

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	8
1 Teorie PCG	9
1.1 Procedurální generování obsahu (PCG)	9
1.1.1 Výhody procedurálního generování	9
1.1.2 Nevýhody a výzvy procedurálního generování	10
1.1.3 Procedurální generování ve hrách	10
1.2 Metody procedurálního generování map	11
1.2.1 Space Partitioning	11
1.2.2 Agent-Based generování	12
1.2.3 Grammar Algorithms	14
1.2.4 Výškové mapy	14
1.2.5 Fraktálové algoritmy	16
1.2.6 Wave Function Collapse (WFC)	16
1.2.7 GAN (Generative Adversarial Networks)	17
1.3 Porovnání metod	18
2 Implementace mapy	21
2.1 Generování herního pole	21
2.2 Implementované metody	21
2.2.1 Náhodná výšková mapa	21
2.2.2 Interpolovaná výšková mapa	22
2.2.3 Gradientní šum	22
2.2.4 Přiřazení typů terénu	23
2.2.5 Vizualizace mapy	24
3 Návrh	25
3.1 Návrh strategické hry	25
3.1.1 Prostor	26
3.1.2 Objekty, atributy a stavy	27
3.1.3 Akce hráče	31
3.1.4 Pravidla hry	33
3.1.5 Skill and Chance (Dovednost a náhoda)	36
3.1.6 Shrnutí	36
4 Implementace základních mechanik	37
4.1 Implementace prototypu herních mechanik	37

4.1.1	Architektura a účel prototypu	37
4.1.2	Třída <code>SpravceHry</code>	38
4.1.3	Třída <code>Hrac</code>	39
4.1.4	Třída <code>Jednotka</code>	40
4.1.5	Třída <code>Budova</code>	40
5	Simulace	42
5.1	Simulace a testování	42
5.1.1	Cíle simulací	42
5.1.2	Metriky pro sběr dat a analýzu	42
5.1.3	Simulace	43
	Závěr	45
	Literatura	45
	Přílohy	48
A	Název přílohy	48

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 PCG

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 generován celý vesmír 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 *D 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 možné 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.

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 jsou logické rozložení mapy, hratelnost a rychlost generování.

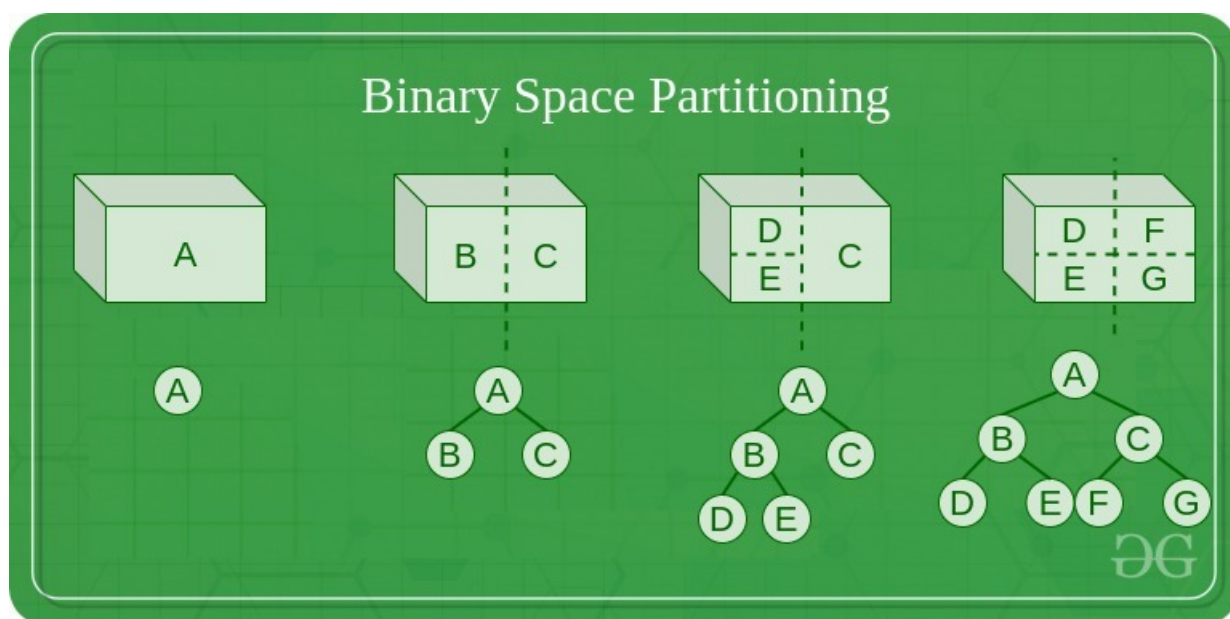
Algoritmy byly mimo jiné zpracovány na základě: [1].

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 na následující straně. Další často využívanou variantou je *quadtree*, kde je prostor rozdělen na čtyři symetrické části. Hlavním 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].



Obrázek 1.1: Ukázka *BSP* [4]

Metoda je běžně využíváno při vykreslování grafických scén, kde se dělící 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].

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, který 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 změni

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 se 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 vícefázového 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 podčásti. 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 jiného objektu. 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 zážitky, ale na druhou stranu může být složité vytvořit univerzální gramatiku, která by pokryla všechny možné herní scénáře. [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 náhodná 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: Nastaví se hodnoty čtyř rohů výškové mapy na náhodné hodnoty.
2. Diamantový krok: Najde se střed čtverce definovaného těmito čtyřmi rohy a nastavíme jeho hodnotu jako průměr těchto čtyř rohů plus náhodná hodnota.
3. Čtvercový krok: Nyní se najde střední body stran čtverce a nastavíme jejich hodnoty na průměr tří hodnot: dvou sousedních rohů a středu čtverce. Znovu se přidá náhodá hodnota.
4. Rekurze: Po prvním kole diamantového a čtvercového kroku se rozdělí 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.

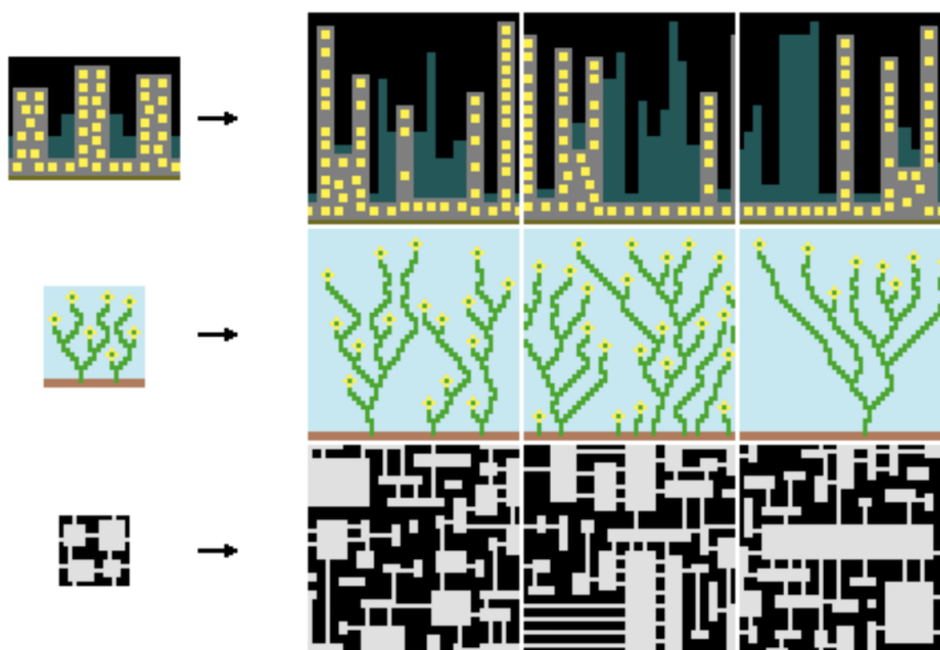
1.2.6 Wave Function Collapse (WFC)

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

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).



