

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:* Cílem tohoto výzkumného úkolu je navrhnout a následně implementovat strategickou hru využívající náhodně generovanou mapu. První část práce se zabývá teorií procedurálního generování obsahu (PCG) a porovnává různé metody pro generování náhodných herních map. V další části je popsán návrh strategické hry, včetně jejího herního světa, mechanik a pravidel. Dále jsou zde popsány implementované algoritmy pro generování map, jako jsou metody výškových map, a implementace základních herních mechanik a umělé inteligence. V poslední části jsou popsány simulace, které ověřují vyváženost herních prvků a kvalitu náhodně generovaných map pomocí metody Monte Carlo.

*Klíčová slova:* procedurální generování obsahu (PCG), Náhodné generování map, Návrh hry, Herní mechaniky, Monte Carlo simulace

# Obsah

<b>Úvod</b>	<b>8</b>
<b>1 Teorie PCG</b>	<b>9</b>
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	13
1.2.4 Výškové mapy	14
1.2.5 Fraktálové algoritmy	15
1.2.6 Wave Function Collapse (WFC)	16
1.2.7 GAN (Generative Adversarial Networks)	17
1.3 Porovnání metod	17
<b>2 Implementace mapy</b>	<b>20</b>
2.1 Generování herního pole	20
2.2 Implementované metody	20
2.2.1 Náhodná výšková mapa	20
2.2.2 Interpolovaná výšková mapa	21
2.2.3 Gradientní šum	23
2.2.4 Přiřazení typů terénu	23
2.2.5 Vizualizace mapy	24
<b>3 Návrh</b>	<b>26</b>
3.1 Návrh strategické hry	26
3.2 Prostor	27
3.2.1 Návrh herního prostoru	28
3.3 Objekty, atributy a stavy	28
3.3.1 Návrh herních objektů	29
3.4 Akce hráče	33
3.4.1 Návrh herních akcí	33
3.5 Pravidla hry	34
3.5.1 Návrh pravidel	35
3.6 Dovednost a náhoda	36

3.6.1	Návrh dovednosti a náhodnost . . . . .	37
<b>4</b>	<b>Implementace základních mechanik</b>	<b>39</b>
4.1	Implementace prototypu herních mechanik . . . . .	39
4.1.1	Architektura a účel prototypu . . . . .	39
4.1.2	Třída <code>SpravceHry</code> . . . . .	40
4.1.3	Třída <code>Hrac</code> . . . . .	41
4.1.4	Třída <code>Jednotka</code> . . . . .	42
4.1.5	Třída <code>Budova</code> . . . . .	42
<b>5</b>	<b>Simulace</b>	<b>44</b>
5.1	Úvod do simulací . . . . .	44
5.1.1	Cíle simulací . . . . .	44
5.2	Teorie simulace . . . . .	44
5.2.1	Monte Carlo metoda v kontextu hry . . . . .	45
5.2.2	Reprodukovatelnost a správa náhodnosti . . . . .	45
5.3	Fáze 1: Testování vyváženosti jednotek v izolovaných scénářích . . . . .	45
5.3.1	Popis testovaných scénářů . . . . .	45
5.3.2	Metodika . . . . .	46
5.3.3	Duely . . . . .	46
5.4	Fáze 2: Testování náhodného herního pole . . . . .	57
5.4.1	Metodika . . . . .	59
5.4.2	Testování měřítka šumu . . . . .	59
5.4.3	Testování různých prahových hodnot . . . . .	63
	<b>Závěr</b>	<b>66</b>
	<b>Literatura</b>	<b>66</b>
	<b>Přílohy</b>	<b>69</b>
<b>A</b>	<b>Tabulky výsledků simulací</b>	<b>69</b>

# Úvod

Tento výzkumný úkol se zabývá vývojem strategické hry, která využívá náhodně generované herní pole. Výsledky projektu budou následně využity jako základ pro diplomovou práci. Pro generování náhodné mapy bude použito procedurální generování obsahu (PCG). PCG je v metoda pro vytváření velkého množství dat, například právě ve hrách kde je zapotřebí velké množství obsahu, kde by ruční výroba byla časově i finančně neefektivní. Výzkumný úkol má za cíl prozkoumat problematiku PCG se zaměřením na jeho uplatnění při tvorbě herních světů. Součástí úkolu je návrh strategické hry využívající náhodně generované herní pole, implementace generování herního pole s využitím procedurálních technik, implementace základních herních mechanik a také simulace navržených herních principů za účelem ověření hratelnosti.

V první kapitole rozeberu co to je procedurální generování obsahu (PCG) a k čemu se využívá, jeho výhody a nevýhody. Následně se zaměřím na metody PCG vhodné pro generování náhodných herních map. Porovnáám jejich výhody a nevýhody a zvolím metodu kterou implementuji pro mou hru.

Ve druhé kapitole se podrobně zaměřím na implementaci zvolené PCG metody. Podrobněji zmíním použité algoritmy a ukážu výsledné mapy.

Ve třetí kapitole zmíním základy teorie návrhu videoher. Na základě těch pak připravím návrh mé strategické tahové hry a specifikuji její pravidla.

Ve čtvrté kapitole pak implementuji základní herní mechaniky strategické hry. Zároveň připravím jednoduchou herní umělou inteligenci, která umožní v poslední kapitole provést simulace a optimalizovat navržené atributy herních objektů.

V poslední páté kapitole využiji implementovaný prototyp k simulování her. Pomocí metody Monte Carlo optimalizuji atributy herních objektů, tak aby byla hra vyvážená. Zároveň pomocí metody Monte Carlo optimalizuji parametry pro generování náhodných map tak, aby byly mapy co nejčastěji průchodné a strategicky zajímavé.



# 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

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 [1].

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 [1]. Podobně ve hře *No Man's Sky* je generován celý vesmír s miliardami planet, z nichž každá má unikátní prostředí, faunu a flóru [1].

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. [2]

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. [1]

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].

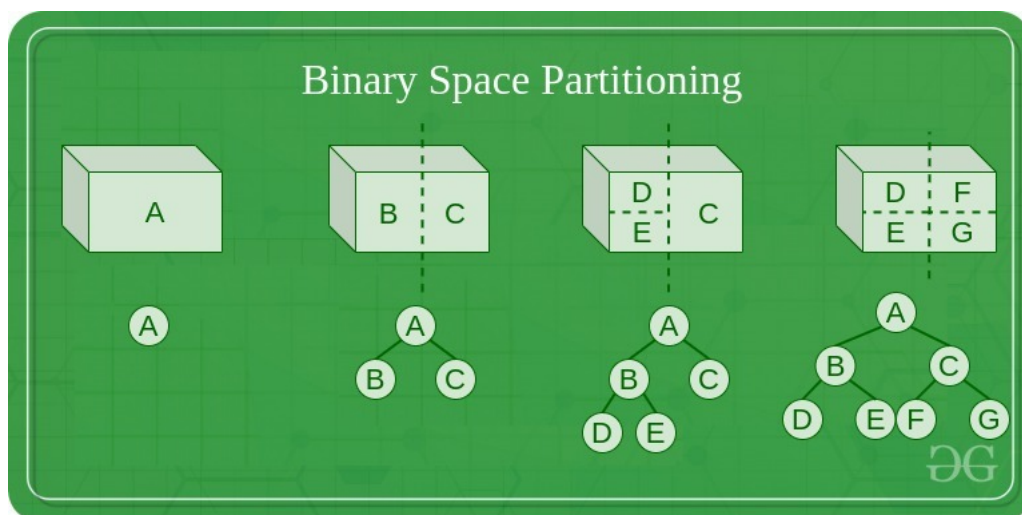
### 1.2.1 Space Partitioning

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. 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].

Metoda je běžně využívána při vykreslování grafických scén, kde se dělící roviny vybírají na základě polygonů, aby optimalizovaly výpočty. Pro renderování scény



Obrázek 1.1: Ukázka *BSP*. Převzato z: [4]

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 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 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 naskládáné 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 podobně jako je vidět v ukázce 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 (Ukázka: 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 nepřírozně 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 algorithmus* (diamantovo-čtvercový algoritmus). Jedná 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 náhodná 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.

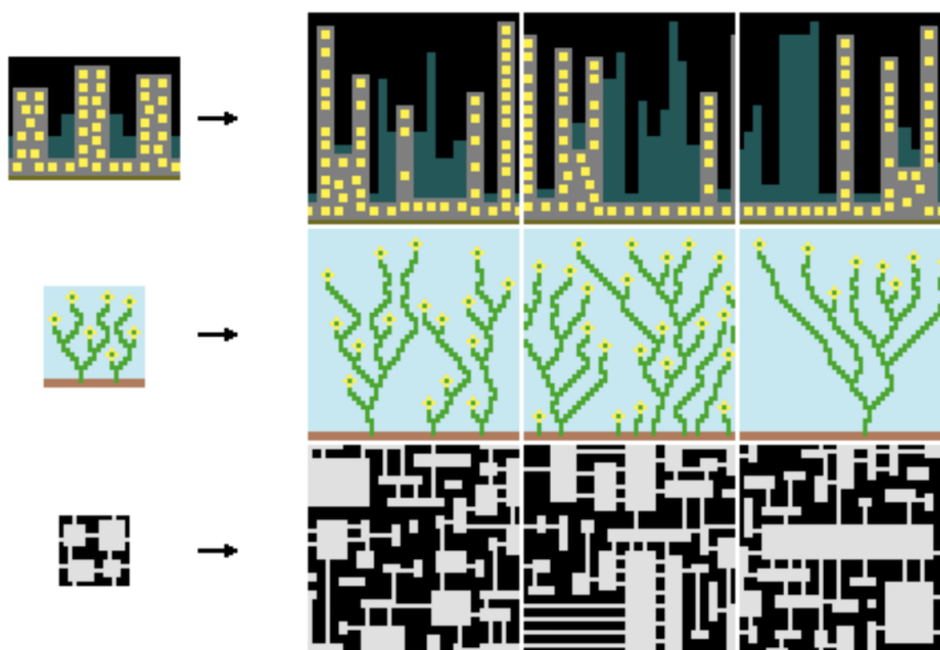
*WFC* začíná s mřížkou buněk, které mají na začátku všechny možné hodnoty. Algoritmus postupně odebírá možnosti na základě stavu okolních buněk a pravidel definovaných na základě vzoru. Tento princip se v algoritmu používá k určení konkrétních hodnot pro buňky mřížky, což vede k postupnému „zúžení“ prostoru 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).
3. U vybrané buňky se zúží možnosti podle pravidel, čímž se její hodnota „zhroutí“ 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]





Obrázek 1.2: Příklad vzorů a výsledků *WFC*. Převzato z: [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 popisuje výhody a nevýhody jednotlivých metod generování.

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.

