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

GENEROVÁNÍ 2D MAP PRO POČÍTAČOVÉ HRY

2D MAP GENERATION FOR COMPUTER GAMES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **KRYŠTOF GLOS**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL MATÝŠEK**
SUPERVISOR

BRNO 2021

Abstrakt

Tato práce se zaměřuje na jednotlivé metody procedurálního generování z hlediska herního průmyslu, zejména v oblasti vytváření map. Cílem této práce, je vytvoření hry žánru Colony-sim, využívající procedurálního generování obsahu.

Zaměřil jsem se na analýzu různých způsobů vytváření obsahu a rozbor metod pomocí kterých lze automatizovaně generovat herní prostředí.

Zvolenou metodou pro tvoření herní oblasti byl Perlinův šum, s následným využitím výškových map. V práci představuji algoritmus, jenž na základě vstupních parametrů generuje funkční, unikátní a neopakující se mapy, což zajišťuje opakovatelnou hratelnost. Klíčovým výstupem této práce, je hra implementující procedurální algoritmus generující mapy, který je aplikovatelný v širokém spektru 2D her.

Abstract

This work focuses on individual methods of procedural generation in the context of the gaming industry, particularly in the realm of map creation. The aim of this study is to develop a Colony-sim genre game utilizing procedural content generation.

I have concentrated on analyzing various methods of content creation and scrutinizing techniques through which gaming environments can be automatedly generated.

The chosen method for crafting the gaming area was Perlin noise, coupled with the subsequent utilization of height maps. In this work, I present an algorithm that, based on input parameters, generates functional, unique, and non-repeating maps, ensuring repetitive playability. The key outcome of this study is the implementation of a game incorporating a procedural algorithm for map generation, applicable across a broad spectrum of 2D games.

Klíčová slova

Procedurální generování obsahu, 2D mapy, 2D hry, počítačové hry, hry o přežití

Keywords

Procedural content generation, 2D maps, 2D games, computer games, survival games

Citace

GLOS, Kryštof. *Generování 2D map pro počítačové hry.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Matýšek

Generování 2D map pro počítačové hry

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Kryštof Glos
21. ledna 2024

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

Obsah

Úvod

Procedurálně vytvářený obsah v herním průmyslu je velmi důležitou součástí her už po několik let. Mnoho her postavených na tomto principu, se již prosadilo na trhu a stále více se uplatňuje. Náhodné vytváření obsahu se používá například na tvoření herních map, věcí v místnosti, skládání různých dopředu vytvořených místností tak aby vznikla jedinečná mapa. Takto implementované hry mají výhodu v opakované hratelnosti a nepředvídatelnosti.

Cílem této práce je návrh a vytvoření prototypu 2D hry, v herním enginu Unity, založené na procedurálním generování herního obsahu. Samotný generační algoritmus pracuje s Perlinovým šumem a výškovými mapami, ty slouží k vytvoření elevace různých bodů na mapě.

Inspirací na téma hry je počítačová hra RimWorld, která se řadí do her typu Colony-sim. Jedná se o žánr, ve kterém hráč ovládá skupinu lidí, kolonii, kterou se snaží, pomocí dobrého vedení, dovést k nějakému danému cíli. V navržené hře je úkolem hráče dostat kolonisty do stavu prosperity a mimo stav nouze o potravu a zdroje. Pohyb kolonistů využívá takzvané NavMesh agenty, kteří spolupracují s vytvořenou navigační sítí (NavMesh).

Vytváření mapy je stěžejní částí celé hry, určuje zdroje pro hráče a tudíž částečně i obtížnost celé hry. Ve hře se objevují nepřátelé, kteří mají jednoduchou umělou inteligenci a jsou jednou z překážek, se kterou se hráč musí během hry vypořádávat. Implementace generačních algoritmů, AI nepřátel a dalšího je popsána v kapitole **??**.

Hra je implementována v herním enginu Unity, které je v dnešní době, jedním z nejvíce používaných vývojových nástrojů pro vývoj her. Mezi další oblíbená vývojová prostředí se řadí například, GameMaker, Unreal Engine, Godot. O vývoji v jednotlivých enginech pojednává kapitola 2.

Po dokončení implementační části bylo zahájeno experimentování s herními systémy a následovné uživatelské testování. Průběh a výsledky této části práce jsou dále popsány v kapitole **??**.

Kapitola 1

Vytváření obsahu v herním světě

Ve světě her jsou dvě možnosti jak přistupovat ke tvoření obsahu, jedním z těchto způsobů je tradiční nebo také mechanické, je jednodušší, vzhledem k tomu že není potřeba žádný algoritmus, ale pracnější. Další možností vytváření obsahu je pomocí metod implementující náhodné nebo také procedurální generování obsahu.

Hry které mají pouze dvě dimenze se nazývají 2D (z anglického two dimensions). Je mnoho žánrů 2D her, RPG (role playing game) hry na hrdiny s příběhem, strategií, Co-op (kooperační) které jsou postavené na spolupráci více hráčů, survival (hry o přežití), colony-sim (z anglického colonization simulation) které mají simulovat kolonizaci ovládanou hráčem a tak dále. Tato bakalářská práce se zabývá hrou žánru colony-sim. Je mnoho způsobů jak vyvíjet takovou hru, nejčastěji se používají takzvané herní enginy, které takové vyvíjení hry ulehčují a jsou na to stavěné.

