

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA JADERNÁ A FYZIKÁLNĚ INŽENÝRSKÁ

Katedra softwarového inženýrství

Studijní program: Aplikace informatiky v přírodních vědách

Specializace: —



Vývoj strategické hry na náhodně generované mapě

VÝZKUMNÝ ÚKOL

Vypracoval: Bc. Štěpán Bezděk

Vedoucí práce: RNDr. Zuzana Petříčková, Ph.D.

Rok: 2025

České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství

Akademický rok 2024/2025

ZADÁNÍ VÝZKUMNÉHO ÚKOLU

Student: Bc. Štěpán Bezděk
Studijní program: Aplikace informatiky v přírodních vědách
Název práce: Vývoj strategické hry na náhodně generované mapě

Pokyny pro vypracování:

1. Nastudujte problematiku procedurálního generování obsahu, především se zaměřením na využití při tvorbě herních světů v počítačových hrách.
2. Navrhněte strategickou hru, která bude využívat náhodně generované herní pole.
3. Implementujte generování herního pole pomocí procedurálních technik.
4. Implementujte základní herní mechaniky navržené hry (např. jednotky, ekonomika, boj).
5. Proveďte simulaci navržených herních principů a ověřte hratelnost navržené hry.

Doporučená literatura:

- [1] Salen, K., Zimmerman, E. Rules of Play: Game Design Fundamentals. MIT Press.
- [2] Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press.
- [3] Scheibenpflug, A. Evolutionary Procedural 2D Map Generation using Novelty Search.
- [4] Shaker, N., Togelius, J., Nelson, M. J. Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer.
- [5] Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., Worley, S. Texturing & Modeling: A Procedural Approach. Morgan Kaufmann.

Jméno a pracoviště vedoucího práce:

RNDr. Zuzana Petříčková, Ph.D.

Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Jméno a pracoviště konzultanta práce:

Ing. Vladimír Jarý, Ph.D.


Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Datum zadání výzkumného úkolu: 21. 10. 2024

Termín odevzdání výzkumného úkolu: 25. 8. 2025

V Praze dne 21. 10. 2024


.....
vedoucí práce


.....
garant programu


.....
vedoucí katedry

Prohlášení

Prohlašuji, že jsem svůj výzkumný úkol vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne

.....

Bc. Štěpán Bezděk

Poděkování

Děkuji ... za ...

Bc. Štěpán Bezděk

Název práce:

Vývoj strategické hry na náhodně generované mapě

Autor: Bc. Štěpán Bezděk

Studijní program: Aplikace informatiky v přírodních vědách

Specializace: –

Druh práce: Výzkumný úkol

Vedoucí práce: RNDr. Zuzana Petříčková, Ph.D.

Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze

Konzultant: Ing. Vladimír Jarý, Ph.D.

Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Abstrakt: Popis VÚ česky

Klíčová slova: Klíčová slova

Obsah

Úvod	7
1 Teorie	8
1.1 Procedurální generování obsahu (PCG)	8
1.1.1 Výhody procedurálního generování	8
1.1.2 Nevýhody a výzvy procedurálního generování	9
1.1.3 Procedurální generování ve hrách	9
1.2 Metody procedurálního generování map	10
1.2.1 Space Partitioning	10
1.2.2 Agent-Based generování	11
1.2.3 Grammar Algorithms	13
1.2.4 Výškové mapy	14
1.2.5 Fraktálové algoritmy	15
1.2.6 Wave Function Collapse (WFC)	15
1.2.7 GAN (Generative Adversarial Networks)	16
1.3 Porovnání metod	17
2 Návrh	18
2.1 Návrh strategické hry	18
3 Implementace mapy	19
3.1 Implementace generování herního pole	19
4 Simulace	20
4.1 Simulace a testování hry	20
Závěr	21
Literatura	21
Přílohy	24
A Název přílohy	24

Úvod

Zde napište text úvodu (1-3 strany, nerozdělujte na podkapitoly) nebo jej vložte ze samostatného souboru: např. příkazem `\input{uvod.tex}`.

Tato šablona je určena **pro elektronické odevzdání VÚ** (od roku 2023). Pokud byste si VÚ chtěli vytisknout, tak změňte řádky 1 a 10 ve zdrojovém kódu této šablony (\Rightarrow budou správně nastavené „prázdné zadní stránky“).

Kapitola 1

Teorie

1.1 Procedurální generování obsahu (PCG)

Zpracováno na základě: [1] [2].