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

# 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

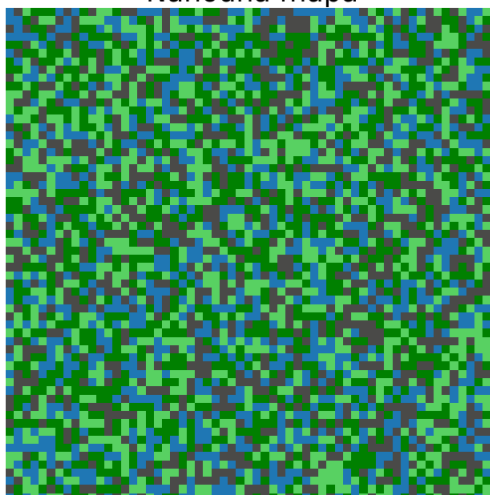
Výškové mapy jsem, jak už bylo zmíněno, implementoval třemi různými způsoby.

#### 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 (na ukázce: 2.1) vrátí. Výsledná mapa je vidět na obrázku 2.1

Náhodná mapa



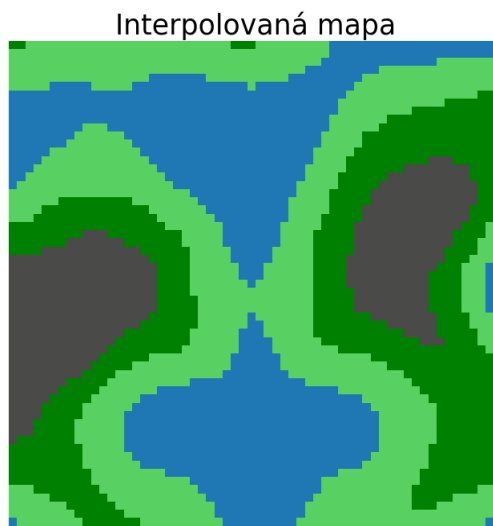
Obrázek 2.1: Ukázka mapy vygenerované metodou náhodné výškové mapy.

```
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 v knihovnu *scipy* jak je vidět v ukázce 2.2. Funkce nakonec opět vrací matici čísel mezi nula a jedna.



Obrázek 2.2: Ukázka mapy vygenerované metodou interpolace výškové mapy.

```

1 def interpolovane_pole(big_rows, big_cols, small_rows, small_cols,
2   min_value=0, max_value=1):
3     # Vytvoření menší mřížky
4     small_grid = nahodne_pole(small_rows, small_cols, min_value,
5   max_value)
6     # Indexy pro malou a velkou mřížku
7     small_x = np.linspace(0, big_cols - 1, small_cols)
8     small_y = np.linspace(0, big_rows - 1, small_rows)
9     big_x = np.arange(big_cols)
10    big_y = np.arange(big_rows)
11    # Vytvoření seznamu souřadnic
12    small_points = np.array([(x, y) for y in small_y for x in
13    small_x])
14    small_values = small_grid.flatten()
15    big_points = np.array([(x, y) for y in big_y for x in big_x])
16    # Doplnění hodnot seznamu interpolací
17    big_list = griddata(small_points, small_values, big_points,
18    method='cubic')
19    # Převedení na mřížku
20    big_grid = big_list.reshape(big_rows, big_cols)
21    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, jak je vidět na obrázku 2.2.

### 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**, která 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 přirozeněji vypadající krajinu, jak je vidět na obrázku 2.3.

### 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` (Ukázka: 2.3).



Obrázek 2.3: Ukázka mapy vygenerované metodou gradientní výškové mapy.

```

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

```

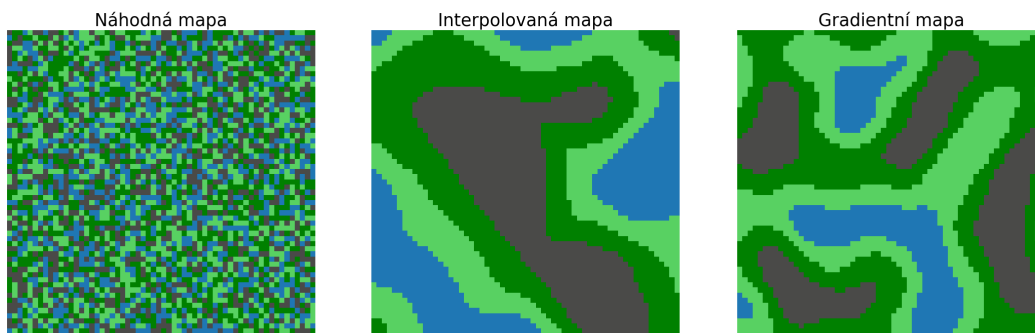
Ukázka 2.3: Převádějící výškovou mapu na konkrétní terény

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` (Ukázka: 2.4).





