VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HRA V UNITY S VYUŽITÍM GENETICKÝCH ALGORITMŮ

GENETIC ALGORITHMS BASED GAME IN UNITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **RADOMÍR BÁBEK**
AUTHOR

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

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

153371

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Bábek Radomír**
Program:	Informační technologie
Název:	**Hra v Unity s využitím genetických algoritmů**
Kategorie:	Počítačová grafika
Akademický rok:	2023/24

Zadání:

1. Prozkoumejte techniky herního vývoje v prostředí Unity.
2. Nastudujte tématiku genetických algoritmů a k tomu vztažené operátory.
3. Navrhněte počítačovou hru v prostředí Unity s využitím prvků genetických algoritmů.
4. Implementujte navrženou hru.
5. Zhodnoťte dosažené výsledky a možnosti publikace hry.
6. Vytvořte demonstrační video.

Literatura:

- Kora, Padmavathi & Yadlapalli, Priyanka. (2017). Crossover Operators in Genetic Algorithms: A Review. International Journal of Computer Applications. 162. 34-36. 10.5120/ijca2017913370.
- Schmitt, Lothar M.. “Theory of genetic algorithms.” *Theor. Comput. Sci.* 259 (2001): 1-61.
- Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. 2017. Fast genetic algorithms. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17). Association for Computing Machinery, New York, NY, USA, 777–784. https://doi.org/10.1145/3071178.3071301
- Marsili Libelli, S., Alba, P. Adaptive mutation in genetic algorithms. *Soft Computing* **4**, 76–80 (2000). https://doi.org/10.1007/s005000000042

Při obhajobě semestrální části projektu je požadováno:
Body 1, 2, 3 a prototyp bodu 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Vlnas Michal, Ing.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2023
Termín pro odevzdání:	9.5.2024
Datum schválení:	9.11.2023

Abstrakt

Tato práce se zaměřuje na oblasti herního vývoje a genetických algoritmů. Popisuje návrh a implementaci hry PlantEVO, ve které figurují speciální bojové rostliny. Úkolem hráče je pochopit fungování genetického algoritmu a využít ho co možná nejefektivněji při šlechtění svých roslin. Následně je síla jeho rostlin otestována v online soubojích proti ostatním hráčům. Souboje probíhají ve stylu tahové strategie. Hra využívá architekturu klient-server. Klien je naprogramován v herním enginu Unity, server využívá architekturu REST API.

Abstract

This thesis focuses on the areas of game development and genetic algorithms. It describes the design and implementation of the game PlantEVO, which features special fighting plants. The player's task is to understand the workings of the genetic algorithm and to use it as efficiently as possible in breeding his plants. After then the strength of his plants is tested in online battles against other players. The battles are designed in a turn-based strategy combat style. The game uses a client-server architecture. The client is programmed in the Unity game engine, the server uses the REST API architecture.

Klíčová slova

genetické algoritmy, digitální hry, mutace, křížení, selekce, fitness funkce, genetické operátory, Unity, hry pro více hráčů, serverové architektury, REST API, klient-server, herní žánry, historie herního vývoje, PlantEVO, tahová strategie, rostliny

Keywords

genetic algorithms, digital games, mutation, crossover, selection, fitness functions, genetic operators, Unity, multiplayer games, server architectures, REST API, client-server, game genres, game development history, PlantEVO, turn-based strategy, plants

Citace

BÁBEK, Radomír. *Hra v Unity s využitím genetických algoritmů*. Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Vlnas

Hra v Unity s využitím genetických algoritmů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Vlnase. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Radomír Bábek
9. května 2024

Poděkování

Děkuji vedoucímu práce Ing. Michalu Vlnasovi za rychlou a vyčerpávající zpětnou vazbu k jednotlivým kapitolám, za nápady ve fázi herního návrhu, užitečné rady a pozitivně naladěné vedení po celý průběh práce. Dále děkuji rodině a přátelům za podporu.

Obsah

Kapitola 1

Úvod

Evoluční vědy jsou fascinující vědní obor. Již tisíce let se lidstvo snaží nalézt odpovědi na otázky za vznikem života. I přestože řada klíčových otázek zůstává bez odpovědi, poznání lidstva neustále roste. Důkazem toho je například aktuální porozumění v oblasti genetiky, kterou lidstvo poměrně detailně zmapovalo. Obor informačních technologií nezůstává tomuto vědění uzavřen. Za pomocí evolučních a genetických principů vytváří účinný nástroj v podobě genetického algoritmu. Ten dokáže řešit širokou škálu rozličných problémů, tato práce ho však plánuje využít čistě pro zábavu. Jeho motivem zájmu jsou speciální bojové rostliny, které hráč pěstuje a šlechtí. Fascinaci lidstva nad životem a genetikou lze totiž převést do herních světů stejně jako cokoliv, co dokáže projít z lidské představivosti na papír.

Tato práce se snaží být syntézou oblastí genetických algoritmů a digitálních her. V obou se snaží poskytnout základní přehled do daného oboru. Dále zasahuje do oblasti herních serverů, kdy popisuje technologie a principy využívané při jejich tvorbě.

Výsledkem práce je hra PlantEVO, ve které figuruje genetický algoritmus jako jeden z klíčových prvků hry. Jeho využití probíhá při šlechtění rostlin, které jsou zobrazeny pomocí 3D modelů. S rostlinami lze následně bojovat proti ostatním hráčům v online soubojovém systému přes internet.

V kapitole 2 jsou vysvětleny základní pojmy genetických algoritmů nutné pro jejich pochopení. Dále jsou popsány základní principy genetických operátorů, které jsou využívány při běhu algoritmu. V kapitole 3 se práce zabývá všeobecným přehledem v oblasti herního vývoje, více technicky se zaměřuje na žánr her pro více hráčů, kde jsou popsány populární technologie a principy, které lze využít pro jejich sestrojení. Kapitola 4 představuje hru PlantEVO a popisuje její herní mechaniky. Tvorba hry je popsána v kapitole 5, ta je rozdělena na programování serveru a herního klienta.

Kapitola 2

Úvod do genetických algoritmů

Genetický algoritmus patří mezi algoritmy inspirovanými přírodou. Jeho původním autorem je John Henry Holland, který v roce 1975 sepsal poznatky svého výzkumu v knize *Adaptation in Natural and Artificial Systems* [8]. Autor se zamýšlí nad způsobem, jak zefektivnit proces tvorby a optimalizace algoritmů a popisuje využití evoluce a přirozeného výběru v této problematice. Tyto teorie definované Charlesem Darwinem v sobě nesou cenné informace o tom, jak se s hledáním řešení na širokou škálu rozličných problémů vypořádává příroda. Díky tomu, že chápání evoluce (a s ní i proces reprodukce jedinců) je na úrovni, kdy je možné jednotlivé klíčové aspekty vyjmenovat a popsat, lze také evoluci v omezené míře algoritmizovat a simulovat.

Do konce 20. století bylo téma genetických algoritmů několikrát znovu navštíveno, jejich využití se postupně zpopularizovalo zejména v odvětvích strojového učení, prohledávacích algoritmů a optimalizačních úloh.

2.1 Definice

Genetické algoritmy jsou heuristické modely učení založené na principech evoluce a selektivního křížení [7].

Při popisu genetického algoritmu lze mluvit o jeho tzv. kanonické formě (viz algoritmus 1). Ta definuje základní sled operací, které se většinou v jakékoliv formě genetického algoritmu objevují. Na začátku proběhne inicializace počáteční populace, nad kterou algoritmus pracuje. Následuje opakování operací selekce, křížení a mutace v tomto pořadí, dokud nenastane ukončovací podmínka. Ta může být definována nejmenší postačující kvalitou řešení, konečným počtem iterací, či jinou, například časovou, podmínkou.

Genetický algoritmus je vždy definován nad určitým problémem. Často se pomocí něj řeší problémy, u kterých je příliš mnoho kombinací na manuální vyzkoušení, ale jednotlivá řešení lze snadno kvalitativně odlišit. Výstupem genetického algoritmu je nejlepší nalezené řešení problému dosažené pomocí simulace evoluce.

Algoritmus 1 Kanonická forma genetického algoritmu [4]

1: Nechť P je populace jedinců; $\mu = |P|$; μ je sudé číslo
2: **Inicializace:** náhodně inicializuj jedince $\gamma_i \in P, i = 1, \ldots, \mu$.
3: **Ohodnocení:** každému jedinci v P přiřaď odpovídající fitness.
4: **repeat**
5: **Selekce:** vyber μ jedinců z P do intermediární populace P' (*mating pool*).
6: **for** $i = 1$ to $\mu/2$ **do**
7: Náhodně vyber dva jedince $p_1, p_2 \in P'$ za rodiče.
8: **Křížení:** s pravděpodobností p_{cross} vytvoř dvojici potomků ψ_1, ψ_2,
9: **if** ke křížení nedošlo **then**
10: pak $\psi_1 = p_1$, $\psi_2 = p_2$.
11: **end if**
12: **Mutace:** s pravděpodobností p_{mut} mutuj každý gen ψ_1, ψ_2.
13: **Nahrazení:** potomky ψ_1, ψ_2 postupně nahrazuj jedince v P.
14: **end for**
15: **Ohodnocení:** každému jedinci v P přiřaď odpovídající fitness.
16: **until** ukončující kritérium

Reprezentace jedince

Jedním z kroků implementace genetického algoritmu je návrh genetického kódování jedince. Toto kódování bývá nejčastěji ve formě seznamu bitů či celých nebo reálných čísel. Jedinec ve tvaru genetického kódu je v kontextu genetických algoritmů často nazýván jako chromozom, jednotlivé bity či čísla bývají označovány jako geny. Jedno uspořádání genů odpovídá jednomu konkrétnímu řešení problému. Při definování chromozomu je nutné vzít v úvahu všechny možnosti, které mohou v problému nastat, a nelze je hned ze začátku vyloučit jako nevyhovující řešení a tedy i z problému samotného. Nejlepší nalezený jedinec bývá zpravidla výstupem genetického algoritmu.

V kontextu reprezentace jedince se dále definují 2 spolu související pojmy: genotyp a fenotyp. Genotypem se označuje kódování bez přidaného sémantického významu v podobě seznamu hodnot použitých k zápisu konkrétního řešení. Mimo jiné lze genotyp podrobit genetickým operátorům křížení a mutace (tyto operátory jsou podrobně popsány v sekcích 2.2 a 2.3).

Fenotyp naopak definuje význam jednotlivých genů a překládá jej na sémantickou reprezentaci. Jedince ve formě fenotypu lze kvalitativně ohodnotit pomocí fitness funkce, která je popsaná v sekci 2.1.

Fenotyp a genotyp mohou mít stejnou syntaktickou podobu, není tomu však pravidlem. V případě odlišnosti jejich syntaxe je třeba vytvořit mapovací funkci, která dokáže přeložit genotyp jedince na fenotyp a obráceně.

Populace

Genetické algoritmy patří do skupiny populačně orientovaných algoritmů. Již bylo zmíněno, že základním principem evoluce je přirozený výběr. Ten je třeba aplikovat nad skupinou jedinců. Touto skupinou je v kontextu genetických algoritmů populace. V hlavní smyčce algoritmu se volají operátory selekce, křížení a mutace nad jedincemi vybraných z populace

podle kritériích pro přirozený výběr. Výsledkem jedné iterace je obměna aktuálních jedinců populace za nové, kteří vznikli při reprodukci vybraných jedinců předcházející populace.

Při spuštění algoritmu je třeba populaci inicializovat a sestrojit tzv. počáteční populaci. Té lze v praxi dosánout například naplněním populace náhodnými jedinci, lze však využít i jiné techniky inicializace, které mohou být efektivnější pro konkrétní problém.

Jakmile dojde k nahrazení všech jedinců aktuální populace, nastává vznik nové generace. Každá generace má pořadníkové číslo, které začíná na čísle 0 nebo 1 pro generaci definovanou počáteční populací, a s každou další obměnou jedinců se toto číslo zvyšuje. U správně nastaveného genetického algoritmu lze s rostoucím počtem generací sledovat postupný růst průměrné hodnoty fitness funkce jedinců v rámci populace.

Funkce Fitness

Funkce fitness se využívá při simulování přirozeného výběru v operátoru selekce. Jejím vstupem je kandidátní řešení ve formě fenotypu, výstup pak ukazuje kvalitu jedince v kontextu daného problému. Hodnota fitness bývá přirovnávána k šanci jedince na přežití a rozhoduje o jeho budoucí reprodukci.

Nalezení vhodné fitness funkce je kritická součást tvorby genetického algoritmu. Kvalita funkce je definována její objektivitou. Neobjektivní fitness nesprávně označuje méně schopné jedince za schopné, a tím koreluje k neoptimálnímu řešení. Funkce je sestavena dle žádoucích kritérií pro hledané řešení optimalizačního problému.[1] Je nutné tato kritéria dobře znát, pravděpodobně však nebude možné sestrojit funkci, která by neobsahovala určitou úroveň generalizace. V opačném případě by nejspíše bylo možné hledané řešení nalézt bez použití genetického algoritmu. Platí však, že objektivnější fitness funkce koreluje k řešení rychleji než fitness funkce neobjektivní.

Vyjma statického výpočtu se v roli fitness funkce využívá i například výsledek běhu simulace. Při výběru složitější funkce je nutné uvážit dobu potřebnou pro její výpočet. Ta by měla být co možná nejnižší. Tato hodnota se totiž násobně projeví v celkové délce běhu algoritmu.

2.2 Genetické operátory

Název „genetické operátory“ je v literatuře používán pro tři základní operace genetického algoritmu. Jedná se o operace selekce, křížení a mutace. Je možné sestavit i genetický algoritmus, který nevyužívá všech tří operátorů a při generování nových řešení využívá například pouze operátor mutace či křížení. Běžnou praxí je také přidávání vah k operacím křížení a mutace, kdy výběr váhy je jedním z klíčových nastavení pro běh genetického algoritmu.

Selekce

Operátor selekce simuluje přirozený výběr a slouží k výběru nejlepších jedinců z celkové populace do tzv. mating pool [4]. Ta se dá definovat jako seznam, do kterého se dle vybrané strategie selekce vkládají jedinci z populace, až do chvíle, kdy se velikost mating pool rovná velikosti populace. Jedinec může být přidán do mating pool i vícekrát. Jedinci z mating pool se následně náhodně párují a podrobují procesu reprodukce, který je podrobněji popsán v sekci 2.2.

[1] U živých organizmů fitness odpovídá úspěšnosti organizmu ve svém přirozeném prostředí.

V oblasti selekce je definován takzvaný selekční tlak. Čím vyšší je selekční tlak, tím více selekce propaguje jedince s lepší fitness do mating pool. Příliš nízký selekční tlak často znamená neefektivní až náhodný výběr, příliš vysoký však znamená možnost uváznutí v lokálním maximu a stagnaci výpočtu. Optimální míra selekčního tlaku se pro každý problém liší a často je nutné pro její nalezení odzkoušet více variant.

O výběru jedinců se zpravidla rozhoduje na základě výsledků funkce fitness. Existují různé strategie výběru a neexistuje ideální strategie selekce pro všechny problémy. Tyto strategie jsou vyjmenovány a popsány v následujících odstavcích [15, 4, str. 36].

Rank

Selekce typu rank nejprve seřadí populaci dle fitness hodnot od nejlepšího jedince po nejhoršího. Jedinci jsou do mating pool následně voleni stochasticky, kdy se pravděpodobnost výběru jedince odvíjí od jeho pořadí v seřazené posloupnosti.

Fitness proportional selection

Fitness proporional selection je stochastický typ selekce. Často je také spojen s označením vážená ruleta. Jedinec je vybrán na základě pravděpodobnosti, která je dána poměrem fitness hodnoty jedince k součtu hodnot fitness všech jedinců populace. Tato metoda není výhodná pro použití, pokud mají jedinci malý rozdíl hodnot fitness. Pravděpodobnosti výběru jedinců se poté vzájemně blíží a situace připomíná náhodný výběr.

Turnaj

Turnajový výběr náhodně vybere dvě dvojice jedinců, v každé z nich poté porovná fitness hodnoty. Z dvojic se vybere vždy jedinec s nejvyšší hodnotou fitness. Podle další implementace algoritmu jsou buď dva nejlepší jedinci z turnaje ihned použiti pro křížení, či jsou přidáni do mating pool pro pozdější náhodný výběr jedinců a křížení. Je také možné využít více jedinců než dva pro počáteční porovnávání, čímž se zvýší selekční tlak.