Procedurální generování je metoda algoritmického vytváření dat. Procedurálně vytvořená data se od ručně vytvořených dat liší v tom, že jsou generována kombinací malého množství ručně vytvořených vstupních dat, na jejichž základě počítač podle daného algoritmu vytváří komplexnější výstup. Tento přístup je vhodný zejména pro aplikace, kde je vyžadováno velké množství různorodého obsahu nebo kde je ruční tvorba neefektivní.

Hlavní výhodou procedurálního generování je úspora času, lidských zdrojů a tím i finančních nákladů. Navíc umožňuje vytvořit obsah, který může být přizpůsoben konkrétním požadavkům nebo preferencím uživatele. Generovaný obsah je dynamický, což otevírá možnosti pro personalizaci zážitků, jako je například tvorba jedinečných map ve hrách nebo simulace náhodných prostředí.

1.1.1 Výhody procedurálního generování

Procedurální generování nabízí mnoho výhod, které ho činí atraktivním pro různé aplikace, hlavně ve vývoji her a designu:

- **Redukce nároků na úložiště:** Namísto ukládání velkého množství dat je možné ukládat pouze algoritmus a jeho vstupní parametry.
- **Variabilita:** Algoritmus může generovat nekonečné množství unikátních výsledků.
- **Dynamika:** Obsah se může přizpůsobovat v reálném čase na základě uživatelských interakcí nebo jiných faktorů.
- **Úspora lidských zdrojů:** Omezuje potřebu ruční práce při tvorbě rozsáhlých prostředí.

1.1.2 Nevýhody a výzvy procedurálního generování

Ačkoliv má procedurální generování mnoho výhod, existují také určité nevýhody a výzvy:

- **Kontrola kvality:** Výstup algoritmu nemusí vždy splňovat očekávání nebo být konzistentní s požadovanými normami.
- **Složitost implementace:** Návrh a ladění algoritmů mohou být časově náročné.
- **Předvídatelnost:** Přílišná náhodnost může vést k obsahu, který nedává smysl nebo není použitelný.
- **Závislost na vstupních datech:** Kvalita výstupu je silně ovlivněna kvalitou a rozmanitostí vstupních dat.

1.1.3 Procedurální generování ve hrách

Kapitola zpracována s pomocí: [1] [2].

Procedurální generování obsahu hraje klíčovou roli v moderním herním vývoji, zejména díky schopnosti efektivně vytvářet rozsáhlé a rozmanité herní světy. Tento přístup je oblíbený především pro hry s otevřeným světem, jako jsou *Minecraft*, *No Man's Sky* nebo série *Rogue-like* her, kde je kladen důraz na variabilitu a dynamické prostředí.

Procedurální generování umožňuje vytvořit herní světy, které jsou pro hráče vždy jedinečné, což zvyšuje atraktivitu a prodlužuje životnost hry. Například v *Minecraftu* se každý nový svět skládá z unikátní kombinace biomů, terénních útvarů a zdrojů, což hráče motivuje k dalšímu průzkumu. Podobně ve hře *No Man's Sky* je generováno celé univerzum s miliardami planet, z nichž každá má unikátní prostředí, faunu a flóru.

Hlavní výhodou využití PCG v herním designu je schopnost generovat obsah přizpůsobený hráčově zkušenosti. Například v akčních hrách mohou algoritmy procedurálního generování upravovat obtížnost úrovní v reálném čase na základě výkonu hráče. Tento adaptivní přístup může zajistit, že hra zůstane náročná, ale zároveň nefrustrující, což zvyšuje angažovanost hráče.

Dalším využitím PCG je tvorba náhodných misí nebo úkolů. Ty mohou být generovány tak, aby obsahovaly různé cíle, nepřátele nebo interaktivní objekty. Tento přístup umožňuje vytvořit dynamický herní zážitek, kdy žádná mise nebo průběh hry nejsou zcela stejné. Tento princip se často využívá například ve hrách typu *Diablo* nebo *Borderlands*, kde jsou zbraně a další vybavení generovány procedurálně.

I přes četné výhody s sebou procedurální generování přináší i určité výzvy. Například ve hře *No Man's Sky* čelili vývojáři kritice za přílišnou repetitivnost obsahu, přestože byl generován procedurálně. Dalším problémem je zajištění smysluplnosti a konzistence herního světa, což vyžaduje pečlivě navržené algoritmy a pravidla.