Obrázek 2.4: Zobrazení vykreslených map generovaných všemi metodami.

```

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. Na obrázku 2.4 jsou vidět zobrazené grafy různých metod.

# Kapitola 3

## Návrh

### 3.1 Návrh strategické hry

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 [13].

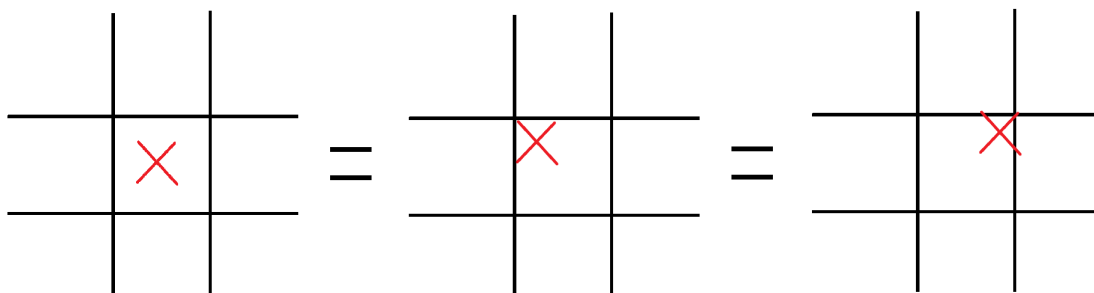
Základní představa, se kterou jsem začínal před samotným systematickým návrhem, byla taková, že hra bude 2D tahová strategie se správou surovin. Hráči budou ovládat a vylepšovat své jednotky, spravovat zdroje a budovat infrastrukturu, přičemž se budou snažit dosáhnout vítězství nad soupeřem. Herní svět je reprezentován mapou složenou ze čtvercových políček, typ terénu na políčku 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: [15], 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.

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



Obrázek 3.1: Ekvivalence různých značení sudoku.

## 3.2 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 odfiltrout veškeré vizuální prvky a zaměřit se pouze na abstraktní uspořádání prostoru.

Kniha *The Art of Game Design: A Book of Lenses* ([15]) 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 diskrétním dvoudimenzionálním poli je "piškvorky" (v rámci příkladu předpokládám herní plochu  $3 \times 3$ ), 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. [14]

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. [15]

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. [15] [14]

### 3.2.1 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.

## 3.3 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 hotelem.

Atributy mohou být statické nebo dynamické [15]:

- 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*. [15] [13]

- 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.

### 3.3.1 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. Skutečné způsobené poškození se pohybuje v rozsahu definovaném minimálním a maximálním poškozením jednotky a je dále ovlivněno šancí na kritický zásah a šancí na uhnutí cílové jednotky. Poškození je redukováno obranou cíle.
	Dosah	Vzdálenost, na kterou může jednotka útočit.
	Základní rychlost	Maximální počet políček, která může jednotka urazit za kolo.
	Násobitel kritického zásahu	Hodnota, kterou se násobí způsobené poškození v případě kritického zásahu.
	Šance na uhnutí	Pravděpodobnost, s jakou se jednotka může vyhnout přichozímu útoku a utrpět nulové poškození.
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 které 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.



## 3.4 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 [15]:

**Operační akce** jsou základní činnosti, které může hráč přímo vykonat, například pohyb jednotky nebo útok. [15]

**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. [15]

**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. [14]

### 3.4.1 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 ( <i>poškození = útok - obrana</i> ) 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 základy jako obranné pozice.

### 3.5 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 exploitačním 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 [15]:

- 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 [15]:

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. [14]

Nejdůležitějším pravidlem jsou cílové podmínky. Každá hra musí mít jasně definovaný cíl, který hráče motivuje a určuje, kdy hra končí. Dobrý cíl by měl podle [15] splňovat:

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

### 3.5.1 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 porážení 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	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). Hra automaticky vyhodnocuje útoky s ohledem na náhodnou variabilitu poškození (mezi minimálním a maximálním poškozením útočníka), šanci na kritický zásah (která násobí způsobené poškození) a šanci na uhnutí cílové jednotky (která může vést k nulovému poškození). Poškození je redukováno obranou cíle. Pokud životy jednotky klesnou na nulu, je automaticky odstraněna. Hra automaticky končí, jakmile některý hráč přijde o všechny jednotky a budovy a tedy nemůže provést žádnou další akci.

## 3.6 Dovednost a náhoda

V herním designu je klíčové pochopit a vyvážit interakci mezi dovedností hráče a prvky náhody. Jesse Schell ve své knize *The Art of Game Design: A Book of Lenses*

zdůrazňuje, že jak dovednost, tak náhoda jsou nezbytné pro vytvoření poutavého a znovu hratelného herního zážitku.

**Dovednost** v kontextu her odkazuje na schopnost hráče dosahovat lepších výsledků prostřednictvím učení, praxe a strategického myšlení. Hry založené primárně na dovednosti odměňují hráče za jejich schopnost rozpoznávat vzorce, plánovat dopředu, efektivně provádět akce a přizpůsobovat se měnícím se podmínkám. Příkladem dovedností může být taktické umístění jednotek, efektivní správa zdrojů nebo rychlé rozhodování pod tlakem. [15]

Přílišná závislost na dovednosti však může vést k tomu, že bude hra nepřístupná pro nové hráče nebo se stane předvídatelnou a repetitivní.

**Náhoda** zavádí do hry nepředvídatelnost a variabilitu. Může se projevat v mnoha formách, jako jsou hody kostkou, náhodné události, nepředvídatelné chování protivníků nebo náhodné generování herního světa. Náhoda může hru oživit, vytvořit dramatické momenty a zajistit, že žádné dvě hry nebudou identické, což zvyšuje znovu hratelnost. [15]

Nicméně, přílišná náhoda může vést k pocitu, že rozhodnutí hráče nemají smysl, a hra se stane čistě závislou na štěstí, což může být frustrující a podkopávat strategickou hloubku.

**Vyvážení dovednosti a náhody** je umění. Cílem je navrhnout hru tak, aby dovednost hráče umožňovala zmírnit dopady nepříznivé náhody nebo naopak využít příznivou náhodu. Náhoda by měla vytvářet zajímavé volby a scénáře rizika/odměny, spíše než negovat strategická rozhodnutí hráče. [14] [15]

### 3.6.1 Návrh dovednosti a náhodnost

V kontextu této strategické hry je dovednost hráče a prvky náhody pečlivě integrovány, aby se dosáhlo dynamického a znovu hratelného zážitku.

**Vliv dovednosti hráče:** Dovednost hráče se ve hře projevuje především v:

- **Strategickém plánování:** Schopnost hráče efektivně spravovat zdroje, budovat armádu s vhodnou kompozicí a plánovat dlouhodobé ofenzivní či defenzivní operace.
- **Taktickém provedení:** Optimální umístění jednotek na mapě, výběr nejvhodnějších cílů pro útok s ohledem na jejich typ a stav, a efektivní využití terénních výhod pro boj i pohyb.

**Vliv náhody:** Náhoda je do hry integrována na několika úrovních, aby zajistila variabilitu a nepředvídatelnost herních průběhů, což je klíčové pro smysluplné simulace Monte Carlo:

- **Procedurálně generovaná mapa:** Jak je popsáno v Kapitole 2, herní mapa je generována náhodně. Každá hra tak začíná na jedinečném herním poli s odlišným rozložením terénu a zdrojů, což nutí hráče i AI k adaptaci strategií.
