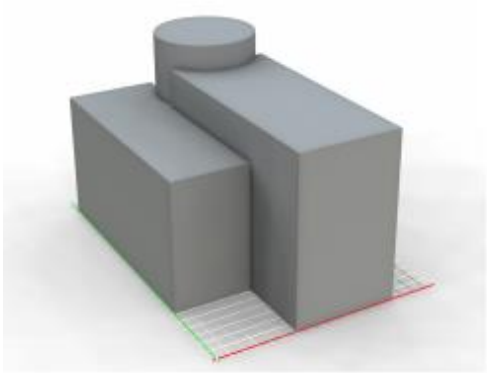
Počítačem generovaná architektura (CGA), angl. Computer Generated Architecture, vytváří modely budov ve vysoké kvalitě s důrazem na detail. Princip generování objektů pracuje na základě Modelování podle gramatik, které pracuje s různou kombinací tvarů. Nejvíce důležité geometrické atributy každého objektu jsou umístění určené polohou bodu P, orientace určená třemi ortogonálními vektory X, Y a Z, popisující společně s bodem P souřadnicový systém objektu, a velikost určená vektorem měřítka S (11).



Obr. č. 3 – Objekt definovaný bodem P, třemi osami X, Y, Z a velikostí S, která je definovaná pomocí ohraničení, ve kterém se nachází daný objekt (12).

2.2.1 Proces generování modelů budov

Objekty reprezentující budovy se skládají z konečného množství základních tvarů. Proces generování může začít s libovolnou konfigurací těchto tvarů. Mezi běžně používanou kombinaci objektů patří krychle, kvádr a válec, které jsou zobrazeny na obrázku č. 4. Generování modelu je rozloženo do několika fází. Začíná se vytvořením kostry objektu, dále následuje přidání střechy, oken a dveří. Výhodou postupu této metody je vytvoření hierarchické struktury, která je součástí modelovacího procesu objektu.

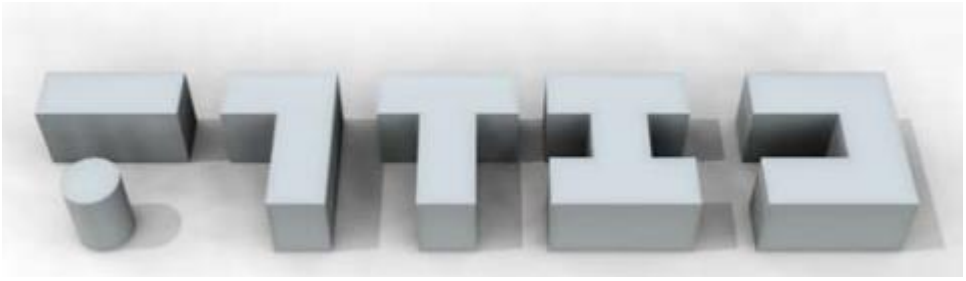


Obr. č. 4 – Model budovy složen ze tří geometrických tvarů (12).

2.2.2 Vytvoření kostry objektu

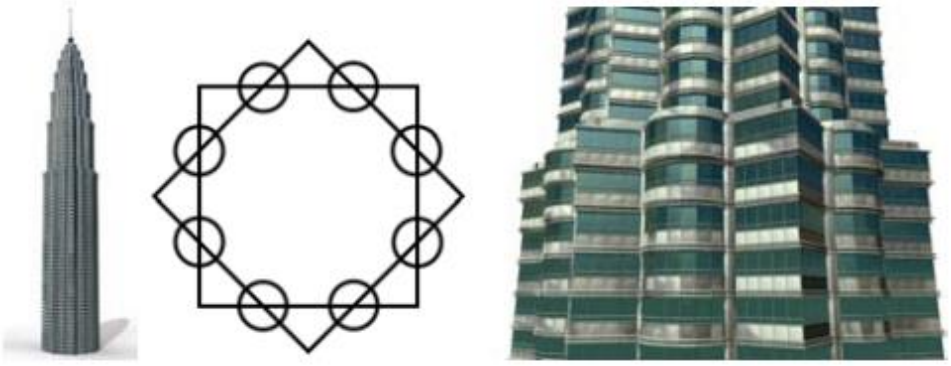
Základní postup generování kostry objektu je založen na vyhodnocení půdorysu budovy, podle kterého se určují další části. Kostra objektu se skládá ze základních grafických primitiv, která pomocí změny měřítka, změnou pozice, složením či

rozdělením vytváří základní model stavby. Příklady různých tvarů jsou zobrazeny na obrázku č. 5.



Obr. č. 5 – Různé typy kostry objektu (11).

Dalším krokem je využití libovolných transformací, především rotací a translací, tvarů a spojení s dalšími modely pro tvorbu složitějších objektů. Samotná konstrukce budov pomocí sjednocení tvarů je vhodně využita při modelování mrakodrapů, které je zobrazeno na obrázku č. 6.



Obr. č. 6 – Rekonstrukce jedné z věží z dvojice, tzv. Petronas towers z centra Malajsie (11). Celkový vzhled, půdorys budovy je složen ze dvou základních grafických primitiv, kvádrů a válců.

2.2.3 Generování střechy

Podle typu kostry objektu je určena i příslušná střecha, která splňuje předpoklady pro daný objekt. Pro složitější objekty, které se mezi sebou protínají, určujeme pravidla, která zajistí správné umístění střechy.

Výsledkem může vzniknout kombinace tvarů, které se neprotínají, částečně protínající a plně spojené. Pomocí tohoto postupu získáváme informace, které využijeme při generování oken, dveří a dalších detailů. Základní výstup generování střechy je zobrazen na obrázku č. 7.

Obr. č. 7 – Podle typu tvaru je vytvořena střecha objektu (11).

2.2.4 Umístění detailů

Třetím krokem vývoje budov je umístění detailů na povrch modelu. Těmito detaily jsou u budov především okna a dveře. Ke kostře objektu je připojeno potřebné množství oken a dveří s důrazem na bezkolizní umístění v prostoru. V případě spojení více jednoduchých objektů je nutné zajistit generování oken a dveří pouze na viditelná místa, což znamená, že okna ani dveře nebudou umístěny v místě, ve kterém se budovy spojují a budou umístěny pouze na místech, ve kterých nejsou jednotlivé objekty v kolizi (11).

Pravidla, která se aplikují na model budovy lze zapsat podobným způsobem jako na obrázku č. 8, který popisuje modifikaci tvaru s názvem „fac“ a jeho nahrazení tvary s názvy „floor“. Pravidlo pro modifikaci tvaru se skládá z:

- identifikátoru pravidla
- tvaru, který má být modifikován
- podmínky
- tvar, který má vzniknout

1: $\text{fac}(h) : h > 9 \rightsquigarrow \text{floor}(h/3)\ \text{floor}(h/3)\ \text{floor}(h/3)$

Obr. č. 8 – Příklad zápisu pravidla pro modifikaci tvaru (11).

Po aplikaci všech potřebných pravidel dochází ke generování modelu budovy, který je viditelný na obrázku č. 9.

Obr. č. 9 – Vygenerovaná budova pomocí předem definovaných pravidel (11).

Výsledkem vývoje je objekt, který lze využít pro znázornění modelu budovy, případně může být vygenerovaný model dále upravován změnou parametrů.

Po vygenerování budovy a jejím umístění do scény lze za účelem optimalizace provést změnu úrovně detailu daného modelu. Při generování scény o větších rozměrech jsou tak vzdálenější objekty vykresleny v kvalitě vzhledem ke vzdálenosti od objektu.

2.3 Změna úrovně detailu

Algoritmus změny úrovně detailu (LOD), z angl. Level of detail, nachází využití především z důvodu úspory výpočetního výkonu. Lidské oko má schopnost zaměřovat pozornost především na objekty, které jsou v dostatečné blízkosti. Proto je princip LOD metod založen na omezení detailu zobrazení vzdálených objektů snížením kvality zobrazení, např. redukcí počtů trojúhelníků, ze kterých jsou vytvořeny (15).

Při využití algoritmu změny úrovně detailu je možné dosáhnout snížení počtu vykreslovaných trojúhelníků, čímž dochází ke zrychlení renderování scény při zachování vyhovující vizuální kvality výsledného zobrazení. Na obrázku č. 10 je ukázán výstup využití algoritmu pro změnu úrovně detailu, který zachovává tvar objektu, pouze snižuje úroveň detailu (4).

Obr. č. 10 – Využití algoritmu změny úrovně detailu v prostředí Unreal Engine (4).

2.4 Parametrizace generování scény

Parametrizace generování scény vstupním parametrem, tzv. seedem, je jednou z metod procedurálního modelování. Na základě vstupního parametru je vygenerován určitý výstup, který je výsledkem konkrétního algoritmu. Konkrétní hodnotou vstupního parametru je dosaženo určitého výsledku a při každém generování se stejným vstupním

parametrem bude i výsledný model stejný. Podoba vstupního parametru je různá, může se jednat o sekvenci čísel nebo o kombinaci čísel a znaků, které jsou předány programu, který vstup zpracuje a vrátí odpovídající výstup (8).

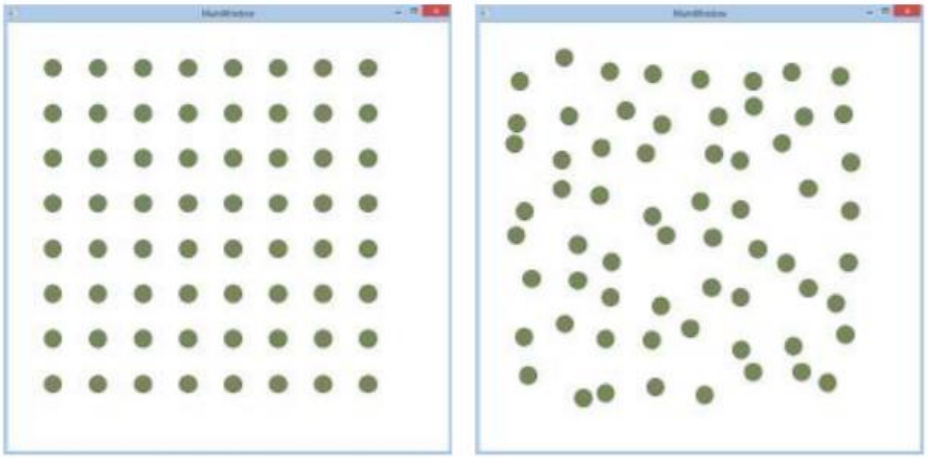
S využitím parametrizace generování scény se lze setkat u procedurálně generovaných her, mezi které patří například Minecraft nebo Lego Worlds, u kterých je možné určit seed ještě před vygenerováním scény (8). Využitím tohoto postupu lze při každém spuštění hry a zadáním stejných parametrů dosáhnout stejného výstupu, který není závislý na čase nebo typu hardwaru, na kterém je generování scény prováděno. Pomocí této metody lze vyvíjet dynamicky se měnící scénu o různé velikosti a není nutné celou velikost scény uchovávat na disku nebo na jiném médiu.