Procedurální generování textur a zvuků

Procedurální generování nachází své uplatnění také při tvorbě textur a zvuků, což významně přispívá k redukci nároků na úložný prostor. Textury povrchů, jako jsou dřevo, kámen nebo kov, mohou být generovány pomocí algoritmů, jako je **Perlin Noise** nebo **Simplex Noise**. Také je tímto přístupem generovat vegetaci, tak aby každý strom a keř vypadal unikátně například pomocí **Grammar Algorithms**. Tento přístup nejen šetří místo na disku, ale také zvyšuje variabilitu, protože každá textura může být jedinečná.

Podobně lze procedurálně generovat zvuky, jako je ambientní hudba nebo efekty prostředí (například šum deště, větru či zvuky zvířat). Použitím technik syntézy zvuku místo ukládání statických zvukových souborů je možné dynamicky přizpůsobit zvukové efekty aktuálním podmínkám ve hře, čímž se nejen šetří úložiště, ale také zlepšuje ponoření hráče do herního prostředí.

Tento přístup umožňuje nejen vytváření efektivnějších her z hlediska úložných nároků, ale také otevírá dveře k větší míře personalizace a přizpůsobení herního obsahu konkrétním hráčům.

1.2 Metody procedurálního generování map

V rámci této práce využijeme procedurální generování specificky pro vytvoření nové herní plochy pro každou hru. V této části se podíváme na některé možné techniky pro generování herních map. Hlavními požadavky na algoritmy je logické rozložení map, hratelnost a rychlost generování.

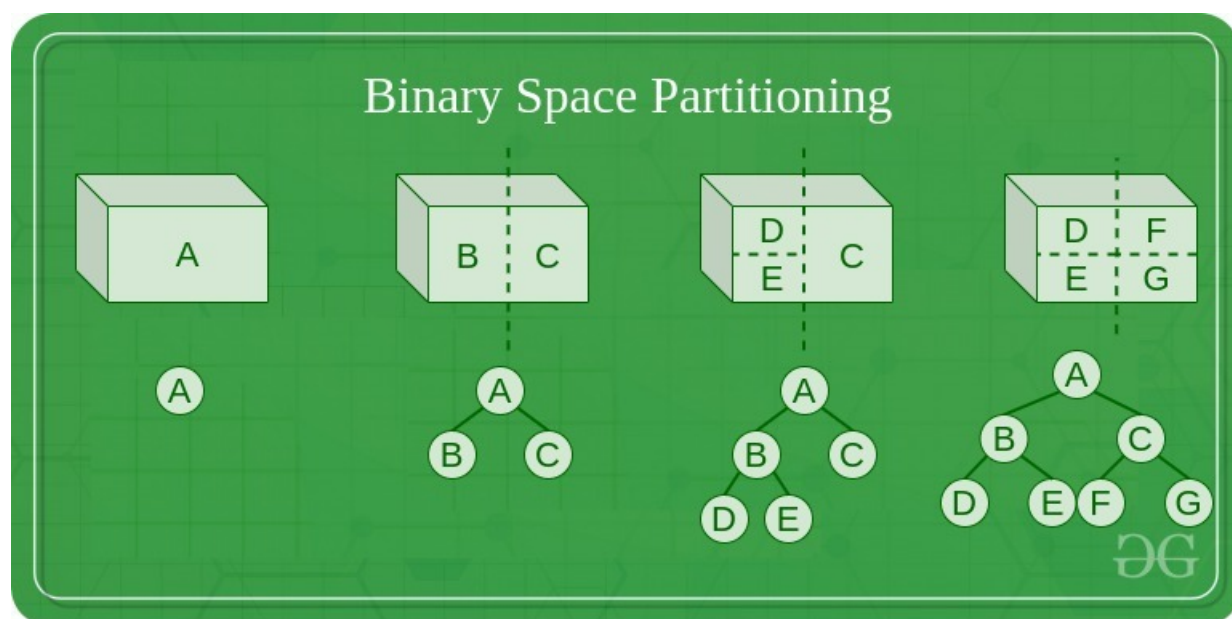
Algoritmy byly mimo jiné zpracovány na základě: [1].

1.2.1 Space Partitioning

Dělení prostoru je metoda, která rekurzivně rozděluje herní mapu na menší, disjunktní oblasti, přičemž každý bod mapy náleží právě jedné z těchto oblastí. Výsledkem je hierarchická struktura, obvykle reprezentovaná stromem, která umožňuje rychlé vyhledávání informací a manipulaci s daty.