- **Náhoda v souboji:**
  - **Variabilní poškození:** Při každém útoku je způsobené poškození náhodně vybráno z definovaného rozsahu. To znamená, že i identické jednotky útočící na stejný cíl mohou mít mírně odlišné výsledky, což zvyšuje nepředvídatelnost jednotlivých střetů.
  - **Kritické zásahy:** S pevnou globální šancí může útok způsobit kritický zásah, který násobí způsobené poškození. Tyto momenty vysokého dopadu přidávají do bojů napětí a dynamiku.
  - **Šance na uhnutí:** Cílová jednotka má šanci se útoku zcela vyhnout a utrpět nulové poškození. Tento prvek náhody ovlivňuje přežití jednotek a nutí hráče zvažovat pravděpodobnosti při plánování útoků.

# Kapitola 4

## Implementace základních mechanik

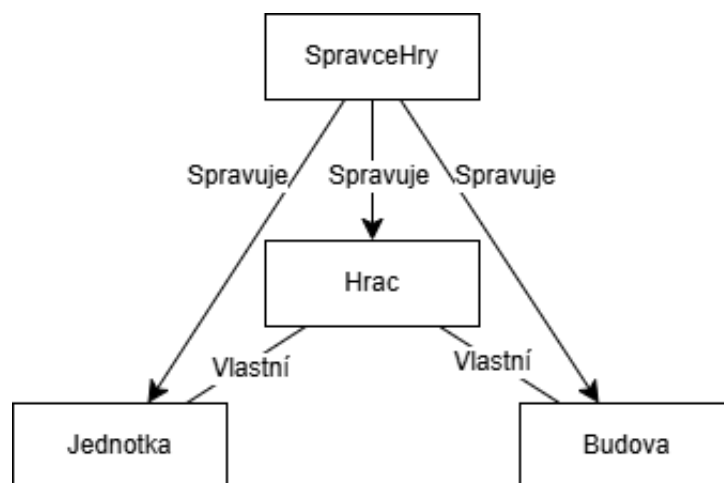
### 4.1 Implementace prototypu herních mechanik

#### 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 (diagram tříd: 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_utoke` a `stavba_a_verbovani_ai`, vyhodnocuje akce jednotek hráče v daném tahu. Rozhodování AI je ovlivněno globálními vahami, které definují preference



pro různé typy akcí a cílů. Pokud jsou v dosahu útoku jednotky AI více nepřátelských cílů, AI vybere cíl pravděpodobnostně na základě těchto vah (např. s vyšší prioritou pro střelecké jednotky nebo budovy). Specifickou výjimkou jsou jednotky s větším dosahem (střelci), které před útokem kontrolují, jestli nestojí nepřítel vedle nich, a pokud ano, pokusí se od nepřítele 'utéct', aby jim nehrozil protiútok. Pokud jednotka nenajde žádný cíl k útoku ani cestu k nepřátelské základně, může se s určitou pravděpodobností pohnout náhodným směrem. Po pohybu jednotek, na základě aktuálního zisku surovin a s ohledem na globální váhy, 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 počítá poškození s náhodnou variabilitou (vybírání hodnoty mezi `min_damage` a `max_damage` útočící jednotky). Dále zohledňuje šanci na kritický zásah (s globální konstantou `CRITICAL_HIT_CHANCE` a násobitelem `crit` útočnicka) a šanci na uhnutí cílové jednotky (`uhyb` obránce). Poškození je redukováno obranou bránící se jednotky, s případnými modifikátory terénu. Metoda `proved_protiutok` umožňuje bránící se jednotce odpovédět na útok, pokud je útočník v jejím dosahu. Protiútok je vyhodnocen stejnou logikou jako standardní útok, což zajišťuje konzistentní aplikaci náhodných bojových faktorů.

#### 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 Úvod do simulací

V této kapitole popíšu průběh simulací zaměřených na získání dat pro vyvážení parametrů jednotek, budov a herního pole. Pro získání robustních dat budou v simulacích využívány konkrétní scénáře a metoda Monte Carlo. Část dat budu v textu uvádět v tabulkách, kompletní data jsou dostupná jako CSV soubory v příloze.

Simulace budou probíhat ve dvou hlavních fázích:

- **Fáze 1: Testování vyváženosti jednotek v izolovaných scénářích.** Tato fáze se zaměří na souboje jednotlivých jednotek a menších skupin v kontrolovaných prostředích.
- **Fáze 2: Testování náhodného herního pole.** V této fázi bude analyzován vliv parametrů generování mapy na charakteristiky herního pole.

#### 5.1.1 Cíle simulací

Hlavními cíli těchto simulací jsou:

- Systematicky testovat atributy jednotek, produkci budov a ceny herních prvků.
- Nalézt vyvážené nastavení herních parametrů, kde žádný prvek není výrazně dominantní nebo zbytečný.
- Analyzovat charakteristiky generovaného herního pole v závislosti na jeho parametrech a posoudit, jak ovlivňují strategické možnosti.

### 5.2 Teorie simulace

Při provádění simulací budu využívat přístup založený na Monte Carlo metodě. Tato metoda spočívá v opakovaném náhodném vzorkování a simulaci s cílem získat

statisticky robustní odhady chování komplexních systémů.

### 5.2.1 Monte Carlo metoda v kontextu hry

V kontextu této hry je metodu Monte Carlo možné využít díky prvkům náhodnosti. Konkrétně se jedná o variabilní poškození, kritické zásahy a šanci na uhnutí v bojovém systému, a také o náhodný pohyb a vážený výběr cíle v rozhodování jednoduché AI vytvořené pro simulace. Tyto prvky zajišťují, že i při identických počátečních podmínkách se jednotlivé herní průběhy mohou výrazně lišit.

### 5.2.2 Reprodukovatelnost a správa náhodnosti

Pro zajištění reprodukovatelnosti výsledků simulací je klíčová správa generování náhodných čísel. To znamená, že pro samotný sběr dat pro Monte Carlo analýzu je nezbytné, aby každý simulační běh byl nezávislý a zároveň reprodukovatelný. Toho je dosaženo použitím ID dané simulace jako *seedu* pro generátor náhodných čísel (`random.seed(simulation_id)`). Tento přístup zajišťuje, že získaná data skutečně reprezentují rozložení možných výsledků v náhodném prostředí a zároveň umožňuje přesnou reprodukci jakéhokoli konkrétního simulačního běhu pro detailní analýzu.

## 5.3 Fáze 1: Testování vyváženosti jednotek v izolovaných scénářích

Tato fáze se zaměří na testování jednotlivých typů jednotek proti sobě v kontrolovaných bojových situacích. Cílem je získat detailní data o výkonu každé jednotky a dopadu náhodnosti na výsledky soubojů.

### 5.3.1 Popis testovaných scénářů

V rámci Fáze 1 budou pro testování vyváženosti jednotek použity předpřipravené scénáře poskytující kontrolované prostředí pro analýzu výkonu jednotek.

- **Scénář: Rovina**

- **Popis mapy:** Velká obdélníková mapa (např. 15x15 polí typu 'P' - pláň).
- **Obecné umístění jednotek:** Jednotky budou začínat v opačných rozích mapy (např. (0, 0) a (14, 14)).
- Scénář je určen ke srovnání atributů jednotek v otevřeném prostoru, kde prostředí nemá žádný zásadní vliv.

- **Scénář: Pohoří**

- **Popis mapy:** Obdélníková mapa (např. 15x15) s horami ('H') rozdělujícími mapu napůl, oblepenými lesy ('L').
- **Obecné umístění jednotek:** Jednotky budou začínat v opačných rozích mapy (např. (0, 0) a (14, 14)).
- Horský terén ('H') modifikuje obranu jednotek stojících na něm (+2 obrana) a zároveň zásadně zpomaluje pohyb. Lesy ('L') podobně snižují rychlost pohybu jednotek. Scénář je určen ke srovnání schopností jednotek v komplexnějším terénu, který ovlivňuje pohyb a bojové schopnosti jednotek.

### 5.3.2 Metodika

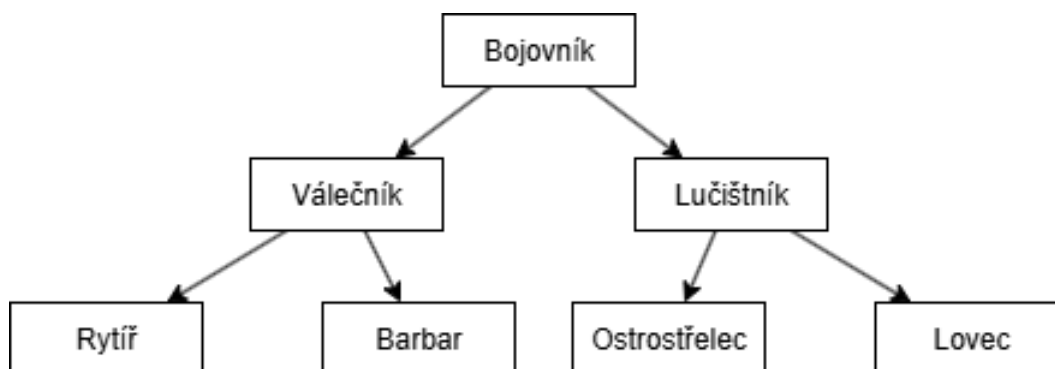
Postup simulací bude následující:

1. **Definice sady atributů:** Nastaví se konkrétní sada atributů pro porovnávané jednotky.
2. **Spuštění simulací:** Pro definovanou sadu atributů se provede 1000 simulací v rámci každého scénáře ("Rovina" a "Pohoří").
3. **Sběr a uložení dat:** Data z každého jednotlivého simulačního běhu jsou uložena ve strukturovaném formátu (CSV). Ukládá se zvlášť souhrnné informace o simulaci a také detailní informace o stavu jednotek v každém kole.
4. **Opakování:** Kroky 1-3 se opakují pro několik dalších sad atributů, aby se shromáždila dostatečná data pro komplexní porovnání.