1.1 Způsoby generování obsahu

V této části porovnáme mechanické generování obsahu s procedurálním, vysvětlíme co je lepší kdy použít a jaké známe metody pro procedurální generování. Následující tři body popisují faktory, které je třeba zvážit při rozhodování, zda bude využita nějaká procedurální metoda generování, nebo bude lepší použít manuální design úrovní a obsahu:

Žánr tvořené hry Při vývoji například takové hry z prvního pohledu (FPS) hry, u které záleží hlavně na ovládání a souboji hráče proti hráči, není třeba vytvářet mapy a další obsah procedurálně. Většinou stačí vytvořit například pouze jednu úroveň a v takovém případě není třeba používat procedurální generování. Při hrách, které závisí na okolí, surovinách a přežití kde každá okolnost nějak ovlivňuje hráče, už procedurální generování hraje větší roli. Zároveň pro vytváření obsahu jako je text, může být PG velmi obtížnou volbou, neboť v RPG hrách bývá textová část velmi důležitá a generování textu je stále dost obtížné

Opakovaná hratelnost Některé hry jsou dělané tak, že čím déle hráč hraje stejnou úroveň (anglicky level), tím více se zlepšuje a je za to například odměňován je právě manuální tvoření úrovní lepší než procedurální. Naopak u jiných her, které mají úrovně procedurálně generované, většinou hráč danou úroveň zvládne, pokračuje na další a nepředpokládá se že se k ní bude ještě vracet.

Aspekt designu hry Jestliže hra závisí z velké části na jedné úrovni s jejími mechanikami, vlastnostmi a obsahem, pak je lepší ji vytvářet mechanicky a doladit všechny vlastnosti a interakce s hráčem.

1.1.1 Mechanické generování obsahu

Mechanický typ generování je jedním z nejobvyklejších tvoření obsahu ve hrách. Používá se převážně v žánrech, jako je RPG (Role Play Game), RTS (Real Time Strategy) a další, ve kterých pozice objektů a struktura mapy hraje velkou roli a bez lidské tvorby by nebylo dosaženo potřebných výsledků. Toto tvoření lze interpretovat jako proces u něhož se návrhář za pomoci různých nástrojů, které postupně aplikuje, snaží dosáhnout požadovaného výsledku. Jde tedy o metodu ručního vytváření obsahu kde designér, nebo grafik navrhuje a postupně vytváří úroveň, či jinou část hry tak, aby vyhovovala potřebám, ať už se jedná o pozici stromu, nebo o to co hráči sděluji NPC.

Díky tomuto přístupu nehrozí nesrovnalosti ve výsledku, například se nemůže stát že určitá část mapy bude nedostupná, nebo úplně nesmyslná. Další výhodou je, že výsledek bude přesně takový jak byl naplánovaný, avšak takovéto tvoření je u větších her, kde je nutná opětovná hratelnost, velmi časově náročné.

1.1.2 Procedurální generování obsahu

Tento typ vytváření obsahu se používá ve více žánrech, ale asi nejznámější z hlediska generování map je Roguelike, kde každá nová hra má unikátní náhodnou mapu. Procedurální generování se ovšem nepoužívá pouze na generování map, ale také na vytváření objektů, jako jsou stromy, textury, animace, text a další. Procedurální generování obsahu není úplně to stejné, jako náhodné generování obsahu. Více do hloubky je tento typ modelování krajiny a textur rozebrán v kapitole Procedurální generování.

V zásadě je to proces obrácený jak u mechanického generování obsahu. Uživatel sice stále definuje různé nástroje, které jsou použity pro vytváření obsahu, ale nikoli pro své vlastní použití, ale naopak je vytváří pro algoritmus. Uživatel dále určuje pravidla podle kterých se generátor musí řídit tak, aby se dobral kýženého výsledku.

Obrázek 1.1: Příklad hry Roguelike žánru jménem Binding of Isaac, vpravo nahoře je vidět mapa dungeonu, která je procedurálně vygenerovaná.

Příklad procedurálního generování vegetace

Simulace lesních prostředí má mnoho aplikací, od zábavní po výzkumné modelování. Pro lepší představu uvádím příklad od Newlandse a Zaunera. [14]

Program má funkci procedurálního generátoru lesů na mapě, cílem tohoto programu je náhodně naskládat stromy s rozumnými rozestupy od sebe. Metody pro vegetaci se dají opět řadit jako procedurální a mechanické. Neprocedurální přístup nabízí velkou kontrolu nad celým procesem a velký detail modelů, výměnou za cenu vyžadujícího značného množství uživatelských zásahů, často na úrovni vysazování rostliny po rostlině. V případě procedurálního generování, se jedná o modely které jsou zkonstruovány algoritmicky a kontrolovány skrze parametrické hodnoty, bez nutnosti vyšší úrovně manuálního vstupu nebo specializace v daném oboru.