Elitismus

Elitismus je strategie, která se obvykle využívá v kombinaci s jinou metodou selekce. Při zavedení elitismu je zajištěno zachování jedinců nejvyšší hodnotou fitness. Ti jsou beze změny zahrnuti i v další generaci. Tato strategie je výhodná, jelikož nejvyšší hodnota fitness následujících generací nikdy nebude menší než u generace předchozí. Zavedení elitismu ovšem přináší riziko uváznutí v lokálním maximu.

Křížení

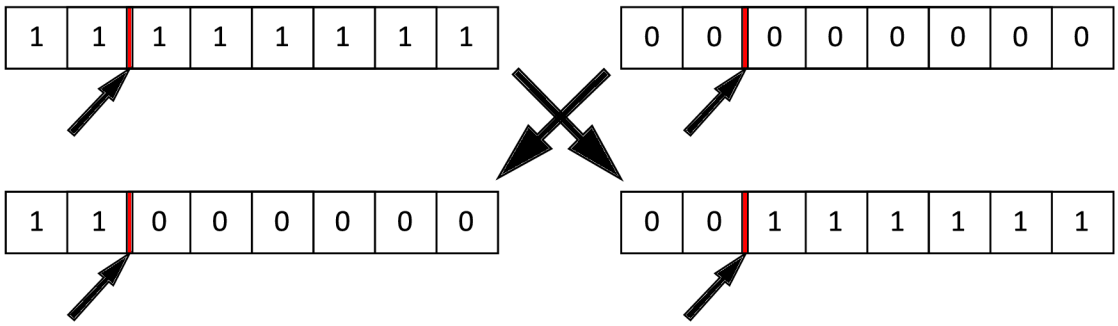
Operátor křížení (také známý jako rekombinace) simuluje reprodukci jedinců. Jeho vstupem jsou zpravidla[2] dva chromozomy jedinců náhodně vybrané z mating pool. Výstupem jsou poté 2 noví jedinci, kteří jsou kombinací chromozomů rodičů. Ti se následně v kanonické formě genetického algoritmu (viz Algoritmus 1) podrobují mutaci. Po provedení mutace probíhá postupné nahrazování stávajících jedinců v populaci. Operátor křížení není povinný. V případě, že není křížení prováděno, se označují jako noví jedinci kopie vybraných jedinců z mating pool bez žádné provedené změny.

[2]Byly prozkoumány i strategie křížení více než dvou rodičů, více lze přešíst v [6].

Výsledek operace záleží na použité strategii křížení. Zde je výčet některých populárích strategií, které se v genetických algoritmech objevují.

Jednobodové křížení

Jednobodové křížení (viz obrázek 2.1) se vykonává nad dvěma chromozomy ideálně v podobě řetězců. Spočívá ve dvou krocích. Nejprve je náhodně vybrána pozice v rámci délky chromozomu. Oba chromozomy, označeny jako rodiče, jsou následně v dané pozici rozděleny na 2 části. Křížení se dokončí spojením počáteční části prvního chromozomu s koncovou částí druhého, zbývající části jsou spojeny stejným způsobem. Výsledkem operace jsou dva noví jedinci, kteří obsahují vlastnosti obou rodičů.

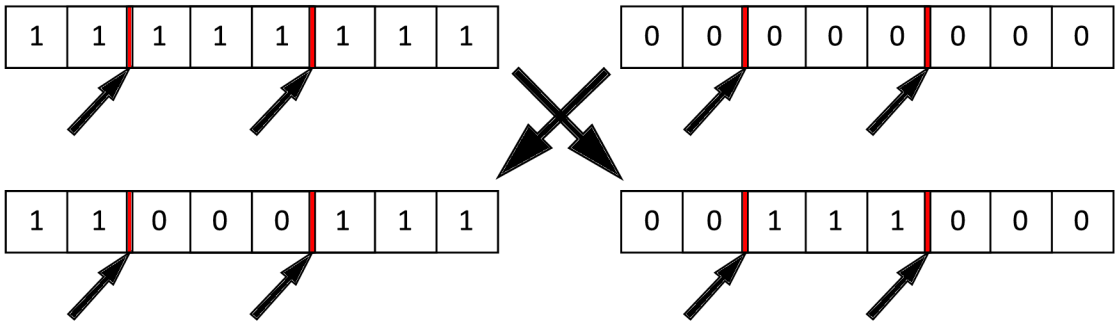


Obrázek 2.1: **Jednobodové křížení** – v náhodném bodě dochází k prohození částí chromozomů.

Jednobodové křížení je nejzákladnější strategií z populárních metod křížení [11]. Při výběru optimálního bodu dochází k propagaci nejlepších vlastností obou rodičů na jejich potomky, v opačném případě výběr špatného bodu dokáže poškodit souhrnnou kvalitu jednoho nebo obou rodičů. Tato strategie byla přesto prokázána jako efektivní pro řešení široké škály problémů a je dobrým startovním bodem při sestavování genetického algoritmu.

N-bodové křížení

Vícebodové křížení probíhá na podobném principu jako křížení jednobodové, je však vybíráno vícero bodů, ve kterých se chromozomy rodičů dělí. Spojování jedinců pak následuje podobná pravidla jako u jednobodového křížení.

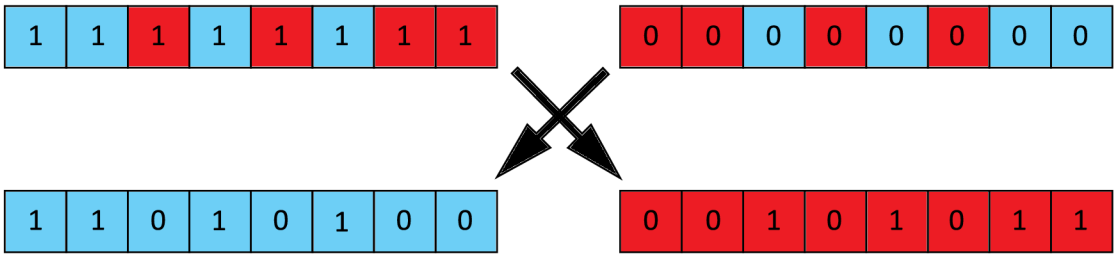


Obrázek 2.2: **2-bodové křížení** – ve dvou náhodných bodech dochází k prohození částí chromozomů.

Tato strategie může být užitečná v případě, kdy ponechání začátku a konce chromozomu jednoho z rodičů má pozitivní dopad na výkon, v mnoha jiných případech se však může projevit jako méně efektivní než jeho jednobodová varianta. Na obrázku 2.2 je demonstrována ukázka pro křížení s výběrem dvou bodů.

Uniformní křížení

V případě uniformního křížení nejsou vybírány žádné body ani nedochází k dělení chromozomů rodičů. Místo toho je šance na předání genů vyhodnocena pro každý gen zvlášť bez ohledu na pořadí a okolí genů v řetězci rodiče. V praxi tento operátor využívá předgenerovanou bitovou masku o délce počtu genů. Bit v masce odpovídá rozhodnutí o předání genu, každému genu odpovídá jeden bit. Při 1 na bitové pozici v masce přebírá první z potomků gen prvního rodiče, při 0 přebírá gen rodiče druhého. Druhý potomek pak získává geny obráceně. Při každém křížení dvou jedinců je generována nová bitová maska. Uniformní křížení je využíváno v případě, kdy není žádoucí do procesu křížení zahrnout důsledky okolí genů v podobě řetězce, nebo je náročné geny v podobě řetězce reprezentovat. Varianta uniformního křížení je zobrazena na obrázku 2.3.



Obrázek 2.3: **Uniformní křížení** – lze provádět pomocí bitové masky nebo za pomoci vyhodnocení náhodného přenosu s 50 % pravděpodobností pro každý gen.

Částečně mapované křížení

Částečně mapované křížení [9, 11] je nejčastěji využívanou formou křížení, má však definované omezující podmínky pro jeho využití. Je aplikováno na nebinární chromozomy, kdy záleží na neopakování se jednotlivých hodnot genů v rámci potomka. Jeho využití se prokázalo efektivní například při řešení problému obchodního cestujícího. Ten spočívá v nalezení co nejkratší možné cesty při projití všemi vrcholy ohodnoceného grafu, kdy ohodnocení hran značí délku cesty z vrcholu do vrcholu. Právě podmínka jediného průchodu každým z vrcholů činí z částečně mapovaného křížení ideální strategii pro řešení tohoto problému.

Operátor funguje následovně: Na začátku se náhodně vyberou dva body v rámci délky chromozomu. Vnitřní vyseknutá část, která vznikne dělením přes vybrané body, je následně přenesena na potomky, přičemž je dodrženo zachování pozic genů (částečné mapování). Dalším krokem je vyplnění prázdných pozic nových chromozomů. Postupně, pro každého potomka zvlášť, jsou ve směru zleva doprava dosazovány na prázdné pozice geny opačného rodiče.

Důležité je ovšem neporušit podmínku jedinečnosti hodnot genů v rámci chromozomu. Pokud nastane případ, kdy chromozom již obsahuje gen, který je právě dosazován, je nahlédnuto, jaká hodnota náleží na stejné pozici genu druhého potomka. Touto hodnotou je

poté nahrazena aktuální hodnota, kterou se snažíme dosadit. Proces je opakován, dokud není nalezena hodnota dosud neobsažená v daném chromozomu.

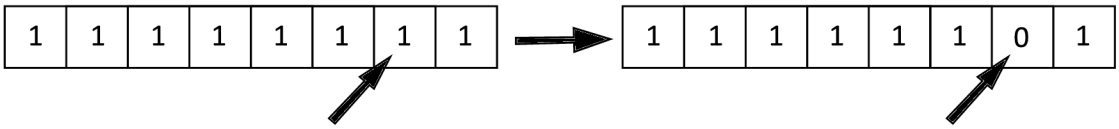
2.3 Mutace

Mutace je operátor genetického algoritmu, který nahrazuje hodnoty genů v chromozomu za náhodné hodnoty. Zpravidla je aplikován po operátoru křížení před vložením jedince do nové generace. Při vzniku genetických algortimů byla mutace zavedena k zabránění vzniku populace, která již není schopna dalšího vývoje. Mutace v tomto případě slouží jako nástroj pro překonání lokálních maxim. Tohoto efektu dociluje obohacením kombinačních schopností operátoru křížení o schopnost vyjmenovávání nových řešení pomocí menších změn v jedinci [8, str. 111]. Moderní genetické algoritmy však běžně využívají mutaci k zrychlení výpočtu, které může dosahovat až exponenciálních hodnot [5].

K mutaci je obvykle přiřazena míra, se kterou se operátor stochasticky aplikuje. Ta je obvykle malá a statická, vyhodnocuje se pro chromozom jako celek. Různé implementace operátoru mutace mohou mít definované další míry, kdy nejčastější z nich je pravděpodobnost mutace pro každý gen.

Jednobodová mutace

Nejzákladnějším typem mutace je tzv. jednobodová mutace. Ta v rámci chromozomu vybere právě jeden gen a jeho hodnotu změní na jinou náhodnou. V případě binárních chromozomů by se tato změna projevila v negaci bitu na náhodné pozici. Tato operace je znázorněna na obrázku 2.4.



Obrázek 2.4: Jednobodová mutace – oproti standardní vícebodové mutaci je vždy při aplikaci jednobodové mutace vybrána jedna náhodná pozice genu v chromozomu. Na obrázku je vyobrazena mutace binárního chromozomu, hodnota genu na vybrané pozici je invertována.

Standardní (vícebodová) mutace

Vícebodová mutace je nejčastěji využívanou implementací operátoru mutace. Využívá předem definovanou míru mutace, která se aplikuje pro rozhodování o mutaci každého z genů. Pomocí této míry se následně v cyklu vyhodnocují pravděpodobnosti aplikování mutace pro každý gen. Při pozitivním vyhodnocení mutuje gen na aktuální pozici. Obecným doporučením pro výběr míry mutace pro chromozom ve tvaru řetězce o délce n je hodnota $1/n$ [5], což v průměru odpovídá efektu jednobodové mutace, v praxi se ale často využívají i vyšší míry kvůli pozitivnímu dopadu na výkon.

2.4 Využití

Přestože využití genetických algoritmů se v průběhu let ustálilo především v oblasti řešení optimalizačních úloh, genetické algoritmy lze využít také v oblasti strojového učení, kde prokázaly svou efektivitu například ve schopnosti odhadovat nejefektivnější váhy neuronů [7].

Aplikace genetického algoritmu nad problémem vyžaduje definovat a naprogramovat následující podčásti:

- Definice chromozomu
- Funkce pro generování náhodných řešení
- Fitness funkce
- Operátor mutace
- Operátor křížení
- Operátor mutace
- Volba velikosti populace
- Ukončovací podmínka.

Knapsack problem – demonstrační řešení

Knapsack problem je jeden nejčastěji zmiňovaných problémů, na kterém se demonstrují optimalizační algoritmy. Je definován následovně: Existuje množina přemětů, každý z nich má svou váhu a hodnotu. Cílem úlohy je naplnit batoh, který unese pouze omezenou váhu co nejhodnotnějšími předměty.

- Chromozom se definuje jako binární řetězec o délce počtu předmětů. 1 na bitové pozici předmětu znamená, že předmět je zahrnut v batohu.
- Funkce pro generování nových řešení vytváří náhodné binární řetězce o délce počtu předmětů.
- Funkce fitness iteruje přes řetězec chromozomu. Spočítá sumu vah a hodnot zahrnutých předmětů. Pokud váha překoná maximální kapacitu batohu, výsledné ohodnocení jedince je 0.
- Selekční strategii lze zvolit libovolně z výčtů strategií v předcházejících sekcích. Naprosto validní volbou je například vážená ruleta nebo turnajový výběr, obě tyto strategie jsou posány v sekci 2.2.
- Jako operátor křížení je nejjednodušší zvolit jednobodovou nebo uniformní variantu. Jednobodová varianta je vhodná i u operátoru mutace. Lze také snížit váhu křížení a naopak zvýšit váhu mutace. Tato kombinace způsobí náhodné přidávání či odebírání předmětů z batohu v aktuální populaci.
- Ukončovací podmínka může být definována jako konečný počet generací, nebo ustálení deviace řešení v populaci s 95% podobností.

Tyto podsoučásti lze následně složit ve výslednou implementaci dle kanonické formy genetického algoritmu viz Algoritmus 1. Cílem této práce ovšem není knapsack problem řešit, pouze demonstruje úvahy, které vedou k sestavení řešení.

Využití v průmyslu

Nejsilnější stránkou genetických algoritmů je jejich univerzálnost. Ta se nejvíce projevuje na širokém spektru odvětví, kde se dá s jejich využitím setkat. Mezi častá odvětví, kde se s jejich výskytem setkáváme patří [1, str. 13]:

- Logistika – tvorba optimální trasy pro přepravu zboží. Liší se od problému obchodního cestujícího faktem, že přiřazujeme části grafu vícero kurýrům (osobám).
- Strojírenský návrh – funkce fitness je často vyhodnocována spouštěním simulace a zkoumá chování komponenty s parametry jednotlivých řešení.
- Ekonomika – optimalizace daňových politik, ceny výrobků produktů, simulace zákona nabídky a poptávky, ...
- Tvorba rozvrhů – od školních rozvrhů po rozvrhy směn či komerčních letů. Je definována sada omezení, genetický algoritmus se snaží najít řešení, které sadě co nejvíce vyhovuje.
- A mnoho dalších odvětví, ...

I přestože koncept využití evoluce v programování je světu známý již 50 let, v tomto oboru stále dochází k získávání nových poznatků a jeho konečné možnosti nejsou stanoveny. Vývoj genetických algoritmů leží v rukou budoucích průkopníků, a zůstává na nich, kam až evoluce ve světě informačních technologií pronikne.