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

3. U vybrané buňky se zúží možnosti podle pravidel, čímž se její hodnota „zhroutlí“ na jednu konkrétní.
4. Zúžení možností 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 vzorů. 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 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

Popsal jsem několik možných přístupů generování herních map. V této části porovnám popsané metody generování map a zvolím vhodnou metodu, kterou implementuji v rámci tohoto výzkumného úkolu.

Cílem je generovat herní mapu pro svět složený z políček nebo dlaždic pro strategickou hru, tedy výstup by měl připomínat přirozeně vypadající mapu světa. Tedy metody, které vytvářejí nesouvislé nebo lineárně rozdělené výsledky nejsou vhodné (nebudu se snažit generovat města, pouze krajinu).

Zároveň, jelikož pro každou hru bude generována nová unikátní mapa, by bylo vhodné, aby generování probíhalo v rozumném čase, tedy metody s nízkou výpočetní náročností budou preferované.

Tabulka 1.1 na následující straně popisuje výhody a nevýhody jednotlivých metod generování.

Metoda	Výhody	Nevýhody
Space Partitioning	Logická a hierarchická struktura Snadné dosažení konzistentních výsledků Vhodné pro vnitřní prostory	Výsledky mohou působit pravouhle Nevhodné pro organické mapy
Agent-Based generování	Přirozené, nelineární výsledky Velká variabilita výsledků Snadné přizpůsobení chování agentů	Náročné ladění parametrů Vysoká míra náhodnosti Nepředvídatelné výsledky
Grammar Algorithms	Vysoká míra kontroly Vhodné pro generování rozvržení úrovní	Náročné na definování pravidel Rychle narůstá složitost pravidel Nevhodné pro přirozené terény
Výškové mapy	Dobře reprezentují reálnou topografii Plynulé přechody mezi výškovými úrovněmi Snadná implementace Snadná kombinace s dalšími technikami	Náhodné generování vede k nereálným výsledkům Nevhodné pro generování budov a interiérů
Fraktálové algoritmy	Přirozeně vypadající krajina Efektivní pro nekonečné terény Relativně nenáročné na výkon	Omezená kontrola Výsledky mohou být homogenní
Wave Function Collapse	Konzistentní výsledky Vhodné pro mapy s pravidelnými vzory Snadné využití existujících vzorů	Výpočetně náročné Složitý vstupní dataset Může skončit ve stavu bez platného řešení
Generative Adversarial Networks	Realistické a variabilní výsledky Adaptivní na různé typy prostředí	Velké množství trénovacích dat Výpočetně náročné Obtížně laditelné

Tabulka 1.1: Porovnání metod generování prostředí

Po porovnání metod generování map jsem se rozhodl využít *metod výškových map*. Hlavním důvodem je, že tato metoda by měla být schopna generovat přirozeně vypadající krajinné útvary při poměrně nízké výpočetní náročnosti. Využití interpolace nebo gradientních metod umožní vytvářet plynulé přechody mezi různými typy terénu, což povede k realisticky vypadajícímu světu.

Mou další volbou by byla metoda *Wave Function Collapse (WFC)* případně *Generative Adversarial Networks (GAN)*, jelikož tyto metody mají potenciál vést k ještě realističtějším a více rozmanitým výsledkům. *WFC* by zajistilo více vzorů v prostředí, zatímco *GAN* by umožnilo generovat zcela nové mapy na základě trénovacích dat. Nicméně, obě tyto metody jsou výpočetně náročnější a vyžadují velká trénovací data, což je činí nevhodnými pro můj výzkumný úkol. Proto tyto metody ponechám jako případné možné rozšíření do diplomové práce.

Kapitola 2

Implementace mapy

2.1 Generování herního pole

Pro generování herního pole jsem zvolil metodu výškových map. Výšková mapa je dvourozměrná matice reálných čísel, kde každé číslo představuje výšku v daném bodě. Výškovou mapu následně převedu na konkrétní terénní typy, jako jsou voda, pláně, lesy a hory.

V této kapitole popíšu implementaci generování výškových map třemi způsoby:

- Náhodná výšková mapa – nejjednodušší metoda, která však vytváří nerealistický terén.
- Interpolace hrubé mřížky – metoda, která vyhlazuje terén pomocí interpolace mezi body náhodné mřížky.
- Gradientní šum – generování přirozeně vypadajícího terénu pomocí Perlinova šumu.

2.2 Implementované metody

Výškové mapy jsem, jak už bylo zmíněno, implementoval třemi různými způsoby.

2.2.1 Náhodná výšková mapa

Tuto metodu jsem implementoval především proto, že je velmi jednoduchá a může sloužit jako základní "benchmark" pro srovnání s ostatními metodami. Výsledky této metody nejsou příliš dobré, protože nevytváří realistické terénní struktury. Výsledné struktury nejsou realistické ani praktické pro hraní.

Celá implementace spočívá v generování náhodných hodnot a jejich uložení do dvourozměrného pole, které pak funkce vrací 2.1.

```

1 def nahodne_pole(rows, cols, min_value=0, max_value=1):
2     return np.random.uniform(low=min_value, high=max_value, size=(
        rows, cols))

```

Ukázka 2.1: Kód generující pole pro náhodnou mapu

2.2.2 Interpolovaná výšková mapa

Lepšího výsledku lze dosáhnout interpolací náhodné mřížky. Tato metoda spočívá v tom, že vygeneruji menší náhodnou mřížku, její hodnoty rozmístím pravidelně do větší mřížky a prázdné hodnoty pak získám interpolací hodnot z menší mřížky. Pro samotnou interpolaci využívám v kódu 2.2 knihovnu *scipy*. Funkce nakonec opět vrací matici čísel mezi nula a jedna.

```

1 def interpolovane_pole(big_rows, big_cols, small_rows, small_cols,
    min_value=0, max_value=1):
2     # Vytvoření menší mřížky
3     small_grid = nahodne_pole(small_rows, small_cols, min_value,
        max_value)
4     # Indexy pro malou a velkou mřížku
5     small_x = np.linspace(0, big_cols - 1, small_cols)
6     small_y = np.linspace(0, big_rows - 1, small_rows)
7     big_x = np.arange(big_cols)
8     big_y = np.arange(big_rows)
9     # Vytvoření seznamu souřadnic
10    small_points = np.array([(x, y) for y in small_y for x in
        small_x])
11    small_values = small_grid.flatten()
12    big_points = np.array([(x, y) for y in big_y for x in big_x])
13    # Doplnění hodnot seznamu interpolací
14    big_list = griddata(small_points, small_values, big_points,
        method='cubic')
15    # Převedení na mřížku
16    big_grid = big_list.reshape(big_rows, big_cols)
17    return big_grid

```

Ukázka 2.2: Kód generující iterpolovanou výškovou mapu

Interpolace vytvoří plynulejší terén bez ostrých přechodů mezi výškami, ale výsledky mají stále tendenci být hranaté a ne zcela přirozeně vypadající.

2.2.3 Gradientní šum

Gradientní šum je komplikovanější metoda, která vytváří přirozenější struktury terénu než předchozí dvě metody. Konkrétně se inspiroji metodou **Perlinův šum**, který generuje plynulé změny hodnot v prostoru, čímž vytváří realistické krajiny s kopečky, údolími a horami.

Perlinův šum funguje na principu interpolace gradientních vektorů. Výsledkem je spojitá mapa hodnot, kde se výška plynule mění bez ostrých přechodů, ale vznikají přirozeně vypadající útvary.