Jednou z nejvíce prozkoumanou modelační technikou postavenou na rekurzivních hierarchiích je koncept L-systémů. Existuje mnoho typů a rozšíření těchto systémů[17], včetně stochastických, parametrických, diferenciálních [18], citlivých na okolí [19], nebo otevřených [13].

Distribuce stromů na scéně lze udělat například pomocí simulace ekosystému. To vytváří více realistickou distribuci vegetace, než náhodné rozestavení, protože vznikají přirozená chování, jako je shlukování druhů a oblasti negativního růstu kolem stromů. [14] Jedná se o simulace, ve kterých každá rostlina žije vlastní život, je ovlivňována ostatními rostlinami a vnějšími podmínkami. [2]

Při Inicializaci simulace se pro každý specifický druh stromu vytvoří určité množství stromů (pro výpočet slouží rovnice 1.1) ve věku $a_i \in \langle 0, a_M \rangle$- kde a_M je maximální věk stromu pro daný druh a jsou náhodně rozestavěny po oblasti od největšího po nejmenší. [14]

$$n_I = \frac{d_0 \cdot \rho}{n_T} w^2 \tag{1.1}$$

Hodnota n_I představuje počet instancí k vytvoření, w je šířka scény, d_0 původní hustota pro objekt a ρ je parametr hustoty jednotlivých objektů.

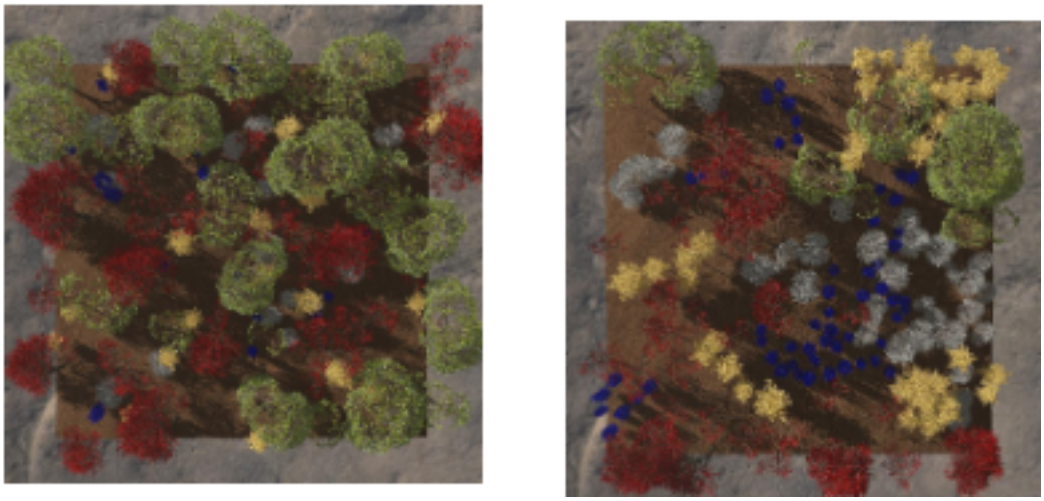
Po úspěšné inicializaci simulace běží N kroků, kde N_Y kroků utváří rok. Pro každý krok se děje následující:

1 Pokud je konec roku, všechny stromy vysejí počet nových rostlin, v kruhu okolo nich.

2 U každého páru stromů je rostlina s menší důležitostí odstraněna.

3 Stromy starší než jejich maximální věk, jsou považovány za mrtvé a odstraněny.

4 Všechny rostliny rostou (jejich věk se zvýší o 1).



(a) n=0 (b) n=1000

Obrázek 1.2: Vzhled ekosystému po inicializaci a rozložení vegetace po prvních 1000 iteracích. Je vidět že se vytváří shluky rostlinných druhů, tyto klastry zůstávají, pouze se mění jejich pozice. Obrázky převzaty z práce[1].

[[chyba odkazu]]

1.2 Procedurální generování v herním průmyslu

Algoritmů na generování obsahu existuje mnoho, každý používá jiné nástroje, ale všechny se musí podrobovat pravidlům která stanovuje programátor a podle kterých se řídí. Je více způsobů a míst kde se dá procedurální generování uplatnit, různé způsoby a důvody jsou popsány v této kapitole.

1.2.1 Text

Skoro všechny hry používají text. Z důvodu že každá informace v textu musí odpovídat realitě, je nutno velké množství omezení pro generování. Například když je v textu informace že král je mrtvý [11], musí být toto tvrzení pravdivé.

[1] https://arxiv.org/pdf/2208.01471.pdf

Velkým plusem procedurálního generování textu je vyprávění [4], takto vytvořené příběhy jsou často kreativnější a zajímavější, než ty co by vytvořil člověk, neboť lidé mají sklony psát příběhy které již slyšeli, nebo ze svých zkušeností, což dost omezuje nápady.