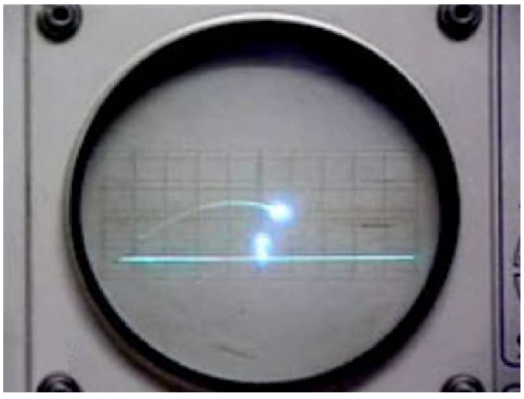
Kapitola 3

Hry pro více hráčů a serverové architektury

Herní průmysl je poměrně mladý obor, který vzniká po druhé světové válce. Od svého vzniku vzbuzuje vášeň jak ze strany tvůrců her, tak od hráčů, zároveň s sebou nese silně kompetitivní prostředí. Na platformě Steam, která slouží jako prodejní trh s počítačovými hrami, každým rokem přibyde více než 10 tisíc nových her. Na vývojáře je tedy kladen tlak vymýšlet neustálé inovace a vytvářet nové neotřelé hry, které zaujmou více než jejich konkurence. V této kapitole budou nejprve popsány počátky herního vývoje. Následně se otevře téma herních žánrů a více rozvedou technologie spojené s žánrem her pro více hráčů.

3.1 Historie herního vývoje

Videohry začaly vznikat v 50. letech 20. století na univerzitách za účelem demonstrace technologického pokroku. Jednou z prvních her, která vznikla s účelem pobavení hráče, byla hra **Tennis for two** (1958). Jejím autorem je William Higinbotham. Svou hru provozoval na plně analogovém počítači a byla zobrazována na osciloskopu Národní laboratoře Brookhaven [3].



Obrázek 3.1: Hra Tennis for two (1958) na osciloskopovém displeji.[2]

Obrázek 3.2: Hra Spacewar! (1962) zobrazena na vědeckém počítači PDP-1.[4]

[2]Obrázek převzat z https://www.imdb.com/title/tt0401821/
[4]Obrázek převzat z [3]

V roce 1962 Steve Russell vytváří hru **Spacewar!** [2]. Hra běžela na raném interaktivním mini počítači od DEC (Digital Equipment Corporation), který využíval displej typu katodové trubice a vstup přes klávesnici. Spacewar! je určena pro dva hráče, cílem hry je sestřelení protihráče využití gravitace hvězdy ve středu herního pole. Spacewar! je první hrou, která se veřejně rozšířila a zpopularizovala. Zdrojový kód hry byl šířen napříč studenty a institucemi, zanedlouho ji bylo možné najít na téměř všech výzkumných počítačích v USA. Hra ovšem autorovi nevykázala žádný výdělek. Hru *Tennis for two* lze vidět na obrázku 3.1, hra *Spacewar!* je vyobrazena na obrázku 3.2.

Arkádové hry

Inspirováni Spacewar! zakládají v roce 1972 Nolan Bushnell a Ted Dabney společnost *Atari studios*. Ze začátku se specializují na výrobu **arkádových herních automatů**. Každý automat obsahuje pouze jedinou hru, kterou si hráč může zahrát po vhození kovové mince. Jedna z prvních her, která byla představena na arkádových automatech, byla hra **Pong** (1972). Tato hra pro dva hráče spočívá v ovládání platformy, kterou se hráč snaží pozicovat pod dopadající míček, a tím ho odrazit na stranu soupeře. Hráč, kterému míček propadává mimo hrací pole, prohrává. Hra se dočkala neočekávaného úspěchu, kdy jeden z prvních automatů krátce po jeho spuštění zaznamenal poruchu z důvodu přeplnění kapacity boxu na čtvrtdolarové mince.

Herní konzole

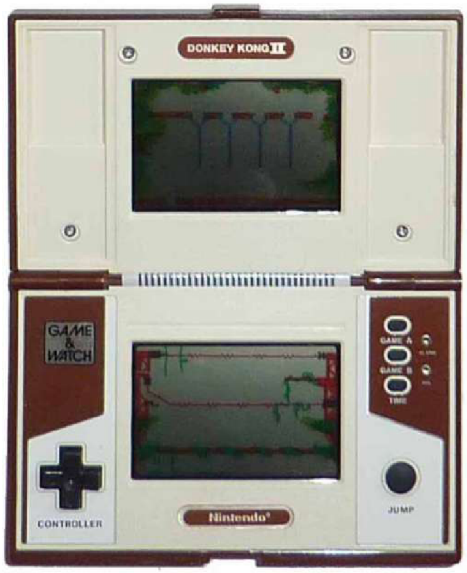
Kromě arkádových automatů se na počátku 70. let začínají objevovat první **herní konzole**. Jedná se většinou o dedikované systémy, které umí přehrávat fixní sadu her, oproti herním automatům se však pyšní svou dostupností a skladností.

První herní konzoli vytvořila americká společnost *Magnavox*. Svou konzoli nazvala **Magnavox Odyssey** (1972). Mimo jiné hry obsahovala tato konzole také hru **Table Tennis**. Jelikož Odyssey vyšla krátce před hrou Pong, Magnavox žaloval Atari za porušení patentových práv. Spor byl rozhřešen v mimosoudním vyrovnání, kdy Atari po zaplacení 700 tisíc dolarů získalo licence na patenty společnosti Magnavox [10, str. 47]. Tento spor byl mimořádně důležitý, jelikož ukázal důležitost patentů a duševního vlastnictví v nově vznikajícím kompetitivním průmyslu.

Další revoluce nastává vydáním herní konzole **Atari 2600**, která umožňuje načíst hru z videoherních kazet **cartridge**. Herní konzole nyní nejsou limitovány sadou her při vydání, herní společnosti a nezávislí vývojáři také mohou vydávat své hry bez nutnosti distribuce herního hardwaru. Nejhranější hrou na Atari 2060 je hra **Space Invaders** (1978), která do dnešního dne (po připočtení inflace) zůstává nejvýdělejčnější videohrou na světě.

Přenosné herní konzole

Do hry konzolových gigantů se postupně přidávají dnes velké herní společnosti jako *Nintendo* nebo *Sega*. Nintendo v roce 1980 vydává konzoli **Game & Watch** (viz obrázek 3.3), která je jednou z prvních handheld herních konzolí. Její návrhář, Gunpei Yokoi, inspirován cestujícím ve vlaku, který si pro zkrácení dlouhé chvíle hrál se svou kalkulačkou, přenáší svou vizi do reality a posouvá přenositelnost herních konzolí na novou úroveň.



Obrázek 3.3: Herní konzole **Game & Watch**[6] je typická pro dvě LCD obrazovky a *handheld* design. Tento styl lze spatřit i u pozdějších konzolí společnosti Nintendo (například konzole *Nintendo DS*). Nejpopulárnější hry uvedené na Game & Watch jsou Donkey Kong, The Legend of Zelda a Mario Bros. LCD displej ovšem limituje konzoli na fixní animace, konzole se tedy prodávají převážně jako dedikovaný systém pro jedinou hru.

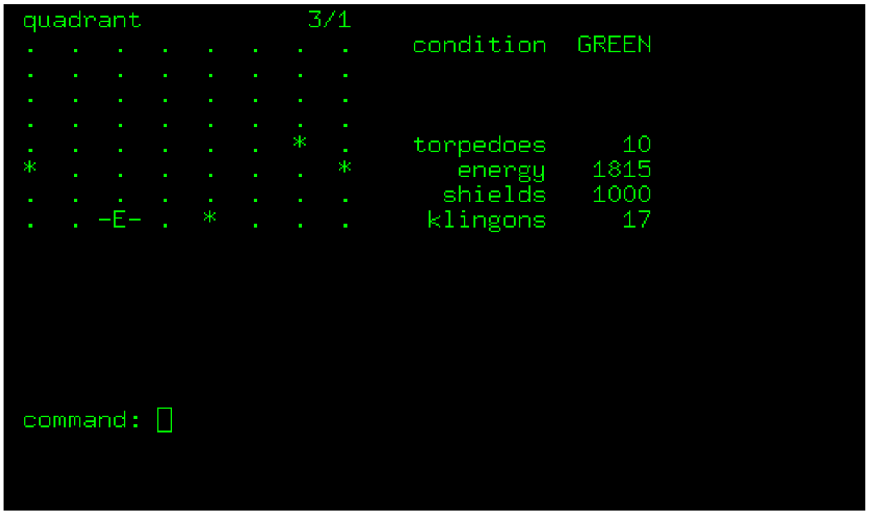
Obrázek 3.4: Game Boy[8] obsahuje minimalistické ovládání s tlačítkami umožňující pohyb do 4 směrů, dvě akční tlačítka *A* a *B* a dvě speciální tlačítka *SELECT* a *START* často určené pro přístup k akcím herního menu. Displej, z prvu černobílý, byl později obohacen 15-bitovou RGB barevnou paletou v novém vydání konzole s názvem *Game Boy Color*.

[6]Obrázek převzat z https://en.wikipedia.org/wiki/Game_%26_Watch
[8]Obrázek převzat z https://en.wikipedia.org/wiki/Game_Boy#/media/File:Game-Boy-FL.jpg

V roce 1989 vydává Nintendo konzoli **Game Boy** (viz obrázek 3.4), která představuje kombinaci cartridge systému s handheld designem. Konzole je vybavena stereo výstupem pro sluchátka a také možností propojení dvou konzolí přes fyzický kabel čímž umožňuje zážitek hry pro více hráčů. Nejznámější hry představené na Gameboy jsou Tetris, nebo Pokémon (tato hra bude podrobně přestavena v popisku obrázku 3.8).

Počítačové hry

První počítačové hry byly vytvářeny na výpočetních počítačích vědeckých institucí a vysokých škol. Jejich výpočetní čas byl však příliš drahý, to se ovšem změnilo příchodem PC (Personal computer), jejichž účelem je osobní využití v kancelářích a domácnostech. Přestože cena osobních počítačů byla na počátku výroby vysoká, počítačové hry vznikaly zároveň s herními konzolemi již v 70. letech 20. století. Většinou se jednalo o textové hry inspirované slovní hrou Dungeons & Dragons nebo deskovými hrami. Na obrázku 3.5 lze vidět textovou strategii *Star trek*. Počítačové hry se více zpopularizovaly v 80. letech s příchodem počítačů od firmy *IBM* a také vydáním počítače *Macintosh* v roce 1984 společností *Apple*.



Obrázek 3.5: **Star trek (1971)**[10] je jednou z prvních textových her, které se objevily na stolních počítačích. Jedná se o strategickou hru, jejíž cílem je zničení nepřátelských klingonských lodí. Hráč ovládá hru pomocí psaní do příkazového pole.

Současnost

V rámci této práce není dostatek prostoru pro kompletní shrnutí historie herního vývoje, ta je podrobněji popsána například v knize *Ultimate history of videogames* [10], ze které tato práce čerpá.

V současnosti (rok 2024) koexistují proudy konzolových a počítačových her vedle sebe. Konzolovými giganty dnešní doby jsou společnosti *Microsoft* se sérií konzolí *Xbox*, *Sony* se sérií *Playstation* a *Nintendo*, které láká svou přenosnou konzolí *Switch* (viz obrázek 3.6).

[10]Obrázek převzat z https://en.wikipedia.org/wiki/Star_Trek_%281971_video_game%29#/media/File:Star_Trek_text_game.png

Obrázek 3.6: **Současná generace konzolí**, zleva Playstation 5, Xbox Series X a Nintendo Switch.

Za zmínku stojí i hraní her na mobilních telefonech, které je možné označit za nejpopulárnější variantu handheld hraní.

Důležitým aspektem moderních her je využití internetu, čímž vzniká velké množství nových herních žánrů. Moderní herní hardware běžně zvládá přehrávat více než 60 snímků za vteřinu, v oblasti grafiky dochází ke snaze v aplikování moderních technologií, jako je například *ray tracing* nebo *motion capture*, které ještě více přibližují herní světy realitě.

Herní průmysl od svého vzniku čím dál více vstupuje do běžného života moderního člověka. Popularitu her dokazuje například existence tzv. *E-sportů*, kdy se ve hrách kompetitivního charakteru pořádají turnaje s gigantickými výherními částkami. Vznikají tak profesionální týmy a organizace, které se herních turnajů účastní.

Moderní herní vývoj

O vývoj her samotných se starají velké společnosti, ale i malé herní studia vytvářející **Indie hry**. Tento pojem je zkrácené pojmenování pro *Independently-developed game*, kdy je hra vyvýjena často s nízkým rozpočtem menším počtem vývojářů. Tuto skutečnost se hra snaží kompenzovat svou originalitou a jedinečnými herními mechanikami. Velká studia oproti tomu prezentují tzn. **AAA** hry, které běžně dosahují multimilionových rozpočtů. Tyto hry spolu často nesou několikahodinovou hratelnost, zároveň jsou kladeny větší nároky na grafickou nebo zvukovou stránku hry.

Herní engine

Samotný vývoj probíhá v tzv. **herních enginech**. Využití enginu násobně usnadňuje vývoj díky předem předprogramované sadě funkcí prakticky využitelných pro herní vývoj. Engine poskytuje grafické prostředí, fyziku, ale také cenné nástroje sloužící například pro správu herních animací, či zajištění vícejazyčnosti v textu a mluveném slově. Moderní enginy zároveň nabízí platformově nezávislý vývoj, který v praxi šetří obrovské sumy času i peněz při přenosu her na novou generaci konzolí či zcela jinou platformu.

Mezi nejznámější moderní enginy patří **Unreal engine**, **Unity** nebo **Godot**. Výběr enginu předurčuje dobu a složitost vývojového procesu. I proto herní společnosti pro své hry i dnes často vyvíjejí enginy na míru.

Herní engine Unity

Unity[11] je populární herní engine umožňující rychlý platformově nezávislý vývoj her. Poskytuje grafické prostředí pro herní světy a nabízí širokou škálu nástrojů usnadňující herní vývoj. Mezi ně patří například nástroje pro tvorbu agentů umělé inteligence nebo nástroje pro tvorbu uživatelských rozhraní.

Hra přepíná mezi **scénami**, které se skládají z množiny **herních objektů**. Ty následně obsahují množinu **komponent**, které udávají jejich umístění v prostoru, vykreslování a chování. Zdrojové kódy jsou nazývany jako **skripty** a jsou psány v jazyce **C#**. Skript se chová jako komponenta a lze ho připojit k hernímu objektu. Unity následně skripty vyhodnocuje za běhu programu, čímž dochází k ovlivňování herního prostředí. Vývojové prostředí enginu Unity lze vidět na obrázku 3.7.

Obrázek 3.7: Prostředí herního enginu **Unity**[12].

3.2 Úvod do herních žánrů

Dosud byla probrána spíše technologická stránka posunu herního průmyslu. Hry jsou totiž přímo ovlivněny časem jejich vzniku a dostupnými technologiemi, ať už ve způsobu ovládání nebo v jejich velikosti, která je limitována hardwarem. Například konzole **Game boy** (viz obrázek 3.4) umožňovala přehrávat pouze 4 zvukové kanály tónů s omezenou bitovou šířkou, což ovlivnilo hudební skladatele, kteří do her pro tuto platformu vytvářeli herní hudbu. S technologickým pokrokem v průběhu let počet omezení klesá a naopak roste svoboda, které mohou vývojáři využít při tvorbě svých her. Dochází k postupnému vymezování herních žánrů, do kterých lze jednotlivé hry zařadit. Příští odstavce slouží pro přiblížení některých populárních herních žánrů a prezentaci rozličnosti světa videoher.

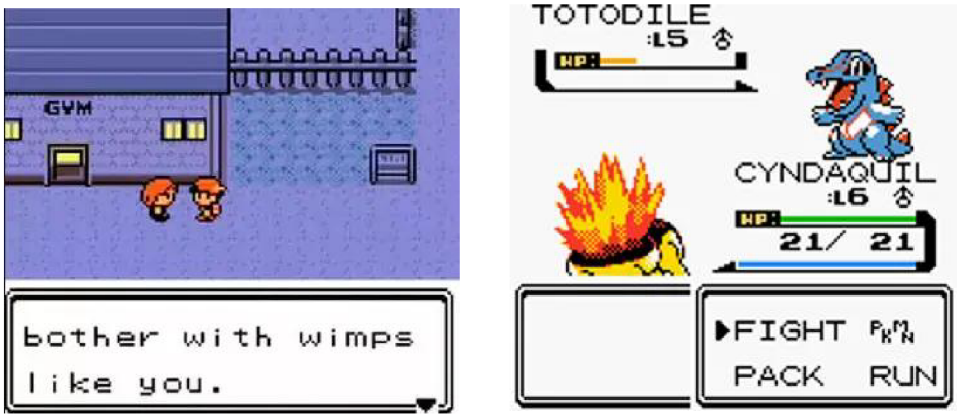
[11] https://unity.com

[12] Obrázek převzat z https://www.megavoxels.com/learn/what-is-the-unity-game-engine/

Strategické hry

Strategické hry jsou široký herní žánr, který se dělí na velké množství podkategorií. Ať už se jedná o tahové strategie, karetní hry nebo hry žánru *RTS (Realtime strategy)*, tyto hry spojuje klíčový prvek systematického a logického rozhodování s cílem přechytračení soupeře a následné výhry. Napětí se oproti akčním hrám ukrývá v nejistotě.