Hlavním principem této metody je postupné dělení základního prostoru, dokud nejsou splněny předem stanovené podmínky, jako je například minimální velikost buňky nebo místnosti. Tento přístup nachází uplatnění nejen při generování herních map, ale také v grafických aplikacích, jako je ray-tracing, kde umožňuje efektivně organizovat a vykreslovat objekty ve scéně. [3].

Nejběžnější metoda pro rozdělování prostoru je *binární dělení prostoru (BSP)*. Jde o proces, který opakovaně rozděluje prostor na dvě části, dokud nejsou splněny určité požadavky. Ukázka *BSP* je zobrazena na obrázku 1.1. Další často využívanou variantou je *quadtree*, kde je prostor rozdělen na čtyři symetrické části. Hlavním



Obrázek 1.1: Ukázka *BSP* [4]

rozdílem mezi *quadtree* a *BSP* je, že v *BSP* nemusí být na rozdíl od *quadtree* všechny buňky symetrické a buňky v *BSP* navíc nemusí mít stejnou orientaci [4].

Metoda je běžně využíváno při vykreslování grafických scén, kde se dělicí roviny vybírají na základě polygonů, aby optimalizovaly výpočty. Pro renderování scény je například důležité zajistit, že každý uzel stromu obsahuje polygony, které lze vykreslit efektivně. [4].

Princip dělení prostoru na disjunktní oblasti je také ideální pro vytváření místností nebo oddělených oblastí v herním levelu. Celý level je reprezentován jako kořen *BSP*, jehož prostor je dělen, dokud nejsou splněny stanovené podmínky (například minimální velikost místností). *BSP* zajišťuje, že se oblasti nebudou překrývat [3].

1.2.2 Agent-Based generování

Nejjednodušší *agent-based* metoda přistupuje ke generování úrovní tak, že vygeneruje agenta který "vykopává" chodby a vytváří oblasti v souvislé sekvenci. Na rozdíl od přístupu dělení prostoru se jedná spíše o mikro přístup ke generování to vede k organicky vypadajícím výsledkům, které však mohou být chaotické a může docházet k překrývání místností.

Těmto problémům se dá do nějaké míry zabránit optimálním nastavením pravidel agenta, ale to je náročné a efekt změny parametrů je obtížné odhadnout bez testování.

Pro jednoduché generování úrovní existují dva hlavní přístupy: "slepého" agenta a agenta s "přehledem" ("look-ahead")

Slepý agent

Metoda se slepým agentem je stochastická a funguje tak, že: Agent začíná v náhodném bodě dungeonu, a náhodně se vybere směr (nahoru, dolů, vlevo nebo vpravo). Agent začne "kopat" tímto směrem, a každý vykopaný dlaždicový bod dungeonu je nahrazen "chodníkovou" dlaždicí. Po prvním "vykopání" je 5% šance, že agent změní směr (vybere nový náhodný směr) a další 5% šance, že agent umístí místnost náhodné velikosti. Každým dalším dlaždicovým bodem ve směru, který se shoduje s předchozím, se šance na změnu směru zvyšuje o 5%. Každým dalším dlaždicovým bodem bez přidání místnosti se šance na přidání místnosti zvyšuje o 5%. Když agent změní směr, šance na změnu směru se sníží na 0%. Když agent přidá místnost, šance na přidání místnosti zůstává na 0%.

Agent s přehledem

Metoda s agentem s přehledem řeší problémy slepé metody jako jsou překrývající místnosti a slepé chodby tím, že agenta informuje o vzhledu úrovně, tedy agent může kontrolovat co se stane když přidá místnost, zda dojde k propojení místností. Zároveň také agent díky přehledu může zvolit směr tak aby se dostal do oblasti kde nehrozí překrytí a tedy nehrozí, že všechny místnosti úrovně budou naskládány na sobě. [3]

Vícefázové generování terénu

V kontrastu s těmito přístupy, existuje další metoda, která využívá agentů k vytváření složitějších terénů, a to prostřednictvím více fázového procesu.

Tento přístup využívá tři hlavní fáze generování terénu:

1. Fáze pobřeží: V této fázi velké množství agentů pracuje na vytvoření obrysu pevninské masy, která může být obklopena vodou.
2. Fáze terénu: V této fázi agenti definují vlastnosti mapy, jako je tvarování hor, nížin a pláží.
3. Fáze eroze: V této fázi agenti vytvářejí řeky, které erodují terén a spojují horské oblasti s oceánem.