5. **Analýza dat:** Teprve po shromáždění dat z více sad atributů se provede komplexní analýza a porovnání výsledků.

### 5.3.3 Duely

Primárním cílem je posoudit, zda jsou jednotky na stejné "vývojové úrovni" vzájemně vyrovnané, což je klíčové pro férové a strategicky zajímavé herní prostředí. Každý duel bude testován v různých terénních podmínkách (scénářích), aby se zohlednil vliv prostředí na bojové výsledky.

Zároveň budou jednotky testovány proti jednotkám z nižší úrovně, aby byla zachována vyváženost mezi jednotlivými úrovněmi a jednotky na vyšší úrovni bylo stále nutné využívat ve hře strategicky. Teoretický vývojový strom je vidět na obrázku 5.1



Obrázek 5.1: Teoretická ukázka postupu jakým se jednotky budou v plné hře moci vyvíjet.

### Válečník vs Lučištník

Tento duel porovnává Válečníka, první vývojový stupeň pro boj na blízko s vyšší obranou, a Lučištníka, což je vývojový stupeň vedoucí k boji na dálku s nižší obranou, ale schopností útočit z bezpečné vzdálenosti. Obě jednotky představují základní archetypy bojových jednotek a očekává se, že budou dostupné v rané fázi hry.

Jelikož se jedná o první vývojový stupeň a na této úrovni jsou tedy jen tyto dvě jednotky, očekával bych, že bude souboj vyrovnaný s tím, že válečník bude mít mírnou výhodu na rovině, kde mu nebude nic bránit v rychlém útoku na lučištníka. Zatímco lučištník bude vyhrávat v komplexním terénu, jelikož bude mít více času útočit z bezpečné vzdálenosti.

*Hráč 1* ovládá jednotku *Válečník* a *Hráč 2* má jednotku *Lučištník*.

Ze simulace první sady atributů (sada: 1; tabulka: 5.1) bylo vidět, že souboj trvá neúměrně dlouho velikosti mapy (15x15 polí). Souboj na hladkém terénu trval průměrně téměř deset kol, proto jsem se v první řadě zaměřil na zvýšení rychlosti. Efekt na rychlost hry postupně klesal se zvyšujícím se atributem rychlosti.

Spolu s tím jsem logicky předpokládal, že bude zapotřebí zvyšovat dosah Lučištníka. Výsledky simulací ukázaly, že zvýšení dosahu téměř nebylo zapotřebí.

Výsledky jsou názorněji vidět z grafů: 5.2 který zobrazuje poměr vítězství na rovné mapě a 5.3 který zobrazuje poměr vítězství v komplexním terénu. Pro další simulace tedy jako základ zvýším atributy rychlosti a o jeden bod zvýším dosah lučištníka (sada: 6).

V dalších simulacích odrážejících se od atributů sady 6 z tabulky 5.1, které jsou zopakovány v tabulce 5.3 jako sada 1, už jsem se zaměřil na vyvažování ostatních atributů jednotek za účelem prozkoumat, jak úpravy atributů ovlivní výsledky simulací, tak abych nakonec dosáhl vyvážení základních jednotek.

Tabulka 5.3 ukazuje vyzkoušené varianty úprav atributů. V tabulce 5.4 je pak vidět, že největší fluktuace výsledků způsobuje úprava *obran*y jednotek. Naopak rov-

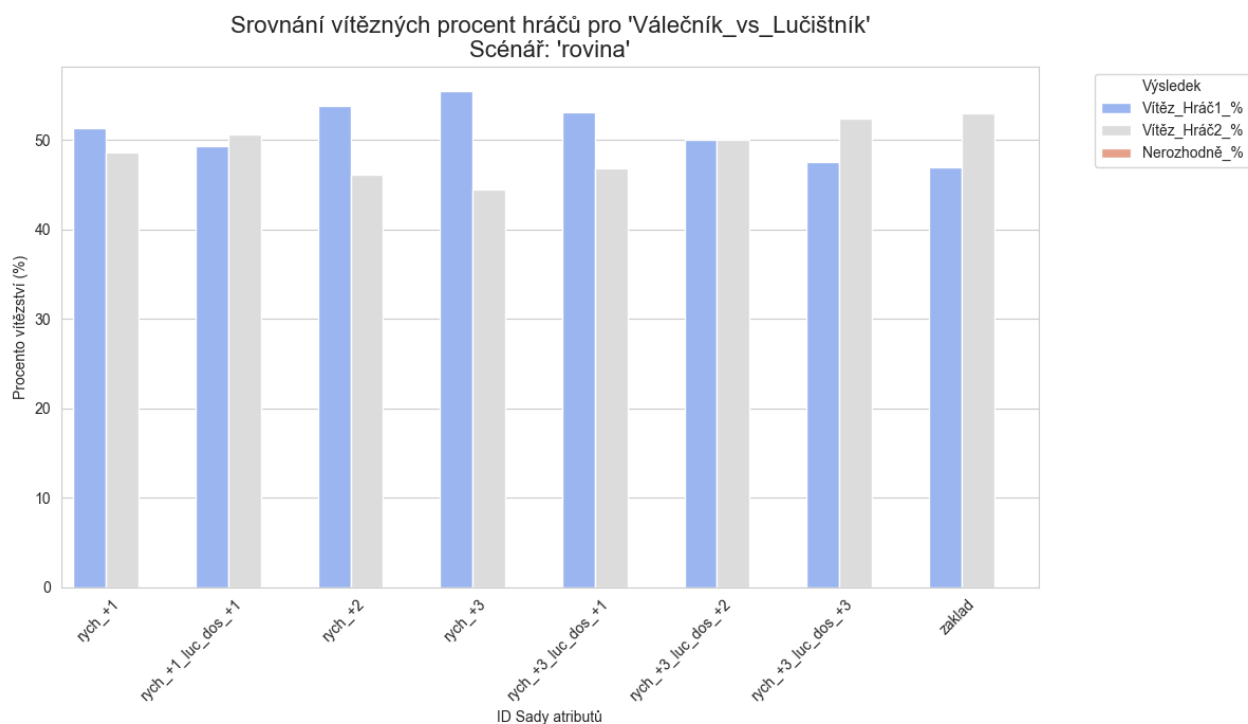
Sada atributů	ID	Typ	Útok Min	Útok Max	Obrana	Rychlost	Dosah	Krit. mult.	Úhyb	Max. životy
zaklad	1	valecnik	6	8	4	3	1	1.2	0.05	15
zaklad	1	lucistnik	6	8	3	3	4	1.2	0.05	10
rych_+1_luc_dos_+1	2	valecnik	6	8	4	4	1	1.2	0.05	15
rych_+1_luc_dos_+1	2	lucistnik	6	8	3	4	5	1.2	0.05	10
rych_+1	3	valecnik	6	8	4	4	1	1.2	0.05	15
rych_+1	3	lucistnik	6	8	3	4	4	1.2	0.05	10
rych_+2	4	valecnik	6	8	4	5	1	1.2	0.05	15
rych_+2	4	lucistnik	6	8	3	5	4	1.2	0.05	10
rych_+3	5	valecnik	6	8	4	6	1	1.2	0.05	15
rych_+3	5	lucistnik	6	8	3	6	4	1.2	0.05	10
rych_+3_luc_dos_+1	6	valecnik	6	8	4	6	1	1.2	0.05	15
rych_+3_luc_dos_+1	6	lucistnik	6	8	3	6	5	1.2	0.05	10
rych_+3_luc_dos_+2	7	valecnik	6	8	4	6	1	1.2	0.05	15
rych_+3_luc_dos_+2	7	lucistnik	6	8	3	6	6	1.2	0.05	10
rych_+3_luc_dos_+3	8	valecnik	6	8	4	6	1	1.2	0.05	15
rych_+3_luc_dos_+3	8	lucistnik	6	8	3	6	7	1.2	0.05	10

Tabulka 5.1: Tabulka sad atributů upravující rychlost souboje Válečníka a Lucíštníka



Scénář: Rovina			Scénář: Hora	
ID	Poměr vítězství Hráče 1 [%]	Počet kol (průměr, min–max)	Poměr vítězství Hráče 1 [%]	Počet kol (průměr, min–max)
1	47.0	9.76 (6–22)	39.5	15.22 (8–37)
2	51.4	7.66 (5–14)	44.2	13.48 (6–27)
3	49.4	7.47 (5–14)	34.0	13.45 (6–30)
4	53.8	6.54 (5–12)	52.2	10.00 (5–25)
5	55.5	5.67 (4–12)	49.9	9.95 (5–23)
6	53.1	5.56 (4–10)	36.1	10.14 (5–23)
7	50.0	5.41 (4–10)	27.5	9.94 (5–23)
8	47.6	5.32 (4–9)	19.1	9.71 (5–22)

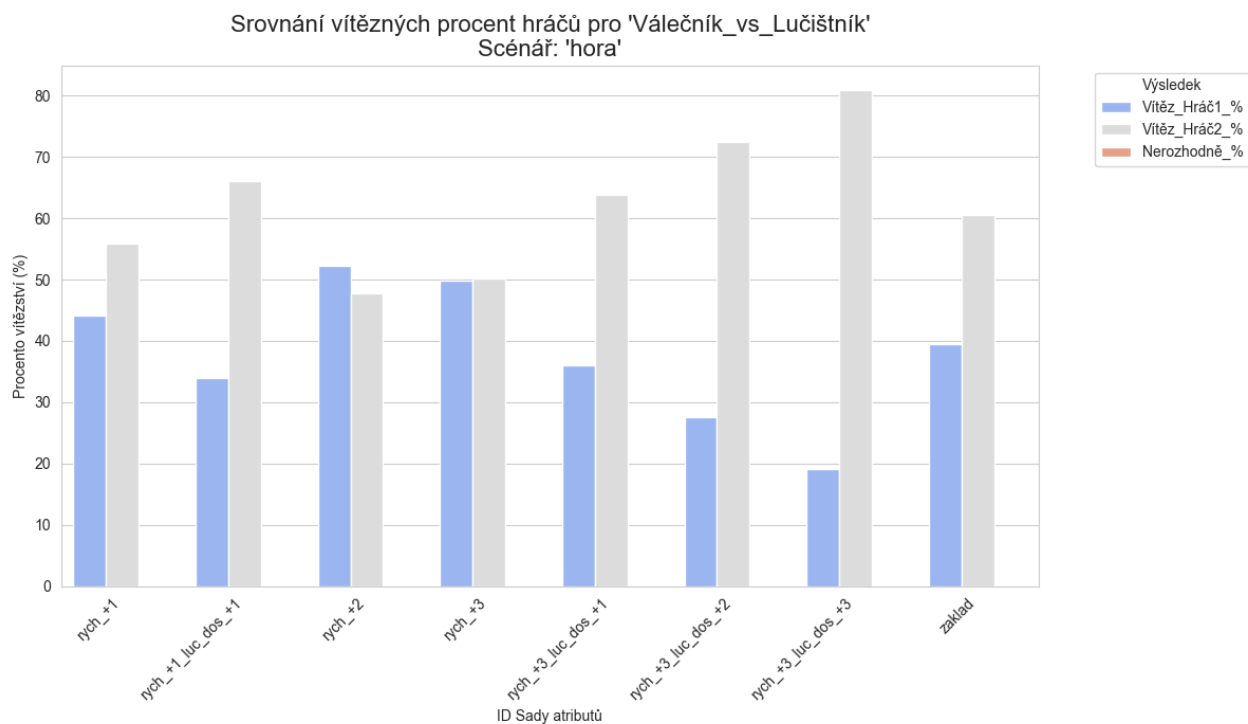
Tabulka 5.2: Tabulka výsledků po úpravách rychlosti pohybu Válečníka a Lučištníka



Obrázek 5.2: Poměr vítězství duelu Válečníka a Lučištníka na rovině.

Sada atributů	ID	Typ	Útok Min	Útok Max	Obrana	Rychlost	Dosah	Krit.	Úhyb	Životy
zaklad2	1	valecnik	6	8	4	6	1	1.2	0.05	15
zaklad2	1	lucisnik	6	8	3	6	5	1.2	0.05	10
val_minAtk_+1	2	valecnik	7	8	4	6	1	1.2	0.05	15
val_minAtk_+1	2	lucisnik	6	8	3	6	5	1.2	0.05	10
val_maxAtk_+1	3	valecnik	6	9	4	6	1	1.2	0.05	15
val_maxAtk_+1	3	lucisnik	6	8	3	6	5	1.2	0.05	10
val_minAtk_-1_maxAtk_+1	4	valecnik	5	9	4	6	1	1.2	0.05	15
val_minAtk_-1_maxAtk_+1	4	lucisnik	6	8	3	6	5	1.2	0.05	10
luc_minAtk_+1	5	valecnik	6	8	4	6	1	1.2	0.05	15
luc_minAtk_+1	5	lucisnik	7	8	3	6	5	1.2	0.05	10
luc_minAtk_-1	6	valecnik	6	8	4	6	1	1.2	0.05	15
luc_minAtk_-1	6	lucisnik	5	8	3	6	5	1.2	0.05	10
luc_maxAtk_+1	7	valecnik	6	8	4	6	1	1.2	0.05	15
luc_maxAtk_+1	7	lucisnik	6	9	3	6	5	1.2	0.05	10
luc_minAtk_-1_maxAtk_+1	8	valecnik	6	8	4	6	1	1.2	0.05	15
luc_minAtk_-1_maxAtk_+1	8	lucisnik	5	9	3	6	5	1.2	0.05	10
val_def_-1	9	valecnik	6	8	3	6	1	1.2	0.05	15
val_def_-1	9	lucisnik	6	8	3	6	5	1.2	0.05	10
val_def_+1	10	valecnik	6	8	5	6	1	1.2	0.05	15
val_def_+1	10	lucisnik	6	8	3	6	5	1.2	0.05	10
luc_def_-1	11	valecnik	6	8	4	6	1	1.2	0.05	15
luc_def_-1	11	lucisnik	6	8	2	6	5	1.2	0.05	10
luc_def_+1	12	valecnik	6	8	4	6	1	1.2	0.05	15
luc_def_+1	12	lucisnik	6	8	4	6	5	1.2	0.05	10
val_luc_minAtk_-1_maxAtk_+1	13	valecnik	5	9	4	6	1	1.2	0.05	15
val_luc_minAtk_-1_maxAtk_+1	13	lucisnik	5	9	3	6	5	1.2	0.05	10
luc_minAtk_-1_val_luc_maxAtk_+1	14	valecnik	6	9	4	6	1	1.2	0.05	15
luc_minAtk_-1_val_luc_maxAtk_+1	14	lucisnik	5	9	3	6	5	1.2	0.05	10
luc_hp_-1	15	valecnik	6	8	4	6	1	1.2	0.05	15
luc_hp_-1	15	lucisnik	6	8	3	6	5	1.2	0.05	9
luc_rych_-1	16	valecnik	6	8	4	6	1	1.2	0.05	15
luc_rych_-1	16	lucisnik	6	8	3	5	5	1.2	0.05	10

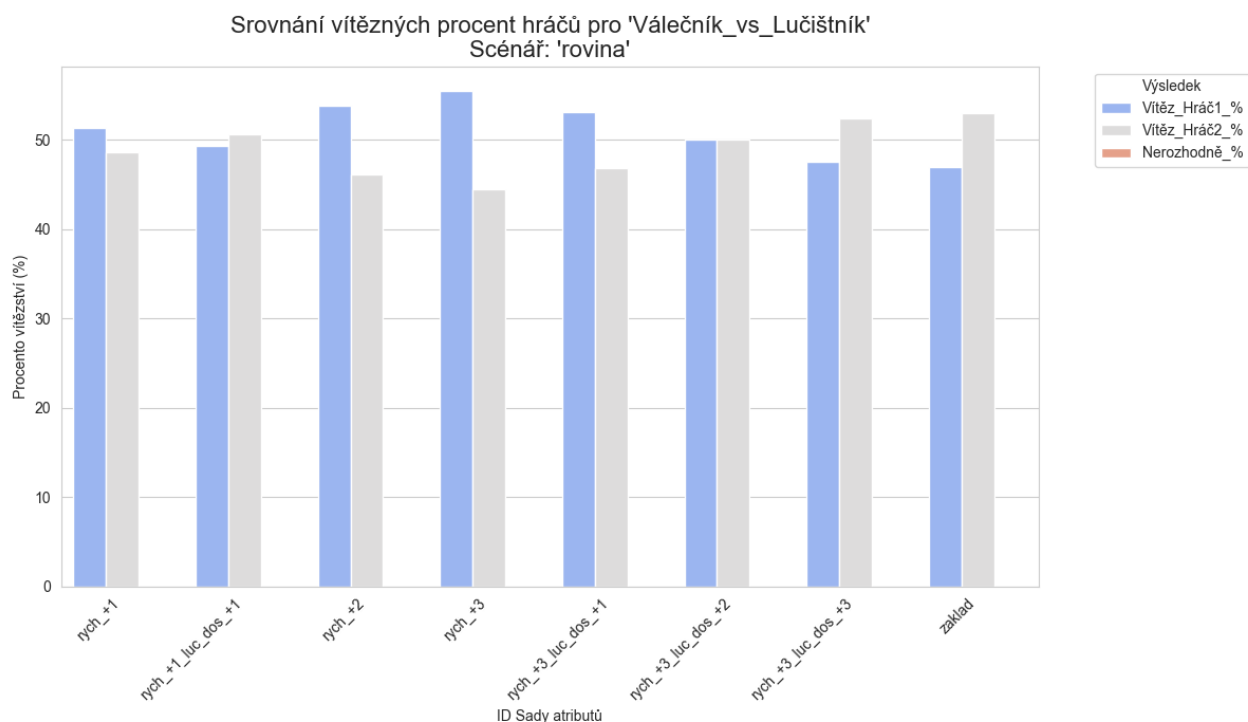
Tabulka 5.3: Tabulka sad atributů souboje Válečníka a Lucíštníka



Obrázek 5.3: Poměr vítězství duelu Válečníka a Lučištníka na horské mapě.

Scénář: Rovina			Scénář: Hora	
ID	Poměr vítězství Hráče 1 { % }	Počet kol (průměr, min–max)	Poměr vítězství Hráče 1 { % }	Počet kol (průměr, min–max)
1	53.1	5.56 (4–10)	36.1	10.14 (5–23)
2	62.2	5.44 (4–10)	46.7	10.09 (5–23)
3	64.7	5.41 (4–10)	47.9	10.06 (5–23)
4	52.0	5.49 (4–10)	38.0	10.11 (5–23)
5	33.2	5.25 (4–10)	17.2	9.36 (5–21)
6	75.6	5.86 (4–12)	58.4	10.52 (5–28)
7	35.1	5.26 (3–11)	20.7	9.36 (5–22)
8	53.9	5.54 (4–10)	34.1	9.96 (5–26)
9	22.6	5.03 (3–9)	9.0	8.89 (5–21)
10	90.3	6.07 (4–10)	85.1	11.16 (6–32)
11	79.3	5.23 (4–9)	61.8	9.93 (5–23)
12	24.7	5.69 (4–10)	12.8	10.24 (5–23)
13	52.9	5.48 (3–10)	36.6	9.92 (5–26)
14	64.4	5.40 (3–10)	44.3	9.83 (5–26)
15	64.3	5.41 (4–10)	47.9	10.07 (5–23)