2.5 Vzorkování podle mřížky s parametrem

Vzorkování podle mřížky s parametrem je jednou z metod procedurálního modelování, které zajišťuje možnost rozmísťovat prvky pravidelně s definovaným parametrem. Mezi podobné metody patří také pravidelné vzorkování prvků podle mřížky, které je zobrazeno na obrázku č. 11 na levé straně. Vzorkování prvků podle mřížky je možné realizovat pomocí dvou cyklů, při kterém jeden z cyklů slouží pro rozmísťování prvků podél osy X a druhý cyklus slouží pro rozmísťování prvků podél osy Y.

Oproti pravidelnému vzorkování se vzorkování podle mřížky s parametrem liší v umístění jednotlivých prvků. Při vzorkování je využívaný parametr nazývaný „jitter", jehož hodnota určuje rozestup oproti původnímu umístění. Při zvyšující hodnotě parametru se zvyšuje odstup od původní pozice, který je vypočítám podle náhodné funkce. Vzorkování podle mřížky s parametrem je zobrazeno na obrázku č. 11 na pravé straně (13).

Mezi výhody vzorkování podle mřížky s parametrem patří snadná implementace a zvýšení přirozeného vzhledu, který lze využít například při vytváření scény, ve které se mají nacházet stromy nebo další části vegetace.

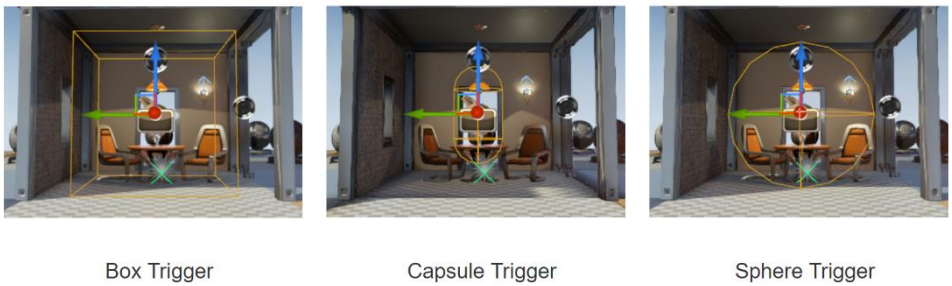


Obr. č. 11 – Vzorkování podle mřížky pravidelné a s parametrem (13).

2.6 Dynamické generování pomocí triggerů

Dynamické generování je jedním ze způsobů, který v závislosti na pozici pozorovatele určuje, které části scény se generují. Pomocí triggerů je možné do scény umísťovat takové prvky, které při interakci s pozorovatelem spouští určitou událost. Mezi takové spouštěče událostí patří například kolize s jiným prvkem scény.

Trigger může mít podobu jednoho ze tří tvarů, kterými jsou kvádr, kapsle a koule. Tyto tvary mají stejné vlastnosti a liší se pouze oblastí využití. V případě řešení kolizí osob je často využíván trigger, který má podobu kapsle. Tento trigger ve tvaru kapsle je obvykle umístěn okolo určitého prvku, kterým může být například pozorovatel (6). Na obrázku č. 12 jsou uvedeny tři podoby triggerů.



Obr. č. 12 – Typy triggerů (6).

Za předpokladu, že je možné umístit prvky, reagující na události, do scény, pak lze tyto události vyhodnocovat. Triggery umožňují, aby byly umístěny například do prostoru části města a pokaždé, kdy by pozorovatel prošel skrz daný trigger, vygenerovala by se další část města.

3 Prostředky pro procedurální modelování scény

Popis prostředků se skládá z přehledu postupů a nástrojů procedurálního modelování, které jsou dostupné v prostředí Unreal Engine 4. Jedná se o souhrn informací o vývojovém prostředí, především se zaměřením na popis postupů, pomocí kterých lze realizovat procedurální modelování 3D scény.

Kromě prostředí Unreal Engine 4 existuje také prostředí s názvem Unity, které přináší také určité postupy a nástroje pro procedurální modelování. Unity je herní engine, který byl vytvořen společností Unity Technologies. První verze enginu vznikla v roce 2005, přičemž od té doby se stala plně multiplatformní. Mezi nástroje pro tvorbu procedurálního obsahu v prostředí Unity patří například UnityScript, pomocí kterého je možné vytvářet 2D a také 3D scénu (18).

3.1 Unreal Engine

Unreal Engine byl vytvořen společnosti Epic Games, jehož první verze vznikla v roce 1998 (9). Od té doby vzniklo několik dalších verzí, které nabízí modernější způsoby pro tvorbu 3D scény. Prostředí Unreal Engine má zajištěnou podporu pro interakci ve virtuální realitě a to od roku 2012 (7).

3.2 Skriptování v Unreal Engine 4

Prostředí umožňuje využívat klasické datové struktury a vývoj za pomoci vizuálního skriptování. Vizuální skriptování, tzv. „Blueprints Visual Scripting“ je postup generování 3D scény, který podporuje objektově orientovaný přístup, pomocí kterého lze vytvářet libovolné třídy a kooperaci mezi nimi. Veškeré třídy, které jsou definovány tímto přístupem, jsou známé jako „Blueprints“ (1).

Skriptování pomocí Blueprints v Unreal Engine je extrémně flexibilní a výkonný nástroj, který umožňuje celou řadu konceptů a nástrojů, které jsou snadno rozšiřitelné. Typů vizuálního skriptování je několik a liší se především podle svého účelu. Mezi základní typy vizuálního skriptování patří:

- Blueprint Class
- Level Blueprint
- Blueprint Interface

Typ „Blueprint Class“, který popisuje standardní třídu, umožňuje definovat určitou funkčnost, jako alternativu k tvorbě kódu. Třída může být definovaná i jako speciální typ „Aktér“ a následně být umístěna jako instance do prostoru 3D scény.

Typ „Level Blueprint“ působí jako globální graf, který se skládá z jednotlivých událostí. Patří do něj konkrétní třídy a funkce, lze si ho představit jako jednu velkou scénu. Pomocí streamovacího mechanismu je zajištěno, že je možné přepínat mezi jednotlivými „Levely“ a lze zajistit, že výsledná aplikace se může skládat z mnoha různých scén.

Dalším typem je „Blueprint Interface“, který reprezentuje kolekci funkcí, které nejsou v rámci interface implementovány, ale je možné, aby byly přidány do ostatních tříd, resp. „Blueprint Class“, ve kterých je rozvedena konkrétní implementace. Pomocí tohoto přístupu je možné zajistit škálovatelnost instancí, které tento interface využívají.

Pro účely procedurálního generování je nutné uvedené typy využívat, k čemuž jsou určeny speciální prvky, pomocí kterých je možné definovat komponenty, provést inicializační kroky v rámci aplikace a další operace jako například odpověď na různé druhy událostí. Mezi tyto speciální prvky patří:

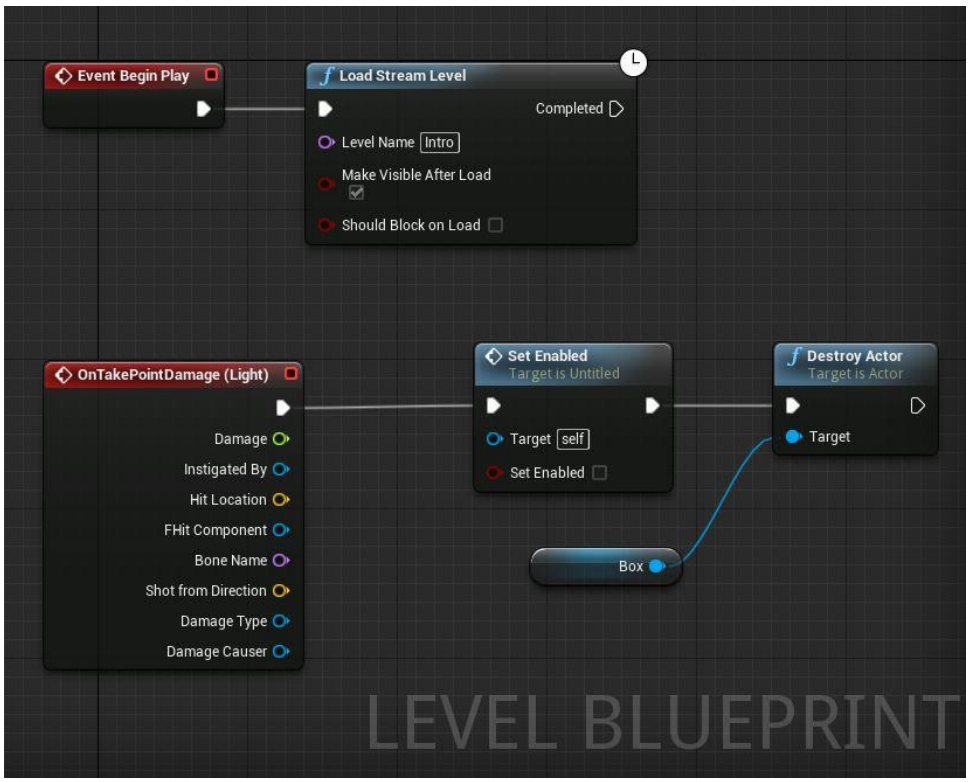
- Construction Script
- Event Graph

Aby bylo možné zařídit, že každá instance z třídy bude při svém vytváření provádět určitou činnost, je využíváno speciálního prvku „Construction Script“, ve kterém je možné zajistit, aby každá instance byla generována s jinými vstupními parametry. Vstupní parametry mohou být určeny pomocí různých kritérií nebo pomocí náhodného generování či náhodného generování v určitém rozsahu hodnot. Na obrázku č. 13 je viditelný vývoj a použití konstrukčního skriptu, u kterého je do scény umísťovaný objekt pomocí funkce „AddStaticMesh“.

Obr. č. 13 – Construction Script v prostředí Unreal Engine 4 (2).

Možnost provádět určitou činnost při vytváření instancí je vhodným způsobem, jak ovlivnit například vzhled instance, ale aby bylo možné i během životního cyklu každé instance provádět také reakce na určité události, je nutné používat prvek „Event Graph“, tzv. graf událostí. Pomocí grafu událostí se dá reagovat na různé typy událostí, kterými může být například kolize s jinými instancemi nebo provést určitou událost při spuštění

aplikace nebo provádět určitou událost v pravidelných intervalech, tzv. „Event Tick“. Provádění určité činnosti při spuštění aplikace může být například načtení tzv. Levelu, resp. „Level Blueprint“ pomocí funkce „LoadStreamLevel“, které je zobrazeno na obrázku č. 14.



Obr. č. 14 - Event Graph a reakce na událost

Procedurální generování přináší rychlejší a systematičtější tvorbu scény, se kterou se můžeme setkat v prostředí Unreal Engine 4, ve kterém je možné využít nástroje pro tvorbu procedurálního obsahu. Na obrázku č. 15 je vygenerovaný terén pomocí nástroje na tvorbu terénu, tzv. „Grass tool“. Na obrázku č. 16 je uvedený výstup, který byl vygenerován pomocí nástroje na tvorbu vegetace, tzv. „Procedural foliage tool“ (3).

Obr. č. 15 – Procedurálně vygenerovaný terén (3). Obr. č. 16 – Procedurálně generovaná vegetace (3).