Každý agent v tomto systému je schopen vidět aktuální výšku jakéhokoli bodu na mapě a může tyto body modifikovat podle potřeby. Tímto způsobem agenti aktivně formují terén a přítomnost jiných agentů může způsobit změny v okolním prostředí. Pro ovládání životnosti agenta je každý agent vybaven určitým počtem tokenů, které spotřebovává při vykonávání akcí. Tento mechanismus dává designérovi možnost ovlivnit, jak bude terén generován.

Designér může ovlivnit makro-vlastnosti mapy tím, že určí počet agentů v každé fázi a počet tokenů, které budou agenti mít k dispozici. To umožňuje vytvářet různé typy terénů, jako jsou například pobřežní oblasti, hory nebo řeky.

Tento přístup, využívající několik fází a různých typů agentů (pobřežní agenti, agenti hor, agenti řek, atd.), dává designérovi větší kontrolu nad výsledným terénem, což může vést k zajímavějším a strukturovanějším mapám než u jednodušších metod založených na jednom agentovi. [5]

1.2.3 Grammar Algorithms

Gramatiky jsou obecně způsob, jak popsat strukturu a rozebrat její pod části. Například věta může obsahovat podmět a ten může obsahovat přídavná jména. *Gramatiky* lze ale také aplikovat mimo mluvené jazyky. Jednoduchým příkladem je binární strom, který může být strukturován následovně 1.1. [7] [6]

```
1  BRANCH ->
2      node BRANCH BRANCH |
3      leaf
```

Ukázka 1.1: Příklad gramatiky: Strom

Tento typ algoritmu však nemůžeme generovat nové struktury. Pro generování nového obsahu můžeme takový algoritmus upravit přidáním nějakých pravděpodobností. Algoritmus pak s určitou pravděpodobností generuje jednotlivé části struktury – například věty nebo jiné objekty. Můžeme naznačit na ukázce stromu, váhy nastavíme tak, že je dvakrát vyšší pravděpodobnost vygenerování listu než rozdělení uzlu 1.2. [6]

```
1  BRANCH ->
2      node BRANCH BRANCH [weight=1] |
3      leaf [weight=2]
```

Ukázka 1.2: Příklad gramatiky: Strom s váhami

Gramatiky jsou silným nástrojem pro generování herního obsahu, protože poskytují strukturovaný způsob, jak popsat a vytvářet různé herní prvky, od prostorových uspořádání po specifické herní objekty. Tento přístup je podobný tomu, jak generativní gramatiky popisují strukturu přirozených jazyků. Jak bylo uvedeno dříve, generativní gramatiky používají konečnou sadu rekurzivních pravidel k definování větších struktur z menších částí, což se ukazuje jako efektivní způsob generování komplexních herních prostorů.

Například je možné použít grafovou gramatiku pro generování topologie úrovní, kde uzly reprezentují místnosti a hrany mezi nimi určují jejich propojení. Tento přístup je velmi flexibilní, protože umožňuje přizpůsobit generování úrovní specifickým parametřům, jako je obtížnost, velikost nebo zábavnost. Tento vysoký stupeň kontroly je výhodný v kontextu her, kde je potřeba mít pod kontrolou herní zážitek, ale na druhou stranu může být složité vytvořit univerzální gramatiku, která by pokryla všechny možné herní scénáře.

Když se vrátíme k použití pravděpodobnostních gramatik, jako ukázáno v předchozím příkladu s binárním stromem, přidání váhových faktorů umožňuje algoritmu vytvářet výsledky s různou pravděpodobností. Tato metoda může být efektivně aplikována na generování úrovní, kde například existují různé pravděpodobnosti pro

výskyt místností určité velikosti nebo pro přítomnost specifických herních objektů. Tímto způsobem lze vytvářet náhodné, ale zároveň logické a konzistentní herní prostory, které odpovídají požadavkům designu. [3]

1.2.4 Výškové mapy

Terén může být snadno reprezentován jako dvourozměrná matice reálných čísel, kdy šířka a výška matice odpovídají rozměrům x a y a hodnoty v jednotlivých buňkách představují výšku v daném bodě. Takové matici se říká *výšková mapa (heightmap)*. [8]

Náhodný terén

Nejjednodušší metodou jak výškovou mapu vytvořit, je použít generátor náhodných čísel a matici jednoduše vyplnit náhodnými čísly. Tato metoda vytvoří výškovou mapu kterou je teoreticky možné vykreslit, ale výsledná mapa nevypadá jako mapa terénu, spíše jako náhodné výčnělky. V reálném terénu se výšky mění plynule, to znamená, že výška v jednom bodě logicky souvisí s výškami v okolí, náhodný generátor však výšky generuje nezávisle na ostatních a to vytváří nepřírozeně vypadající terén.