Odlehčenější strategie mohou být například strategie budování města (například hra Cities Skylines) nebo karetní hry (např. Hearthstone). Z tahových strategií je možné zmínit sérii her Pokémon (viz obrázek 3.8). Z žánru RTS lze zmínit například hru Stronghold Crusader (viz obrázek 3.9).



Obrázek 3.8: **Pokémon Gold** (1999), původně vydáno na konzoli Game Boy (viz obrázek 3.4), je kombinací tahové strategie a RPG. Hráč se stává trenérem příšerek nazývaných jako Pokémoni, které sbírá a trénuje v soubojích. Hlavním cílem hry je porazit šampióna regionu a stát se nejlepším trenérem Pokémonů. Souboje jsou provedeny v tahovém systému, kdy se hráči vzájemně střídají v útoku.

Obrázek 3.9: **Stronghold Crusader** (2002)[14]patří mezi strategie v reálném čase (RTS). V tomto žánru čas běží přirozeně a hráč s ním musí pracovat tak, aby stihnul co nejvíce akcí za co nejkratší čas. RTS jsou často hrány kompetitivně *online* vícero hráči.

Simulátory

Simulátory zakládají na věrnosti a reálnosti. Jejich využití přesahuje herní svět, využívají se například při profesním trénování nebo pro účely vzdělávání. Mohou být ale určeny pouze pro zábavu. Populárními simulátory jsou například Microsoft Flight Simulator (viz obrázek 3.10), Euro Truck Simulator nebo Farming Simulator.

Obrázek 3.10: **Microsoft Flight Simulator (2020)**[16] využívá satelitní snímky, díky kterým vytváří 3D modely krajiny celého světa napříč různými obdobími. Zprostředkovává realistické výpočty aerodynamiky a fyziky v různých podmínkách. Obsahuje také výukový režim, který vysvětluje základy létání nováčkům.

RPG

Role-playing games zkráceně RPG představují herní žánr stavěný na živém vtahujícím světě a budování postavy. Hráč by měl mít možnost vžít se do role hlavního hrdiny a mít pocit, že akce ve světě provádí právě on. Rozsáhlé RPG hry bývají většinou produktem AAA studií, příkladem může být hra Final Fantasy zobrazená na obrázku 3.11.

RPG hry jsou často propojovány s online hratelností. Do extrému tento prvek vyhání tzv. **MMORPG** (*Massive multiplayer online RPG*) hry, které si zakládají na živém světě plném reálných hráčů. Mezi populární MMORPG patří například hra World of Warcraft.

[14] Obrázek převzat z https://games.tiscali.cz/preview/krizaci-a-stronghold-crusader-17964

[16] Obrázek převzat z https://www.cnet.com/tech/gaming/microsoft-flight-simulator-takes-flight-on-xbox-game-pass-tuesday/

Obrázek 3.11: **Final Fantasy VII (1997)**[18] bývá označováno za první AAA titul, který kdy vznikl, a představuje klasiku z žánru RPG. Hra je zasazena do fantasy světa, který je na pokraji ekologického zhroucení. Hlavní postava a parta spolubojovníků se vydávají na cestu za její záchranou.

Další žánry

Existuje spoustu dalších žánrů, jako jsou například adventury, závodní hry, sportovní hry, rytmické hry, hororové hry, nebo rozsáhlý žánr akčních her. V praxi je běžné, že hra obsahuje prvky více žánrů. Tento jev je žádoucí, jelikož hra má vyšší potenciál zaujmout širší publikum, je ovšem důležité, aby žánrové provázání dávalo smysl.

3.3 Hry pro více hráčů

Multiplayerové hry, neboli hry pro více hráčů, bývají považovány za vlastní herní žánr. K hernímu návrhu je totiž od začátku nutné přistupovat odlišně. V průběhu procesu programování může být dodatečné zakomponování multiplayeru obtížné a časově náročné. Hry pro více hráčů lze podle formy implementace rozdělit na **lokální** a **online** multiplayer. Další dělení tohoto žánru je na hry **kooperativní** a **kompetitivní** podle jejich cíle a principu.

[18] Obrázek převzat z https://www.idisplayit.co.uk/news/film-and-tv/final-fantasy-vii-rebirth-launch-preview/

Split-screen

Jednou z forem provedení multiplayeru je **splitscreen**. Jedná se o lokální formu multiplayeru, kdy hra rozdělí svoji obrazovku na více částí, přičemž každá část náleží jednomu hráči. Problémem splitscreenu bývá v praxi příliš malá obrazovka nebo nízké rozlišení obrazu. Hra v režimu splitscreen je vyobrazena na obrázku 3.12.

Obrázek 3.12: **Mario Kart 8 Deluxe (2017)**[19] v režimu splitscreen pro 4 hráče.

Peer to peer

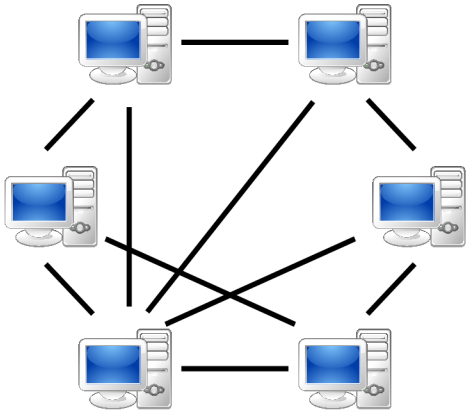
Peer to peer je decentralizovaná forma online připojení. Hráči se na sebe připojují přímo bez přičinění třetí strany. Data (v podobě *packetů*) se posílají na IP adresu počítače hráče, kterému jsou určena. Tato varianta hry pro více hráčů se využívá zejména z důvodu odstranění nákladů za dedikované servery.

Existuje více typů implementace peer to peer. V prvním typu je jeden z hráčů označen jako *host*, ostatní hráči se k němu následně chovají jako k serveru. Packety ostatních hráčů jsou směrovány na hosta, ten jim naopak zasílá aktualizace herního stavu. Tato varianta je závislá na rychlosti připojení a výpočetním výkonu stroje hostujícího hráče.

Druhou variantou je připojení, kdy mezi hráči neexistuje host ani server. Vzniká chaotický systém, ve kterém jsou packety zasílány napříč všemi zařízeními. Tento typ komunikace se využívá například pro přenos hlasových dat a je vyobrazen na obrázku 3.13.

Nevýhodou peer to peer připojení je zabezpečení a autentizace. Ze síťového hlediska může nastat interference s ochranou *firewall* nebo *NAT* a hráči mohou mít problémy s navázáním připojení. Tento problém je ovšem ve většině případech řešitelný. Druhým problémem je autentizace a ochrana před podváděním. Bez přítomnosti třetí strany nexistuje jednoduchý způsob, který by ověřil platnost každé akce.

[19] Obrázek převzat z https://www.theverge.com/2017/5/3/15506778/mario-kart-8-deluxe-split-screen-local-multiplayer-joy-con

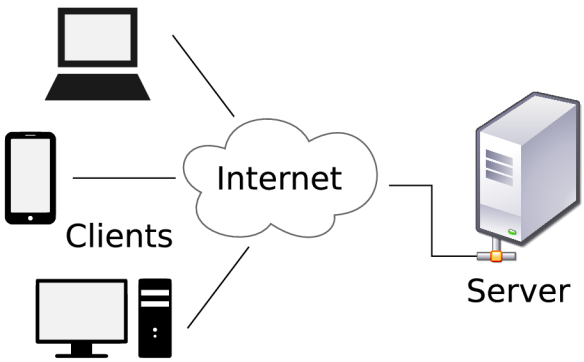


Obrázek 3.13: Architektura **peer to peer**[21]. Zařízení si mezi sebou posílají data bez centrálního bodu komunikace.

Dedikované servery

Dedikované servery zajišťují centralizovanou formu online hraní. Jsou zpravidla provozovány přímo provozovateli her a umožňují připojení velkého množství hráčů. Jejich využití je časté zejména u kompetitivních her, kdy je důležitá stabilita připojení a ochrana proti podvádění. Hra na zařízení hráče slouží jako klient, který navazuje připojení se serverem, a následně reaguje na data, která mu jsou od serveru zasílána. Server nemusí být schopen plného grafického zobrazování na své straně, u videoher s důležitostí fyziky a kolizí však často provozuje graficky méně náročnou verzi prostředí, například bez světla a textur. Klienti následně svůj herní stav synchronizují dle stavu serveru. Architektura klient-server je vizualizována na obrázku 3.14.

Méně výpočetně náročnou implementací serveru jsou servery v podobě aplikačního rozhraní, které fungují jako logické aplikace bez grafického zobrazování na straně serveru. Server tohoto typu implementuje také hra PlantEVO, která je výsledkem této práce.



Obrázek 3.14: Architektura **klient-server**[23]. Zařízení se dorozumívají přes centrální bod v podobě serveru.

[21] Obrázek převzat z https://cs.m.wikipedia.org/wiki/Soubor:P2P-network.svg
[23] Obrázek převzat z https://en.m.wikipedia.org/wiki/File:Client-server-model.svg

Využití dedikovaných serverů s sebou nese možnosti bezpečné autentizace a stabilního připojení. Od jejich využití odrazuje cena služeb poskytovatele produkčního prostředí, které mohou dosahovat až desetitisíců korun za měsíc.

3.4 Serverové architektury

Tvorba serverů vyžaduje rozsáhlou škálu znalostí jako je porozumění síťovým protokolům a technikám přenosu dat. Neméně důležité je vědět, jak data na serveru správně uchovávat zabránit jejich poškození či ztrátě. Tato sekce se věnuje teoretickým znalostem potřebných v rámci práce, avšak neklade si za cíl být celkovým přehledem aktuálního stavu poznání.

Databáze

Databázový systém [14] je kolekce programů, které slouží k ukládání a správě dat. Zároveň umožňuje efektivní získávání uložených dat v případě nutnosti jejich použití. Dle pravidel danými konkrétním databázovým modelem definujeme struktury, které umožňují ukládat objekty. Databázových modelů existuje více, mezi nejznámější patří hierarchický, síťový a relační databázový model.

Relační databázový model je v dnešní době nejpoužívanějším modelem. Umožňuje poměrně snadné definice databázových objektů, které ukládá do tzv. tabulek. Mezi jednotlivými tabulkami lze definovat vazby, které umožňují provádět operace nad širším kontextem dat.

Operace v relačním databázovém modelu se řídí jazykem **SQL** (*Structured Query Language*), který poskytuje nástroje pro aktualizaci databázového schématu a datovou manipulaci. Klíčovou funkcionalitou SQL je také dotazování, které umožňuje efektivní získávání dat z databáze. Dotazy mohou být specifikovány s různými podmínkami, to umožňuje cílit na přesně definovanou množinu dat relevantních pro konkrétní situaci. SQL dále umožňuje správu přístupových práv různých uživatelských skupin.

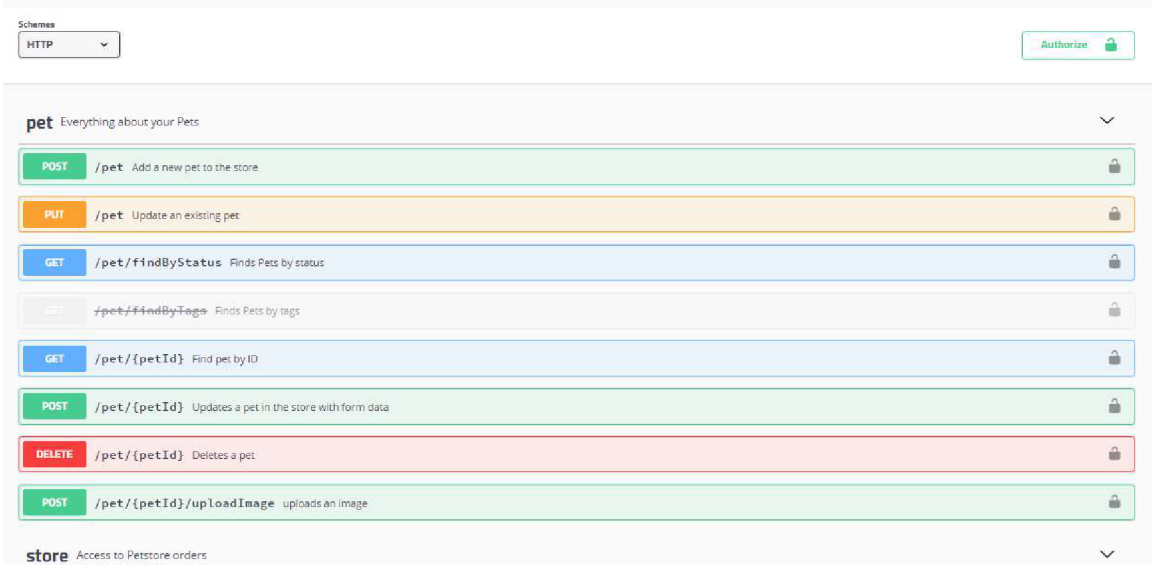
REST API

REST API (*Representational State Transfer Application Programming Interface*) [12] je nástroj k definici komunikačního rozhraní serveru. Vyznačuje se *code-on-demand* charakteristikou, kdy komunikaci začíná klient a server na požádání odpovídá. Další z klíčových charakteristických vlastností je bezstavovost, kdy komunikace mezi serverem a klientem zaniká dokončením požadavku. Klient tedy vždy posílá všechny kontextové informace potřebné k provedení dotazu, jako je například identita uživatele.

REST API plně využívá protokol **HTTP 1.1** (*HyperText Transfer Protocol*). Své rozhraní definuje pomocí HTTP metod a odpovídá v souladu se standardními návratovými kódy protokolu. Pro vizualizaci a dokumentaci REST API slouží nástroj Swagger UI, který lze vidět na obrázku 3.15.

Protokol HTTP

Protokol HTTP je základní protokol sloužící k přenosu hypertextového obsahu po internetu. Jeho nejrozšířenějším využitím je přenos webových stránek, kdy webový prohlížeč slouží jako nástroj k odesílání HTTP dotazů. Ty se dělí na několik typů dle použité metody protokolu. Nejzákladnější metodou je dotaz GET, který slouží k získání dat ze serveru. Dalšími metodami jsou například metody POST, PUT nebo DELETE, které manipulují s daty



Obrázek 3.15: **Swagger UI**[25]je nástroj sloužící k dokumentaci REST API. Na obrázku lze vidět sadu metod, které dané aplikační rozhraní implementuje. Ke každé metodě API je zároveň přiřazena metoda HTTP a cesta sloužící k zavolání metody.

na serveru. Konkrétní chování serveru není metodou definované a je čistě implementačně závislé. Je ovšem vhodné se držet základních konvencí, ke kterým metody nabádají.

Společně s dotazem lze zasílat také metadata ve formě hlaviček. Hlavičky (Headers) přímo ovliňují chování HTTP dotazu, obsahují například typ obsahu, maximální dobu dotazu, informace o zařízení nebo údaje umožňující autentizaci uživatele.

HTTP protokol (mimo jiné) definuje návratové kódy odpovědí, jejichž podrobný výčet je možné nalézt v [13]. Kódy jsou trojciferná čísla, přičemž první číslice udává obecný typ odpovědi, zbývající 2 číslice poté identifikují konkrétní odpověď. Obecné typy návratových kódů jsou definovány následovně:

- 1xx – informace
- 2xx – úspěch
- 3xx – přesměrování
- 4xx – chyba klienta
- 5xx – chyba serveru.