4 Návrh řešení

Pro testování konceptu parametrického 3D modelování jsou navrženy dva modely scény. První model je průchod dynamicky generovaným městem na základě pravidel parametrizace scény. Výslednou scénu lze využít například při vizualizaci architektonických konceptů nebo v cestovním ruchu. Druhý model je průchod galerií s obrazy, u kterých je možné měnit texturu jednotlivých obrazů z libovolných externích fotografií. Tím je možné personifikovat scénu podle potřeb jednotlivých uživatelů. Případné analyzování působení na uživatele lze aplikovat v oboru psychologie.

4.1 Skladba 3D scény

Scéna se skládá z outdoor a indoor části, v každé z těchto částí se nacházejí instance jednotlivých tříd, které můžeme nazývat elementy. Tyto elementy jsou zodpovědné za konkrétní činnosti, mezi které patří například průchod scénou pro instance chodců, kontrola rozmístění oken a dveří pro instance budov nebo také kontrola jízdy pro instance vozidel. Celou scénu lze tedy vnímat jako spolupráci jednotlivých částí.

4.1.1 Outdoor scéna

Outdoor scéna je reprezentována jako průchod městem, jehož části jsou procedurálně generovány. Jednotlivé části jsou složeny z grafických elementů, které mohou být dále parametrizovány. Některé elementy jsou umístěny do scény staticky, např. Cesta, Křižovatka, Budova, Vegetace, jiné se mohou pohybovat, např. Chodec, Vozidlo.

Pro vytvoření cesty je využito základního modelu kostky s vhodnou transformací do podoby, která tvoří část silnice. Po stranách jsou připojeny ze stejného modelu kostky také tvary, které reprezentují chodník. Přidání vhodných textur s motivem silnice a chodníku doplňuje vzhled.

Vytvoření křižovatky je navrženo pro možnosti připojení čtyř cest, samotná křižovatka je reprezentována stejným modelem jako cesta a také doplněna o chodníky, jen s rozdílem zvolených textur.

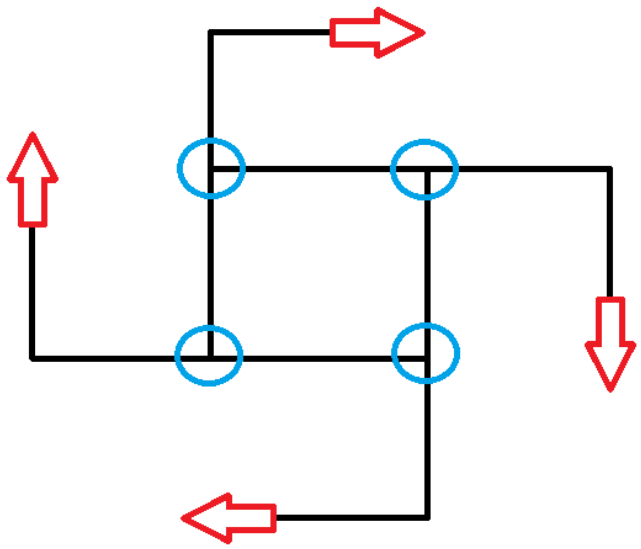
Vytvoření budovy se odvíjí, od již vytvořených cest a křižovatek, a to z toho důvodu, že budova je generována do okolí cest a je kladen důraz na předcházení kolizím mezi již vygenerovanou cestou a budovou. Nemělo by tedy docházet k tomu, že budova bude vygenerována na stejné místo jako je cesta nebo křižovatka.

Generování vegetace je záležitost především mimo město, jedná se o tři druhy stromů a trávu. Mimo město není nutné řešit kolize s elementy, které se nachází ve městě, je však nutné dbát na to, aby stromy neprotínaly jiné stromy. Vegetace se nenachází pouze mimo město, ale i přímo ve městě v okolí budov jsou umístěny skupiny keřů. Vhodně určené souřadnice zajišťují, že nedojde ke kolizím s budovou nebo s jinou částí modelu. Každý keř se skládá z několika částí, jako první část je plocha, na které se keř nachází, která je doplněna o texturu trávy, další částí je vytvoření plotu okolo keře a následuje vytvoření samotného keře. Pro vizualizaci keře je zvolen defaultní model, který nabízí prostředí Unreal Engine 4.

Poslední dva elementy ve městě jsou chodec a vozidlo, kteří na rozdíl od ostatních mají odlišnou vlastnost a to, že se pohybují. Pro chodce i vozidlo je nutné řešení kolizních stavů, při kterých se střetnou s jinými elementy. Za předpokladu, že se střetne chodec s vozidlem, je zajištěno jejich odstranění ze scény. Z toho důvodu se dají očekávat náročnější požadavky na výpočetní výkon, především z důvodu, že každý chodec a každé vozidlo nejsou staticky umístěny.

4.1.2 Indoor scéna

Indoor scéna je reprezentována jako průchod galerií s obrazy, ve které jsou jako hlavní elementy chodba a obraz. Samotná chodba obsahuje jednotlivé obrazy, které jsou umístěny na stěnách chodby. Model scény se skládá ze statické a dynamické části. Statická část je ta část scény, která je již vygenerovaná a dynamická část scény se vygeneruje až v okamžiku, kdy uživatel vstoupí do konkrétního místa. Je tím zajištěno, že není nutné generovat před spuštěním velký obsah dat a zároveň může uživatel procházet relativně daleko v galerii. Samotné generování není stále stejným směrem, ale je zajištěno, aby po určitém intervalu došlo k tzv. odbočce. Další chodby se generují rovně, o 90° doleva nebo doprava. Určení, zda se bude chodba generovat rovně, doleva nebo doprava je v první řadě závislé na náhodě, ale před každým výběrem je zvážena možnost, jestli je zvolený směr možný. Pokud by ve směru generování byla již vygenerovaná chodba, došlo by ke kolizi, a proto je pomocí vytvořené funkce této možnosti zabráněno a je vybraná jiná možnost. Pokud je již galerie rozsáhlejší a všechny možnosti by vytvářely kolize, je chodba zakončena a uživatel dále stejným směrem nepokračuje.



Obr. č. 17 – Mapa průchodu galerií

Na obrázku č. 17 je zobrazena mapa průchodu galerií, ve které jsou znázorněny modré kruhy a červené šipky. Modré kruhy jsou místa, do kterých pokud uživatel vstoupí, začne se dynamicky generovat další scéna ve směru šipky, u které je modrý kruh.

Statická část scény je složena z chodeb, které jsou spojeny do čtverce, který v místech spojení (v rozích) plynule navazuje na možnost vstoupit do části dynamické.

Dynamická část se skládá z konečného množství chodeb, při jejichž průchodu se před uživatelem generuje další množství chodeb.

4.2 Parametry generování scény

Parametry generování scény určují tvorbu modelu a výsledný vzhled celé scény. Pro různorodost výsledné scény je zavedeno celkem deset parametrů, které může uživatel před spuštěním modifikovat a také několik dalších parametrů, které jsou určeny náhodnou funkcí, která má definované rozpětí. Z celkem deseti parametrů je jich devět vymezeno pro modifikaci outdoor scény a jeden parametr pro modifikaci indoor scény. Některé parametry je také možné měnit v průběhu průchodu scénou a odlišit tak vzhled již vygenerované statické scény od dynamicky generované. Přehled parametrů a jejich typu ukazuje tabulka č. 1.

Tabulka č. 1 – Přehledová tabulka parametrů

Účel parametru	Zkratka	Typ scény	Parametr	
			statický	dynamický
Určení četnosti výskytu křižovatek	P_{vk}	outdoor	✓	✓
Určení rozměrů budov	P_{rb}	outdoor	✓	
Určení počtu pater budov	P_{pb}	outdoor	✓	✓
Určení počtu kvádrů tvořících budovy	P_{ktb}	outdoor	✓	✓
Určení velikosti mezer mezi keři	P_{mkb}	outdoor	✓	
Určení velikosti mezer mezi okny	P_{mob}	outdoor	✓	
Určení výšky střešní stěny budov	P_{sb}	outdoor	✓	
Určení počtu osob ve skupině	P_{os}	outdoor	✓	✓
Určení počtu chodců a vozidel	P_{chv}	outdoor	✓	✓
Určení hustoty vegetace mimo město	P_{vmm}	outdoor	✓	
Určení velikosti galerie	P_{vg}	indoor	✓	

Parametr pro určení četnosti výskytu křižovatek P_{vk} a počtu budov v bloku

První z parametrů pro modifikaci outdoor scény je parametr četnosti výskytu křižovatek a počtu budov v bloku. Parametr P_{vk} určuje velikost čtvercového bloku, ve kterém jsou jednotlivé budovy. Zároveň je tak určena i četnost křižovatek mezi jednotlivými bloky. Jako hodnota parametru je nastavena délka hrany čtvercového bloku v intervalu <3,10>. Druhá mocnina hodnoty parametru určuje počet budov v bloku.

Po určení hodnoty parametru P_{vk} dochází k vygenerování odpovídajícího počtu budov, a to v povoleném rozpětí, při kterém je nejnižší počet budov v bloku vymezen na devět a maximální počet budov v bloku na sto budov. Jako blok je definován prostor, který je mezi dvěma křižovatkami. Parametr P_{vk} nepřímo určuje, jak často se vyskytuje

křižovatka, protože pokud je dodrženo pravidlo, že generování budovy se odvíjí, od již vytvořených cest a že budovy jsou generovány do okolí cest, pak čím více je budov v bloku, tím větší bude rozpětí mezi křižovatkami. Parametr P_{vk} je určen před vygenerováním scény, ale zároveň je možné tento parametr ovlivňovat i během průchodu městem, dále uvedeno v 4.3.

Parametr pro určení rozměrů budov P_{rb}

Parametr pro určení rozměrů budov je určen náhodnou funkcí, která funguje na principu výběru náhodné hodnoty z vymezeného rozpětí. Vzhledem k tomu, že hodnotu přímo neovlivňuje uživatel, ale určuje jí vnitřní funkce, není nutné pro každou budovu zadávat tyto hodnoty.

Rozměry, které parametr určuje, jsou dva, a to velikost budovy na ose X a Y. Z důvodu možnosti určovat z kolika kvádrů jsou tvořeny budovy (dále uvedeno v Parametr pro určení počtu kvádrů tvořících budovy) je nutné brát v úvahu skutečnost, že pokud je budova složena z jednoho kvádru, tak velikost tohoto kvádru nebude stejná jako v případě, kdy bude tato budova složena z více kvádrů. V případě, že bude budova složena z více kvádrů, bude součet velikostí jednotlivých kvádrů roven velikosti budovy, která je složena z jednoho kvádru. Je tím zajištěno, že celková velikost budovy bude stejná, nezávisle na tom, z kolika kvádrů je složena. Velikost budovy na obou osách lze zapsat jako:

VelikostBudovy[osa X] = NáhodnáHodnotaVRozpětí() * P_{ktb}

VelikostBudovy[osa Y] = NáhodnáHodnotaVRozpětí()

Důvodem pro náhodné určování těchto hodnot je především skutečnost, že při vyšším počtu budov by bylo náročné zadávat pro každou budovu její rozměry. Jako další možnost by připadala v úvahu varianta, při které by tento parametr mohl být modifikován uživatelem, který by určil jednu hodnotu, a budovy by se poté generovaly podle stejného parametru. Problémem tohoto přístupu by byl fakt, že by všechny budovy měly stejný rozměr na ose X a Y. Proto je hodnota parametru pro určení rozměrů budov určena náhodnou funkcí.