16	56.0	6.09 (4–11)	40.2	9.09 (5–24)

Tabulka 5.4: Tabulka výsledků simulací soubojů Válečníka a Lučištníka



Obrázek 5.4: Poměr vítězství duelu Válečníka a Lučištníka na rovině.

Útok Min	Útok Max	Obrana	Rychlost	Dosah	Krit.	Úhyb	Životy
5	8	3	5	1	1.2	0.05	12

Tabulka 5.5: Tabulka základní atributů Bojovníka před začátkem simulací.

noměrné rozšíření intervalu poškození vedlo k téměř nulovým změnám v poměru výsledků a k minimální změně trvání hry. Na grafech 5.4 a 5.5 jsou pak znovu vidět poměry vítězství u jednotlivých úprav atributů.

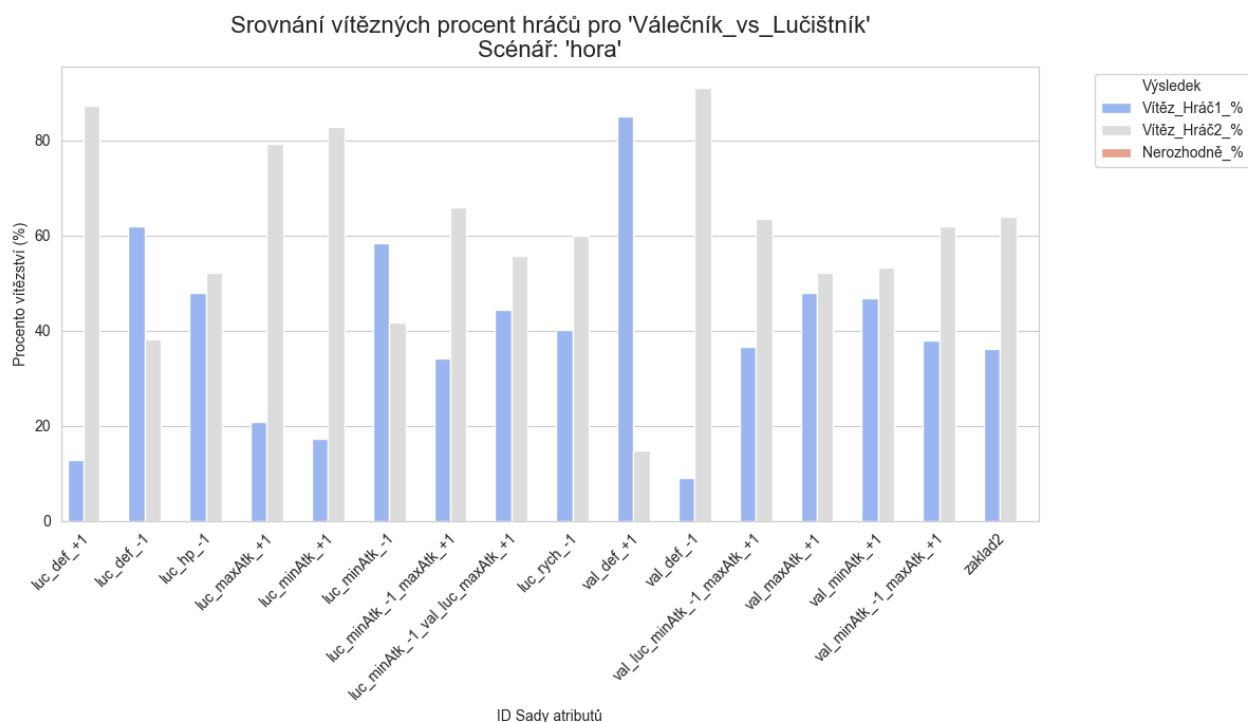
Jako nejvyváženější úprava se nakonec ukázala varianta snížení *rychlosti* Lučištníka o jedna, což vedlo k drobnému zvýšení úspěšnosti Válečníka v obou terénech, tak, že síly jednotek v tomto stavu považují za vyvážené, s tím, že Lučištník více benefituje z komplexního terénu. Stále existující nevyváženost výsledků bude kompenzována tím, že jednotka Lučištník bude dražší na naverbování.

## Bojovník vs Válečník a Bojovník vs Lučištník

Vyvážení duelu Válečník vs Lučištník použiji jako odrazový můstek pro zbytek simulací. V této části budu vyvažovat základní jednotku Bojovník proti jeho vývojem Válečníkovi a Lučištníkovi. Jelikož jsou Válečník i Lučištník přímý vývoj Bojovníka, očekávám, že by měli v podstatě vždy zvítězit, na druhou stranu je potřeba, aby tam byla alespoň šance, takže budu výsledek směřovat na deseti až dvaceti procentní vítězství Bojovníka.

Atributy Bojovníka, od kterých jsem začínal, jsou vypsány v tabulce: 5.5.

	Válečník: Rovina				Lučičník: Rovina				
			Hora				Hora		
ID	Vítězství Bojovník [%]	Průměrný počet kol	Vítězství Bojovník [%]	Průměrný počet kol	Vítězství Bojovník [%]	Průměrný počet kol	Vítězství Bojovník [%]	Průměrný počet kol	Průměrný počet kol
zaklad	1.8	5.43	4.4	8.4	5.0	5.34	3.2	8.46	
boj_uhyb_0.1	2.8	5.53	6.5	8.53	7.7	5.44	4.1	8.6	
boj_rych_+1	1.1	4.99	7.0	9.23	5.0	5.02	3.7	7.68	
boj_minAtk_+1	3.6	5.41	10.1	8.37	8.1	5.32	6.1	8.45	
boj_hp_+3	5.7	5.85	10.9	8.92	13.1	5.69	7.5	9.02	
boj_def_+1	9.3	5.97	18.3	9.63	16.6	5.83	13.1	9.47	
boj_atk_-1+1	3.6	5.41	7.7	8.39	8.8	5.32	6.0	8.44	



Obrázek 5.5: Poměr vítězství duelu Válečníka a Lučištníka na horské mapě.

Útok Min	Útok Max	Obrana	Rychlost	Dosah	Krit.	Úhyb	Životy
5	8	4	5	1	1.2	0.05	12

Tabulka 5.6: Tabulka atributů Bojovníka po dokončení simulací.

Většina úprav atributů nevedla k zásadním změnám v úspěšnosti bojovníka. Z dat je vidět, že největší vliv mělo zvýšení *životů* a zvýšení *obranu*. Zvýšení životů na patnáct (sada: *boj\_hp\_+3*) je těsně pod požadovanou úrovní úspěšnosti, zvýšení životů o jeden nebo dva by pravděpodobně vedlo k ideální úspěšnosti; na druhou stranu by pak Bojovník měl nejen více životů než Lučištník, ale také více než Válečník. Což z hlediska hierarchie vylepšení nedává úplně smysl.

Z toho důvodu vidím jako ideálnější variantu sadu se zvýšenou obranou Bojovníka (*boj\_def\_+1*). Tedy do hry budou použity atributy v tabulce: 5.6.

Tímto nastavením se Bojovník stává dostatečně odolným, aby měl alespoň minimální šanci na vítězství proti svým pokročilejším vývojm, aniž by narušil celkový balanc herních jednotek.

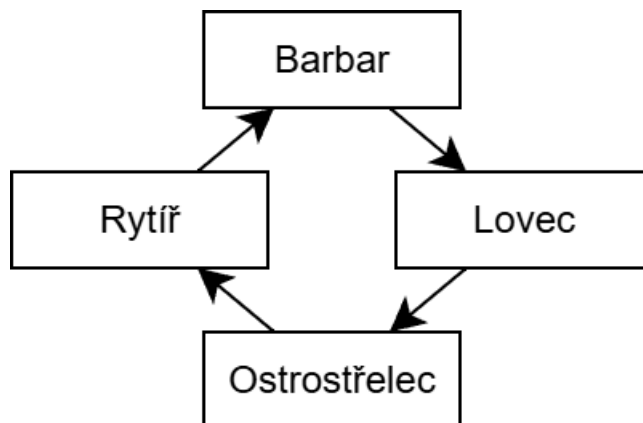
### Simulace jednotek třetí vývojové úrovně

Po vyvážení jednotek první i druhé úrovně, budu v této části provádět simulace za účelem vyvážení jednotek třetí úrovně jak mezi sebou, tak proti jednotkám druhé úrovně.

Třetí úroveň zahrnuje čtyři jednotky: Rytíře, Barbaru, Ostrostřelce a Lovce. Jelikož

	Útok Min	Útok Max	Obrana	Rychlost	Dosah	Krit.	Úhyb	Životy
barbar	8	12	3	7	1	2.5	0.1	15
rytíř	7	10	6	4	1	1.5	0.01	20
lovec	6	8	2	8	4	2.0	0.2	12
ostrostřelec	8	10	2	6	8	1.5	0.05	10

Tabulka 5.7: Tabulka základních atributů jednotek třetí úrovně.



Obrázek 5.6: Diagram ideálního rozložení sil jednotek třetí úrovně mezi sebou.

je zde více jednotek s různými zaměřenými, neočekávám od nich na rozdíl od druhé úrovně vyváženost, spíše doufám, že různé rozložení atributů povede k asymetrickému rozložení sil (silnější/slabší jednotka vůči ostatním) což bude vyžadovat různé strategie pro každou jednotku.

Zároveň je nutné změny testovat proti jednotkám z druhé vývojové úrovně. Protože opět požadují, aby bylo pro slabší jednotky alespoň teoreticky možné vyhrát. V tomto případě bych si představoval úspěšnost mezi deseti a třiceti procenty.

Základní atributy jednotek jsou uvedené v tabulce: 5.7. Moje očekávání toho, která jednotka bude silná proti které, jsem naznačil na obrázku 5.6. Zda je tato představa realistická, se uvidí v průběhu simulací.

Tabulka 5.8 zobrazuje poměr vítězství duelu mezi jednotkami na třetí úrovni, procenta určují poměr vítězství první jednotky ve sloupci.

Základní atributy (sada: 1.) se ukázaly příliš vysoké ve srovnání s jednotkami z druhé úrovně. Tedy prvním krokem bylo všechny jednotky mírně oslabit.

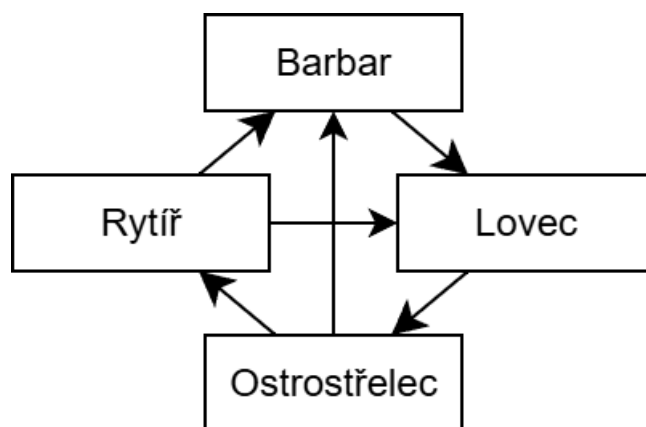
Po oslabení jednotek se ukázalo, že nebude možné upravit atributy tak, aby byly síly jednotek specificky účinné pouze proti jedné konkrétní jednotce stejné úrovně. Přesto se mi po dalších úpravách atributů podařilo dosáhnout výsledků, které považuji za přijatelné (sada: 6.).

Oproti původní představě došlo k rozdělení jednotek třetí úrovně na silnější a slabší. Konkrétně Barbar a Lovec se ukázali jako slabší, respektive více specializované, takže primitivní AI v simulacích nedokáže jejich silné stránky využít ideálním způsobem.

Vítězství Hráč 1 [%]	Barbar Lovec		Barbar Ostrostřelec		Ostrostřelec Lovec		Rytíř Barbar		Rytíř Lovec		Rytíř Ostrostřelec	
	Rovina	Hora	Rovina	Hora	Rovina	Hora	Rovina	Hora	Rovina	Hora	Rovina	Hora
<b>1.</b>	62.6	57.0	44.5	14.1	20.6	22.6	85.3	84.6	99.9	99.4	58.5	34.3
<b>2.</b>	51.8	31.2	31.7	14.6	28.5	33.5	70.2	70.8	91.6	78.8	25.6	10.3
<b>3.</b>	52.5	33.0	38.2	17.3	28.5	34.4	57.5	52.2	91.6	78.4	25.6	10.3
<b>4.</b>	64.3	38.8	43.4	21.8	28.5	34.4	43.2	41.8	87.8	69.9	16.9	6.4
<b>5.</b>	53.1	26.8	27.7	12.1	23.7	30.7	65.4	64.5	92.9	76.7	21.1	8.0
<b>6.</b>	60.3	34.5	27.7	12.1	22.6	22.3	65.4	64.5	95.8	84.9	21.1	8.0

Tabulka 5.8: Tabulka výsledků duelů jednotek třetí úrovně.





Obrázek 5.7: Diagram výsledného rozložení sil jednotek třetí úrovně mezi sebou.

Jednotky Rytíř a Ostrostřelec se ukázali jako efektivnější, tomu samozřejmě přispívá to, že jejich ideální využití je snadné. Výsledný diagram síly proti ostatním je vidět na obrázku: 5.7.

Rozdíl v síle budu opět kompenzovat tím, že lepší jednotky budou dražší jak v pořizovací ceně, tak v ceně za údržbu. Tím se zajistí, že hráči budou déle shromažďovat suroviny na verbování silnějších jednotek a budou nuceni efektivněji využívat jednotky slabší.

Tabulka 5.9 ukazuje, že úpravy atributů vedou k minimálním změnám v délce soubojů. Souboje silnějších jednotek jsou zároveň obecně kratší (výjimkou jsou souboje Rytíře, které trvají podobně dlouho jako souboje slabších jednotek).