Tuto metodu jsem prvně implementoval pomocí knihovny *noise* jako vzorový výsledek a následně jsem si také napsal vlastní implementaci.

Vlastní implementace

Mnou napsaná funkce generuje Perlinův šum v několika krocích:

1. Vytvoření mřížky gradientových vektorů – Každému bodu v hrubé mřížce (menší ze dvou mřížek, vytvořené podle zvolené hodnoty *scale*) je přiřazen náhodný gradientový vektor. Tento vektor určuje, jak se bude hodnota šumu měnit v okolí daného bodu.
2. Výpočet skalárních součinů – Pro každý bod na jemné mřížce se vypočítá skalární součin mezi gradientním vektorem a vektorem k bodu, jehož hodnotu chceme určit.
3. Použití funkce fade – Hodnoty jsou vyhlazeny pomocí speciální funkce fade, která zajišťuje plynulý přechod mezi body mřížky a zabraňuje vzniku ostrých přechodů.
4. Interpolace mezi sousedními hodnotami – Hodnoty se interpolují mezi čtyřmi nejbližšími gradientními body, čímž vznikne plynulý přechod mezi oblastmi s různou výškou.
5. Normalizace – Nakonec se hodnoty normalizují aby výsledné hodnoty dobře vycházeli mezi hodnoty nula a jedna.

Použitím různých měřítek (*scale*) je možné ovlivnit rozmanitost a velikost útvarů vygenerované krajiny.

Výslednou matici hodnot, lze použít jako výškovou mapu pro tvorbu herního terénu. Výsledná mapa má plynulé přechody mezi různými výškami a vytváří přirozeně vypadající krajinu.

2.2.4 Přiřazení typů terénu

Po vygenerování výškové mapy je potřeba převést hodnoty na jednotlivé typy terénu, k tomu je použita funkce `cislo_na_policko` 2.3.

```
1 def cislo_na_policko(grid):
2     mapa = np.empty_like(grid, dtype='str')
3     for i in range(grid.shape[0]): # Počet řádků
4         for j in range(grid.shape[1]): # Počet sloupců
5             if grid[i][j] < 0.25:
6                 mapa[i][j] = "V" # Voda
7             elif grid[i][j] < 0.5:
8                 mapa[i][j] = "P" # Pláně
9             elif grid[i][j] < 0.75:
10                 mapa[i][j] = "L" # Les
```

```

11         else:
12             mapa[i][j] = "H" # Hory
13     return mapa

```

Ukázka 2.3: Převádějící výškovou mapu na konkrétní terény

Aktuální prahové hodnoty jsem zvolil experimentálně, ale mohou být upravovány po testování v simulaci, aby byl terén vyvážený a poskytoval vhodné herní prostředí.

2.2.5 Vizualizace mapy

Matici terénů vycházející z funkce `cislo_na_policko` pak zobrazují pomocí *matplotlib* pomocí funkce `zobraz_mapu` 2.4.

```

1 def zobraz_mapu(mapa):
2     """
3     Barevně vykreslí terénní mapu.
4     """
5     # Definice barev pro jednotlivé typy terénu
6     barvy = {
7         "V": "#1f77b4", # Modrá - Voda
8         "P": "#58d162", # Světle zelená - Pláně
9         "L": "#0c3b10", # Zelená - Les
10        "H": "#4a4a48", # Šedá - Hory
11    }
12
13    # Převedení mapy na numerickou matici s indexy
14    text_to_index = {"V": 0, "P": 1, "L": 2, "H": 3}
15    index_map = np.vectorize(text_to_index.get)(mapa)
16
17    # Vytvoření barevné mapy
18    cmap = ListedColormap(barvy.values())
19
20    # Vykreslení mřížky
21    plt.figure(figsize=(8, 8))
22    plt.imshow(index_map, cmap=cmap, interpolation='nearest')
23
24    plt.show()

```

Ukázka 2.4: Kód generující pole pro náhodnou mapu

Výstupem je čtyřbarevný graf složený ze čtvercových dlaždic.

Kapitola 3

Návrh

3.1 Návrh strategické hry

Zpracováno na základě: [15] [14] [13].

V této kapitole rozeberu postup návrhu jednoduché strategické hry hrané na náhodně vygenerované mapě.

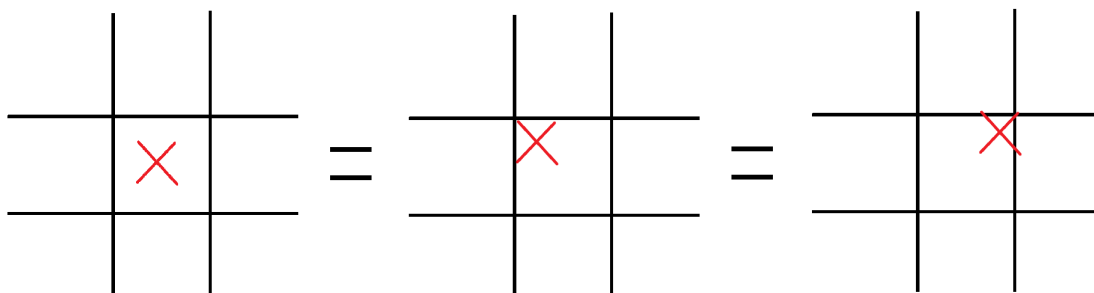
Návrh strategické hry vyžaduje systematický přístup k definování jejích klíčových prvků. Aby byla hra hratelná, vyvážená a pokud možno i zábavná, je zapotřebí systematicky rozebrat a promyslet návrh všech částí hry.

Základní představa, se kterou jsem začínal před samotným systematickým návrhem, byla taková, že hra bude 2D tahová strategie se správou surovin. Hráči budou ovládat a vylepšovat své jednotky, spravovat zdroje a budovat infrastrukturu, přičemž se budou snažit dosáhnout vítězství nad soupeřem. Herní svět je reprezentován dlaždicovou mapou, typ terénu na dlaždici ovlivňuje pohyb jednotek, možnosti výstavby a dostupnost zdrojů.

Pro návrh hry využívám metodologii z knihy *The Art of Game Design: A Book of Lenses* od Jesseho Schella, která strukturuje herní design do několika klíčových kategorií:

- Prostor – Jak je herní svět uspořádán a jak se v něm hráči pohybují.
- Objekty, atributy a stavy – Jaké herní entity existují, jaké mají vlastnosti a jaké stavy mohou při hře nastat.
- Akce hráče – Jaké interakce může hráč provádět a jak ovlivňují hru.
- Pravidla hry – Jaká jsou omezení a podmínky vítězství.
- Dovednost a náhoda – Jaký je poměr mezi strategickým rozhodováním a prvky náhody.

Tento rámec poskytuje ucelený pohled na návrh hry a pomáhá zajistit, aby všechny prvky dohromady tvořily soudržný a dobře vyvážený celek.



Obrázek 3.1: Ekvivalence různých značení sudoku.

Ve zbytku kapitoly rozeberu podrobněji právě tyto herní prvky a konkretizuji herní návrh.

3.1.1 Prostor

Každá hra obsahuje nějaký herní prostor, ve kterém se celá hra odehrává. Ten obecně vymezuje herní lokace a určuje, jak jsou mezi sebou propojeny. Z pohledu herních mechanik je považován prostor za matematickou konstrukci, tedy je potřeba odfiltrovat veškeré vizuální prvky a zaměřit se pouze na abstraktní uspořádání prostoru.