1.2.2 Krajina a úrovně

Nejvíce obvyklý obsah který se ve hrách generuje, a který je zároveň hlavním zaměřením této práce, jsou krajiny a úrovně. Generování lokací, úrovní, nebo obsahu mapy lze jak u 2D her, tak u 3D her. Za úroveň nebo oblast lze označovat otevřené, třeba krajina s lesy, i uzavřené prostranství, vnitřek budovy, nebo jeskyně. Tato část PG je rozvedená a podrobněji popsaná v následující kapitole **??**.

1.2.3 Textury

Jednou z nejčastějších metod, která se používá na tvoření textur, jsou L-systémy, nebo Perlinův šum, který je detailněji popsán v sekci 3.3.

1.2.4 Zvuky a hudba

Většina her má soundtrack a zvukové efekty. Soundtrack obvykle nemá nijak zvlášť přísná pravidla, ale zvukové efekty musejí být výstižné a odpovídající akci v daný moment.

Jukebox [3] je model který dokáže generovat hudbu se zpěvem v originální nezpracované formě zvukových dat, s délkou v řádu minut, i s určením žánru a vokálního stylu. Modelů jako je tento již existuje více, avšak zatím to nejsou plně hodnotné soundtracky pro hry a ještě chvíli potrvá, než bude možné jednoduše vygenerovat hudbu a efekty pro hru pomocí pouhého nástroje.

Kapitola 2

Enginy na vývoj her

Herní engine představuje platformu složenou z interagujícího softwaru, který dohromady vytváří integrovaný celek a umožňuje spouštění samotných her. Herní engine se skládá z několika částí s přesně specifikovanou funkcionalitou: rendering, fyzika, síťování, zvuk atd. [15]

Platforem na vývoj her existuje mnoho, některými z nich jsou například Unity, Construct 2, MonoGame, Unreal Engine, nebo GameMaker Studio 2. Každý herní engine je v něčem jiný a tudíž se hodí na jiné žánry, nebo styly her. Při rozhodování, který engine použít, se z hlediska vývojáře musí zohlednit vícero faktorů, například podporovaný programovací jazyk, nebo platformu na kterou je hra vyvíjena. [24]

2.1 Construct 2

Tento engine dovoluje lidem, kteří nejsou programátoři, vytvářet 2D hry. Používá drag and drop editor pro všechnu logiku založenou na událostech a chování. Může být rozšířen a skriptován pomocí JavaScriptu.

I když Construct tvrdí že zveřejňují hry na většině mobilních a desktopových platformách, jejich primární cíl je HTML5/JavaScript. Tudíž jakákoliv verze která není ve vyhledávači je obsažena v DOM a obalovacím rozhraní umožňujícím použití JavaScriptu. Tato architektura obecně snižuje výkon. [1]

2.2 Unreal engine

Podporuje multiplatformní vydávání her, jmenovitě DirectX, OpenGL, nebo WebGL.

Jedná se o engine který je zdarma, ale pouze pro nekomerční užití a ve všech ostatních případech licencování softwaru za malé předplatné a licenční poplatek. Přes to, že původně byl vyvinut pro podporu *Unreal* her z první osoby a neměl tolik nástrojů jako Unity, vyrostl Unreal Engine do podoby velmi výkonného enginu, schopného podporovat jakýkoli žánr her. [1]

Skriptování je v enginu pomocí jazyka C++.

2.3 Unity

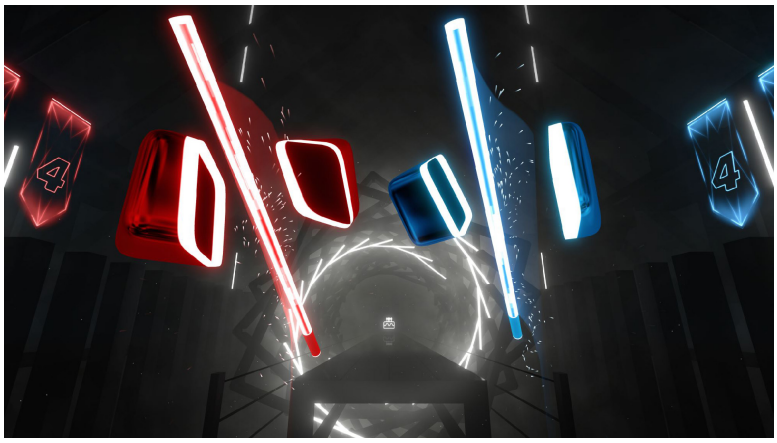
Unity je multiplatformní herní engine, podporující vývoj her na Windows, Linux i macOS. Vyvinula ho společnost Unity Technologies v roce 2005. Oproti jiným herním enginům podporuje vývoj her ve 2D, 3D, rozšířenou realitu (AR), nebo virtuální realitu (VR).

Unity má více plánů které mohou vývojáři využívat, jedná se o personal, pro, enterprise a industry. Ze základu je pro všechny vývojáře zdarma k užívání, ale po překročení 100 000$ za posledních dvanáct měsíců, je nutné pořídit si jeden z placených plánů. Tyto plány mají i jisté výhody, jako je třeba Havok Physics, přidávající robustní detekci kolizí a fyzikální simulace, technickou podporu, nebo prioritní frontu na zákaznickou podporu. [23]

Další výhodou je integrované fyzikální jádro PhysX, které pracuje v reálném čase a je vyvíjeno společností Nvidia. Toto jádro zahrnuje efektivní podporu pro multithreading a využívá akceleraci fyzikální simulace prostřednictvím GPU. Programování je v enginu zařízené pomocí jazyka C#.