## 5.4 Fáze 2: Testování náhodného herního pole

Tato fáze se zaměřuje na testování funkčnosti procedurálně generovaných náhodných map. Je nutné ověřit, zda jsou náhodně vygenerované mapy pro hru vhodné a použitelné. Současně budu kontrolovat vybrané mapy z vizuální stránky, zda vypadají přirozeně.

Cílem této fáze je prověřit, jak spolehlivě generátor map funguje, se zaměřením na tři hlavní oblasti:

- Rozložení terénů: Budu analyzovat procentuální zastoupení jednotlivých typů terénu (voda, pláně, lesy, hory) na generovaných mapách. To pomůže posoudit, jak se rozložení terénů a s nimi spojených zdrojů mění a zda je dostatečně variabilní.
- Průchodnost a konektivita: Zde se zaměřím na to, zda jsou mapy průchozí. Budou zkoumány cesty mezi náhodnými body na mapě, s cílem detekovat zda dochází ke generování neprůchodných map.
- Vizuální inspekce: Kromě kvantitativních dat provedu i vizuální kontrolu map, aby se posoudila jejich estetika, přirozenost a celková strategická zajímavost.

Průměrný počet kol	Barbar Lovec	Barbar Ostrostřelec		Ostrostřelec Lovec		Rytíř Barbar		Rytíř Lovec		Rytíř Ostrostřelec	
		Hora	Rovina	Hora	Rovina	Hora	Rovina	Hora	Rovina	Hora	Rovina
1.	4.31	7.04	3.78	5.53	3.29	3.95	5.41	8.07	5.96	10.28	7.62
2.	4.39	7.72	4.82	6.37	3.35	4.01	5.7	8.89	5.94	10.77	7.19
3.	4.4	7.7	4.8	6.36	3.35	4.02	5.64	8.77	5.94	10.78	7.19
4.	4.2	7.66	4.78	6.35	3.35	4.02	5.54	8.66	5.87	10.57	6.84
5.	3.91	7.04	4.32	5.75	3.32	3.99	5.38	8.49	5.74	10.52	6.96
6.	3.85	6.98	4.32	5.75	3.3	3.89	5.38	8.49	5.54	10.16	6.96

Tabulka 5.9: Tabulka průměrné délky duelů jednotek třetí úrovně.

Tato kontrola pomůže odhalit problémy, které by se v číslech nemusely projevit, například opakující se vzory nebo vizuální nesrovnalosti.

### 5.4.1 Metodika

Postup bude následující:

#### 1. Definování ovladatelných parametrů:

- rozměr herního pole,
- měřítko šumu (`scale`), které ovládá velikost náhodného gradientního pole, podle kterého se pak mapa generuje
- a prahové hodnoty, které definují rozdělení terénních typů.

#### 2. Generování tří map pro vizuální inspekci:

Vygenerují se tři náhodné mapy s danými parametry a ty budu zkoumat zda vypadají přirozeně, herně zajímavě a férově.

#### 3. Generování map pro kvantitativní kontrolu:

Pokud vygenerované mapy projdou vizuální kontrolou (nebude s nimi zásadní problém), vygeneruje se tisíc map, ke každé se uloží poměr množství políček jednotlivých typů. Dále bude pro každou mapu náhodně vygenerováno sto dvojic startovních a cílových bodů, u nichž se zaznamená úspěšnost nalezení cesty.

### 5.4.2 Testování měřítka šumu

V první řadě začnu testovat krajní případy `scale`, pak otestuji případy mezi extrémy a na těch nejlepších budu testovat změny prahových hodnot.

Na začátek používám vyrovnané prahové hodnoty:

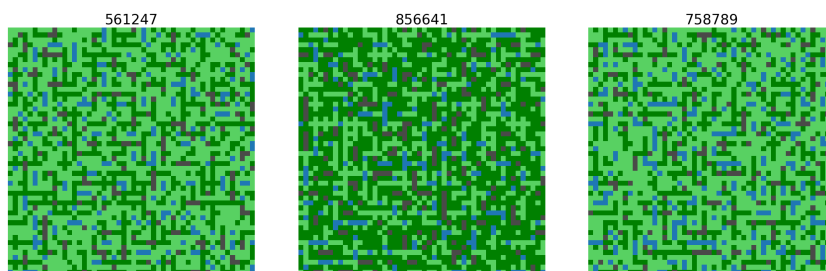
- Voda:  $[0 - 0.25)$
- Pláně:  $[0.2 - 0.5)$
- Lesy:  $[0.5 - 0.75)$
- Hory:  $[0.75 - 1]$

#### Malé `scale`

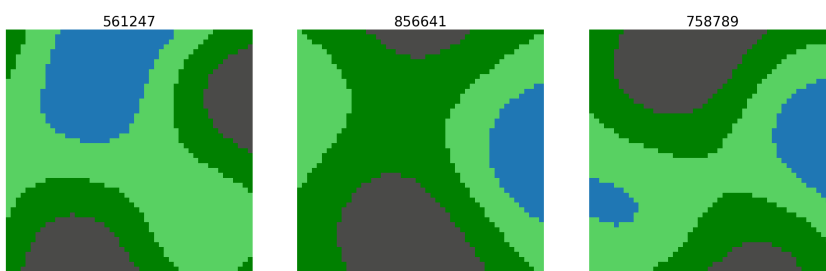
Malé měřítko šumu znamená, že základní náhodná mřížka, ze které dostávám výslednou mapu, je téměř stejně velká jako výsledná mapa, a tedy vygenerované struktury na mapě jsou malé.

Jako malé "hraniční" `scale` jsem použil `scale = 2` na mapě  $50 \times 50$ .

Na obrázku 5.8 je vidět, že struktury jsou tak malé, že mapy vypadají jako náhodné, takže nemá smysl s těmito parametry pokračovat.



Obrázek 5.8: Mapy vygenerované pro vizuální hodnocení. S nadpisy určujícími seed použití pro jejich vygenerování. `scale = 2`



Obrázek 5.9: Mapy vygenerované pro vizuální hodnocení. S nadpisy určujícími seed použití pro jejich vygenerování. `scale = 50`

### Velké scale

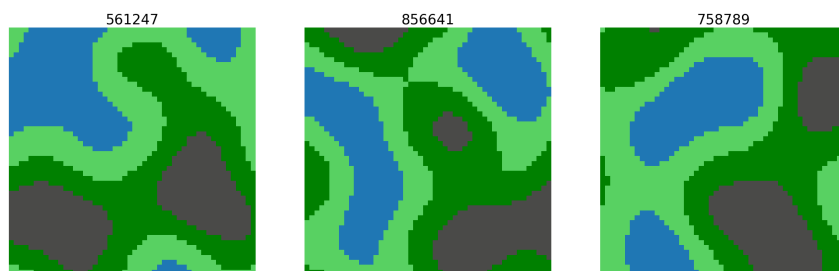
Naopak velké měřítko šumu vede k minimální velikosti náhodné mřížky, "interpolace" mezi malým počtem bodů vytváří velké souvislé struktury. Jako velké `scale` jsem použil `scale = 50` na mapě  $50 \times 50$ .

Na obrázku 5.9 je vidět, že výsledný terén vypadá poměrně přirozeně, ale z herního hlediska není moc strategicky zajímavý. Ačkoliv to stále není ideální výsledek, provedu pro toto nastavení kvantitativní sběr dat.

### Střední scale

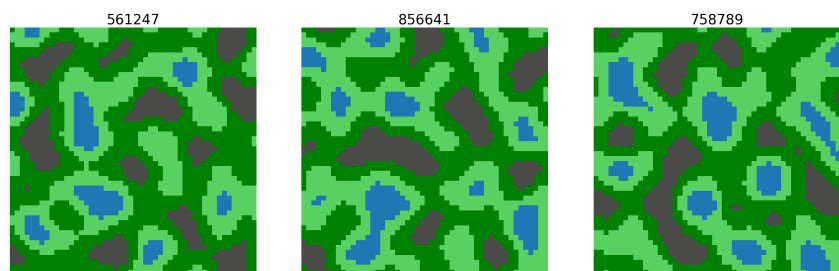
Velké měřítko šumu se ukázalo jako výrazně lepší varianta, takže použiju další největší číslo, kterým je možné celočíselně dělit velikost herní mapy. To na mapě  $50 \times 50$  znamená `scale = 25`.

Na obrázku 5.10 je vidět, že výsledný terén opět vypadá poměrně přirozeně, ale navíc už má určitou úroveň komplexity, která jej činí vhodným pro využití jako herní mapa.



Obrázek 5.10: Mapy vygenerované pro vizuální hodnocení. S nadpisy určujícími seed použitý pro jejich vygenerování.

`scale = 25`



Obrázek 5.11: Mapy vygenerované pro vizuální hodnocení. S nadpisy určujícími seed použitý pro jejich vygenerování.

`scale = 10`

### Další varianta `scale`

Zmenšení `scale` vedlo ke zvýšení komplexity, otázka je, zda další zmenšení nepovede k chaotické mapě, podobně jako tomu bylo u malého `scale`. Další celočíselné dělení mapy  $50 \times 50$  je `scale = 10`.

Výsledné mapy, jak je vidět na obrázku 5.11, jsou podle očekávání mnohem komplexnější, což je samozřejmě lepší pro hru ze strategického hlediska. Na druhou stranu už mapa začíná působit poněkud přelácaně, takže menší měřítko šumu již nebudu testovat.

### Zvětšená mapa

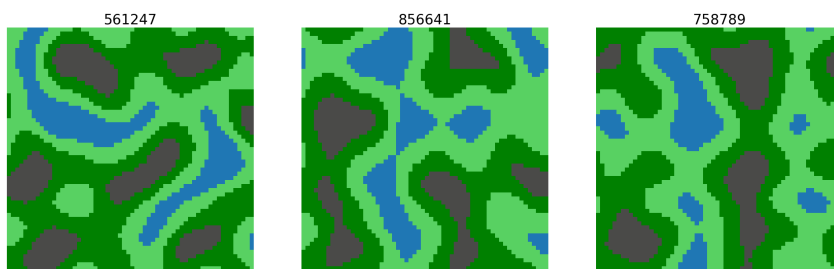
Jelikož měřítko šumu 25 je spíše nedostatečně komplexní pro hru a měřítko 10 začíná působit chaoticky, respektive neorganicky, zkusím upravit velikost, tak abych mohl použít měřítko někde mezi tím. Tedy použiji `scale = 20` na mapě  $60 \times 60$ .

Výsledné mapy, na obrázku 5.12 jsou, jak se dalo očekávat, velmi podobné mapám s měřítkem 25, pouze působí více přirozeně, tím, že jsou struktury komplexnější.



Obrázek 5.12: Mapy vygenerované pro vizuální hodnocení. S nadpisy určujícími seed použitý pro jejich vygenerování.

`scale = 20`



Obrázek 5.13: Mapy vygenerované pro vizuální hodnocení. S nadpisy určujícími seed použitý pro jejich vygenerování.

`scale = 15`

### Menší `scale` na větší mapě

Jako poslední simulaci zkusím ještě jednou zmenšit měřítko šumu. Tedy použiji `scale = 15` na mapě  $60 \times 60$ .

Mapy na obrázku 5.13 působí velmi organicky a jsou strategicky dostatečně komplexní, i přesto nepůsobí příliš chaoticky. Tedy bych tuto variantu považoval po vizuální stránce za nejlepší.

### Shrnutí

Testy ukázaly, že měřítko šumu (`scale`) výrazně ovlivňuje podobu generovaných map. Příliš malé měřítko (`scale = 2`) vedlo k chaotickým mapám, příliš velké (`scale = 50`) sice působilo přirozeně, ale postrádalo strategickou hloubku.

Optimální výsledky na mapě  $50 \times 50$  poskytl `scale = 25`. Při větší mapě  $60 \times 60$  se jako nejvhodnější ukázalo `scale = 15`, které dosahuje nejlepší rovnováhy mezi vizuální přirozeností, komplexitou a průchodností.