Parametr pro určení počtu pater budov P_{pb}

Parametr pro určení počtu pater budov je dalším z parametrů, které může uživatel před spuštěním modifikovat a zároveň se jedná o parametr, který je možné ovlivňovat také při průchodu městem. Jako hodnotu parametru je možné zvolit v rozpětí od jedné do deseti, přičemž je vycházeno z reálného světa, ve kterém je například jednopatrová

budova taková budova, která má přízemí a nad ním jedno patro. Rozpětí od jedné do deseti je zvoleno z určitých důvodů, mezi které patří především nároky na výpočetní výkon a pak také z důvodu, že nejčastěji se lze setkat v reálném světě s budovami, které mají počet pater v tomto rozpětí.

Podle počtu pater budovy je určeno, jaký vzhled bude mít střecha budovy. Za předpokladu, že počet pater budovy je menší než 2, bude střecha šikmá, v opačném případě bude střecha rovná, dále uvedeno v Parametr pro určení výšky střešní stěny budov. Důvodem pro nastavení tohoto pravidla je snaha o přiblížení se skutečnosti a při pohledu na reálný svět je obvyklé, že výškové budovy mají rovnou střechu, a naopak nižší budovy mají šikmou střechu.

Parametr pro určení počtu kvádrů tvořících budovy P_{ktb}

Parametr pro určení počtu kvádrů tvořících budovy je stejně jako parametr P_{pb} dalším z parametrů, které může uživatel před spuštěním modifikovat a ovlivňovat také při průchodu městem. Jedná se o parametr, který nabývá hodnot v rozpětí od jedné do tří, což znamená, že výsledná budova se může skládat z jednoho až tří kvádrů.

Pokud je určena hodnota parametru P_{ktb}, je možné přejít k samotnému generování zvoleného počtu budov a provést vyhodnocení, z kolika kvádrů se má budova skládat. Za předpokladu, že se budova skládá z jednoho kvádru, je podle vstupních parametrů vytvořena budova. Mezi vstupní parametry patří pozice nebo také velikost kvádru. Pokud je však budova složena z více kvádrů, je nutné znát pozici a velikost všech kvádrů a zajistit jejich vhodné umístění. Postup je takový, že pokud se bude budova skládat například ze dvou kvádrů, dojde k vygenerování prvního kvádru a jeho transformace je předána ke generování druhého kvádru. Druhý kvádr se bude nacházet na boku prvního kvádru a jeho umístění bude možné spočítat jako:

DruhýKvádr[X pozice] = PoziceX(PrvníKvádr) + (ZměnaMěřítkaX(PrvníKvádr) * 50) + (ZměnaMěřítkaX(DruhýKvádr) * 50)

Je tím zajištěno, že při změně velikosti nebo pozice první budovy je druhá budova umístěna na odpovídající pozici. Podobný postup je zvolen při generování budovy, která se skládá ze tří kvádrů.

Parametr P_{ktb} se řídí pravidlem „Čím více kvádrů, tím menší rozměr jednotlivých kvádrů“, který zajišťuje, že při narůstajícím počtu kvádrů tvořících budovy se snižuje velikost jednotlivých kvádrů. Snižování velikosti je proto, aby samotná budova byla podobně velká nezávisle na počtu kvádrů, z kterých se skládá. Důvodem pro vznik

parametru P_{ktb} je snaha o zajištění možnosti generovat různorodou scénu. Při zavedení možnosti měnit, z kolika kvádrů se skládá budova, je zajištěno, že budovy se nebudou lišit pouze jejich velikostí, ale i celkovým vzhledem.

Parametr pro určení velikosti mezer mezi keři u budov P_{mkb}

Parametr pro určení velikosti mezer mezi keři u budov je parametr, který může uživatel modifikovat před spuštěním a který slouží k určení, jak velké mezery budou mezi keři, které se nachází u budov. Rozpětí mezer je určeno tak, aby v minimálním množství zajistilo, že mezi keři bude mezera o velikosti jednoho keře a v maximálním množství mezera o velikosti šesti keřů. Hranice rozpětí jsou vymezeny na základě celkového vzhledu a to tak, aby v minimálním množství byla mezi keři alespoň minimální mezera a v maximálním množství bylo v okolí budov alespoň pár keřů.

Parametr pro určení velikosti mezer mezi okny budov P_{mob}

Každá budova v outdoor scéně má okna, které mají určité rozestupy od okolních oken. Rozestupy mezi okny určuje parametr P_{mob}, který je přičítán při přidávání oken k jejich pozici na konkrétní ose. Pokud je přidáváno okno na přední nebo zadní stranu, je parametr přičítán k X-ové souřadnici každého okna, zatímco při přidávání okna na boční strany je parametr přičítán k Y-ové souřadnici každého okna.

Parametr pro určení výšky střešní stěny budov P_{sb}

Parametr pro určení výšky střešní stěny budov určuje vzhled střechy budovy a podle hodnoty parametru P_{pb} je určeno, jaká střešní stěna se vygeneruje. Za předpokladu, že je budova nižší než dvě patra, bude střešní stěna do špičky. Střešní stěna tak bude připomínat klasickou střechu a změny parametru P_{sb} budou aplikovány na změnu měřítka střechy na Z-ové ose. Střecha tak bude se zvyšujícím parametrem P_{sb} více špičatější. Pokud bude budova vyšší, bude střecha reprezentována střešní stěnou po stranách budovy, jejíž výška bude upravována parametrem P_{sb}. Úprava střešní stěny bude spočívat ve změně její pozice a měřítka na Z-ové ose. Při zjišťování, na které souřadnice umístit střešní stěnu například při zvýšení parametru, je možné aplikovat vzorec:

StřešníStěna[Z pozice] = (P_{pb} * PoziceZ(Kvádr) + (ZměnaMěřítkaZ(Kvádr) * 50)) + (ZměnaMěřítkaZ(StřešníStěna) * 50)

Vzhledem k tomu, že střešní stěna je řešena pro každý kvádr, je pro konkrétní střešní stěnu předána jako parametr souřadnice na Z-ové ose pro odpovídající kvádr.

Parametr pro určení počtu osob ve skupině P_{os}

Parametr pro určení počtu osob ve skupině je jedním z parametrů, který lze měnit i dynamicky během průchodu outdoor scénou. Je možné nastavit hodnotu parametru v intervalu <1,5>, která zajistí možnost určit velikost skupin, ve kterých osoby prochází scénou. Se zvyšujícím se parametrem roste hustota osob v celé scéně.

Změny parametru P_{os} by se měly projevit na plynulosti průchodu scénou, stejně tak jako parametr P_{chv}, se kterým společně určují hustotu provozu ve městě.

Parametr pro určení počtu chodců a vozidel P_{chv}

Parametr pro určení počtu chodců a vozidel určuje, jak často mají být chodci a vozidla do scény generovány. Pomocí parametru P_{chv} tak je určen celkový počet chodců a vozidel ve scéně, protože čím častěji se chodci a vozidla generují, tím více jich ve scéně bude za určitý časový úsek.

Po určení parametru P_{chv} je v pravidelných intervalech generováno další množství chodců a vozidel. Velikost intervalu je rovna hodnotě parametru P_{chv}.

Parametr pro určení hustoty vegetace mimo město P_{vmm}

Vegetace mimo město se skládá především z jednotlivých stromů, které mezi sebou mají určité odchylky, resp. mezery. Pokud jsou tyto mezery snižovány nebo zvyšovány, je ovlivněna celková hustota vegetace mimo město.

Hodnota parametru P_{vmm} je povolena v intervalu <0,70>, přičemž mezi jednotlivými hodnotami je rozdíl vždy 10. Parametr P_{vmm} tedy může nabývat například hodnoty 30 procent, 50 procent nebo 70 procent.

Parametr pro určení velikosti galerie P_{vg}

Parametr pro určení velikosti galerie je jediným parametrem, který ovlivňuje vzhled indoor scény. Vzhled indoor scény ovlivňuje parametr P_{vg} určováním velikosti čtvercové plochy statické části galerie. Při vzniku statické části galerie se využívá cyklu, který po určitém intervalu změní směr generování o 90° doprava. Právě zmíněný interval je roven hodnotě parametru P_{vg}. Hodnota parametru je povolena v rozpětí od 40 do 80, přičemž tato hodnota určuje délku hrany čtvercové plochy galerie.

4.3 Dynamické generování scény

Dynamické generování scény je řešeno jak v rámci outdoor scény, tak také v rámci indoor scény. V obou případech je řešeno pomocí triggerů. Každý trigger je umístěn uvnitř jednotlivých instancí tříd, jako například v části cesty v outdoor scéně nebo části chodby v indoor scéně. Při průchodu triggerem je vygenerována další část scény, která je před uživatelem. Aby při průchodu scénou bylo dynamické generování více realistické, je v obou případech řešeno odlišně. V případě outdoor scény je přidána mlha, která snižuje viditelnost, a proto například generování části cesty před uživatelem není tak rozpoznatelné. V případě indoor scény je architektura chodeb navržena tak, aby uživatel neviděl mimo prostor indoor scény.

Dynamické generování scény je jedním ze způsobů, jak pomocí změny parametrů generování scény lze dosáhnout možnosti měnit celkový vzhled scény. Parametrů, které je možné dynamicky měnit je celkem pět, proto je možné jejich modifikací zařídit odlišnosti v jednotlivých částech města. Je tak možné v určité části města zajistit převahu nízkých budov ve velkém množství, zatímco v jiné části mohou vznikat vysoké budovy v menším množství, ale s odlišným počtem kvádrů, ze kterých se skládají. Je tak také možné v určité části města zvýšit výskyt křižovatek a v jiné části města jej snížit pomocí parametru P_{vk}. Pro zajištění možnosti dynamicky modifikovat jednotlivé parametry jsou vytvořeny klávesové zkratky, pomocí kterých lze zvýšit nebo snížit hodnoty jednotlivých pěti parametrů.

4.4 Vzorkování vegetace podle mřížky s parametrem

Vzorkování vegetace podle mřížky s parametrem využívá principů jedné z metod procedurálního modelování, které se nazývá Vzorkování podle mřížky s parametrem. Tato metoda přináší možnost, jak vhodně vytvořit část 3D scény, ve které jsou stromy a případně další části vegetace. Samotné vzorkování je využíváno v outdoor scéně do prostoru mimo město.

Pro vznik realisticky vypadající scény mimo město je nutné, aby jednotlivé stromy nebyly v řadách, ale byly umístěny jako v reálném světě. Zároveň je důležité zvolit takový postup, který zajistí rychlé generování stromů. Aby bylo tyto pravidla možné dodržet, je zvolen princip, který pro určení polohy každého stromu sčítá tyto hodnoty:

- Výchozí pozice

- Mezery mezi stromy
- Náhodně určený parametr (tzv. jitter)

Výchozí pozice je nastavena na statickou hodnotu a určuje počátek, odkud se mají jednotlivé stromy generovat. Výchozí pozici lze chápat jako spodní levý roh celé mřížky. Hodnota mezer mezi stromy obsahuje výchozí hodnotu, která je v každém jednotlivém kole cyklu vynásobena indexem cyklu a přičtena k výchozí pozici. Náhodně určený parametr je získán z náhodné funkce, která má vymezené minimum a maximum. Hodnota maxima je určena jako třetina velikosti mezer mezi stromy a hodnota minima je vypočítána jako hodnota maxima krát mínus jedna.