Z unity vzešlo mnoho oblíbených herních titulů, řadí se mezi ně hra ve virtuální realitě Beat Saber 2.2, mobilní hra Pokémon Go 2.3, nebo multiplatformní Cuphead 2.1.

Obrázek 2.1: Ukázka hry Cuphead



Obrázek 2.2: Ukázka hry Beat Saber

Obrázek 2.3: Ukázka mobilní hry Pokemon Go

Kapitola 3

Procedurální generování

Tato kapitola popisuje metody pro generování geometrie, vegetace, celulární automaty, perlinův šum a použití procedurálního generování v herním průmyslu, do detailu popisuje PG krajiny, generování fraktálů.

Procedurální modelování je téma které se aktivně zkoumá už přes čtyřicet let. Myšlenka je, jak již bylo zmíněno, aby obsah který se vytvářel ručně, dal modelovat pomocí navržené procedury automaticky. Takovýto přístup se již uplatnil na generování například textur, geometrických modelů, zvukových nahrávek, nebo animací. V roce 1980 se začalo pracovat s různými metodami na vytváření terénu, jako hory, pláně a jezera. Začal se také řešit růst rostlin a obecně práce s přírodou. [22]

Roden and Parberry [21] pojmenovávají tento druh algoritmů *amplifikační algoritmy (amplification algorithms)*, přijímají menší množství vstupních informací, které zpracují a vracejí větší objem dat na výstupu. Hendrikx et al. [7] pojímají procedurální generování jako alternativu k mechanickému navrhování obsahu, ale kladou důraz na zdokonalování a přidávání parametrů umožňujících zásah návrháře do takto vygenerovaných objektů.

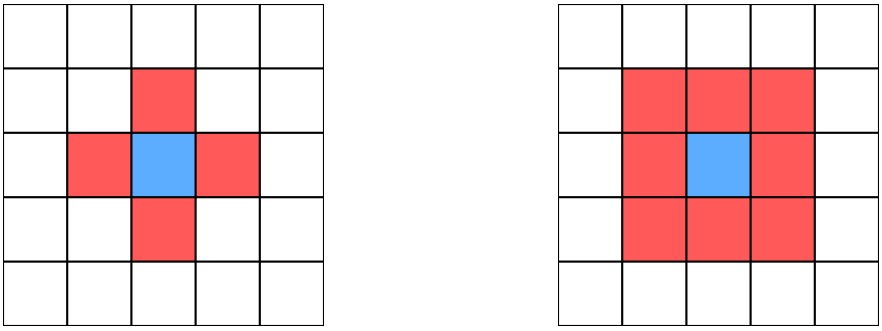
3.1 Celulární automaty pro PG

Celulární automaty se ve hrách používají intenzivně zejména pro modelování týkajícího se systémů v prostředí, jako jsou teplo, oheň, déšť, tlak a exploze. Zatím podle průzkumu nejsou známé žádné hry, které by postavili generování celého 2D herního světa, pouze pomocí celulárních automatů. Momentálně existují webové stránky, které možnost generování malých map pomocí mřížek navrhují, ale není jich mnoho a neexistuje žádné spolehlivé ohodnocení těchto algoritmů. [8]

Původně byly CA vymyšleny Johnem Von Neumannem jako formální model sebereprodukujících se organismů. Šlo o dvou-dimenzionální celulární automat, kde každá buňka, tzv. cell, je malý čtverec na velkém čtverečkovaném papíru. Každá buňka má dva možné stavy, černý a červený, které jsou určeny jejich sousedstvím. V John Von Neumannově teorii, je sousedství tvořené čtyřmi přilehlými čtverci a na obrázku 3.1a jsou vyznačeny červenou barvou. [5]

Nejznámější celulární automat byl vytvořen v roce 1970 britským matematikem Johnem Hortonem Conwayem, který se jmenoval Game Of Life. Stejně jako Von Neumannův byl i tento automat dvou-dimenzionální a mohli nabývat pouze hodnot živá, nebo mrtvá. Využívá Moorovo sousedství, které oproti Von Neumannově považuje za sousední buňky všech osm přilehlých, vyobrazené na obrázku 3.1b. Fungování automatu je následovné, buňka zůstává

naživu, pokud má dvě, nebo tři sousedící buňky živé. Což simuluje, že buňka nepřežije pokud je osamělá, ale zároveň pokud je okolí přeplněné organismy, tak je utlačována. Další pravidlo je, že pokud je libovolná buňka mrtvá, může se "narodit", pokud jsou v sousedství alespoň tři živé buňky. Toto pravidlo má simulovat rození, kde každá buňka musí mít tři rodiče. Automat díky těmto jednoduchým pravidlům dokáže vytvářet simulace které působily jako živý organismus. [5]