Kniha *The Art of Game Design: A Book of Lenses* rozděluje herní prostory podle několika parametrů:

- Diskrétnost vs. kontinuita – Prostor může být buď diskrétní, nebo kontinuální. Diskrétní prostor je tvořen pevně stanovenými, oddělenými místy. Kontinuální prostor umožňuje pohyb v plynulém, nekonečném rozsahu.
- Počet dimenzí – Každý herní prostor má určitý počet dimenzí, které definují jeho rozsah a strukturu. Prostor může být jednorozměrný, dvourozměrný nebo dokonce třírozměrný.
- Ohraničenost a propojení oblastí – Prostor může být uzavřený, s pevně definovanými hranicemi, nebo otevřený, umožňující pohyb hráče nebo herních prvků mimo hranice.

Příklad hry hrané na diskretním dvoudimenzionálním poli je "piškvorky" (v rámci příkladu předpokládám herní plochu 3×3), kde herní plocha je rozdělena na devět oddělených políček. Herní deska je zobrazena jako souvislý prostor, ale z hlediska herních mechanik bereme v potaz pouze těchto devět specifických míst, která můžeme znázornit jako uzly v síti. Tedy dokud je jednoznačně rozeznatelné, do kterého políčka zapadá, jsou si mechanicky ekvivalentní, jak je naznačeno na obrázku 3.1.

Hra monopoly je pak příklad jednodimenzionální hry. I když je herní deska vizuálně uspořádána do tvaru čtverce, když se odstraní grafické prvky, můžeme vidět, že hra umožňuje pohyb po jediné řadě políček, propojených v cyklické smyčce. V tomto případě se každé políčko na desce chová jako bod nulové dimenze, ačkoliv vizuálně vypadají některé čtverce odlišně, jejich funkce se neliší.

Herní prostor může také zahrnovat „prostory v prostorech“, což se často vyskytuje v počítačových hrách. Například může existovat venkovní prostor (kontinuální, dvou-rozměrný), ale hráč může narazit na ikony, které představují města nebo jeskyně. Tyto ikony přecházejí do zcela oddělených prostorů, které s venkovním prostorem nejsou přímo propojené, což je příkladem prostorového uspořádání, které je více založeno na mentálních modelech hráčů než na geografické realitě.

Návrh herního prostoru

Jak jsem již zmínil, navrhovaná hra bude 2D tahová strategie, tedy herní prostor bude dvoudimenzionální a diskrétní, složený ze čtvercových políček propojených po horizontální a vertikální ploše. To bude mít zásadní vliv na pohyb jednotek po mapě a tedy i veškerá strategická rozhodnutí hráčů.

Diskrétnost herního prostoru usnadňuje měření vzdáleností, po které se jednotky mohou po desce pohybovat, a také omezuje rozložení budov na jednu budovu na políčko. To samé platí pro jednotky, což umožňuje strategické tahy, jako blokování pohybu jednotek nepřítele.

Ve hře nebudou žádné podprostory ani oblasti, které by hráči mohli navštívit jako separátní oblasti. Všechny interakce probíhají na jednom herním poli, přičemž všechny herní mechaniky se soustředí na strategické využití této plochy.

Z hlediska návrhu a pozdější implementace tedy herní prostor uvažuji jako dvourozměrné diskrétní pole, kde každé „políčko“ představuje bod s nulovou dimenzí. Tento pohled by měl zjednodušit návrh pravidel a interakcí mezi hráči a prostředím, což umožňuje efektivnější rozvoj herních taktik.

3.1.2 Objekty, atributy a stavy

V herním prostoru se pak nacházejí objekty, se kterými hráči v průběhu celé hry manipulují nebo s nimi interagují pomocí jiných objektů. Může se jednat například o herní figurky, postavy, karty, nebo samotné herní prostředí. Objekty lze chápat jako „podstatná jména“ herních mechanik.

Každý objekt má nějaké atributy, které popisují jeho vlastnosti. Atributy lze chápat jako „přídavná jména“. Například auta v závodní hře mohou mít atributy jako *maximální rychlost* a *aktuální rychlost*.

Každý herní objekt má **stav**, který určuje jeho vlastnosti a chování ve hře. Stav objektu se skládá z jeho atributů, což jsou jednotlivé charakteristiky objektu. Například figurka v šachu má atribut „mód pohybu“, který může nabývat stavů jako

„volně se pohybující“, „v šachu“ a „matovaná“. V Monopoly má každý pozemek atribut „počet domů“, jehož stav se může měnit mezi 0 až 4 domy nebo hotel.

Atributy mohou být statické nebo dynamické:

- Statické atributy – Nemění se v průběhu hry. Například barva figurky v dámě nebo maximální rychlost auta.
- Dynamické atributy – Mohou se měnit v závislosti na akcích hráčů či mechanikách hry. Například aktuální rychlost auta se mění podle řízení hráče, životy jednotky klesají při útoku.

Každý herní objekt tak může mít kombinaci statických a dynamických atributů. Například v šachu má figurka statický atribut „barva“, ale dynamický atribut „pozice na šachovnici“, který se mění během hry.

Je důležité si uvědomit, že objekty ve hře často interagují mezi sebou. Například, jednotka může útočit na jinou jednotku, budova může produkovat suroviny, nebo terénní prvek může ovlivňovat pohyb jednotek. Tyto interakce by měly být navrženy tak, aby dávaly smysl a byly pro hráče srozumitelné.

Stav objektu není nutně viditelný celý – některé atributy mohou být skryté nebo viditelné jen pro určité hráče.

Důležitou součástí herního návrhu tedy je rozhodnout, kdo má přístup k jakým informacím. Pro jednoduchost rozlišíme informace na *veřejné*, *částečně skryté* nebo *zcela skryté*.

- Veřejné informace – Všechny atributy a jejich stavy jsou viditelné pro všechny hráče. Například v šachu oba hráči vidí všechna pole a figurky na hrací desce, takže jediným tajemstvím je přemýšlení soupeře.
- Částečně skryté informace – Někteří hráči znají určitou informaci, ale jiní ne. Například ve hře poker někteří hráči viděli kartu, zatímco jiní ne.
- Zcela skryté informace – Existují atributy, které zná pouze samotná hra. Například v počítačových hrách mohou být některé části světa před hráčem skryté, dokud je neodhalí.

Rozhodnutí o tom, kdo má přístup k jakým informacím, zásadně ovlivňuje herní strategii a atmosféru. Hry jako poker jsou postavené na utajení a odhadu soupeřových karet, zatímco v šachu mají hráči k dispozici informace o stavu celé herní plochy, a jedinou neznámou je strategie protivníka.

Návrh herních objektů

V této části se podle probraných principů pokusím nastínit jednotlivé objekty a definovat atributy těchto objektů.

Ve hře se nachází několik typů objektů, se kterými hráči mohou manipulovat nebo s nimi interagovat. Základním objektem ve hře jsou **políčka** ze kterých je složené herní pole a určují podmínky pro pohyb jednotek a stavbu budov. **Budovy**, produkují suroviny, případně poskytují další služby a plní jiné strategické funkce. Jako poslední jsou **jednotky** což jsou pohyblivé objekty ovládané hráčem, které hráč využívá k různým činnostem, jako je boj, nebo těžba surovin. Každý z těchto objektů má své specifické atributy a stavy, které určují jejich vlastnosti a možnosti ve hře.

Co se informací týče, nakonec jsem se rozhodl, že hra nebude mít skryté informace, tedy hráči od začátku uvidí celý herní prostor a pohyby protivníka.

Políčka Políčka představují základní stavební jednotku mapy, na níž se odehrává hra. Každé políčko má několik atributů, popisujících jeho vlastnosti a interakce s ostatními objekty. Tyto atributy jsou:

- Statické:
 - **Pozice na mapě** – Souřadnice určující umístění políčka v herním prostoru.
 - **Název** – Název terénu.
 - **Zpomalení** – Určité typy terénů mohou snižovat pohybovou rychlost jednotek.
 - **Bonusová obrana** – Některé terény mohou zvyšovat obranu jednotek na daném políčku.
 - **Získatelné suroviny** – Typ surovin které je možné na políčku získat.
- Dynamické:
 - **Obsazenost jednotka** – Na políčku se může nacházet pouze jedna jednotka, tento atribut tedy bude bránit vstupu jiné jednotky na políčko.
 - **Obsazenost budova** – Na políčku může stát pouze jedna budova.