Výsledná pozice může být upravena parametrem P_{vmm}, pomocí kterého je určeno procento výskytu stromů. Z hlediska průchodu scénou je také aplikován algoritmus, který se nazývá změna úrovně detailu. Pomocí tohoto algoritmu je zajištěno, že vzdálenější stromy nejsou vykreslovány a je dosaženo plynulejšího průchodu scénou.

Cílem je vznik takové scény, která zajišťuje, že stromy neprotínají jiné stromy a že za využití náhodně určeného parametru lze každému stromu určit unikátní pozici působící realisticky.

5 Implementace v Unreal Engine 4

5.1 Skladba 3D scény

5.1.1 Outdoor scéna

V rámci outdoor scény patří mezi hlavní elementy především cesty a jednotlivé budovy. Za vznik cest je zodpovědná funkce „AddRoad“ a za vznik budov je zodpovědná třída „ParameterizedBuilding“. Ve třídě pro vznik budov je hlavní logika umístěna v konstrukčním skriptu, který je zavolán pro každou instanci a obsahuje tyto funkce:

- AddGround
- AddWall
- AddDoor
- AddWindow
- AddRoof
- AddBush

Funkce jsou navrženy tak, aby bylo možné podle jejich parametrů určovat vzhled budovy. Jako první je zavolána funkce „AddGround“, která vytváří podstavu budovy. Po vytvoření podstavy budovy je její transformace předána funkci „AddWall“, která na základě vstupního parametru vytvoří kostru objektu. Kostra objektu se může skládat až ze tří kvádrů. Po vytvoření kostry objektu je pomocí funkce „AddDoor“ vybráno vhodné umístění dveří. Umístění dveří určuje náhodná funkce ve vymezeném rozsahu šířky budovy. Transformace dveří je předána funkci „AddWindow“, která slouží k umístění oken. Tato funkce má nejvíce vstupních parametrů, mezi které patří umístění podstavy budovy, změna měřítka kostry objektu, počet pater a materiál pro určení vzhledu budovy. Po vytvoření oken je přidána střecha pomocí funkce „AddRoof“, která stejně jako funkce pro umístění oken přijímá parametr pro počet pater. Pomocí parametru pro počet pater lze určit, kolik má budova pater, a tedy v jaké výšce má být střecha umístěna. Kromě parametru pro počet pater také přijímá transformaci kostry objektu, aby byla střecha umístěna na jednotlivé kvádry, ze kterých se kostra objektu skládá. Na závěr je zavolána funkce s názvem „AddBush“, která podle umístění a změny měřítka podstavy budovy vytváří keře v okolí budov.

Kromě modelu budovy jsou v outdoor scéně také pohybující se chodci a vozidla. V obou případech je za tuto funkčnost zodpovědná funkce, která je v prostředí již zabudovaná a jmenuje se „Event Tick“. Pomocí této funkce je pravidelně v definovaném

intervalu jedné vteřiny volána funkce „AddMovementInput“. Je tak zajištěno, že se chodci i vozidla pohybují a je tím také možné určit, jakým směrem se mají chodci či vozidla pohybovat.

Za předpokladu, že se chodci i vozidla pohybují, je důležité řešit kolize, které mezi nimi mohou vznikat. Pro tyto případy má každá instance okolo sebe trigger, který když protne některý jiný chodec nebo vozidlo, dojde k vyvolání funkce s názvem „OnComponentBeginOverlap“. Reakce na tuto událost je v případě obou kolidujících instancí jejich odstranění ze scény pomocí funkce „DestroyActor“.

5.1.2 Indoor scéna

Implementace indoor scény je složena ze dvou částí a to ze statické a dynamické. Statická část je řešena pomocí funkce s názvem „AddFloorSquare“, která vytváří chodby spojené do čtverce. Realizace chodeb spojených do čtverce je pomocí cyklu, který po dosažení jedné čtvrtiny z celkové délky čtverce změní směr generování o 90 stupňů.

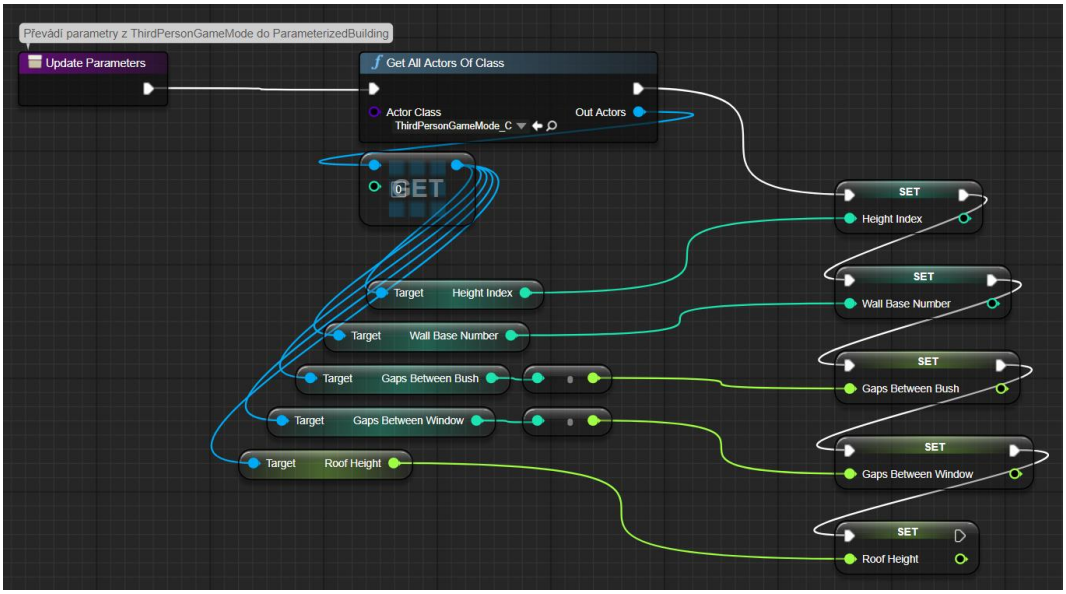
Dynamická část je řešena pomocí triggerů, které jsou umístěny v každé části chodby dynamicky generované scény. Znamená to tedy, že triggery nejsou umístěny ve statické části, ale pouze v dynamické. Za dynamickou část je zodpovědná funkce s názvem „AddFloorTile“, která při každém průchodu triggerem vygeneruje další část chodby před uživatelem. Při generování dalších částí chodeb je aplikováno počítadlo, které když dosáhne hodnoty deset, dojde k náhodnému určení, kterým směrem se bude dále chodba generovat, zda rovně, doleva nebo doprava. Na základě těchto tří možností je nutné zajistit, aby zvolená možnost nevedla k tomu, že se protnou dvě různé chodby. Před každým rozhodováním, kterým směrem se bude dále chodba generovat, se získají souřadnice aktuální části chodby. Poté se určí jedna ze tří možností a ověří se, zda na pozici, kam se má generovat existuje po dalších deseti iteracích počítadla jiná část chodby nebo ne. Pokud existuje, ověří se další možnosti a za předpokladu, že všechny možnosti vedou ke kolizi, vygeneruje se taková část chodby, která již dál nevede.

5.2 Parametry generování scény

Parametry, které jsou určeny pro generování scény, jsou hlavní sadou hodnot, které při běhu aplikace musí existovat. Existence těchto parametrů začíná při každém spuštění aplikace a to pomocí funkce „Event Begin Play“, ve které jsou postupně volány další funkce. Mezi tyto funkce patří také „ResetParameters“, která jako vstupní parametr přijímá externí data a nevrací žádnou hodnotu. Při běhu funkce jsou jednotlivé vstupní

parametry vyhodnoceny podle jejich indexu, a proto by mělo být dodržováno správné pořadí dat, které tato funkce přijímá.

Další zpracování skupiny parametrů nastává při vzniku budovy v outdoor scéně, která pro svou aktualizaci parametrů využívá funkci s názvem „UpdateParameters“. Aktualizace parametrů znamená získání parametrů z jedné třídy do třídy druhé. První třída se nazývá „ThirdPersonGameMode“, která je aktivní po celou dobu běhu aplikace a plní úlohu tzv. herního módu. Druhá třída, která přijímá parametry, se nazývá „ParameterizedBuilding“ a slouží ke tvorbě budovy.



Obr. č. 18 – Funkce pro aktualizaci parametrů

Na obrázku č. 18 je pět parametrů, přičemž na levé straně jsou proměnné třídy herního módu a na pravé straně proměnné třídy ke tvorbě budovy. Význam jednotlivých proměnných vysvětluje tabulka č. 2.

Tabulka č. 2 – Přehled parametrů, které využívá funkce pro aktualizaci parametrů

Název proměnné	Název parametru	Datový typ
HeightIndex	Parametr pro určení počtu pater budov	Celé číslo
WallBaseNumber	Parametr pro určení počtu kvádrů tvořících budovy	Celé číslo
GapsBetweenBush	Parametr pro určení velikosti mezer mezi keři u budov	Celé číslo
GapsBetweenWindow	Parametr pro určení velikosti mezer mezi okny budov	Celé číslo
RoofHeight	Parametr pro určení výšky střešní stěny budov	Desetinné číslo

Pro některé parametry z přehledové tabulky je umožněno, aby byla změněna jejich hodnota pomocí klávesové zkratky během hry. Mezi tyto parametry patří parametr pro určení počtu pater budov a také parametr pro určení počtu kvádrů tvořících budovy. První z parametrů je možné pomocí klávesy „H“ inkrementovat a pomocí klávesy „J“ dekrementovat. Druhý parametr lze inkrementovat klávesou „K“ a dekrementovat klávesou „L“.

5.3 Dynamické generování scény

Dynamické generování scény je implementováno pomocí funkce s názvem „DynamicGeneration“, která je zavolána pokaždé, když uživatel projde triggerem. V rámci této funkce je vyhodnoceno, o jaký trigger se jedná, a podle toho jsou předány vhodné parametry dalším funkcím. Vyhodnocení o jaký trigger se jedná je řešeno pomocí funkce „GetWorldRotation“, která vrátí hodnotu typu vektor. Hodnota vektoru je složena ze tří souřadnic X, Y a Z, ve které je dále vyhodnocena Z-ová souřadnice. Podle Z-ové souřadnice je určeno, kterým směrem uživatel scénou prochází a na základě toho je generována příslušná část scény.

Po průchodu triggerem jsou zavolány další funkce s odpovídajícími parametry, mezi které patří v outdoor scéně funkce „AddRoad“ a „AddParamBuilding“. Kromě dalších funkcí je také po průchodu triggerem zavolána funkce „DestroyComponent“ pro konkrétní trigger. Je tomu tak z důvodu, aby byl každý trigger využitý pouze jednou a nedocházelo tak k nadbytečnému generování scény například pokud by se uživatel vrátil na původní místo.