(a) Von Neumannovo sousedství (b) Moorovo sousedství

Obrázek 3.1: Sousedství z pohledu Von Neumanna a Moora

3.2 L-systémy

L-systémy (Lindenmayerovy systémy) jsou formálním nástrojem [17], který se používá pro modelování vývoje rostlin a buněčných struktur. Tyto systémy byly zavedeny biologem Aristidem Lindenmayerem v roce 1968. [6] Jedná se o paralelní řetězce přepisující systémy, za účelem modelovat růst celulárních organismů. L-systém $\mathcal{L}$ je entice

$$\mathcal{L} = \langle M, \omega, R \rangle,$$

kde M je abeceda L-systému, ω je axiom a R je množina pravidel přepisování. Abeceda obsahuje parametrizované moduly $M = A(P), B(P), \ldots$, kde $P = p_1, p_2, \ldots, p_n$ jsou modulové parametry, jako jsou rotace, zvětšení, zmenšení. Axiom $\omega \in M^+$ je neprázdná sekvence modulů a M^+ jsou všechny možné prázdné řetězce z M. Pravidla pro přepisování mají následující formu:

$$id_1 : A(P) : cond \rightarrow x, x \in M^*,$$
$$id_1 : A(P) : cond \rightarrow x, x \in M^*,$$
$$\ldots$$

kde M^* jsou všechny možné řetězce z M včetně prázdného ϵ. Pravidlo id_i přepisuje znak na levé straně z abecedy $A(P)$ posloupností písmen z pravé strany, pokud je podmínka *cond* pravdivá. Modul, který se nenachází na levé straně pravidla, se nazývá *terminální symbol*, neboť se nemůže dál měnit, všechny ostatní moduly se nazývají *neterminální symboly*.[20]

Každé písmeno má vlastní pravidlo derivace řetězce, ale probíhá paralelním provedením aplikovatelných pravidel, z množiny R pro každé písmeno které obsahuje. Produkční pravidla přepisují začínající symbol sekvencí modulů a pokračují v úspěšných derivacích $\omega \Rightarrow m_1 \Rightarrow m_2 \Rightarrow \ldots$, dokud není možné žádný další modul přepsat (řetězec končí pouze

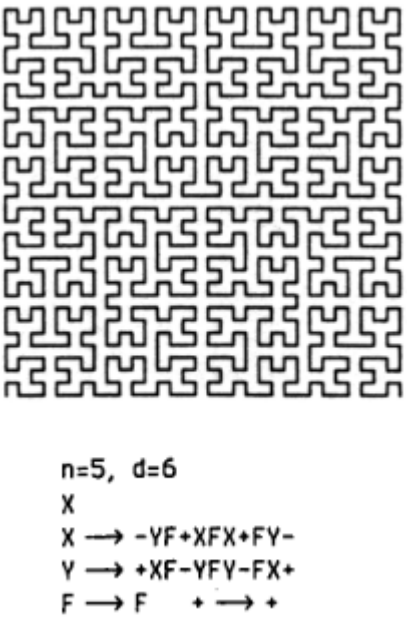
terminálními symboly), řetězec modulů je prázdný, kvůli aplikaci pravidla epsilon, nebo byl proces ukončen kvůli maximálnímu počtu iterací, které stanovil uživatel.[9]

Rekurze nastává v L-systémech tehdy, když se symbol z levé strany objevuje na pravé straně toho stejného pravidla (i když ne přímo). Nedeterminismus povoluje více pravidel pro jeden znak z abecedy. Toto navíc vyžaduje specifikaci pravděpodobnosti jejich aplikace.[10]

3.2.1 Geometrie pomocí L-systémů

Pro tvoření geometrie z textových řetězců, je každý řetězec reprezentován želvou, která vytváří geometrické symboly jako čáry, nebo dokonce 3D geometrii. Ve 2D má želva stav $S(p, 0)$ kde $p = [x, y]$ je její pozice a 0 je směrový vektor, který udává směr jejího pohybu.

Želva sekvenčně čte písmena interpretovaného řetězce z modulu od začátku po konec, kde každé písmeno je interpretované jako příkaz. Písmeno F si překládá jako "pohyb od p směrem k 0 o vzdálenosti d, která je zadaná a nakresli linku mezi starou pozicí a novou."Příkazy $+(\alpha)$ a $-(\alpha)$ mění směr pohybu želvy, jejím otočením doleva, či doprava o α. Cokoliv v závorkách $[M^+]$ je geometricky interpretováno jako větev vygenerované struktury. Výhodou je že ne všechna písmena abecedy musí mít nastavenou geometrickou interpretaci, pokud ji nemají, tak je želva ignoruje. [17]



Obrázek 3.2: Vygenerovaných obrázek L-systémy želví interpretací, jsou zde vidět jednotlivé posuvy i hodnoty o které se posouvá, převzatý ze článku[1]

[[chyba odkazu]]

3.3 Šumy

Šumy se v počítačové grafice využívají například pro přidání kvalitních detailů do syntetický vytvořených obrázků. Perlinův šum, navržený Kenem Perlinem [16], se v dnešní době

[1] http://algorithmicbotany.org/papers/graphical.gi86.pdf

používá ve vytváření procedurálních textur včetně mraků, vln, tornád, raketových cest, atd. Tato kapitola popisuje různé druhy šumů a jejich použití.