Terén není jen grafický prvek, ale významně ovlivňuje hru. Správná volba umístění jednotek může znamenat rozdíl mezi vítězstvím a porážkou – například jednotka stojící na horách má lepší obranu, zatímco husté lesy mohou zpomalit postup nepřátel. Navíc různé druhy terénu určují, jaké budovy lze postavit a jaké suroviny lze těžit.

Budovy Budovy jsou struktury, které hráči staví na mapě za účelem generování zdrojů nebo poskytování jiných výhod. Každá budova má následující atributy:

- Statické:
 - **Pozice na mapě** – Pozice budovy na herní mapě.
 - **Název** – Označení budovy.

- **Vlastník** – Určuje hráče, kterému budova patří. Změna vlastníků během hry nebude možná, pouze zničení nepřátelských budov.
 - **Typ terénu** – Omezení, na kterých typech políček lze budovu postavit.
 - **Produkce za kolo** – množství surovin, které budova generuje za kolo.
 - **Životy max** – Maximální počet životů budovy.
 - **Obrana** – O kolik se zredukuje poškození způsobené útokem.
 - **Cena** – Množství surovin potřebných pro stavbu budovy.
 - **Bonusová obrana** – Budova může zvyšovat obranu jednotek nacházejících se na stejném políčku.
 - **Speciální funkce** – Budova může umožňovat provádění speciálních akcí jako generování nebo vylepšování jednotek.
- **Dynamické:**
 - **Životy** – Aktuální počet životů budovy, pokud klesne na nulu budova je zničena.

Budovy kromě generování surovin poskytují hráči jiné strategické možnosti jako vylepšování jednotek, zvyšování obrany vlastních jednotek nebo blokování postupu nepřítele.

Jednotky Jednotky jsou pohyblivé objekty na mapě, které hráči ovládají a skrze které primárně interagují s herními mechanismy. Každá jednotka má své atributy, které určují její vlastnosti a schopnosti:

- **Statické:**
 - **Název** – Název typu jednotky.
 - **Pozice na mapě** – Kde na herním poli se jednotka nachází.
 - **Vlastník** – hráč, kterému jednotka patří.
 - **Cena** – Množství surovin potřebné pro vytvoření jednotky.
 - **Cena za kolo** – Náklady na udržování jednotky. Množství surovin které jednotka spotřebuje každé kolo.
 - **Životy max** – Maximální počet životů jednotky.
 - **Základní obrana** – Základní obrana jednotky. Redukuje poškození způsobené nepřátelským útokem.
 - **Útok** – Síla útoku. Určuje o kolik se sníží životy nepřátelské jednotky při útoku. Poškození je redukováno obranou.
 - **Dosah** – Vzdálenost, na kterou může jednotka útočit.
 - **Základní rychlost** – Maximální počet políček, která může jednotka urazit za kolo.

- **Dynamické:**
 - **Životy** – Aktuální počet životů jednotky, pokud klesne na nulu jednotka zmizí.
 - **Obrana** – Funkční obrana jednotky, po modifikaci prostředím (Typem terénu nebo budovou.).
 - **Rychlost** – Skutečný počet políček přes který se jednotka může v daném tahu pohybovat, po modifikaci terénem.
 - **Zaměstnaná** – Indikuje, zda jednotka vykonala akci v tomto tahu.
 - **V pohybu** – Indikuje zda se jednotka v tahu pohybovala.
 - **Zkušenosti** – Jednotka může být vylepšena v konkrétní budově po dosažení určitého počtu zkušeností, které získává prováděním akcí odpovídajících jejímu typu (válečníci získávají zkušenosti bojem, pracovníci těžbou surovin nebo opravami budov).

Jednotky představují hlavní způsob, jak hráč ovlivňuje dění na mapě. Každá jednotka má specifickou roli – některé slouží k boji, jiné k těžbě surovin nebo stavbě budov. Postupem času mohou získávat zkušenosti a vylepšovat své schopnosti, což přidává další vrstvu strategického rozhodování. Kromě toho jednotky také každé kolo spotřebovávají množství surovin, tedy hráč musí zvážit, zda si může dovolit postavit velkou armádu slabých jednotek.

3.1.3 Akce hráče

Akce jsou jedním ze základních stavebních kamenů herní mechaniky. Lze je chápat jako "slovesa" hry, jelikož definují, jak může hráč interagovat s herním světem. Rozdělujeme je do dvou hlavních kategorií:

- **Operační akce** jsou základní činnosti, které může hráč přímo vykonat, například pohyb jednotky nebo útok.
- **Výsledné akce** vycházejí z kombinace operačních akcí. Tyto emergentní akce často nejsou přímo definovány pravidly, ale vznikají přirozeně během hry a přispívají k její hloubce.

Operační akce jsou základní mechanismy, které hráč využívá pro interakci s herními mechanismy. V některých hrách je množství těchto akcí omezené, což vede k menší variabilitě herního stylu, zatímco v jiných hrách mají hráči širokou škálu možností, což umožňuje kreativní přístupy. Například ve hře dáma má hráč k dispozici tři základní operační akce:

- Posun kamene vpřed.
- Přeskok soupeřova kamene.

- Pohyb zpět v případě dosažení úrovně krále.

Výsledné akce vznikají kombinací operačních akcí a přispívají ke strategické hloubce hry. Zatímco operační akce jsou pevně dané pravidly, výsledné akce se objevují jako důsledek interakcí mezi hráčem, herním prostředím a protivníky. Ve hře dáma mohou být výslednými akcemi například:

- Ochrana kamene – umístění jiného kamene za něj, aby zabránil zajetí.
- Vynucení tahu soupeře – postavení figurky tak, že soupeř musí provést nevýhodný tah.
- Obětování kamene – nabídnutí figurky soupeři s cílem získat lepší pozici na hrací ploše.

Výsledné akce přidávají hře hloubku a umožňují emergentní chování hráče. Čím větší je poměr výsledných akcí vůči operačním akcím, tím více hra podporuje kreativitu hráče.

Návrh herních akcí

V této části rozeberu obecné operační akce, které bude hráč moci ve hře provádět, a následně zmíním vzniklé výsledné akce.

V rámci tahového systému může hráč během svého tahu každou jednotkou provést pohyb a jednu další akci. Některé akce mohou proběhnout pouze pokud jsou splněny určité podmínky, například pozice jednotky v určité budově. Vzhledem k tahovému systému a interakci mezi jednotkami a terénem mohou vznikat nové strategie, které nelze vždy předvídat.

Operační akce jsou základní činnosti, které může hráč vykonávat, primárně interakcí se svými jednotkami.

- Pohyb – Jednotka se pohne o počet polí odpovídající jejímu typu a terénu.
- Útok – Jednotka zaútočí na nepřátelskou jednotku v dosahu, způsobí poškození (*poškození = útok - obrana*) a pokud nepřítel přežije, provede protiútok.
- Těžba surovin – Pracovní jednotka získá určité množství surovin na základě toho na jakém terénu těžba proběhne.
- Práce – Pracovní jednotka v budově může získávat větší množství surovin, v budově která normálně generuje suroviny pasivně.
- Stavba budovy – Budovatelská jednotka postaví novou budovu na vhodném políčku.
- Oprava budovy – Pracovní nebo budovatelská jednotka obnoví část života poškozené budovy.