Interpolace terénu

Jedním z jednoduchých řešení tohoto problému je použití interpolace. Nejprve se náhodné hodnoty výšek vygenerují na hrubší mřížce, a poté se výšky mezi těmito body dopočítají pomocí interpolace. Ačkoliv tímto způsobem nevzniknou některé přirozené útvary jako útesy, tento přístup vytváří hladší a realističtější terén.

Gradient-based náhodný terén (Šum)

Místo generování výškových hodnot a následné interpolace svahů můžeme generovat rovnou svahy, tedy gradienty změny výšky terénu a z těch následně odvozovat výšky. [8]

*”Gradient noise je způsob generování terénu, kdy náhodné čísla interpretujeme jako náhodné gradienty, což znamená strmost a směr svahů. Tento přístup byl poprvé použit Kenem Perlinem pro film Tron z roku 1982, a proto se někdy nazývá **Perlin noise**.”* – přeloženo z: [1]

Na hrubší mřížce se vygenerují náhodné gradienty, reprezentované vektory dx a dy , které odpovídají sklonu v osách x a y . Gradienty mohou mít kladnou nebo zápornou hodnotu, což umožňuje modelovat stoupající i klesající svahy.

Výšky pak získáme tak, že nejprve do každého bodu mřížky nastavíme hodnotu 0. Pro určení výšky na místech mezi mřížkovými body se podíváme na čtyři sousední body mřížky. Nejprve si představíme, že bychom zohlednili pouze gradient v levém

horním rohu. Jaká by byla výška v aktuálním bodě, pokud by terén rostl nebo klesal jen podle tohoto gradientu? Výška by byla jednoduše hodnota tohoto gradientu vynásobená vzdáleností, kterou jsme urazili podél svahu: strmost na ose x , tedy dx , vynásobená vzdáleností od mřížky ve směru x , plus strmost na ose y , tedy dy , vynásobená vzdáleností ve směru y . Tento výpočet provedeme pro každý z čtyř sousedních bodů mřížky. Výsledkem budou čtyři výšky, které odpovídají situacím, kdy by výška terénu závisela pouze na jednom z těchto gradientů. Ty jednoduše interpolujeme. [9]

Gradientní šum umožňuje vytvářet realistické terény s plynulými změnami výšek.

1.2.5 Fraktálové algoritmy

Fraktální algoritmy umožňují vytvářet realistický terén díky svojí schopnosti generovat detailní rozsáhlé struktury.

Často používanou metodou pro generování fraktálního terénu je *Diamond-square algoritmus* (*diamantovo-čtvercový algoritmus*). Jená se o výpočetně nenáročný a snadno implementovatelný algoritmus. [10]

Algoritmus funguje následovně:

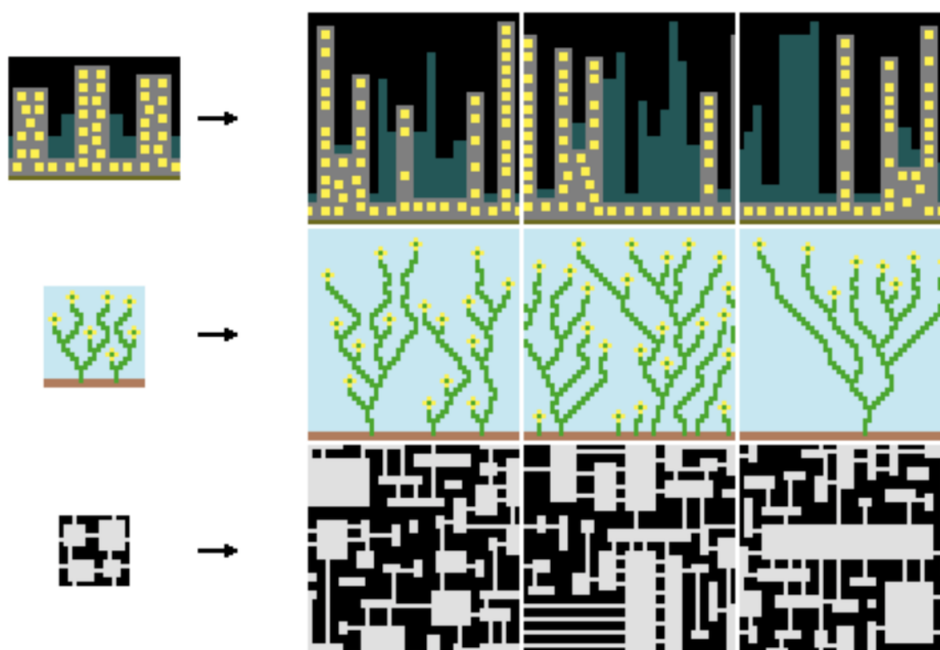
1. Počáteční nastavení: Nastavíte hodnoty čtyř rohů výškové mapy na náhodné hodnoty.
2. Diamantový krok: Najdeme střed čtverce definovaného těmito čtyřmi rohy a nastavíme jeho hodnotu jako průměr těchto čtyř rohů plus náhodná hodnota.
3. Čtvercový krok: Nyní najdeme střední body stran čtverce a nastavíme jejich hodnoty na průměr tří hodnot: dvou sousedních rohů a středu čtverce. Znovu přidáme náhodnou hodnotu.
4. Rekurse: Po prvním kole diamantového a čtvercového kroku rozdělíme čtverec na čtyři menší čtverce. Drsnost se sníží a celý proces se opakuje pro menší čtverce. Tento proces pokračuje, dokud není dosaženo maximálního počtu iterací.