[25]Zdroj obrázku: https://swagger.io/tools/swagger-ui/

Kapitola 4

Návrh strategie PlantEVO

V předcházejících kapitolách byly představeny oblasti genetických algoritmů a herního vývoje. Cílem této práce je vytvořit hru, ve které genetický algoritmus představuje klíčový herní prvek. Výsledkem práce je hra PlantEVO, která spočívá v křížení speciálních bojových rostlin se snahou získat co nejsilnější sbírku rostlin a porazit ostatní hráče.

4.1 Žánr a motiv

PlantEVO je **online kompetitivní tahová strategie**. Svou hratelností míří spíše na platformu mobilních telefonů. Jejím cílem není vtáhnout hráče na několik hodin. Naopak se snaží umožnit krátké herní úseky, které lze několikrát denně opakovat. Hráč plynule střídá 2 role. V první roli prozkoumává svou **sbírku rostlin**, tu se dále snaží co nejlépe šlechtit a získávat nové silnější rostliny. Ve druhé roli **bojuje proti ostatním hráčům**, při výhře v souboji získává herní měnu potřebnou ke šlechtění. Soubojový systém se inspiruje u jiných tahových strategiích, příkladem může být hra **Pokémon**, ta je vyobrazena na obrázku 3.8.

Motiv bojujících rostlin byl v herním průmyslu již zpracován například ve frančíze **Plants vs. Zombies**. PlantEVO je ovšem speciální pro svou genetickou stránku. Dalším unikátním prvkem je **variabilita vizuálů** jednotlivých rostlin, kdy rodiče předávají svůj vizuál na potomky.

4.2 Herní mechaniky

Každá hra obsahuje svou sadu *herních mechanik*, které ji definují. Pokud nejsou zábavné jednotlivé mechaniky, není zábavná ani hra jako celek. PlantEVO obsahuje hned několik herních mechanik, které si vyžadují podrobný popis. Tato sekce slouží k jejich vyjmenování, tím také plánuje docílit přiblížení hry jako celku. Následující definice se týkají abstraktního návrhu a neberou v úvahu výslednou implementaci či zpětnou reflexi jejich návrhu po testování.

Stavba rostliny

Rostlina je definována pomocí **částí** a **atributů**. Již bylo zmíněno, že důležitým prvkem hry je variabilita vizuálů jednotlivých rostlin. Toho PlantEVO dociluje pomocí rozdělení rostliny na menší elementární části. Jedná se o **stonek**, **listy**, **květy** a **speciální detaily** (například trny). Části, ze kterých se rostlina skládá, přímo ovlivňují její schopnosti v soubojích nebo sílu útoků.

Sílu rostliny v soubojích dále ovlivňují speciální atributy. **Atribut** je vlastnost rostliny, která má přiřazenou hodnotu ve formě celého kladného čísla. Hráč se snaží u svých rostlin docílit co nejvyšších hodnot. Všechny atributy jsou vyjmenovány a stručně popsány v tabulce 4.1.

Název	**Definice**
Zdraví (Health)	určuje počet životů rostliny
Síla (Power)	zvyšuje poškození útoků rostliny
Přesnost (Precision)	zvyšuje šanci na kritický zásah
Ohebnost (Flex)	zvyšuje šanci na vyhnutí vůči nepřátelskému útoku
Čerpání slunce (Sundrain)	zvyšuje množství nasbíraného slunce za tah
Čerpání vody (Waterdrain)	zvyšuje množství nasbírané vody za tah
Životnost (Lifespan)	určuje délku života rostliny

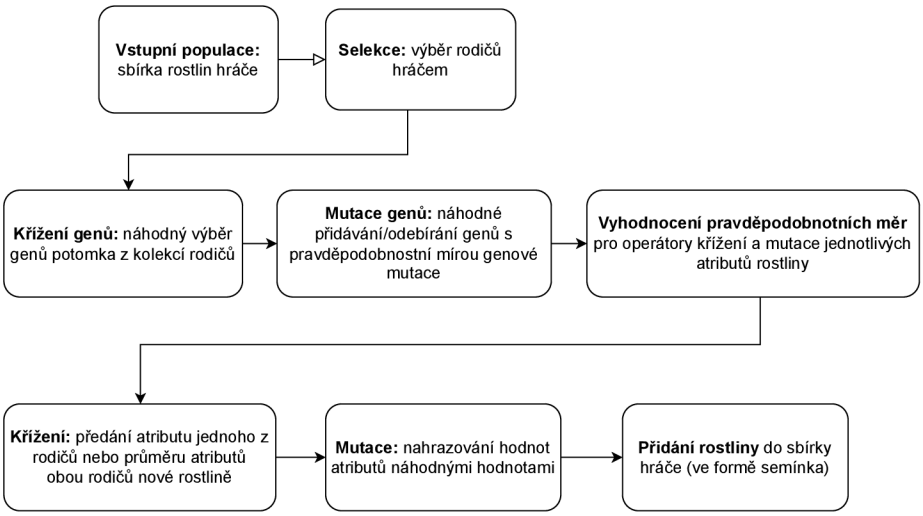
Tabulka 4.1: Výčet a definice atributů rostlin.

Části a atributy definují sílu rostliny. Velkou roli hrají také při šlechtění, kdy se stávají hlavním předmětem genetického algoritmu.

Speciální je atribut *Životnost*, který jako jediný nemá vliv na průběh soubojů. Hra dále v textu představuje mechaniku **hynutí rostlin**, atribut životnosti určuje délku života rostliny. Tato herní mechanice je více popsána v podsekci 4.2.

Genetický algoritmus

Klíčovou herní mechanikou v PlantEVO je genetický algoritmus. Ten zde neslouží k optimalizačním účelům, ale k **simulaci reprodukce**. Je přesto možné jeho chod přirovnat ke **kanonické formě** popsané v algoritmu 1. Na obrázku 4.1 je znázorněn vývojový diagram genetického algoritmu využívaného ve hře. Po něm následuje textový popis jeho funkcionality a definice klíčových prvků. Textový popis záměrně přeskakuje sekci křížení a mutaci genů a následné vyhodnocení pravděpodobnostních měr. Geny jsou kompletně nový koncept, který je představen později v textu a dopodrobna rozebrán v implementační části práce,



Obrázek 4.1: Vývojový diagram genetického algoritmu hry PlantEVO.

Populace jedinců je definována sbírkou rostlin hráče. Tuto populaci je na počátku nutné inicializovat. Zajímavým přístupem by byla počáteční stavba rostlin, kdy by hráč mohl sám rozhodnout, s jakou počáteční populací začne. Pro účely prototypu stačí začít náhodným vygenerování 4 rostlin, kde 2 budou **mužského** a 2 **ženského** pohlaví.

Dále následuje operátor **selekce**. Ten zprostředkovává hráč, který musí vyhodnotit **fitness** svých rostlin a rozhodnout, které rostliny zvolí pro šlechtění. Pro úspěch této mechaniky je nutné interaktivní uživatelské rozhraní, které přehledně a jasně sdělí hráči atributy jednotlivých rostlin a umožní porovnání mezi jednotlivci.

Jakmile hráč zvolí rostliny, nastává fáze **křížení**. Ta probíhá pomocí uniformní strategie (viz sekce 2.2), kdy se postupně iteruje přes všechny atributy rostliny a následně vybírá atribut jednoho z rodičů. U číselných atributů zde vstupuje do hry také operátor **průměru**, který nové rostlině předá průměrnou hodnotu atributů obou rodičů.

Následuje operátor **mutace**, který nejprve probíhá ve své standardní variantě (viz sekce 2.3). Mutované části rostlin jsou nahrazeny za jiné náhodné, číselné atributy jsou vygenerovány náhodně v rozptylu od střední hodnoty všech atributů nové rostliny. Poté ovšem následuje deviace od standardní implementace operátoru mutace v podobě **inkrementů a dekrementů** jednotlivých hodnot atributů potomka. Po aplikaci operátoru se některé číselné atributy mírně zvýší, jiné naopak mírně klesnou. Důvod zavedení tohoto kroku může být prozatím nejasný, hráč ovšem dokáže za pomocí speciálních genů ovlivnit inkrementaci nebo dekrementaci jednotlivých atributů. Funkčnost a princip genů bude objasněn v následující podsekci.

Po dokončení operátoru mutace je rostlina přidána do kolekce hráče ve formě **semínka**, které potřebuje čas na růst. Obměna generací probíhá přirozeně, díky postupnému hynutí hráčových rostlin.

Základní požadavky na genetický algoritmus jsou **objektivita** a částečná **předpovídatelnost**. V opačném případě hráč nebude nad strategickým šlechtěním rostlin vynakládat žádný čas, bude mu totiž připadat, že výsledné rostliny získává naprosto náhodně.

Geny

Aby byl proces křížení zajímavější a ovlivnitelnější, zavádí PlantEVO do procesu šlechtění tzv. **geny**. Gen nenese v terminologii PlantEVO stejný význam jako v terminologii genetických algoritmů, místo toho se jedná o speciální atribut rostliny, který ovlivňuje průběh genetického algoritmu. Toho dociluje pomocí manipulace s vahami genetických operátorů. Rostlina může vlastnit až 9 různých genů, ty získává za pomoci křížení od svých rodičů nebo díky genové mutaci. Příklad genu a jeho efektu je zobrazen na obrázku 4.2.

Obrázek 4.2: **Mutagen** – zvyšuje šanci na mutaci atributu nebo části. Vazba genu na atribut nebo část je znázorněna v pravém dolním rohu ikony genu. Zobrazené mutageny se aplikují z leva na atribut *zdraví*, atribut *čerpání slunce* a části *květu* rostliny.

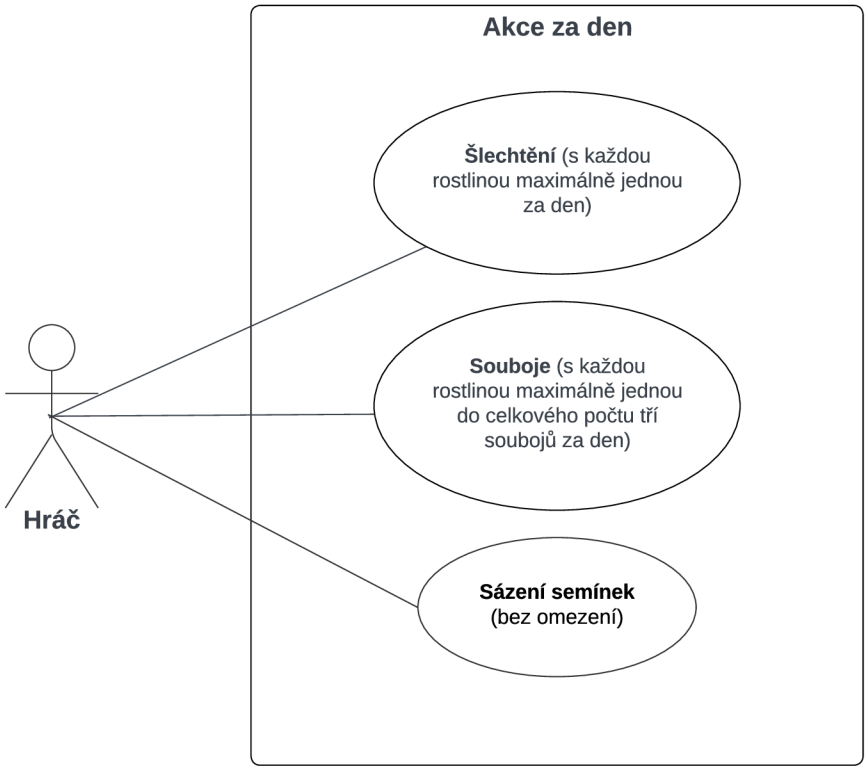
Denní cyklus

Důležitou mechanikou hry je také počítání herních dnů. Některé, zejména mobilní, hry integrují do svých herních mechanik čas skutečného světa, kdy například probíhá simulace stavby budovy odpočtem 6 hodin od začátku vykonávání akce. Dny ve hře PlantEVO ovšem nemají se skutečným časem nic společného a jsou čistě virtuální. Hra se tímto stylem snaží být méně návyková.

Každý den může hráč vykonat omezený počet akcí. Může šlechtit své rostliny, ovšem každou rostlinu pouze jednou za herní den. Dále může s každou rostlinou jednou za den bojovat, ovšem maximálně do provedení 3 soubojů za den. Poté musí hráč využít tlačítko nového dne, čímž dojde ke zvýšení počítadla dnů a obnovení jeho herních možností. Vizualizace akcí za den je vyobrazena pomocí diagramu na obrázku 4.3.

Hráč ovšem musí ze svých dnů vytěžit co možná nejvíce, zvýšení počítadla dnů má totiž neblahý dopad. Každá rostlina dokáže přežít pouze omezený počet dnů, po jejich uplynují následně umírá. Cílem hry je dosáhnout co nejvyššího dne, přičemž při uhynutí všech rostlin hráč prohrává a musí začít od začátku.

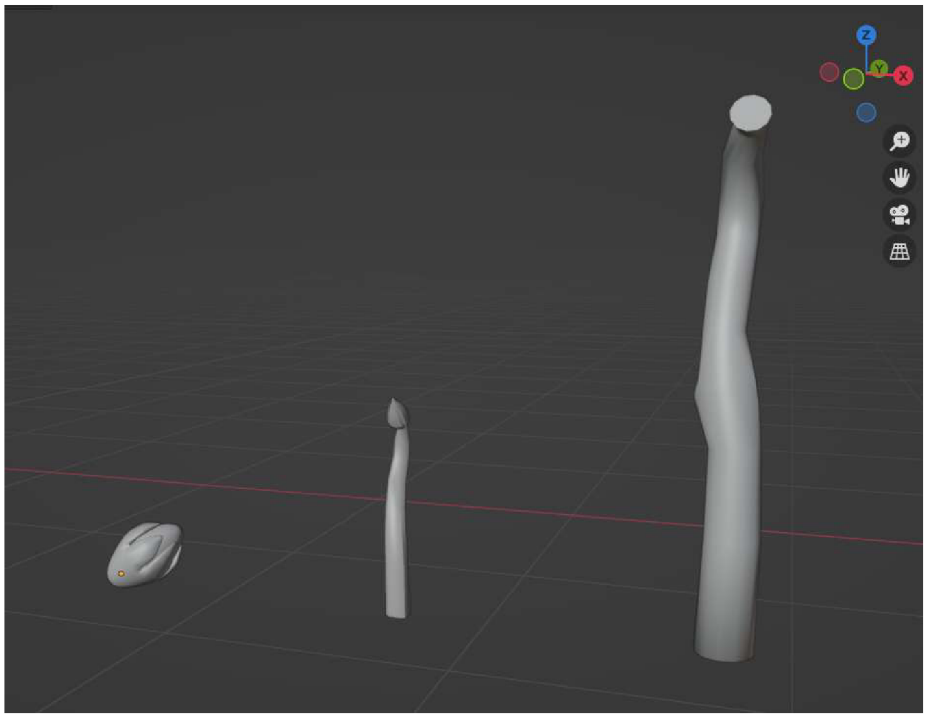
Vyšší dny s sebou zároveň přinášejí další riziko. Hra se do soubojů snaží propojit hráče, kteří se nácházejí (ideálně) ve stejném dni, to znamená, že na křížení své sbírky rostlin měli stejný počet herních dnů. Pokud jeden hráč kříží správně a druhý špatně, tento rozdíl půjde o to více poznat s přibývajícími dny ve hře.



Obrázek 4.3: **Diagram případů použití** popisující akce, které má hráč k dispozici za jeden herní den.

Semínka rostlin

Po úspěšném šlechtění rostlin získává hráč semínko. To je potřeba zasadit a poté vyčkat určitý počet herních dnů na dokončení růstu. Herní návrh počítá se zaléváním pomocí speciálních **konviček**, které by urychlily růst rostliny, nebo se pozitivně projevily na hodnotách atributů. Rostlinu je možné zasadit se zpožděním od šlechtění, jelikož od chvíle dokončení růstu začíná odpočet dnů, po které je rostlina naživu. To přidává prostor pro strategii, kdy hráč musí dopředu zvažovat, jak nepřijít o všechny své rostliny. Růst rostliny je vizualizován pomocí modelů na obrázku 4.4.



Obrázek 4.4: Herní modely rostoucího stonku od semínka v prostředí *Blender*.