- Vylepšení jednotky – Pokud jednotka splní požadavky (např. bojové zkušenosti, existenci potřebné budovy, ...), může být vylepšena na silnější variantu.

Výsledné akce vznikají kombinací operačních akcí a strategického uvažování hráče.

- Obrana klíčových bodů – Hráč umístí jednotky tak, aby blokovaly přístup k důležitým budovám nebo oblastem.
- Napadání zásobování – Cílený útok na pracovníky nebo budovy snižující zdroje nepřítele.
- Taktický ústup – Ústup jednotek do bezpečnější oblasti, například k opravám nebo pod ochranu budov.
- Obklíčení – Koordinovaný útok více jednotek k eliminaci klíčových nepřátel.
- Zdržovací taktika – Hráč strategicky obětuje jednotky nebo využívá terén, aby zpomalil postup nepřítele.
- Ofenzivní opevnění – Hráč staví budovy poblíž nepřátelské základny jako obranné pozice.

3.1.4 Pravidla hry

Pravidla určují, jak se hra hraje, jaké akce může hráč provádět a jaké jsou jeho cíle. Bez jasně definovaných pravidel není možné vytvořit funkční a férové herní prostředí.

Špatně strukturovaná pravidla mohou vést k nevyváženosti, nechtěným exploatačním mechanikám nebo dokonce ke ztrátě zábavnosti. Správně nastavená pravidla zároveň hráče motivují k objevování efektivních strategií k dosažení vítězství.

Pravidla jsou soubor omezení a možností, které definují, jak hráči interagují s herním světem. V každé hře existují různé typy pravidel, která ovlivňují hratelnost:

- Akce hráčů – Co mohou hráči během hry dělat?
- Stav hry – Jak se hra mění v průběhu času?
- Cíle hry – Kdy hra končí a kdo vyhrává?

Tato pravidla společně vytvářejí herní mechaniky, které určují dynamiku hry a poskytují hráčům výzvy k řešení.

David Parlett, britský herní teoretik, rozdělil pravidla do několika kategorií:

- Operační pravidla – Popisují základní akce hráčů, například „hráč se může pohnout o n políček“.

- Základní pravidla – Definují matematický model hry, určují, jak se hra vyvíjí na základě akcí hráčů.
- Chování hráčů – Implicitní pravidla "fair play" a sportovního chování.
- Psaná pravidla – Formálně dokumentovaná pravidla hry.
- Zákony – Další pravidla, která mohou být zavedena například pro turnajovou hru.
- Oficiální pravidla – Spojují psaná pravidla a zákony do jednoho uceleného souboru.
- Doporučená pravidla – Tipy a strategie pro efektivní hraní.
- Domácí pravidla – Úpravy pravidel hráči pro přizpůsobení hry jejich preferencím.

Herní pravidla mohou být ovlivněna herním režimem, který může do hry přidávat komplexitu přidáním nebo odebráním pravidel. Případně úpravou podmínek vítězství.

Nejdůležitějším pravidlem jsou cílové podmínky. Každá hra musí mít jasně definovaný cíl, který hráče motivuje a určuje, kdy hra končí. Dobrý cíl by měl splňovat:

- Konkrétnost – Je jasné, co musí hráči udělat.
- Dosažitelnost – Hráči mají reálnou možnost dosáhnout cíle.
- Odměňující charakter – Splnění cíle přináší uspokojení.

Návrh pravidel

V této části shrnu dosavadní návrh do konkrétních pravidel. Pravidla definují podmínky vítězství, možné akce hráčů a interakce mezi herními objekty.

Primárním cílem hry je poražení protivníka, což je dosaženo zničením všech jeho jednotek a budov. Hra tedy končí, když zůstane pouze jeden hráč.

Sekundární cíle si hráč stanovuje sám, ale zahrnují:

- Efektivní správa ekonomiky – těžba surovin a stavba podpůrných budov.
- Budování armády – rekrutování a vylepšování jednotek.
- Dobývání strategických bodů – kontrola území s cennými zdroji nebo taktickými výhodami.

Objekty Hra obsahuje tři typy herních objektů:

- Jednotky
 - Bojové – primárně slouží k útoku a obraně (např. válečník).
 - Stavební – mohou stavět nové budovy, opravovat poškozené budovy a těžit suroviny.
- Budovy
 - Ekonomické – pasivně produkují zdroje (např. důl).
 - Obranné – poskytují ochranu a strategickou kontrolu území.
 - Vývojové – umožňují vylepšovat jednotky na vyšší úroveň.
- Terén
 - Ovlivňuje pohyb a bojové vlastnosti jednotek. (Např. hory zpomalují pohyb, lesy poskytují krytí, voda je nepřekročitelná.)

Akce V každém tahu může hráč provést s každou svou jednotkou jeden pohyb a jednu akci s tím, že některé akce vyžadují specifické podmínky, například vylepšení vyžadují určité budovy nebo sběr surovin má různý efekt podle terénu. Akce zahrnují:

- Pohyb jednotek
 - Každá jednotka se pohybuje pouze vertikálně a horizontálně, diagonální pohyb není možný.
 - Terén ovlivňuje pohyb jednotek (např. hory snižují jejich pohybovou vzdálenost).
- Útok
 - Hráč vybere útočníka v dosahu.
 - Útok je vyhodnocen podle atributů jednotek (útok vs. obrana).
 - Pokud obránce přežije, provede protiútok.
- Výstavba a interakce s objekty
 - Stavební jednotky mohou budovat budovy na určitých místech.
 - Pracovní jednotky mohou těžit suroviny, případně opravovat budovy

Vynucování pravidel Pravidla jsou automaticky vynucována herním systémem: Hráč může provádět pouze akce, které jsou v dané situaci povolené (např. jednotky nemohou cestovat přes políčka s vodou). Výsledky bojů jsou deterministické, hra tedy automaticky udělí způsobené poškození, případně zruší jednotku, pokud její životy klesnou na nulu. Hra automaticky končí, jakmile některý hráč přijde o všechny jednotky a budovy a tedy nemůže provést žádnou další akci.

3.1.5 Skill and Chance (Dovednost a náhoda)

- Jaký vliv má dovednost hráče na výsledek hry:
 - Strategické plánování.
 - Správné rozhodování a reakce na situace.
- Jaký vliv má náhoda:
 - Procedurálně generovaná mapa jako faktor variabilních podmínek.
- Vyvážení mezi dovednostmi a náhodou.

3.1.6 Shrnutí

- Shrnutí hlavních bodů návrhu.
- Vztah návrhu k procedurálnímu generování map.

Kapitola 4

Implementace základních mechanik

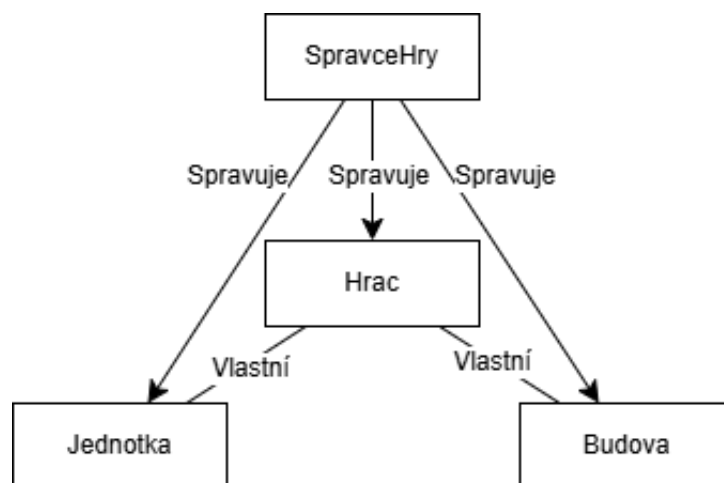
4.1 Implementace prototypu herních mechanik

4.1.1 Architektura a účel prototypu

V tomto prototypu jsem implementoval základní herní mechaniky nezbytné pro provádění simulací a shromažďování dat, na jejichž základě budou v dalších fázích výzkumu rozšiřovány typy jednotek a budov a upravovány jejich statistiky. Architektura prototypu je založena na čtyřech klíčových třídách 4.1, které zajišťují funkčnost herního cyklu a interakci mezi herními objekty:

- **SpravceHry:** Centrální řídicí prvek, který spravuje průběh hry, střídání hráčů a implementuje základní umělou inteligenci. Zajišťuje koordinaci mezi ostatními třídami.
- **Hrac:** Reprezentuje jednotlivého hráče ve hře. Spravuje jeho ekonomiku (suroviny), vlastněné jednotky a budovy.
- **Jednotka:** Představuje herní jednotky s atributy jako útok, obrana, životy a pohyb.
- **Budova:** Reprezentuje hráčské budovy, které v prototypu primárně slouží k produkci surovin. Jejich implementace je v prototypu zjednodušená, spíše abstraktní.