Velikost náhodných hodnot, které používáme v těchto krocích, se nazývá **drsnost**. Větší hodnoty vedou k drsnějšímu terénu, menší hodnoty vytvářejí hladší terén.

Diamond-square algoritmus je využit v mnoha aplikacích, včetně her jako *Minecraft* nebo simulací pro generování krajiny v leteckých simulátorech.

1.2.6 Wave Function Collapse (WFC)

Algoritmus *Wave Function Collapse* je inspirován kvantovou mechanikou. Spočívá ve využití principu superpozice a postupně "kolabuje" políčka do jedné z možností podle daných pravidel.



Obrázek 1.2: Příklad vzorů a výsledků *WFC* [11]

WFC začíná s mřížkou buněk, které mají na začátku všechny možné hodnoty. Algoritmus postupně odebírá možnosti na základě stavu okolních buněk a pravidel definovaných na základě vzoru. Tento princip se v algoritmu používá k určení konkrétních hodnot pro buňky mřížky, což vede k postupnému „zúžení“ prostoru možných konfigurací.

Konkrétně algoritmus funguje takto:

1. Každá buňka mřížky začíná s plnou superpozicí všech možných hodnot.
2. Vybere se buňka s nejnižší entropií (největší nejistotou o její hodnotě).
3. U vybrané buňky se zúží možnosti podle pravidel, čímž se její hodnota „zhroucení“ na jednu konkrétní.
4. Zúžení možnosti jedné buňky ovlivní okolní buňky, čímž se jejich možnosti také zúží.
5. Proces se opakuje, dokud není celá mřížka vyřešena.

WFC se využívá pro generování úrovní, textur a dalších prostorových uspořádání, protože umožňuje vytvářet složité a realistické struktury na základě jednoduchých vzorců. Příklad vzorů a jejich výsledků je vidět na obrázku 1.2. [11]

1.2.7 GAN (Generative Adversarial Networks)

Generativní adversariální síť (GAN) je metoda generování obsahu složená ze dvou komponent – generátoru a diskriminátoru které „bojují“ v procesu učení. Generátor

vytváří realistické výstupy napodobující reálná data. Diskriminátor se pak snaží rozlišit zda jsou data reálná nebo vygenerovaná.

Princip metody *GAN* právě spočívá v interakci mezi generátorem a diskriminátorem, díky které dochází k postupnému učení obou a tedy zlepšování kvality výsledných dat. Tento proces je řízen pomocí zpětné propagace, která umožňuje optimalizovat váhy v obou sítích. Toto vzájemné "soupeření" má za následek, že výsledná data systému *GAN* vypadají realisticky i s menším množstvím vstupních reálných dat.