Soubojový systém

Soubojový systém je plně online a vyhledává náhodné hráče. Preferováni jsou hráči v podobném herním dnu, tato mechanika je popsána v předešlé podsekci o denním cyklu. Souboje jsou podobné hrám Pokémon (viz obrázek 3.8) nebo Final Fantasy (viz obrázek 3.11). Hráč má na výběr z omezeného počtu akcí v podobě útoků. V případě hry PlantEVO jsou útoky rostliny určeny částmi, ze kterých se rostlina skládá. Každý list a květ má přiřazený jeden **útok**, který hráč může v souboji použít. Tyto útoky mohou mít různé efekty od **léčení** po **ubírání** protivníkových **životů**. Dále mohou na soupeře aplikovat **speciální efekty**, které ho v různých ohledech oslabí. Každý útok také stojí určitý počet nasbírané **sluneční** nebo **vodní energie**.

Souboje jsou tvořeny tahovým systémem. Na začátku každého tahu se hráčům doplní sluneční a vodní energie. Velikost doplněné energie se odvíjí podle hodnot atributů *Sundrain* a *Waterdrain*. Následně má hráč možnost útočit nebo svůj tah přeskočit. Poté, co oba hráči vyberou svou akci jsou útoky provedeny v pořadí odvíjejícího se dle atributu rychlosti

použitého útoku. Souboj končí v moment, kdy počet životů jednoho z hráčů klesne na 0. Vítěz získává herní měnu v podobě DNA bodů.

Souboje je nutné konzistentně vyhrávat, v opačném případě hráč dříve nebo později ztrácí všechny DNA body a je neschopen dalšího křížení. Jeden souboj by neměl trvat déle než pár minut.

DNA body – herní měna

Do hry je plánováno přidání obchodu s herními předměty. Jedním z takových předmětů by mohla být pomůcka umožňující klonování rostliny namísto šlechtění. Dále by mohly herní předměty prodloužit její životnost, zesílit rostlinu v soubojích nebo zrychlit proces jejího růstu. Momentální využití herní měny je ovšem limitováno pouze na cenu za šlechtění.

Speciální detaily

Jednou z částí rostlin jsou speciální detaily. Ty mohou mít tvar trnů, léčebných drahokamů nebo například úlomků zrcadla. Speciální detaily rostlinám přidávají **pasivní schopnosti**. Tyto schopnosti nelze aktivovat, aktivují se samy při naplnění určitých podmínek. Například schopnost trnů při každém protivníkově útoku způsobuje zpětné poškození.

4.3 Návrh serveru

Hra je postavena na online prvcích a navržena pro architekturu klient server. Využití této architektury je počítáno s dedikovanými servery z důvodu ochrany vůči podvádění a centralizace hry. Herní server zároveň uchovává data hráčů a omezuje akce nad nimi pouze na ty, které hráč může v jeho aktuálních podmínkách provést. Hráči například nebude umožněno křížit rostliny, které nevlastní.

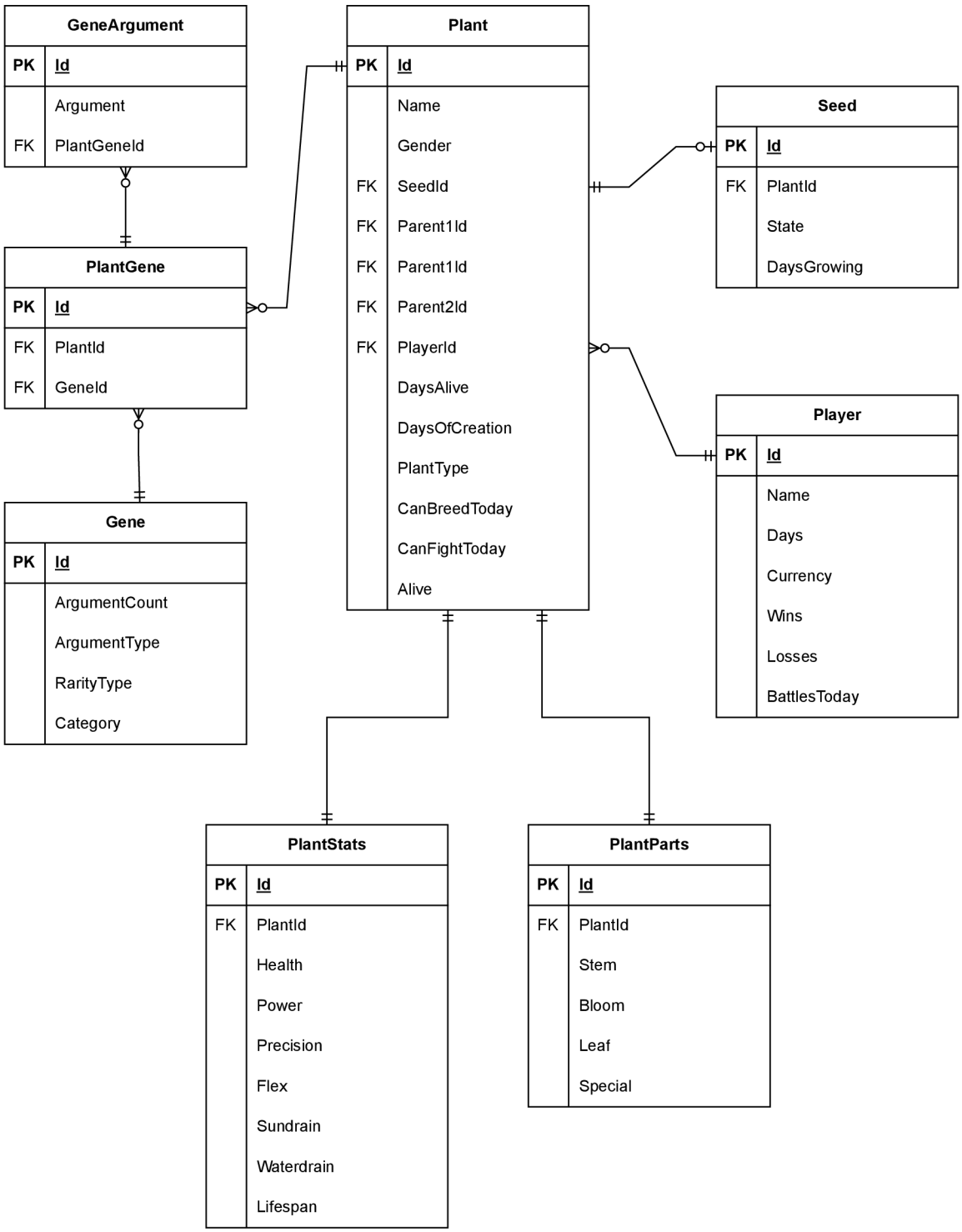
Základní pilíře funkčnosti serveru PlantEVO jsou:

- **Ukládání dat hráčů** – server spravuje data hráčů a obsahuje vždy jako první validní verzi dat.
- **Definice a implementace akčního rozhraní** – hráči mohou zasílat požadavky dle definovaného rozhraní serveru. Požadavek provádí manipulace s daty a vrací výsledek v kontextu dané akce.
- **Vytváření a správa zápasů** – hráči mohou požádat o nalezení zápasu. Server spravuje frontu hráčů a ovládá probíhající souboje.

ER diagram

Hra využívá databázi, která ukládá data hráčů. Základem pro návrh databáze je sestrojení ER (*Entity relationship*) diagramu. Tento diagram popisuje vztahy mezi databázovými entitami a je využíván pro vizualizaci databázového schématu. ER diagram pro databázi hry je k vidění na obrázku 4.5.

Diagram prezentuje tabulku **GeneArgument**, která si žádá krátké objasnění. K některým typům genů je nutné připojit argument, který přidává kontext k efektu genu. Například gen *Keeper*, který zvyšuje šanci na předání atributu potomkům, se může vztahovat



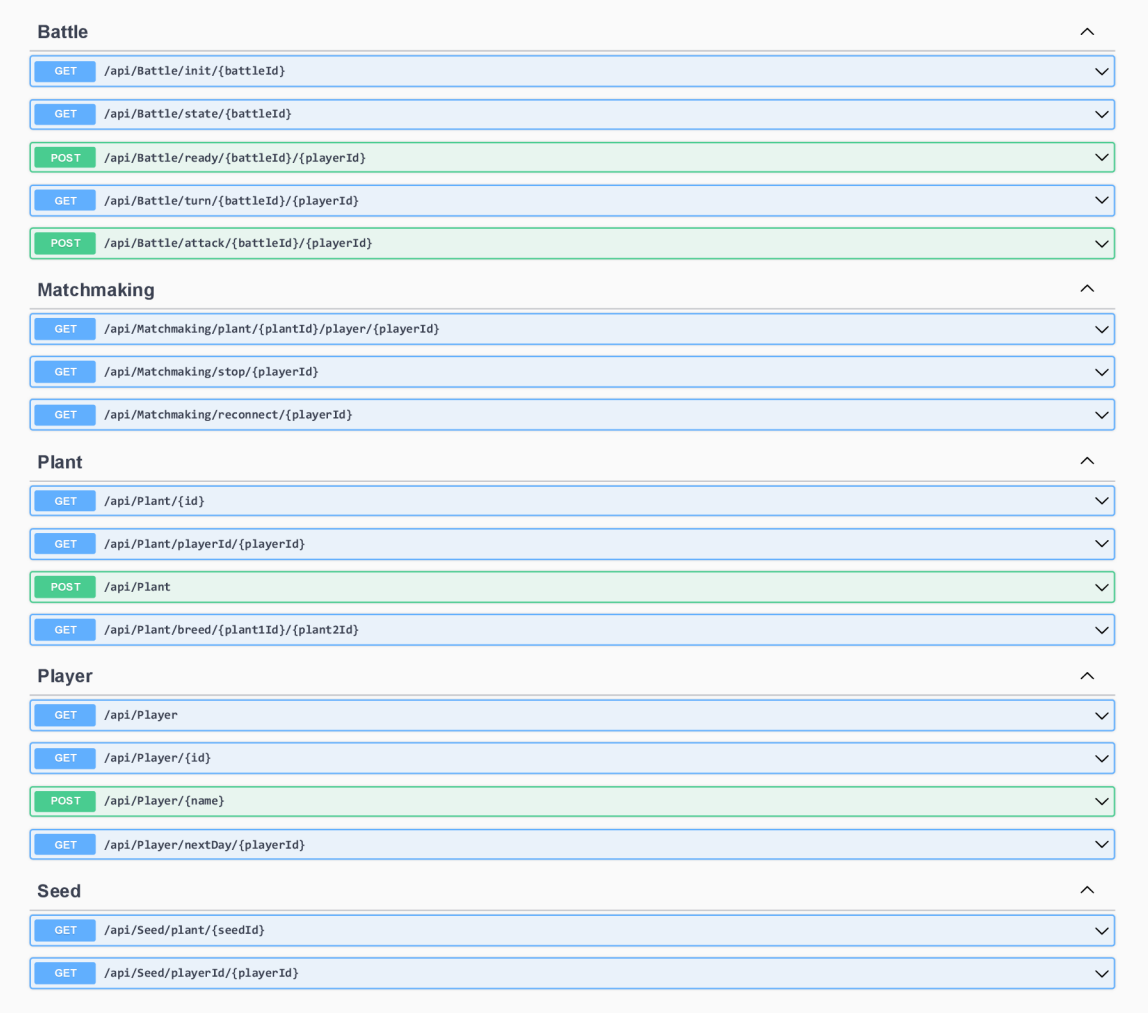
Obrázek 4.5: **ER diagram** serverové databáze.

k jakémukoliv ze sedmi atributů rostliny. To, o který atribut se jedná, je specifikováno pomocí řetězce *Argument*. Tento přístup byl zaveden z důvodu zajištění libovolného využití argumentu, které se odvíjí od konkrétního typu genu.

Ve schématu nejsou zahrnuty tabulky útoků, které by mohly být uloženy v číselníkovém schématu. Návrh předpokládá s uložením dat útoků do statických proměnných serveru. V případě potřeby může být databáze rozšířena a upravena.

Návrh API

Server počítá s využitím REST API. To je rozděleno do 5 sekcí: *Plant*, *Seed*, *Player*, *Matchmaking* a *Battle*. První 3 sekce zprostředkovávají databázové operace nad objektem jejich zájmu. Sekce Matchmaking umožňuje vyhledat zápas nebo jeho vyhledávání zrušit. Sekce Battle umožňuje vykonávat akce v konkrétním zápasu. Návrh aplikačního rozhraní si lze prohlédnout na obrázku 4.6.



Obrázek 4.6: Návrh serverového API zobrazený pomocí *Swagger UI*.

Kapitola 5

Implementace hry

Implementace hry PlantEVO se dá rozdělit na několik fází. Jedná se o tvorbu herních modelů, následně o programování serveru a herního klienta. Server a klient jsou dvě samostatné aplikace, jejichž programování proběhlo odděleně, na některých místech v kódu (zejména v oblasti soubojů) je logika oboustranně provázaná a její programování probíhalo zároveň. Tato kapitola si klade za cíl být přehledem použitých technologií, dále podrobně popsat postupy a algoritmy využité při tvorbě hry.

5.1 Modelování

Důležitou součástí hry je její grafická stránka. PlantEVO využívá 2D grafiku pro zobrazení uživatelského rozhraní, herní objekty jsou dále zobrazovány ve 3D prostoru. Pro tvorbu 3D modelů byl využit nástroj **Blender**[1], který umožňuje tvorbu 3D modelů, animací, textur, procedurálně generovaných materiálů a spoustu dalších funkcí. Herní model v prostředí *Blender* je vyobrazen na obrázku 5.1

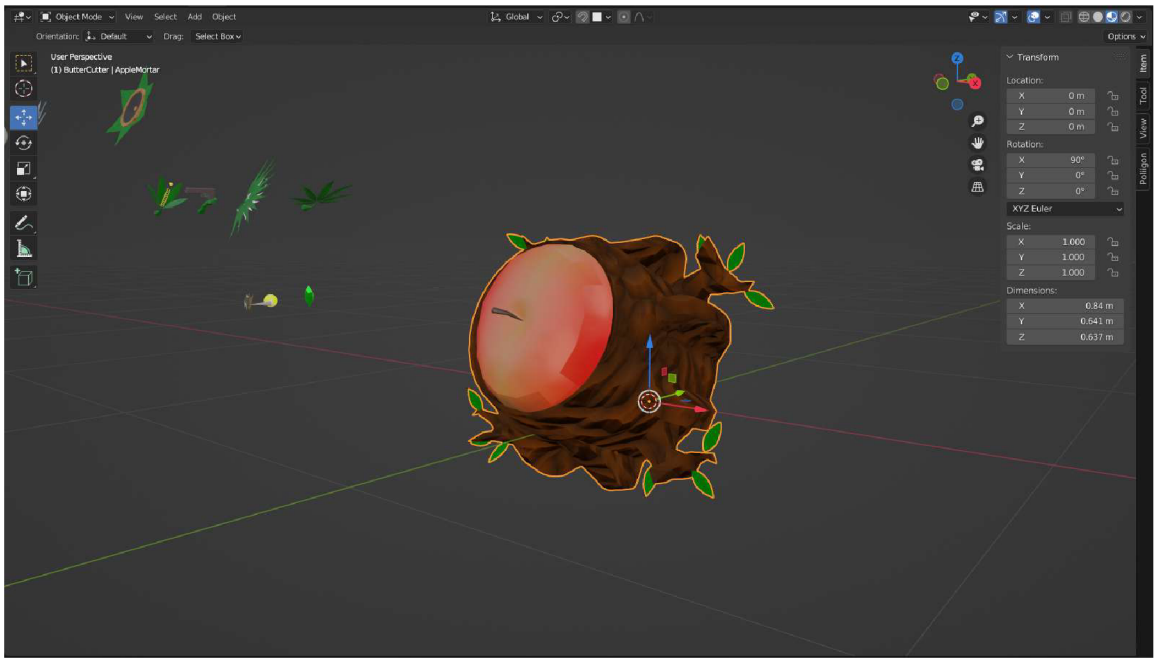
Cílem modelování byla tvorba modelů částí rostlin, které je následně herní engine schopen sestavit do podoby výsledných rostlin. Rozdělení rostliny na elementární části je podrobně popsáno v sekci 4.2. Výsledkem modelování je 19 herních modelů, kdy každému typu části rostliny náleží 4 až 5 různých modelů.