Účelem tohoto prototypu je získat funkční základ pro simulace herních scénářů s různými konfiguracemi jednotek a budov. Získaná data poslouží k informovanému návrhu nových typů jednotek a budov a k úpravě jejich statistik pro dosažení vyvážené hratelnosti v plné verzi hry.



Obrázek 4.1: Zjednodušený diagram tříd.

4.1.2 Třída SpravceHry

Třída `SpravceHry` je hlavním řídicím prvkem celé hry. Zajišťuje koordinaci všech částí systému, spravuje průběh jednotlivých tahů a obsahuje také jednoduchou implementaci umělé inteligence ovládající hráče pro účely testování. Jedná se o centrální bod, který propojuje třídy `Hrac`, `Jednotka` a `Budova` a zabezpečuje jejich souhru v rámci herního cyklu.

Hlavní úlohy třídy zahrnují:

- aktualizaci stavu hry v jednotlivých tazích,
- správu střídání hráčů a ukončení hry při splnění podmínky vítězství,
- rozhodování AI o pohybu a útocích jednotek.

Aktualizace hry. Před začátkem hry je pomocí `inicializace_hry` vytvořeno počáteční rozestavení hráčů včetně jejich základů a startovní ekonomiky (`startovni_domek`).

Pomocí metody `aktualni_hrac` se zjistí, který hráč je aktuálně na tahu. Střídání hráčů je zajištěno metodou `dalsi_hrac`, která po odehrání všech hráčů zvyšuje číslo kola.

Při splnění podmínky vítězství (např. zničení základny) je hra ukončena metodou `konec`, která zaznamená výsledek a ukončí herní cyklus.

Provedení tahu. Metoda `proved_tah` spravuje celý tah aktuálního hráče, včetně aktualizace surovin, vyhodnocení údržby jednotek a provedení akcí AI.

Metoda `kontrola_bojeschopnosti` ověřuje, zda hráč po ztrátách ještě má bojeschopné jednotky.

Rozhodování AI. Jednoduchá AI, implementovaná metodami `ai_tah`, `ai_pohyb_a_atak` a `stavba_a_verbovani_ai`, vyhodnocuje akce jednotek hráče v daném tahu. Jednotky nejprve útočí na nejbližšího nepřítele v dosahu, pokud je k dispozici, nebo hledají nepřátelské jednotky v okolí jednoho kroku. Pokud ani tak nedosáhnou, posouvají se směrem k nepřátelské základně. Po pohybu jednotek, na základě aktuálního zisku surovin, AI rozhoduje o stavbě nových budov nebo verbování nových jednotek.

Verbování a Stavba Nové jednotky jsou vytvářeny metodou `verbovani`, která ověřuje podmínky a najde volné místo v okolí základny.

Stavba budov probíhá obdobně metodou `stavba_budovy`, pouze je zde na pevně daná pozice, jelikož umístění budov neberu v prototypu v potaz, pracuji s nimi jako s abstraktními.

Souboje Metoda `vyhodnot_souboj` řeší útok jednotky na protivníka, včetně protiútoků a odstranění padlých jednotek. Pokud padne základna, hra je ukončena metodou `konec`.

4.1.3 Třída Hrac

Třída `Hrac` reprezentuje jednotlivého hráče ve hře. Uchovává jeho jméno, seznam jednotek a budov, dostupné suroviny. Hlavní odpovědností třídy je správa ekonomiky hráče, zejména manipulace se surovinami, zpracování údržby jednotek a vyhodnocení následků nedostatku zdrojů.

Hlavní úlohy třídy zahrnují:

- správu zásob surovin a jejich změn,
- evidenci vlastněných jednotek a budov,
- zajištění výroby surovin budovami,
- řešení údržby jednotek a následků nedostatku surovin.

Práce se surovinami. Třída umožňuje přidávat nové suroviny pomocí metody `pridej_suroviny` a odebírat suroviny pomocí `odecti_suroviny`. Při odečítání je kontrolováno, zda má hráč dostatek požadovaných zdrojů.

Správa ekonomiky a údržby. Každé kolo může hráč získat nové suroviny produkováné budovami prostřednictvím metody `zisk_z_budov`. Údržba jednotek je řešena metodou `zpracuj_udrzbu`, která odečítá příslušné množství surovin. Pokud hráč nemá dostatek zdrojů, všechny jeho jednotky utrpí ztrátu životů.

Nedostatek jídla. Konkrétním případem správy ekonomiky je hladovění jednotek, implementované metodou `zpracuj_nedostatek_jidla`. Pokud hráč nemá dostatek jídla, jeho jednotky ztrácejí životy.

4.1.4 Třída Jednotka

Třída `Jednotka` reprezentuje jednu funkční jednotku ve hře. Její hlavní úlohou je definovat chování jednotky v rámci herního světa, včetně pohybu, boje a interakce s herním prostředím.

Hlavní úlohy třídy zahrnují:

- uchovávání atributů jednotky (typ, pozice, statistiky, cena, vlastník),
- výpočet možných pohybů na základě rychlosti a terénu,
- realizaci pohybu na zvolenou cílovou pozici,
- vyhledávání nepřátelských jednotek v dosahu útoku,
- provádění útoku na nepřátelské jednotky a přijímání protiútoků,
- správu životů a mechanismus zániku jednotky.

Pohyb. Metoda `vypocet_moznych_pohybu` na základě aktuální pozice, rychlosti jednotky a typu terénu na herní mapě určuje všechny dosažitelné polohy. Při výpočtu zohledňuje překážky, jako je voda nebo přítomnost jiných jednotek. Metoda `proved_pohyb` následně aktualizuje pozici jednotky, pokud je cílová pozice v seznamu možných pohybů.

Boj. Metoda `najdi_cile_v_dosahu` prozkoumá okolí jednotky a vrátí seznam nepřátelských jednotek, které se nacházejí v jejím dosahu útoku. Samotný útok je realizován metodou `proved_utok`, která sníží životy napadené jednotky na základě rozdílu mezi útočnou silou útočníka a obranou bránícího se, s případnými modifikátory terénu. Metoda `proved_protiutok` umožňuje bránící se jednotce odpovédět na útok, pokud je útočník v jejím dosahu.

4.1.5 Třída Budova

Třída `Budova` obecně reprezentuje strukturu vlastněnou hráčem, která plní specifickou funkci v rámci hry. Může se jednat o budovy produkující suroviny, obranné stavby nebo jiné speciální budovy. Třída uchovává informace o typu budovy, její pozici na mapě, vlastníkovi, životech, obraně, produkci surovin a ceně za postavení.

Hlavní úlohy třídy zahrnují:

- uchovávání atributů budovy (typ, pozice, vlastník, životy, obrana, produkce, cena),
- generování surovin.