5.4 Vzorkování vegetace podle mřížky s parametrem

Implementace vzorkování vegetace podle mřížky s parametrem je řešena v rámci funkce s názvem „AddLowTriangleTree“, která je zavolána při každém spuštění aplikace. Jako hlavní jádro této funkce jsou dva cykly, kterým předchází nastavení několika proměnných. Mezi tyto proměnné patří:

- LowTriangleTreeStartXY(float), který zastává výchozí pozici určující počátek
- GapsBetweenLowTriangleTree(int), který určuje mezery mezi jednotlivými stromy

- LowTriangleTreeJitter(float) označující odchylku od původní pozice, která pomocí náhodné funkce nabývá různých hodnot
- LowTriangleTreeLastIndex(int), který určuje velikost prostoru, do kterého se vzorkuje vegetace

Pro vhodné rozložení vegetace v okolí města je nutné pro určení výchozí pozice znát, na kterých souřadnicích se nachází město. Pokud bude například město v okolí souřadnic x=0, y=0, bude to znamenat jako výchozí pozici určit bod, který bude na souřadnicích x=-100 000, y=-100 000. Jak bylo zmíněno v 4.4, výchozí pozice je spodní levý roh celé mřížky, která se v každé iteraci cyklu rozšiřuje až do souřadnic x=100 000, y=100 000.

Mezery mezi jednotlivými stromy jsou vypočítány na základě parametru pro určení hustoty vegetace, který byl zmíněn v 4.2. Hodnota parametru může být například 50, označující procento hustoty vegetace. Pro určení mezer mezi stromy je použitý následující výpočet:

GapsBetweenLowTriangleTree = (100 – VegetationDensity) * 100

Za předpokladu, že hodnota parametru hustoty vegetace je 50, mezery mezi stromy budou 5000.

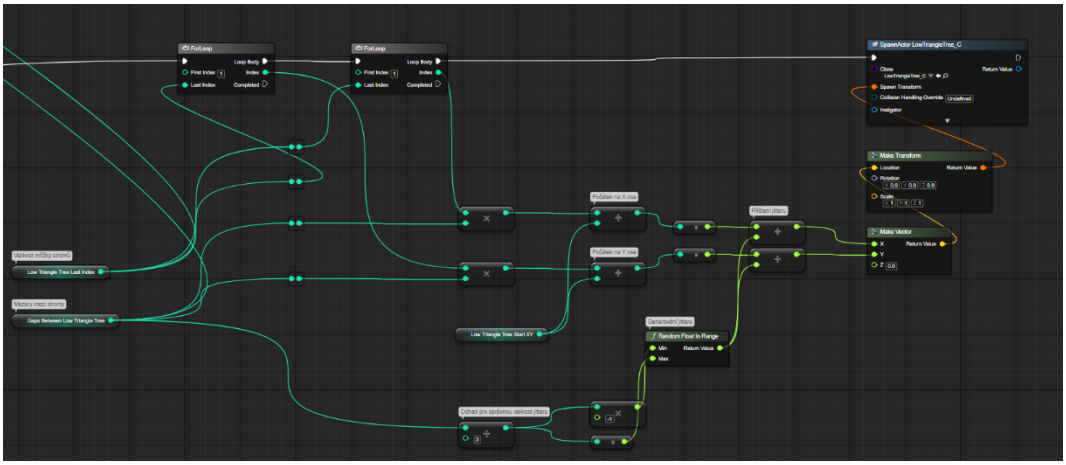
Odchylka od původní pozice je závislá na velikosti mezer mezi jednotlivými stromy, aby nedocházelo ke kolizím s jinými stromy. Hodnota odchylky je určena náhodnou funkcí v určitém rozpětí minima a maxima. Maximum je vypočteno jako hodnota mezer mezi jednotlivými stromy děleno třemi. Minimum je hodnota maxima násobena mínus jedničkou.

Velikost prostoru, do kterého se vzorkuje vegetace, se odvíjí od hodnoty parametru pro určení hustoty vegetace. Je tak zajištěno, že při měnící se hustotě vegetace je velikost mřížky stále stejná.

Po nastavení všech parametrů je přistoupeno k jádru funkce, kterými jsou dva cykly. V případě obou cyklů je první index roven jedné a poslední index roven hodnotě, která je uložena v proměnné LowTriangleTreeLastIndex. V každé části cyklu je vytvořen aktér „LowTriangleTree“ a to na pozici, která je dána předpisem:

LowTriangleTreeStartXY + (GapsBetweenLowTriangleTree * index) + LowTriangleTreeJitter

Jádro funkce je zobrazeno na obrázku č. 19.



Obr. č. 19 – Jádro funkce pro vzorkování vegetace

5.5 Konfigurace parametrů pomocí nadstavbové aplikace

Pro zajištění možnosti parametrizovat scénu je navržen a implementován nástroj, pomocí kterého lze zadat hodnoty jednotlivých parametrů a spustit generování vytvářené 3D scény. Jedná se o nadstavbovou aplikaci vytvořenou v programovacím jazyku C++ s využitím doplňkových knihoven, pomocí kterých je vývoj aplikací snazší a méně časově nákladný. Za pomocí QT frameworku je možné vytvořit konfigurační menu, které dává uživateli možnost rozšířit funkčnost prostředí Unreal Engine 4.

Nadstavbová aplikace slouží k několika specifickým úkolům, pomocí kterých lze dosáhnout vhodné parametrizace scény. Hlavním úkolem je zajistit, aby bylo možné, že každý uživatel aplikace bude moci vybrat určité obrázkové soubory, které se následně budou zobrazovat v indoor scéně při průchodu galerií. V prostředí Unreal Engine 4 nelze snadnou cestou zařídit, aby byly při každém spuštění aplikace uživatelem vybrány jiné obrázkové soubory, z toho důvodu je celá problematika řešena externí kompilací před každým spuštěním.

Výhodou prostředí Unreal Engine 4 je možnost využívat externí kompilaci pomocí spuštění dávkových souborů pro jednotlivé činnosti. V první části je nutné, aby byly obrázkové soubory, které jsou běžně ve formátu JPG převedeny do formátu prostředí, tedy „uasset“. K tomuto úkolu slouží zavolání aplikace UE4Editor.exe pomocí příkazové řádky Windows s určitými parametry. Prvním parametrem je zvolení cesty k projektu, který byl vytvořen v prostředí Unreal Engine 4 s koncovkou „uproject“, dalším parametrem je zvolení konkrétního typu akce, která v tomto případě je „-run=ImportAssets“. Jako další parametr je uvedená cesta k obrázkovému souboru,

který má být do cílové aplikace transformován a to „-source=“CESTA_K_SOUBORU\soubor.jpg““ a jako poslední je nutné určit cílový adresář, kam se má výsledný „uasset“ soubor umístit, který je určený parametrem „-dest=“/CÍL““.

Pokud jsou obrázkové soubory úspěšně transformovány do adresáře dané aplikace, lze spustit samotný dávkový soubor, který provede externí kompilaci. Samotný dávkový soubor má několik parametrů, pomocí kterých je určeno, jak bude aplikace vytvořena.

Tento úkol provádí volání dávkového souboru RunUAT.bat s parametrem BuildCookRun, následné určení cesty k projektu, který byl vytvořen v prostředí Unreal Engine 4 „-Project=“CESTA_K_SOUBORU\SOUBOR.uproject““. Poté je vymezený parametr „NoP4“, „-NoCompileEditor“, „-Distribution“, „-TargetPlatform=Win64“, „-Platform=Win64“, „-ClientConfig=Shipping“, „-ServerConfig=Shipping“, „-Cook“, „-Build“, „-Stage“, „-Pak“, „-Archive“. Následně je nutné určit adresář, do kterého bude cílová aplikace vytvořena „-ArchiveDirectory=“CESTA_K_APLIKACI““ a parametry „-Rocket“, „-Prereqs“ a „-Package“.

6 Výsledky a hodnocení

Výsledky a hodnocení jsou složeny z několika částí, v první řadě se jedná o hodnocení vzhledu výsledné modelované a následně vizualizované scény. V rámci hodnocení vzhledu scény je posouzeno, zda je výsledný model scény rozumný, což znamená, že komunikace navazují, stromy nerostou uvnitř domů a obrazy nejsou pověšeny ve vzduchu. Dále je hodnocena rychlost aplikace, určující, jak parametrizace scény ovlivňuje výpočetní výkon. V rámci testování byl použit osobní počítač s CPU Intel Core i5 2.30 GHz, grafickým adaptérem Intel Graphics 5500, RAM 8 GB a operačním systémem Windows 7 64 bit.

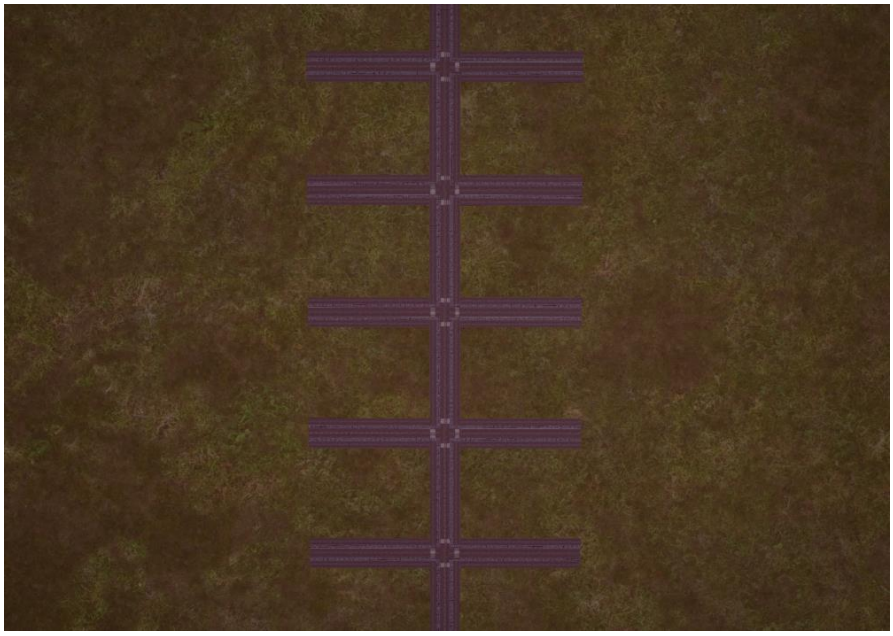
6.1 Vizuální výsledky

Vizuální hodnocení bylo provedeno jednotlivě pro outdoor a indoor scénu. Bylo hodnoceno postupné generování grafických elementů scény na základě zadaných parametrů. Výsledná aplikace byla testována na běžném desktopovém počítači a notebooku, ale i s použitím setu pro virtuální realitu obsahující hlavový displej s ovladači a detektory pohybu. V průběhu testování byly snímány obrázky přímo z prostředí vytvořené aplikace. Ke staticky generovanému modelu scény byly následně přidány dynamické prvky, např. vozidla a chodci v outdoor scéně.

6.1.1 Outdoor scéna

Generování cest a křižovatek

Generování modelu města začíná vytvořením cesty, na které jsou v pravidelných intervalech umístěny křižovatky. Parametry generování nabízí možnost dynamicky měnit délku cest a rozpětí mezi křižovatkami. Na obr. č. 20 je zobrazena vygenerovaná cesta při zachování defaultní hodnoty parametru P_{vk}.



Obr. č. 20 – Generování cesty

Generování budov do prostoru podél cesty

Podle délky a šířky vygenerované cesty dojde k vygenerování odpovídajícího počtu budov. Za předpokladu, že parametr P_{vk} bude zvětšován, bude se zvětšovat i vzdálenost mezi jednotlivými křižovatkami a zároveň počet budov v dané oblasti. Na obr. č. 21 je zobrazena vygenerovaná cesta včetně vygenerování modelů budov.