Mezi různé varianty *GAN* patří například *Deep Convolutional GAN (DCGAN)*, který používá konvoluční vrstvy k lepšímu zachycení prostorových vzorců v datech. Tento přístup je například využít možné využít při generování výškových map, nicméně má některá omezení, jako jsou problémy s velikostí a škálovatelností výstupů. Dále existuje, *Spatial GAN (SGAN)* jedná se o vylepšenou variantu *DCGAN*, která eliminuje plně propojené vrstvy a místo nich používá konvoluční vrstvy s *krokem*, což umožňuje lépe zachovávat prostorové vztahy a zlepšuje efektivitu generování. [12]

1.3 Porovnání metod

- Tabulka s výhodami, nevýhodami a vhodností pro daný projekt.
- Vyhodnocení nejvhodnějších metod pro konkrétní potřeby hry.

Kapitola 2

Návrh

2.1 Návrh strategické hry

- Herní koncept (např. typ hry, pravidla, cíle hráče).
- Popis mapy: Použití dlaždic a jejich vliv na herní mechaniky.
- Navržené mechaniky:
 - Jednotky (pohyb, útok, obrana).
 - Ekonomika (zdroje, produkce).
 - Stavba budov (možnosti na jednotlivých terénech).

Kapitola 3

Implementace mapy

3.1 Implementace generování herního pole

Kapitola 4

Simulace

4.1 Simulace a testování hry

- Popis simulace: Automatizované testy hratelnosti.
- Testovací kritéria:
 - Rovnováha herních mechanik.
 - Rozmanitost generovaných map.
 - Uživatelská přívětivost.
- Výsledky a analýza.

Závěr

Zde napište text závěru (1-3 strany, nerozdělujte na podkapitoly) nebo jej vložte ze samostatného souboru: např. příkazem `\input{zaver.tex}`.

Literatura

- [1] SHAKER, Noor; TOGELIUS, Julian a NELSON, Mark J. *Procedural Content Generation in Games*. Switzerland: Springer International Publishing, 2016. ISBN 978-3-319-42714-0.
- [2] SMITH, Gillian. *An Analog History of Procedural Content Generation*. Paper. 360 Huntington Ave, 100 ME Boston, MA 02115, USA: Northeastern University, 2015.
- [3] ANTONIOS LIAPIS. *Constructive Generation Methods for Dungeons and Levels*. Online. Antoniosliapis.com. 2017. Dostupné z: https://antoniosliapis.com/articles/pcgbook_dungeons.php. [cit. 2025-01-13].
- [4] GEEKS FOR GEEKS. *Binary Space Partitioning*. Online. www.geeksforgeeks.org. 2020, 30 Sep, 2020. Dostupné z: <https://www.geeksforgeeks.org/binary-space-partitioning/>. [cit. 2025-01-13].
- [5] DORAN, Jonathon a PARBERRY, Ian. Controlled Procedural Terrain Generation Using Software Agents. *ReasearchGate*. 2010, vol. 2, no. 2, s. 111 - 119.
- [6] *Procedural Location Generation with Weighted Attribute Grammars*. Bakalářská práce. University of Twente P.O. Box 217, 7500AE Enschede The Netherlands: University of Twente, 2021.
- [7] *Generative Grammars*. Online. Fandom.com. 2016. Dostupné z: https://procedural-content-generation.fandom.com/wiki/Generative_Grammars. [cit. 2025-01-15].
- [8] Travall. *Procedural 2D Island Generation - Noise Functions*. Online. <https://medium.com>. 2018. Dostupné z: <https://medium.com/@travall/procedural-2d-island-generation-noise-functions-13976bddeaf9>. [cit. 2025-01-16].
- [9] *Perlin Noise*. Online. Scratchapixel.com. 2022. Dostupné z: <https://www.scratchapixel.com/lessons/procedural-generation-virtual-worlds/perlin-noise-part-2/perlin-noise.html>. [cit. 2025-01-16].
- [10] BROWN, Adam. *The Maths of Fractal Landscapes and Procedural Landscape Generation*. Online. fractal-landscapes.co.uk. 2002 - 2020. Dostupné z: <https://www.fractal-landscapes.co.uk/maths.html>. [cit. 2025-01-16].

- [11] HEATON, Robert. *The Wavefunction Collapse Algorithm explained very clearly*. Online. Robertheaton.com. 17 Dec 2018. Dostupné z: <https://robertheaton.com/2018/12/17/wavefunction-collapse-algorithm/>. [cit. 2025-01-16].
- [12] SPICK, Ryan J.; COWLING, Peter a WALKER, James Alfred. *Procedural Generation using Spatial GANs for Region-Specific Learning of Elevation Data*. Paper. University of York, UK: University of York, 2019.

Příloha A

Název přílohy

Zde napište text první přílohy nebo jej vložte, např.: `\input{priloha_A.tex}`.