Produkce surovin. Pokud je budova určena k produkci surovin, slovník **produkce** definuje typy surovin a jejich množství, které budova vyprodukuje za jedno herní kolo. Metoda **generuj_suroviny** vrací tento slovník, čímž umožňuje hráči získávat zdroje.

Umístění. V tomto prototypu je pozice budovy pevně daná při jejím vytvoření a nelze ji měnit. Metoda pro stavbu budovy ve třídě **SpravceHry** zajišťuje její umístění na předdefinovanou pozici. V prototypu budovy neinteragují s pohybem jednotek ani mezi sebou, takže jejich přesné umístění nemá funkční dopad.

Speciální funkce a blokování cesty. Budovy mají v prototypu implementovanou pouze schopnost generovat suroviny. Jejich speciální schopnosti a schopnost blokovat cestu jsem se rozhodl implementovat až v rámci celé hry.

Kapitola 5

Simulace

5.1 Simulace a testování

V této kapitole popíšu průběh simulací zaměřených na získání dat pro vyvážení herních mechanik a určení optimálních parametrů jednotek, budov a herního pole. V simulacích budu využívat konkrétní scénáře a také metodu Monte Carlo pro získání robustních dat.

5.1.1 Cíle simulací

Hlavními cíli těchto simulací je:

- Systematicky testovat atributy jednotek, produkci budov a ceny herních prvků v různých scénářích.
- Identifikovat silné a slabé stránky jednotlivých jednotek za různých podmínek.
- Nalézt vyvážené nastavení herních parametrů, kde žádný prvek není výrazně dominantní nebo zbytečný.
- Poskytnout kvantitativní data o výkonnosti herních prvků pro budoucí vývoj pokročilé umělé inteligence.
- Analyzovat charakteristiky generovaného herního pole v závislosti na jeho parametrech.

5.1.2 Metriky pro sběr dat a analýzu

Pro sběr dat a analýzu v těchto simulacích se zaměřím na následující metriky v rámci jednotlivých testovaných scénářů:

- **Míra přežití jednotek:** Procento jednotek daného typu, které přežijí střet.

- **Poškození:** Celkové poškození způsobené jednotkou daného typu během střetu.
- **Zranění:** Celkové poškození, které jednotka daného typu utrpěla.
- **Poměr ztrát:** Poměr mezi ztrátami na straně útočníka a obránce v daném scénáři.
- **Počet kol:** Jak dlouho trvá, než dojde k rozhodnutí (např. zničení všech jednotek jedné strany).
- **Ekonomická efektivita:** Poměr mezi investovanými surovinami a dosaženými výsledky (např. způsobené poškození na jednotku ceny).

5.1.3 Simulace

Při provádění simulací budu využívat přístup podobný Monte Carlo metodě. Tato metoda spočívá v opakovaném náhodném vzorkování a simulaci s cílem získat statisticky robustní odhady chování komplexních systémů. V kontextu této práce to znamená, že jednotlivé testovací scénáře a celkové herní simulace budou spouštěny mnohokrát s potenciálně mírně odlišnými počátečními podmínkami (zejména v pozdějších fázích s náhodně generovanými mapami a prvkem náhody v rozhodování AI), abych získal průměrné výsledky a posoudil stabilitu a vyváženost herních mechanismů.

Fáze 1: Testování vyváženosti jednotek a ekonomických parametrů v izolovaných scénářích

V této fázi se zaměřím na testování jednotlivých typů jednotek proti sobě v kontrolovaných bojových situacích a zároveň budu zkoumat vliv cen jednotek a produkce surovin na jejich efektivitu.

- **Scénáře 1v1:** Budeme simulovat souboje mezi dvěma jednotkami různých typů (např. bojovník vs. lučištník) se stejnou nebo podobnou cenou. Budeme sledovat míru přežití, způsobené a přijaté poškození, abychom zjistili, zda existují výrazné disproporce v jejich efektivitě.
- **Scénáře "armáda proti armádě":** Budeme simulovat střety mezi menšími skupinami jednotek (např. 3 bojovníci vs. 2 lučištníci a 1 bojovník) se srovnatelnou celkovou cenou. Budeme sledovat poměr ztrát a celkovou efektivitu jednotlivých kompozic.
- **Testování cenové efektivity:** Budu analyzovat, jak cena jednotky ovlivňuje její výkonnost v boji a její ekonomickou návratnost v kontextu verbování.
- **Předběžné testování produkce budov:** V rámci těchto scénářů mohu simulovat, jak rychlost získávání surovin ovlivňuje schopnost hráčů nasazovat jednotky v různých typech střetů.

- **Variace atributů a cen:** Na základě výsledků budu upravovat atributy a ceny jednotek a produkci budov a testovat upravené verze v nových scénářích.

Fáze 2: Testování náhodného herního pole

V této fázi se zaměřím na analýzu charakteristik generovaného herního pole v závislosti na parametrech generování mapy. Budu testovat různé kombinace parametrů, jako je velikost mapy, škálování Perlinova šumu a prahové hodnoty pro definici jednotlivých typů terénu. Pro každé nastavení budu provádět opakované generování map a sbírat následující statistiky:

- **Existence cesty mezi základnami:** Pro každou vygenerovanou mapu zjistím, zda existuje alespoň jedna průchodná cesta mezi počátečními pozicemi základen (předpokládám pevně dané startovní pozice pro účely tohoto testování). Budu sledovat procentuální zastoupení map, kde cesta existuje.
- **Délka nejkratší cesty mezi základnami:** Pokud cesta existuje, vypočítám délku nejkratší cesty (například pomocí algoritmu A* nebo Dijkstrova algoritmu s ohodnocením políček podle typu terénu). Budu sledovat průměrnou délku této cesty.
- **Existence "snadné" cesty (bez hor):** Zjistím, jak často existuje cesta mezi základnami, která neprochází přes horský terén ('H'). Pokud taková cesta existuje, zaznamenám její průměrnou délku.
- **Poměr zastoupení jednotlivých typů terénu:** Pro každou vygenerovanou mapu spočítám procentuální zastoupení jednotlivých typů terénu (voda 'V', pláně 'P', les 'L', hory 'H'). Budu sledovat průměrné poměry pro různá nastavení parametrů generování.

Cílem této fáze je identifikovat parametry generování map, které vedou k herním polím s vhodnými charakteristikami z hlediska průchodnosti, strategické hloubky (dané rozložením terénu) a potenciálu pro interakci mezi hráči.

Fáze 3: Testování v celkových herních simulacích na náhodně generovaných mapách

V této fázi se vrátím k simulacím celých her na náhodně generovaných mapách s jednoduchou AI, s již předběžně vyladěnými parametry jednotek a produkce budov z Fáze 1 a s nastavením generování map z Fáze 2, které se ukázalo jako nejvhodnější.

- **Sledování celkové dynamiky hry:** Budu sledovat délku her, frekvenci interakcí mezi hráči a celkový průběh hry.
- **Sbírání dat pro finální doladění:** Budu shromažďovat data o využití jednotek a budov, výsledcích střetů a ekonomickém vývoji hráčů v reálných herních podmínkách pro případné další úpravy parametrů.

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.
- [13] ROGERS, Scott. *Level UP! The guide to great video game design*. John Wiley & Sons, 2010. ISBN 978-0-470-68867-0.
- [14] SALEN, Katie a ZIMMERMAN, Eric. *Rules of Play - Game Design Fundamentals*. 2. Massachusetts Institute of Technology, 2004. ISBN 0-262-24045-9.
- [15] SCHELL, Jesse. *The art of game design: a book of lenses*. Boca Raton : CRC Press, 2008. ISBN 978-0-12-369496-6.

Příloha A

Název přílohy

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