Průměrná průchodnost každého simulovaného nastavení je vidět v tabulce 5.10. Kde je zároveň vidět, že čím je `scale` menší, tím více stoupá průchodnost, což jsem na

	Voda[%]	Pláně[%]	Les[%]	Hory[%]	Existující cesty [%]
<b>scale=50</b>	21.03284	28.81832	28.60976	21.53908	95.78300
<b>scale=25</b>	16.66204	33.66308	33.22772	16.44716	96.33500
<b>scale=10</b>	11.56728	38.70728	38.34388	11.38156	99.73400
<b>60 x 60 scale=20</b>	13.470472	36.481278	36.672889	13.375361	98.658000
<b>60 x 60 scale=15</b>	12.860944	37.137472	37.222042	12.779542	99.289000

Tabulka 5.10: Tabulka průměrných výsledků získaných generováním náhodných map s různou velikostí **scale**

začátku nečekal. Pro testování různých prahových hodnot použiji právě zmíněné mapy se **scale=25** a **scale=15**.

### 5.4.3 Testování různých prahových hodnot

Po identifikaci optimálních měřítek šumu (**scale**), konkrétně **scale = 25** pro mapu  $50 \times 50$  a **scale = 15** pro mapu  $60 \times 60$ , se dál zaměřím na testování prahových hodnot. Ty přímo ovlivňují poměrné zastoupení jednotlivých terénních typů na generované mapě. Cílem je doladit distribuci vody, pláně, lesů a hor tak, aby výsledné mapy byly strategicky vyvážené a herně zajímavé.

#### Shrnutí

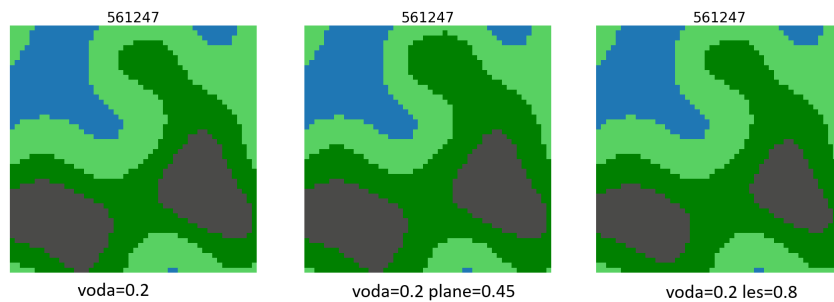
Úprava prahových hodnot nepovede k zásadním vizuálním změnám, pouze upravuje poměry jednotlivých typů terénu, proto budou změny spíše minimální. Cílem bude zvětšit prostor, na kterém se mohou hráči pohybovat, ale zachovat komplexnost terénu, aby bylo strategické manévrování stále užitečné.

Základní prahové hodnoty, od kterých budu upravovat, jsou tyto:

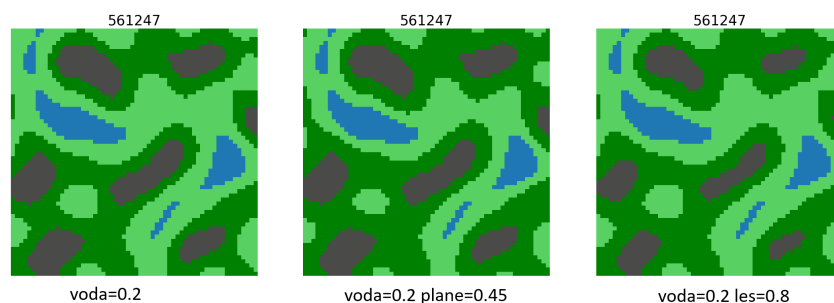
- Voda:  $[0 - 0.25)$
- Pláně:  $[0.2 - 0.5)$
- Lesy:  $[0.5 - 0.75)$
- Hory:  $[0.75 - 1]$

Z tabulky 5.11 je vidět, že podle očekávání, snížení hranice vody vede ke zvýšení úspěšnosti průchodů mapy. Zlepšení průchodnosti je samozřejmě pozitivní, ale spolu s tím klesá komplexnost terénu, jelikož mizí zcela neprůchozí bariéry, proto nemohu zásadně snižovat hranici pro vodu.

Na obrázcích 5.14 a 5.15 je vidět změna vybraných map při změně prahových hodnot. Po zvážení vizuálních výsledků a dat z kvantitativních simulací jsem zvolil jako nejlepší variantu:



Obrázek 5.14: Ukázka map po úpravě hraničních hodnot. `scale = 25`



Obrázek 5.15: Ukázka map po úpravě hraničních hodnot. `scale = 15`

	Voda[%]	Pláně[%]	Les[%]	Hory[%]	Existující cesty [%]
<b>50x50 voda=0.2</b>	11.84624	38.47888	33.22772	16.44716	98.52400
<b>60x60 voda=0.2</b>	8.056778	41.988306	37.771194	12.183722	99.809000
<b>50x50 voda=0.2 plan=0.45</b>	11.84624	31.07976	40.62684	16.44716	98.52400
<b>60x60 voda=0.2 plan=0.45</b>	8.056778	33.313444	46.446056	12.183722	99.809000
<b>50x50 voda=0.2 les=0.8</b>	11.84624	38.47888	37.97944	11.69544	98.52400
<b>60x60 voda=0.2 les=0.8</b>	8.056778	41.988306	41.941556	8.013361	99.809000

Tabulka 5.11: Tabulka průměrných výsledků získaných generováním náhodných map s různými prahovými hodnotami.



- Voda:  $[0 - 0.2)$
- Pláně:  $[0.2 - 0.5)$
- Lesy:  $[0.5 - 0.75)$
- Hory:  $[0.8 - 1]$

která maximalizuje herní prostor, po kterém se mohou jednotky snadno pohybovat, čímž zvyšuje šance jednotek na blízko, ale zároveň stále zachovává dostatečnou rozmanitost, tak aby hráči mohli strategicky využívat prostředí. Zároveň redukce množství hor na mapě oddaluje získání zdrojů pro verbování finálních jednotek, čímž teoreticky nutí hráče využívat jednotky na nižší úrovni.

# Závěr

Hlavním cílem výzkumného úkolu bylo prozkoumat a prakticky aplikovat principy procedurálního generování obsahu (PCG) pro tvorbu herních světů a následně implementovat strategickou tahovou hru využívající náhodně vygenerované mapy.

V první kapitole jsem rozebral problematiku procedurálního generování obsahu (PCG), analyzoval jeho využití, výhody a nevýhody. Na základě detailního srovnání různých metod PCG vhodných pro generování náhodných herních map (např. dělení prostoru, fraktálové algoritmy či komplexnější metody jako wave function collapse) jsem vybral jako ideální pro mé potřeby a omezení metodu výškových map.

Ve druhé kapitole jsem se zaměřil na implementaci zvolené metody výškových map. Tu jsem implementoval ve třech variantách a to konkrétně jako náhodnou výškovou mapu, interpolovanou výškovou mapu a gradientní výškovou mapu. Zároveň jsem v této kapitole zmínil, jak výsledné výškové mapy převádím na políčka s různými typy terénu, a pro účely prototypu jsem představil, jak výsledné mapy vypadají.

Ve třetí kapitole jsem si nastudoval teorii návrhu videoher a na základě zpracovaných znalostí jsem připravil návrh mé strategické hry. Specifikoval jsem herní prostor, objekty a jejich možné stavy a atributy, dále akce hráče, pravidla a cíle hry, a zmínil jsem důležitost dovednosti vs náhody ve hře.

Ve čtvrté kapitole jsem implementoval prototyp obsahující základní herní mechaniky navržené strategické hry, včetně pohybu jednotek, správy zdrojů a bojového systému. Klíčovým prvkem byla také příprava jednoduché herní umělé inteligence, která se stala nezbytnou pro automatizované testování a systematickou optimalizaci v závěrečné fázi projektu.

V poslední páté kapitole jsem využil implementovaný prototyp k simulacím herních scénářů. Konkrétně jsem se zaměřil na duely jednotek, aby jednotky s podobnou hodnotou měly podobné síly. Pomocí metody Monte Carlo jsem optimalizoval atributy herních objektů, čímž jsem dosáhl vyvážení hratelnosti. Dále jsem stejnou metodou optimalizoval parametry pro generování náhodných map, což vedlo ke zvýšení průchodnosti map a zvýšení jejich strategické zajímavosti a variability.

Celkově tento výzkumný úkol poskytl komplexní vhled do procedurálního generování obsahu pro herní účely. Byly navrženy a implementovány systémy pro generování herního prostředí a základní herní mechaniky, jejichž vyvážení bylo empiricky ověřeno a optimalizováno pomocí simulačních metod. Výsledky práce poslouží jako vhodný základ pro další práci v rámci diplomové práce, kde se zaměřím na dokončení hry a využití strojového učení pro herní umělou inteligenci.

# 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](https://antoniosliapis.com/articles/pcgbook_dungeons.php). [cit. 2025-01-13].
- [4] GEEKS FOR GEEKS. *Binary Space Partitioning*. Online. [www.geeksforgeeks.org](http://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](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](http://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

## Tabulky výsledků simulací

	Válečník:				Lučištník:
	Rovina		Hora		Rovina
ID	Vítězství Bojovník [%]	Průměrný počet kol	Vítězství Bojovník [%]	Průměrný počet kol	Vítězství Bojovník [%]
zaklad	1.8	5.43	4.4	8.4	5.0
boj_uhyb_0.1	2.8	5.53	6.5	8.53	7.7
boj_rych_+1	1.1	4.99	7.0	9.23	5.0
boj_minAtk_+1	3.6	5.41	10.1	8.37	8.1
boj_hp_+3	5.7	5.85	10.9	8.92	13.1
boj_def_+1	9.3	5.97	18.3	9.63	16.6
boj_atk_-1+1	3.6	5.41	7.7	8.39	8.8