PlantEVO zatím neobsahuje animace modelů, u většiny svých modelů ovšem k 3D hmotě přidává materiál nebo texturu. Pečení textur (*Texture Baking*) je proces mapování procedurálně generovaných materiálů na model. Výsledkem operace je textura v obrázkovém formátu, kterou lze využít místo výpočetně náročného materiálu. Existuje více typů textur, PlantEVO využívá textury barev, které určují zbarvení modelu, a normálové mapy, které určují lom světla, a tím přidávají povrchu zakřivení. Příklady textur lze vidět na obrázku 5.2.

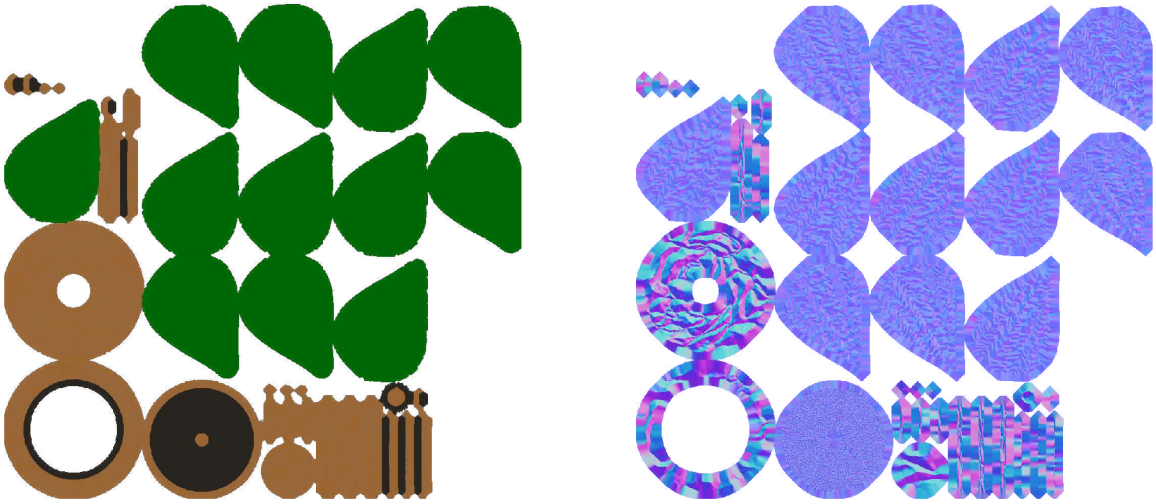
5.2 Sdílená knihovna Common

Server a klient jsou oddělené aplikace, obě jsou ale napsané v programovacím jazyce C#. To umožňuje využití sdílené knihovny, která je sestavena ze zdrojových kódů společných pro obě aplikace. Tato knihovna nese název *Common*. Jejím nejdůležitějším obsahem jsou především **sdílené modely**, které jsou využity při síťové komunikaci pro přenos dat. Dalším důležitým obsahem sdílené knihovny jsou výčty, které vyjmenovávají typy útoků nebo například konkrétní části rostlin.

[1] https://www.blender.org



Obrázek 5.1: **Apple mortar** – dokončený model květu v prostředí *Blender* před exportem do herního enginu.



Obrázek 5.2: **Subblaster** – textura barev a normálová mapa herního modelu květu.

Knihovna Common dále obsahuje složku *BattleProtocol*. Jejím obsahem jsou především definice datových struktur typu *záznam*, které udávají formát a strukturu komunikace v průběhu souboje.

V podslední řade obsahuje knihovna **sdílené konstanty** a převodové funkce atributů rostliny na její skutečné vlastnosti (například převodová funkce atributu životnost na počet dnů, které rostlina přežije).

5.3 Implementace serveru

Server je sestaven z více knihoven tříd. Hlavním faktorem členění je především účel, který knihovna plní. Knihovny lze dále seskupit do vrstev. Server je rozčleněn na 4 vrstvy:

- **Datová vrstva** (*Data access layer*) – pracuje s databází a entitami, které korespondují s databázovými tabulkami. Implementuje funkce *CRUD*[2] repositářů nad tabulkami. Datovou vrstvu tvoří knihova `DAL` (dále využívá modely entit z knihovny `Common.Server`).
- **Obchodní vrstva** (*Business layer*) – oboustranně převádí databázové entity na sdílené modely. Tento krok je důležitý k utajení některých informací, které jsou známé pouze serveru, také umožňuje agregaci databázových tabulek. Dále sdružuje větší počet databázových a jiných logických operací do komplexnějších akcí. Tyto akce se nacházejí uvnitř metod v třídách pojmenovaných jako `Service`. Další typ speciálních tříd jsou třídy typu `Facade`. Ty umožňují komunikaci s datovou vrstvou a tedy přístup do databáze. Fasády oproti repositářům pracují se sdílenými modely. Obchodní vrstvu tvoří knihovna `BL`.
- **Genetická vrstva** – vrstva obsahující veškerou logiku genetického algoritmu včetně implementace genové logiky. Tuto vrstvu tvoří knihovna `Genetic`.
- **Serverová (síťová) vrstva** – vrstva obsahující logiku REST API a výchozí spouštěcí bod aplikace. Jednotlivé vstupní body aplikačního rozhraní volají pouze metody z obchodní vrstvy. Serverovou vrstvu tvoří knihovna `Server`.

Celou architekturu serveru dle jednotlivých knihoven lze vidět na obrázku 5.3. Speciální vrstvou je **testová vrstva**, která není v předchozím výčtu zahrnuta. Její existence totiž není nezbytná při konečném provozu aplikace. Automatické testy jsou více popsány v sekci 5.3.

Entity Framework Core

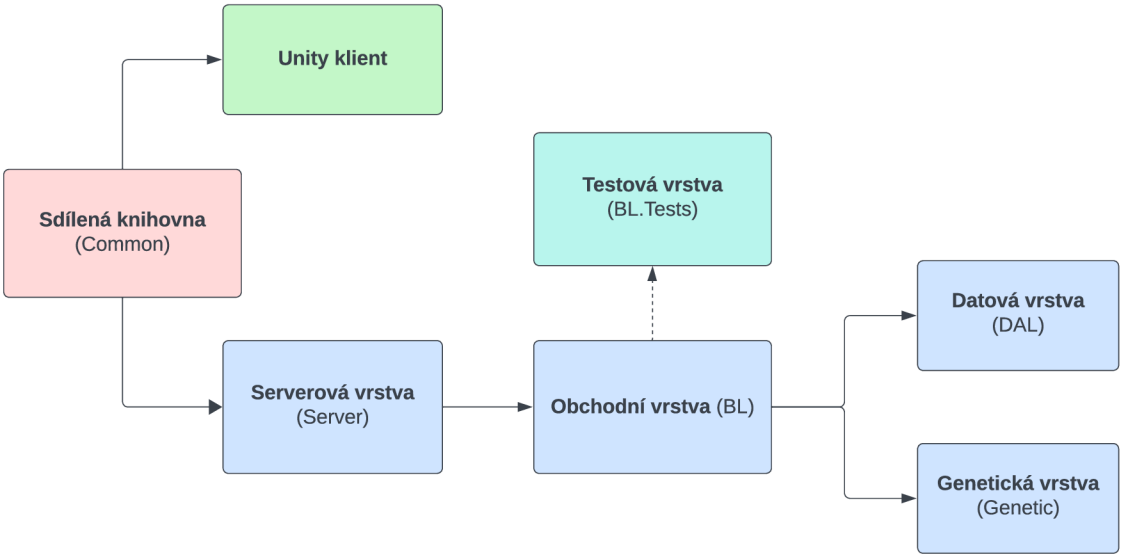
Hlavní knihovnou, která zprostředkovává databázové operace je knihovna *Entity Framework Core*[3]. Tato populární knihovna třetí strany umožňuje využít pro tvorbu databáze namísto jazyka SQL moderní **Code first** strategii, kdy programátor začíná psaním tříd databázových entit, které každá reprezentují jednu databázovou tabulku. Databáze se následně vytvoří automaticky dle definovaných entitních tříd. Dotazování databáze probíhá pomocí metod knihovny.

V serverové aplikaci tyto třídy končí klíčovým slovem `Entity`. Databázové relace jsou definovány v souboru `BPDbContext.cs`, ve kterém probíhá také automatické vkládání statických objektů, jako jsou například typy genů, do databáze. Databáze je při spuštění vytvořena pomocí třídy `SqlServerDbContextFactory`. K obsahu databáze se následně přistupuje pomocí generické třídy `Repository<TEntity>`.

Automapper

Jak již bylo nastíněno, databáze pracuje nad sadou **entit**. Klient a server však komunikují pomocí společných **modelů**. Model je v rámci serveru PlantEVO sada informací uspořádaná do záznamové struktury. Informace jsou získávány pomocí agregace databázových

[2] `create`, `read`, `update` a `delete`
[3] https://learn.microsoft.com/en-us/ef/core/



Obrázek 5.3: Diagram znázorňující **architektonický návrh aplikace**. Tučným písmem je napsán název vrstvy aplikace, v závorce se nachází název knihovny, která vrstvu zaštiťuje. Červenou barvou je znázorněna knihovna `Common`, kterou využívá jak server, tak klient. Modrou barvou jsou znázorněny serverové vrstvy. Knihovna `BL` využívá funkce z Datové a Genetické vrstvy. Knihovna Server následně využívá metody z knihovny `BL`. Zelenou barvou je znázorněn Unity klient.

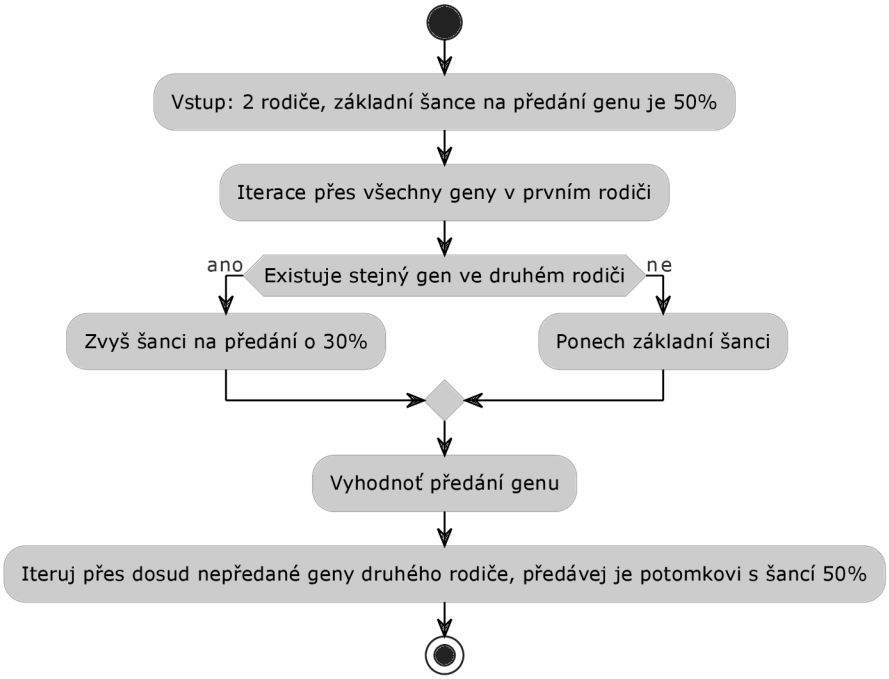
tabulek. Model obsahuje pouze relevatní informace pro daný případ, není tedy nutné zahrnout veškeré informace získané z databáze do modelu. Tím mohou být pro klienta snadno zatajeny určité informace.

Server provádí mapování entit na modely a obráceně, čímž umožňuje propojení datové a obchodní vrstvy. Mapování zprostředkovává knihovna třetích stran **Automapper**[4]. Na celý server připadá jedna instance třídy `Mapper`, ta má informace o tom, jak provádět jednotlivá mapování skrz tzv. `MapperProfiles`. Ty jsou napsané programátorem, v rámci serverových zdrojových kódů aplikace PlantEVO se nachází v knihovně tříd `BL`.

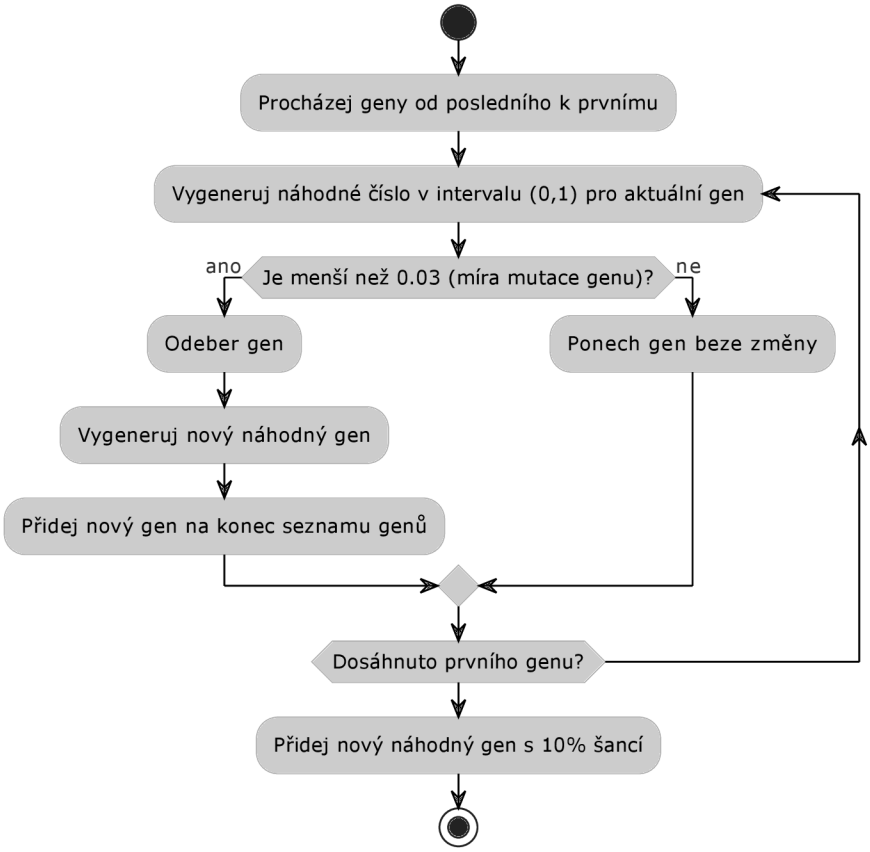
Implementace genetického algoritmu

Genetický algoritmus je realizován v knihovně tříd s názvem `Genetics`. Jeho metody jsou volány z obchodní vstvy. Implementuje rozhraní `IGenetic`, díky tomu ho lze snadno nahradit za jinou implementaci. Vstupní metodou je metoda `Breed`, která předpokládá předešlé nastavení rodičů v rámci instance genetického algoritmu.

[4] https://automapper.org



Obrázek 5.4: Diagram aktivit metody **PickGenes**, která je operátorem křížení v rámci genů rostliny. Výsledkem metody je předání genů rodičů na potomka.



Obrázek 5.5: Diagram aktivit metody **MutateGenes**, která je operátorem mutace v rámci genů rostliny. Metoda zamění některé geny v nové rostlině za jiné náhodné.

Implementace následuje návrh definovaný na obrázku 4.1. Na začátku proběhne předání genů na potomka. Křížení genů probíhá v metodě `PickGenes`, její diagram aktivit je vyobrazen na obrázku 5.4. Mutace genů poté probíhá podle algoritmu na obrázku 5.5.

Po předání genů dochází k vyhodnocení jejich efektů a k modifikaci měr genetických operátorů křížení, inkrementace a mutace. Pro jejich uložení se využívá třída **InfluenceMatrix**. Funkcionalita genů je obsažena v třídách, které dědí z abstraktní třídy `Gene`. Ta definuje virtuální metodu `Execute`, která přijímá instanci třídy `InfluenceMatrix` a modifikuje její hodnoty dle definovaného chování pro konkrétní gen. Tato metoda se volá pro každý gen těsně před zavoláním genetických operátorů, které konečnému potomku přiřadí atributy a části. Po dokončení šlechtění je rostlině přiřazeno jméno. Toto jméno je vždy dvouslovné a je generováno náhodně podle typu květů a listů rostliny. První jméno udává květ, druhé jméno list. Službu generování jmen poskytuje třída `NameGeneratorService`.