Obr. č. 21 – Generování cesty a budov

Při zvýšení hodnoty parametru P_{vk} jsou výsledné vzdálenosti mezi křižovatkami větší a zároveň dojde k vygenerování většího množství budov. Na obr. č. 22 je zobrazena vygenerovaná cesta včetně budov se zvýšenou hodnotu parametru P_{vk}.

Obr. č. 22 – Generování cesty a budov s větší velikostí cesty

Generování chodců a vozidel

Provoz ve městě se skládá z chodců a vozidel. Umístění chodců a vozidel do scény je definováno pomocí souřadnic, které odpovídají chodníku v případě chodců a silnici v případě vozidel. Pomocí směrového vektoru je určen směr chůze nebo jízdy. Každý chodec se řídí pravidly, která zajišťují plynulost pohybu.

Obr. č. 23 – Generování chodců a vozidel

Parametrizace provozu ve městě

Hustota provozu ve městě je určena počtem chodců a vozidel. Za předpokladu, že bude přibývat chodců i vozidel, hustota provozu se bude zvyšovat. Při vysokém provozu se zvyšují nároky na výpočetní výkon a zároveň jsou častěji řešeny kolize mezi chodci a vozidly.

Obr. č. 24 – Provoz ve městě při snížení parametru pro určení počtu chodců a vozidel

Obr. č. 25 – Provoz ve městě při zvýšení parametru pro určení počtu chodců a vozidel

Parametrizace vzhledu budov

Vzhled budov je ovlivněn několika parametry, mezi které patří:

- P_{pb}
- P_{ktb}
- P_{mob}
- P_{sb}

Pomocí těchto čtyřech parametrů je možné zařídit odlišnost vzhledu budov a skladba parametrů může být například: P_{pb}: 1, P_{ktb}: 1, P_{mob}: 400, P_{sb}: 2. Vzhled vytvořeného modelu scény podle daných parametrů je na následujícím obrázku.

Obr. č. 26 – Ukázka parametrizace scény pro nízkopodlažní budovy

Pro odlišení vzhledu lze vytvořit novou scénu, ve které bude skladba parametru tato: P_{pb}: 4, P_{ktb}: 3, P_{mob}: 100, P_{sb}: 1. Viz následující obrázek.

Obr č. 27 – Ukázka parametrizace scény s vysokopodlažními budovami

Parametrizace vegetace

Vegetace se nachází především v oblastech okolí města. Jedná se o tři druhy stromů a trávu. Se snižující hodnotou mezer mezi stromy dochází k častějšímu vzorkování a zvýšení nároků na výpočetní výkon. Stejně tak, jako parametrizace jiných částí scény, tak i parametrizace vegetace je určena na základě konfiguračního souboru, ve kterém uživatel určí její hustotu. Na obrázcích č. 28-30 je uvedený výstup vzorkování vegetace s parametrem P_{vmm}, který nabývá hodnot 30, 50 a 70 procent.

Obr. č. 28, 29, 30 – Výstup vzorkování vegetace

Vzorkování vegetace s parametrem P_{vmm} z pohledu pozorovatele je viditelné na obrázcích č. 31-33.

Obr. č. 31, 32, 33 – Výstup vzorkování vegetace z pohledu pozorovatele

Průchod outdoor scénou s dynamickou změnou parametrů

Při průchodu outdoor scénou lze pomocí klávesových zkratek pravidelně upravovat hodnoty jednotlivých parametrů. Za pomoci dynamického generování scény před uživatelem je možné při průchodu městem za běhu aplikovat zvyšování či snižování parametrů a získat tak možnost dynamicky parametrizovat scénu. Obrázek č. 34 ukazuje scénu před aplikováním dynamické změny parametrů. Následuje obrázek č. 35, na kterém je uveden průchod scénou a několik změn parametrů v jednotlivých částech města.

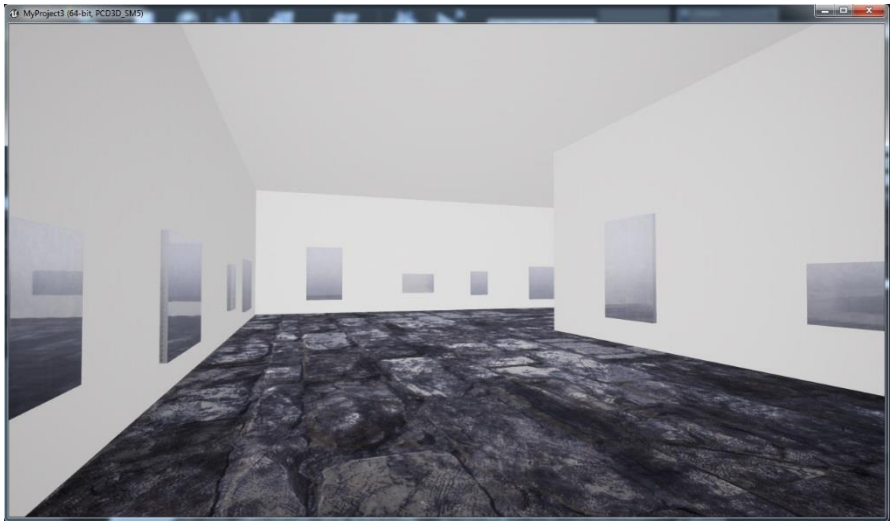
Obr. č. 34 – Průchod outdoor scénou s výchozí hodnotou parametrů

Obr. č. 35 – Průchod outdoor scénou se změnou parametrů

6.1.2 Indoor scéna

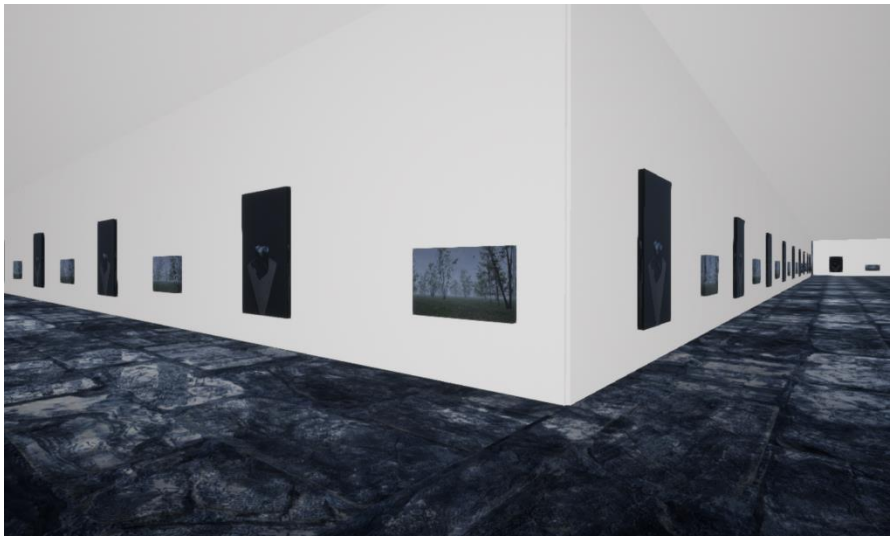
Průchod galerií s obrazy

V rámci indoor scény je implementována galerie, ve které jsou na stěnách obrazy. Do obrazů je možné importovat pomocí nadstavbové aplikace vybrané obrázkové soubory ve formátu jpg. Při pohledu do galerie si lze všimnout, že jednotlivé obrazy nemají stejnou velikost a nejsou ani umístěny v podobné výšce. Před vytvořením každé části chodby je rozhodnuto, jak budou obrazy velké a kde se budou přesně nacházet. Rozměry obrazů jsou určeny náhodnou funkcí s předem definovaným rozsahem, nemůže se tedy stát, že by byl vygenerovaný obraz s nevhodnými rozměry. Umístění obrazu je řízenou podobnou funkcí, která má předem definované rozpětí umístění, kde se může výsledný obraz nacházet a je tak zajištěno, že obrazy nebudou protínat jiné obrazy, strop nebo podlahu.



Obr. č. 36 – Průchod galerií s obrazy bez přidání vlastních obrazů

Před spuštěním aplikace je pomocí nadstavbové aplikace vhodné, aby si uživatel vybral některé ze svých oblíbených obrázkových souborů, které se následně budou zobrazovat v galerii. Cílem průchodu galerií je setkávání se se známými fotografiemi z reálného života, které by si již uživatel za jiných okolností nemusel prohlédnout, ale díky průchodu galerií je bude mít stále v paměti.



Obr. č. 37 – Průchod galerií s obrazy s přidáním vlastních obrazů

6.2 Rychlost zobrazení

Při testování vytvořené aplikace byl kladen důraz na zajištění rychlosti zobrazení scény, která umožnuje uživateli reálnou interakci. Vzhledem k tomu, že v rámci vytvořené aplikace je uživateli umožněno určovat, jak bude scéna vypadat, lze očekávat rozdíly v rychlosti. Rychlosti zobrazení závisí na složitosti scény, především na počtu generovaných budov, chodců či vozidel.

V případě generování budov v outdoor scéně dojde při navýšení parametru P_{vk} k vygenerování více budov. Dojde tedy v určitých situacích k drobnému zpomalení při průchodu městem, a to v takových situacích, kdy jsou tyto budovy dynamicky generovány. Nejedná se pouze o počet generovaných elementů, další příčinou změny rychlosti aplikace je také funkčnost, která zajišťuje dynamické generování scény, která je před uživatelem.

Dynamicky generovaná scéna je generována až v situaci, kdy je uživatel v určitém místě, zatímco staticky generována scéna se generuje ještě před spuštěním aplikace. Lze tedy pozorovat při průchodu galerií v indoor scéně nebo při průchodu městem v outdoor scéně, že v situacích, kdy se generuje nový obsah, je rychlost aplikace snížena. Samotné snížení rychlosti aplikace je ovlivněno především výpočetním výkonem počítače, na kterém je aplikace spouštěna.

Na celé rychlosti aplikace se velmi pozitivně projeví aplikace algoritmu LOD, jehož pomocí je urychleno vykreslování vzdálenějších objektů. Průchod scénou je plynulejší, především při otáčení se nebo při potřebě rychle vykreslit velké části scény je rychlost zobrazení výrazně zvýšena.

6.3 Interakce

Vytvořená aplikace v prostředí Unreal Engine 4 umožňuje uživateli průchod indoor i outdoor scénou a lze jej realizovat na klasickém desktopovém počítači, ale i ve virtuální realitě pomocí sady HTC Vive. Za předpokladu, že uživatel spouští aplikaci přímo z prostředí Unreal Engine 4, je nutné zvolit způsob spuštění vytvářené aplikace.

Obr. č. 38 – Spuštění aplikace v módu virtuální reality (5).

Za předpokladu, že je aplikace spouštěna již po externí kompilaci, nelze tuto možnost nastavit, nicméně při korektním spojení sady HTC Vive s počítačem se aplikace automaticky spustí v módu virtuální reality. Aby se mohl uživatel v indoor i outdoor scéně pohybovat, jsou nastaveny standardní klávesy (W, A, S, D) a podpora myši pro rozhlížení. Pro případ virtuální reality lze využívat MotionController (Left, Right) a pro případ rozhlížení je namísto myši využito brýlí pro virtuální realitu. Na obrázku č. 39 je zobrazeno testování interakce ve virtuální realitě.