3.3.1 Definice šumu

3.3.2 Perlinův šum

3.4 Výškové mapy

Výškové mapy představují dvoudimenzionální mřížky obsahující výškové hodnoty, jež jsou častým prvkem v modelování terénu. Tyto mapy se běžně využívají jako klíčový prvek pro reprezentaci základu terénu v herním průmyslu. Pro tvorbu výškových map existuje mnoho algoritmů.

Jedny z nejstarších algoritmů jsou metody založené na pododdělení. Segmenty v rámci vygenerované hrubé výškové mapy jsou iterativně rozdělovány, kde každá iterace navíc používá kontrolovanou náhodnost k přidávání detailů. Miller [12] popisuje některé varianty všeobecně známé metody středového posunu, ve které se výška nového bodu nastavuje na průměr hodnot jeho rohů v trojúhelníkovém nebo diamantovém tvaru, k němuž je přidán náhodný offset. Každou iterací se rozsah náhodných hodnot offsetu snižuje, podle parametru, který kontroluje hrubost výsledné výškové mapy.

Generování výškových map se v dnešní době provádí převážně pomocí fraktálních generátorů šumu, jako je Perlinův šum, který provádí generování šumu vzorkováním a interpolací bodů na mřížce náhodných vektorů.

Výškové mapy se mohou dále transformovat na základě běžných filtrací obrazu, například vyhlazování (smoothing), nebo simulací fyzikálních jevů, např. eroze. [22]

3.5 Procedurální generování krajiny

Procedurální generování krajiny se řadí ke složitějším tématům PG. Je tomu tak hlavně kvůli tomu, že se k tomuto typu generování většinou používají výškové mapy (height maps), podle kterých se ohodnocují jednotlivé pixely, jak je popsáno v sekci 3.4. Jakmile jsou všechny pixely ohodnoceny, jsou tři různé způsoby jak postupovat:

- ruční vykreslování textur,
- aplikování textur na ručně vytvořené regiony podle výšky,
- vygenerování textur po analyzování výšek na vygenerované výškové mapě.

První metoda je výhodná v tom, že textury ručně vykreslené lze udělat na míru a žádný algoritmus je nemůže replikovat. Bohužel jsou časově náročné a jejich kvalita přímo závisí na schopnostech výtvarníka.

Druhá metoda je podobná, nakreslí se barvy na určitá místa, kam si designer myslí že by se mohly hodit hory, řeky, nebo země. Jedná se o docela obvyklou metodu, která přináší velice kvalitní výsledky, ale stále se jedná o manuální techniku.

Třetí metoda používá data height mapy a vypočítává jakou texturu použít na které místo. Nízké hodnoty (tmavší místa) se používají například jako oceány, střední hodnoty se vyhodnocují jako země/tráva a na vysoké hodnoty (světlá místa) se nanášejí textury hor nebo kamení. Při generování ve 3D hrách lze ještě započítávat takzvané sklony (slopes),

díky kterým je možné oddělovat vyšší plochy od nižších pomocí dalších textur například kamene.

Vzhledem k tomu že první dva postupy vyžadují mechanické vykreslování buď barvy, nebo textur on designéra, nedá se o nich mluvit jako o čistě procedurálních metodách. Naopak postup třetí tomuto popisu zcela odpovídá, neboť kompletně závisí na height mapě a není nutná žádná akce návrháře [4].

Generování oblastí lze rozdělit následovně:

Vytváření půdorysu Zahrnuje vytvoření země, moří, řek, hor, atd. Všechny tyto oblasti jsou otevřeného typu. Také můžeme rozumět pod generováním půdorysu i vytváření uzavřeného typu oblastí, například rozložení jednotlivých místností jeskynního komplexu, vytvoření bludiště a jeho cest.

Přidání vegetace a objektů Tento bod v podstatě navazuje na předchozí, je potřeba abychom měli vytvořený půdorys, aby bylo kam přidávat a rozmísťovat další objekty. Jedná se o bod do kterého spadá vegetace, budovy, atd. Programu se zadávají různá omezení na množství vegetace, místa kam který objekt lze přidávat a další omezení.

(a) Vygenerovaný půdorys oblasti (b) Přidané stromy k půdorysu

Obrázek 3.3: Na obrázcích jsou vidět různé postupy generování obsahu, na obrázku a je vygenerovaný půdorys oblasti i s mořem a skálou, na obrázku b se k tomu přidaly stromy

[[informace o všemožných známých metodách procedurálního generování]]


KEEP
CALM
AND
START
WRITING

Literatura

[1] Andrade, A. Game engines: a survey. *EAI Endorsed Transactions on Serious Games*. EAI. Listopad 2015, sv. 2, č. 6. DOI: 10.4108/eai.5-11-2015.150615.

[2] Beneš, B. A stable modeling of large plant ecosystems. In: *Proceedings of the International Conference on Computer Vision and Graphics*. 2002, s. 94–101.