Obr. č. 39 – Interakce ve VR

Obr. č. 40 – Detail interakce ve VR

6.4 Odezva

Hodnocení odezvy aplikace je především závislé na výpočetním výkonu počítače, na kterém je aplikace spuštěna, odezva aplikace se liší podle uživatelem určených parametrů, podle kterých je sestavena scéna. V případě outdoor scény je vyšší nárok na výpočetní výkon než v případě indoor scény, je to způsobena především množstvím elementů, které se nachází v obou scénách. Za předpokladu, že uživatel určí nízkou hustotu provozu a nízký počet generovaných budov při průchodu outdoor scénou, je při využití výše zmíněné sestavy odezva rychlá. Při zvyšování parametrů, které určují hustotu provozu, se odezva aplikace prodlužuje, stejně tak jako při zvýšení parametru, který určuje množství generovaných budov.

V případě indoor scény se lze setkat se změnou odezvy v případě, kdy uživatel postoupí do části galerie, která se generuje dynamicky, nicméně vzhledem k menšímu množství elementů, než v outdoor scéně jsou v indoor scéně odezvy velmi nízké.

Řešení problému dlouhé odezvy je možné navýšením výpočetního výkonu anebo optimalizací algoritmů, které jsou zodpovědné za generování scény. Jako jedna z možností z vhodných optimalizací je využitý algoritmus změny úrovně detailu (LOD), pomocí kterého se určité části scény generují s nižším detailem a některé části scény se negenerují, především ty, které jsou od uživatele vzdálené.

Jako další možnost z vhodných optimalizací lze využít omezení rozsahu parametrizace, kterou může uživatel určit. Znamenalo by to, že uživatel má omezenou horní hranici počtu budov nebo omezený počet maximální hustoty provozu. Díky snížení počtu chodců a vozidel ve městě dojde k výraznému snížení odezvy aplikace a je tedy možné bez problému procházet outdoor scénou.

6.5 Konfigurace

Konfigurace slouží pro generování parametrizované scény podle parametrů definovaných v konfiguračním souboru. Uživatel tak může určit, jak bude scéna vypadat ještě před jejím vygenerováním. V případě indoor scény je možné zvolit konkrétní obrazové soubory zobrazované při průchodu galerií.

Konfigurační soubor je externí soubor, který je při spuštění aplikace načten a dále zpracován. Pro zpracování souboru v prostředí Unreal Engine 4 je nutné dodržet určité typy formátů. Jako jedna z možností je formát XML, který je možné v daném prostředí načíst a dále zpracovávat. Jednotlivé části XML textu jsou následně přetransformovány do podoby jednotlivých datových typů, podle kterých je určen vzhled scény. Struktura XML textu je tato:

```
<scene>
  <Pvk>9</Pvk>
  <Ppb>4</Ppb>
  <Pktb>3</Pktb>
  <Pmkb>400</Pmkb>
  <Pmob>100</Pmob>
  <Psb>1</Psb>
  <Pos>1</Pos>
  <Pchv>15</Pchv>
  <Pvmm>30</Pvmm>
  <Pvg>40</Pvg>
</scene>
```

7 Závěr

Práce je zaměřena na prozkoumání oblasti počítačové grafiky se specializací na procedurální generování modelu prostorové scény. V úvodní teoretické části jsou shrnuty základní principy a metody pro generování 3D scény a jejich použití v různých oblastech lidské činnosti. Pro možnost odzkoušení principů a testování možností byla v prostředí Unreal Engine navržena a implementována aplikace se dvěma typy prostředí, indoor a outdoor.

Teoretická část shrnula základy pro generování scény především se zaměřením na princip parametrizace scény, který společně s dynamickým generováním umožňují vznik dvou typů prostředí. V prostředí indoor scény je možné personifikovat scénu podle potřeb jednotlivých uživatelů, zatímco prostředí outdoor scény vizualizuje architektonické koncepty. Pro možnost personifikovat scénu bylo v indoor scéně využito možnosti měnit texturu jednotlivých obrazů z libovolných externích fotografií, díky čemuž může každý uživatel určovat, jak bude scéna vypadat. Vizualizace architektonických konceptů byla v rámci outdoor scény zajištěna pomocí celkem deseti parametrů, jejichž hodnotu může uživatel určovat před generováním scény.

Pro odzkoušení principů a testování možností byla vytvořena aplikace, pomocí které lze jednotlivé principy a metody pro generování 3D scény implementovat a následně ověřit, jaké výsledky přinesly. Implementace v prostředí Unreal Engine usnadnila dynamicky generovat scénu pomocí triggerů, zajistila možnost přidávat do scény modely přímo z prostředí anebo z externích zdrojů a umožnila změnu vzhledu scény pomocí načítání hodnot parametrů z externího souboru. Dosažené výsledky pomocí vytvořené aplikace ověřily, že model scény je rozumný. Komunikace navazují, stromy nerostou uvnitř domů a obrazy nejsou pověšeny ve vzduchu. Byla ověřena interakce ve virtuální realitě, která úzce souvisí se směrem dalšího výzkumu.

Za pomoci procedurálního generování lze vytvářet scénu libovolných rozměrů a složitosti a lze jej vnímat jako vhodný směr dalšího výzkumu. Jako vhodné téma by mohla být tvorba realističtější scény, která by umožňovala lepší interakci s uživatelem, případně scénu dále rozšířit o možnost interakce několika uživatelů zároveň. Kromě toho by jako další zajímavé téma mohlo být rozšíření modelovaných prostředí o další typ, který by mohl reprezentovat domov uživatele, kde by se nacházely běžné předměty z lidského života.

Seznam literatury

1. Epic Games, Inc. *Blueprints*, 2015 [cit. 22. 07. 2018]. Dostupné z: <https://docs.unrealengine.com/en-us/Engine/Blueprints>
2. Epic Games, Inc. *Construction Script*, 2015 [cit. 22. 07. 2018]. Dostupné z: <https://docs.unrealengine.com/en-us/Engine/Blueprints/UserGuide/UserConstructionScript>
3. Epic Games, Inc. *Open World Tools*, 2015 [cit. 11. 08. 2017]. Dostupné z: <https://docs.unrealengine.com/latest/INT/Engine/OpenWorldTools/index.html>
4. Epic Games, Inc. *Procedural Buildings*, 2012 [cit. 12. 08. 2017]. Dostupné z: <https://docs.unrealengine.com/udk/Three/ProceduralBuildings.html>
5. Epic Games, Inc. *QuickStart*, 2017 [cit. 29. 07. 2018]. Dostupné z: <https://docs.unrealengine.com/en-us/Platforms/SteamVR/QuickStart>
6. Epic Games, Inc. *Trigger Actors*, 2015 [cit. 21. 07. 2018]. Dostupné z: <https://docs.unrealengine.com/en-us/Engine/Actors/Triggers>
7. Epic Games, Inc. *What is Unreal Engine 4*, 2018 [cit. 22. 07. 2018]. Dostupné z: <https://www.unrealengine.com/en-US/what-is-unreal-engine-4>
8. GREEN, Dale. *Procedural Content Generation for C++ Game Development*; Packt Publishing Ltd.; 2016; ISBN 978-1-78588-671-3
9. GREGORY, Jason. *Game engine architecture*. Second edition. Boca Raton: CRC Press, 2014. ISBN 978-1466560017
10. KELLY, George. MCCABE, Hugh. *A survey of procedural techniques for city generation.* The ITB Journal, 2006, 7.2: 5.
11. MÜLLER, Pascal, et al. *Procedural modeling of buildings.* In: Acm Transactions On Graphics (Tog). ACM, 2006. p. 614-623.
12. MÜLLER, Pascal, et al. *Procedural 3D Reconstruction of Puuc Buildings in Xkipché.* In: VAST. 2006. p. 139-146.
13. NORD, A. *Example Based Procedural Distribution Tool*, 2018 [cit. 21. 07. 2018]. Dostupné z: <http://www.diva-portal.org/smash/get/diva2:763228/FULLTEXT01.pdf>
14. PARCHMANN, R. and DUSKE, J., 2018 [cit. 22. 07. 2018]. *Self-embedding indexed grammars*. Dostupné z: <https://www.sciencedirect.com/science/article/pii/0304397586901477>
15. RÖTTGER, Stefan, et al. *Real-time generation of continuous levels of detail for height fields.* Universität Erlangen-Nürnberg, 1998.

16. SCHWARZ, M., MÜLLER, P. 2015. *Advanced procedural modeling of architecture*. ACM Transactions on Graphics, 34, 4 (Proceedings of SIGGRAPH 2015), Article 107
17. SHAKER, Noor; TOGELIUS, Julian; NELSON, Mark J. *Procedural content generation in games*. New York, NY: Springer Berlin Heidelberg, 2016. ISBN 978-3-319-42714-0
18. Unity Technologies, Products - Unity, 2018 [cit. 22. 07. 2018]. Dostupné z: <https://unity3d.com/unity>
19. ŽÁRA, Jiří, Bedřich BENEŠ a Petr FELKEL. *Moderní počítačová grafika.* Praha: Computer Press, 1998. ISBN 80-7226-049-9.
20. LECHNER, Thomas; WATSON, Ben; WILENSKY, Uri. *Procedural city modeling.* In: In 1st Midwestern Graphics Conference. 2003.

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2017/2018

Studijní program: Aplikovaná informatika
Forma: Kombinovaná
Obor/komb.: Aplikovaná informatika (ai3-k)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Ondřej Tomáš	Novodvorská 1130 179, Praha - Braník	I1500504

TÉMA ČESKY:

Procedurální generování 3D scény v grafických enginech

TÉMA ANGLICKY:

Procedural generation of 3D scene in graphics engines

VEDOUCÍ PRÁCE:

Ing. Bruno Ježek, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl: Prozkoumat oblast počítačové grafiky zaměřenou na procedurální generování prostorové scény. Odzkoušet vybrané postupy a metody ve zvoleném 3D enginu, a navrhnout vhodnou aplikaci pro testování.

1. Prozkoumat principy a metody procedurálního modelování 3D scény.
2. Vytvořit přehled postupů a nástrojů procedurálního modelování dostupných ve vybraném 3D enginu.
3. Prozkoumat příklady užití procedurálního modelování vizuální scény a navrhnout vlastní aplikaci použití a testování popsaných principů.
4. Navrhnout a implementovat algoritmy pro řešení procedurálně generované scény.
5. Navrhnout a provést testování, zda je procedurální generování vhodnou metodou pro zvolené použití.
6. Zhodnotit dosažené výsledky.

SEZNAM DOPORUČENÉ LITERATURY:

GREGORY, Jason. Game engine architecture. Second edition. Boca Raton: CRC Press, 2014. ISBN 978-1466560017
D. Green; Procedural Content Generation for C++ Game Development; Packt Publishing Ltd.; 2016; ISBN 978-1-78588-671-3
Shaker, Noor; Togelius, Julian; Nelson, Mark J. Procedural content generation in games. New York, NY: Springer Berlin Heidelberg, 2016. ISBN 978-3-319-42714-0

Podpis studenta: ..... **Datum:** 19.10.2017

Podpis vedoucího práce: ..... **Datum:** 19.10.2017