[3] Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A. et al. Jukebox: A Generative Model for Music. *ArXiv*. 2020, abs/2005.00341. Dostupné z: https://api.semanticscholar.org/CorpusID:218470180.

[4] Freiknecht, J. *Procedural content generation for games*. Mannheim, 2021. Disertační práce. Dostupné z: https://madoc.bib.uni-mannheim.de/59000/.

[5] Gong, Y. A survey on the modeling and applications of cellular automata theory. *IOP Conference Series: Materials Science and Engineering*. IOP Publishing. sep 2017, sv. 242, č. 1, s. 012106. DOI: 10.1088/1757-899X/242/1/012106. Dostupné z: https://dx.doi.org/10.1088/1757-899X/242/1/012106.

[6] Guo, J., Jiang, H., Benes, B., Deussen, O., Zhang, X. et al. Inverse Procedural Modeling of Branching Structures by Inferring L-Systems. *ACM Trans. Graph.* New York, NY, USA: Association for Computing Machinery. jun 2020, sv. 39, č. 5. DOI: 10.1145/3394105. ISSN 0730-0301. Dostupné z: https://doi.org/10.1145/3394105.

[7] Hendrikx, M., Meijer, S., Van Der Velden, J. a Iosup, A. Procedural Content Generation for Games: A Survey. New York, NY, USA: Association for Computing Machinery. feb 2013, sv. 9, č. 1. DOI: 10.1145/2422956.2422957. ISSN 1551-6857. Dostupné z: https://doi.org/10.1145/2422956.2422957.

[8] Johnson, L., Yannakakis, G. a Togelius, J. Cellular automata for real-time generation of. Září 2010. DOI: 10.1145/1814256.1814266.

[9] Lindenmayer, A. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of theoretical biology*. Elsevier. 1968, sv. 18, č. 3, s. 280–299.

[10] Lindenmayer, A. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*. 1968, sv. 18, č. 3, s. 280–299. DOI: https://doi.org/10.1016/0022-5193(68)90079-9. ISSN 0022-5193. Dostupné z: https://www.sciencedirect.com/science/article/pii/0022519368900799.

[11] Liu, J., Snodgrass, S., Khalifa, A., Risi, S., Yannakakis, G. N. et al. Deep learning for procedural content generation. *Neural Computing and Applications*. jan

2021, sv. 33, č. 1, s. 19–37. DOI: 10.1007/s00521-020-05383-8. ISSN 1433-3058. Dostupné z: https://doi.org/10.1007/s00521-020-05383-8.

[12] Miller, G. S. P. The Definition and Rendering of Terrain Maps. New York, NY, USA: Association for Computing Machinery. aug 1986, sv. 20, č. 4, s. 39–48. DOI: 10.1145/15886.15890. ISSN 0097-8930. Dostupné z: https://doi.org/10.1145/15886.15890.

[13] Měch, R. a Prusinkiewicz, P. Visual models of plants interacting with their environment. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1996, s. 397–410. SIGGRAPH '96. DOI: 10.1145/237170.237279. ISBN 0897917464. Dostupné z: https://doi.org/10.1145/237170.237279.

[14] Newlands, C. a Zauner, K.-P. *Procedural Generation and Rendering of Realistic, Navigable Forest Environments: An Open-Source Tool*. 2022.

[15] Nilson, B. a Söderberg, M. Game Engine Architecture. *Mälardalen University*. 2007.

[16] Perlin, K. An image synthesizer. *SIGGRAPH Comput. Graph.* New York, NY, USA: Association for Computing Machinery. jul 1985, sv. 19, č. 3, s. 287–296. DOI: 10.1145/325165.325247. ISSN 0097-8930. Dostupné z: https://doi.org/10.1145/325165.325247.

[17] Prusinkiewicz, P. Graphical applications of L-systems. In: *Proceedings of graphics interface*. 1986, sv. 86, č. 86, s. 247–253.

[18] Prusinkiewicz, P., Hammel, M. S. a Mjolsness, E. Animation of plant development. In: *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1993, s. 351–360. SIGGRAPH '93. DOI: 10.1145/166117.166161. ISBN 0897916018. Dostupné z: https://doi.org/10.1145/166117.166161.

[19] Prusinkiewicz, P., James, M. a Měch, R. Synthetic topiary. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1994, s. 351–358. SIGGRAPH '94. DOI: 10.1145/192161.192254. ISBN 0897916670. Dostupné z: https://doi.org/10.1145/192161.192254.

[20] Prusinkiewicz, P. a Lindenmayer, A. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.

[21] Roden, T. a Parberry, I. *From Artistry to Automation: A Structured Methodology for Procedural Content Creation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. 151–156 s. ISBN 978-3-540-28643-1.

[22] Smelik, R., Kraker, K. J. de, Groenewegen, S., Tutenel, T. a Bidarra, R. A Survey of Procedural Methods for Terrain Modelling. In:. červen 2009.

[23] Technologies, U. *Compare unity plans: Personal, pro, enterprise, industry*. 2024.

[24] Vohera, C., Chheda, H., Chouhan, D., Desai, A. a Jain, V. Game engine architecture and comparative study of different game engines. In: IEEE. *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. 2021, s. 1–6.