ASP.NET Core – REST API

Serverová vrstva využívá knihovnu třetích stran *ASP.NET Core*[5]. Ta umožňuje tvorbu robustních serverových aplikací včetně aplikačních rozhraní typu REST API. Integrace knihovny začíná již ve vstupním bodě aplikace v souboru `Program.cs`. Ten je rozdělen na tři části. Nejprve jsou definovány služby potřebné pro sestavení aplikace. Po jejím sestavení probíhá konfigurace služeb. Mezi tyto služby se řadí *Dependency Injection* kontejner, databázové připojení nebo *Automapper*. Dále také probíhá načtení vstupních bodů REST API a jiná nastavení webové aplikace. Koncové body a sekce aplikačního rozhraní jsou definovány v souborech dědících ze třídy `ControllerBase` a jsou uloženy ve složce `Controllers`. Jednotlivé třídní metody korespondují s návrhem REST API na obrázku 4.6.

Automatické testy

Server obsahuje automatické testy, které spravuje knihovna třetích stran *XUnit*[6]. Tyto testy byly využity především při balancování útoků. Zároveň slouží jako vstupní bod pro debugovací účely při opravách genetického algoritmu a testují zachování integrity modelů rodičů po jejich výstupu z genetického algoritmu.

5.4 Implementace klienta

Herní klient je implementován v enginu Unity, který byl krátce představen v sekci 3.1. Slouží k zprostředkování hry hráčům pomocí uživatelského rozhraní a 3D modelů. Cílem herního klienta je, aby si hráč téměř neuvědomoval existenci serveru a soustředil se pouze na hru jako celek. V případech, kdy si hráč existenci serveru uvědomuje, většinou nastávají problémy s připojením a jiné znepříjemňující okolnosti. Herní klient se snaží poskytnout hráči intuitivní a zábavné prostředí, které vysvětluje hru přirozeným způsobem.

Herní jádro

Při programování jednotlivých součástí je důležité zvažovat, které prvky se budou v programu objevovat častěji. Uspěchaný přístup má za důsledek kód, který implementuje stejnou věc vícekrát. Správa takového kódu může být náročná a nepříjemná. Dobrou praxí je

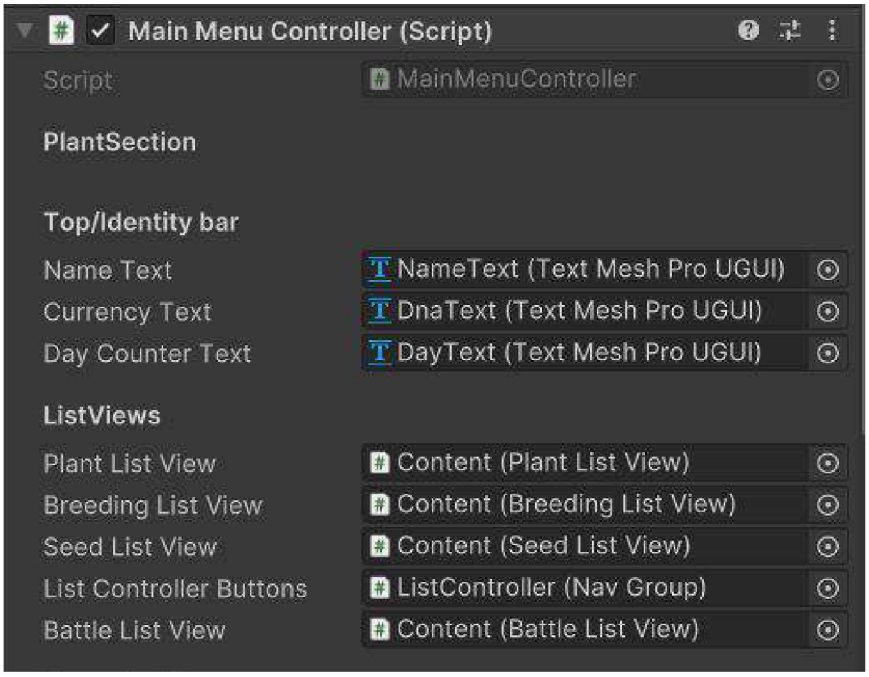
[5] https://dotnet.microsoft.com/en-us/apps/aspnet
[6] https://xunit.net

postupně vytvářet **jádro programu**, které obsahuje **znovupoužitelné komponenty** a **bázové třídy** využitelné při dědičnosti. Pomocí dědičnosti lze poté těžit z již napsaného kódu a zároveň usnadnit pozdější úpravy programu.

- **BaseController**

Herní klient se snaží dodržet základní návrhový vzor **Model-View-Controller**. Modelová vrstva je představena v sekci 5.2. Každá scéna obsahuje jeden *controller*, který zobrazuje modely na *view* komponenty. Tyto komponenty jsou většinou tvořené skriptem obsahující reference na textové a jiné zobrazovací herní objekty a logikou mapující model na komponentu. Komunikace se serverem probíhá vždy pouze z controlleru. Využití controlleru v rámci scény hlavního menu je zobrazeno na obrázku 5.6.

Bázovou třídou pro každý controller je třída `BaseController`, která spravuje identitu přihlášeného hráče a umožňuje nad ní provádět akce jako je například odhlášení.

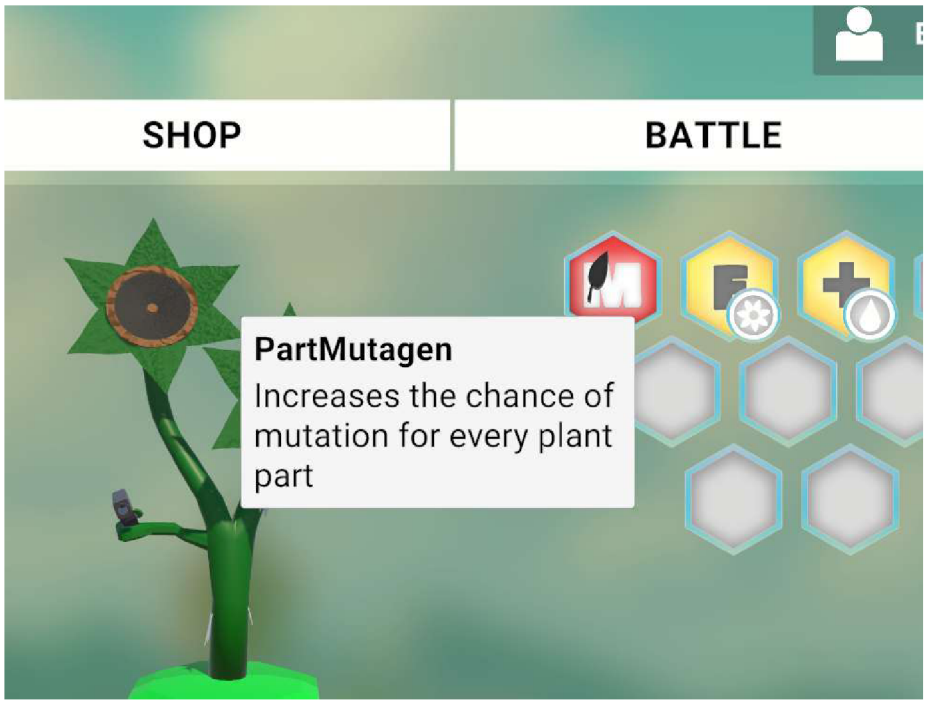


Obrázek 5.6: MainMenuController využívá reference na zobrazovací komponenty, které ukládá dle sekcí, ke které komponenta spadá. Skript je vizualizován v Unity Inspectoru.

- **Tooltipy a nápovědy**

Důležitým prvkem uživatelských rozhraní je **zobrazování nápověd** a herních pravidel. Pro tento účel se využívají tzv. **tooltipy** fungující na principu zobrazení menšího okna s nápovědou při podržení kurzoru nad objektem, ke kterému je vyžadována nápověda. Systém *tooltipů* a nápověd představuje nástroj, který nezahltí hráče textem a zároveň přehledně popíše komplexní prvky hry, jako jsou například významy jednotlivých atributů rostliny.

PlantEVO využívá dva hlavní skripty implementující systém nápověd: **TooltipTrigger** a **TooltipObject**. Skript `TooltipTrigger` je připojen k objektu, který vyvolává zobrazení nápovědy. Po podržení kurzoru nad daným objektem dochází k instanciaci herního objektu, který má připojený skript `TooltipObject`. Ten nastavuje texty nápovědy podle parame-



Obrázek 5.7: **Tooltip** okno s nápovědou vysvětlující princip genu *PartMutagen*.

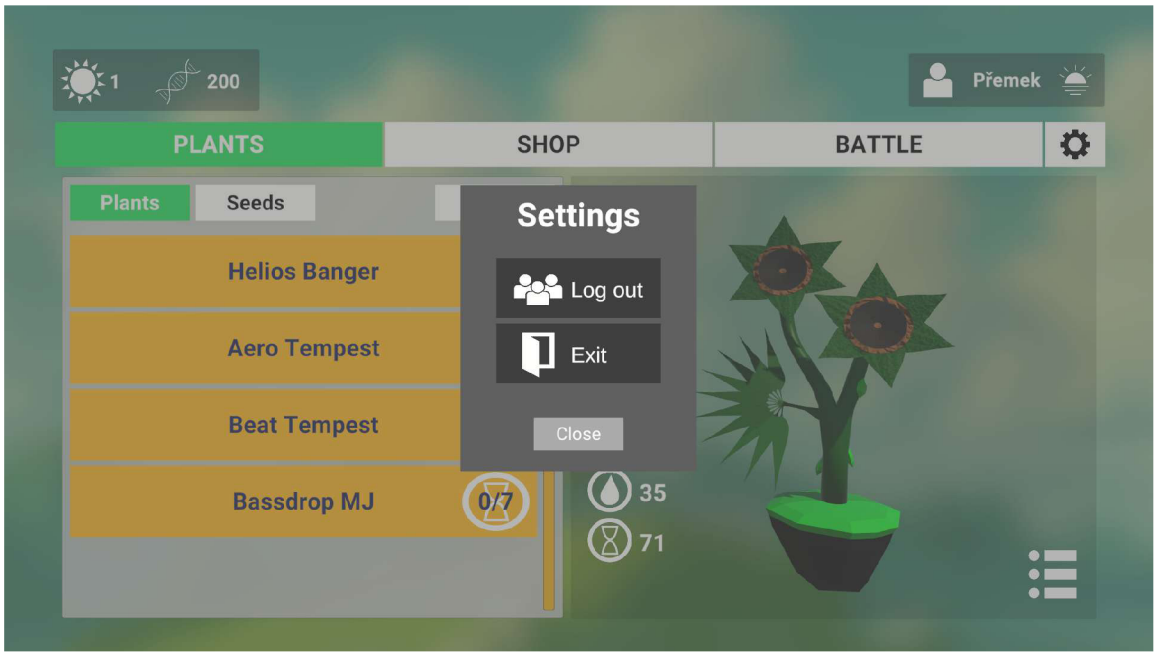
trů vyvolávajícího objektu. Objekty tooltipů jsou vytvářeny na speciálním plátnu jménem `TooltipCanvas`, které je vždy v nastavení *Screen space – Overlay* a překrývá zbytek obsahu scény. Praktická ukázka *tooltipu* je zobrazena na obrázku 5.7.

- **Modalová okna**

Modalová okna jsou dalším důležitým prvkem uživatelských rozhraní. Jejich nesprávné využití může být pro hráče frustrující, jelikož po „vyskočení“ modelového okna jsou blokovány veškeré ostatní akce hry až do jeho uzavření. PlantEVO se snaží jejich využití omezit na případy, kdy je pro další pokračování nutné zadat uživatelský vstup nebo na akce oznamovacího charakteru.

V aktuální fázi vývoje systém modalů zajišťují třídy `ModalPopper` a `ModalBase`. Skript `ModalPopper` lze připojit na objekt, který po zaznamenání události kliknutí otevře připojený modal pomocí metody `Pop()`. Později ve vývoji se však ukázalo praktičtější předávat konkrétní modalová okna referencí v argumentu funkce. Pro tento účel se využívá metoda `Pop(ModalBase prefab)`. Hra prozatím ponechává oba způsoby otevírání modalových oken. Šablony pro modalová okna jsou uloženy mezi herními prostředky (*assety*), metoda `Pop(...)` poté vrací referenci na skript `ModalBase` nové instance modalového okna.

Pokud je vytvářeno modalové okno, které dědí z bázové třídy a obsahuje speciální chování, je nutné jej po instanciaci přetypovat na konkrétní typ Modalu. Modalová okna se otevírají na speciální plátno jménem`ModalCanvas`. Ilustrace využití modalového okna je dostupná na obrázku 5.8.



Obrázek 5.8: **Modalové okno možností** – okno blokující hlavní menu umožňující odhlášení nebo ukončení hry.

· Generické seznamy (ListView, ListItem)

Další důležitou součástí uživatelských rozhraní jsou seznamy umožňující listování. Pro tento účel definuje PlantEVO generické znovupoužitelné skripty **ListView<TModel, TListItem>** a **ListItem<TModel>**. `ListItem` implementuje rozhraní `IListItem`, které má jedinou veřejnou metodu `SetModel`. Po přiřazení modelu se následně volá virtuální metoda `Render`, kterou implementují třídy dědící z `ListItem`. Výsledkem metody `Render` je zobrazení modelu na herní objekt seznamové položky.

Metoda `SetModel` je zpravidla volána ze třídy `ListView`, která opět definuje metodu `Render`. Namísto metody `SetModel` definuje podobnou metodu `SetData`, která jako argument přijímá seznam modelů, který je následně vykreslen v listovacím seznamu. Třída `ListView` má k dispozici referenci na šablonu položky seznamu, která je uložena mezi herními prostředky. Objekty položek seznamu jsou v hierarchii scény vytvořeny jako potomci herního objektu, který má přiřazen skript `ListView`.

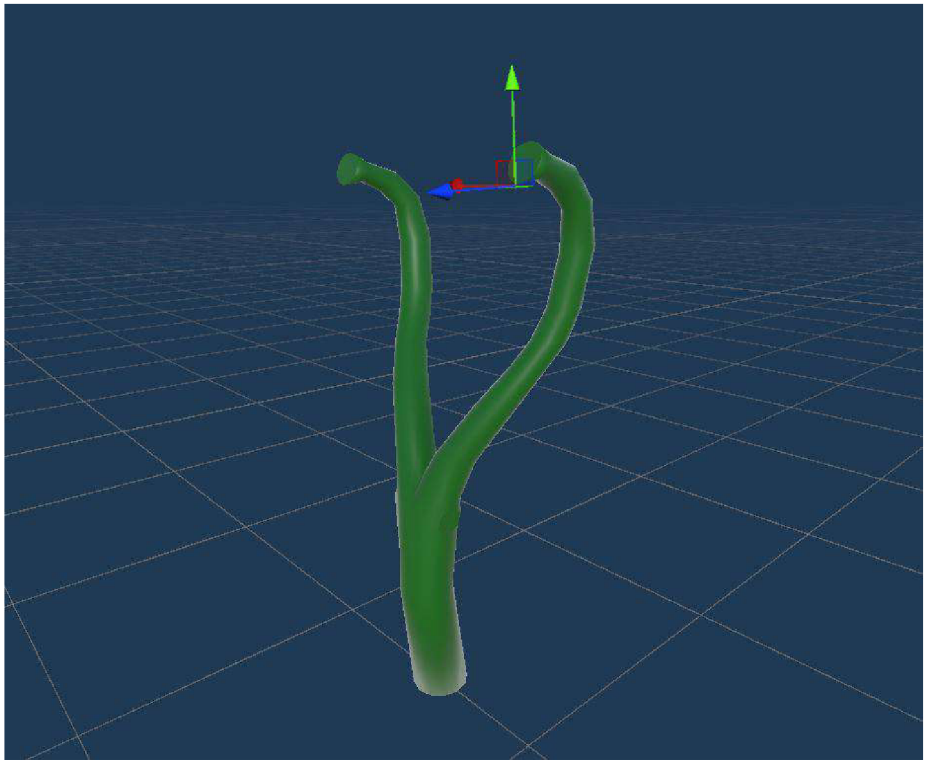
O mechaniku listování v seznamu a pozicování elementů se stará Unity a jeho komponenty pro tvorbu uživatelských rozhraní. Komponenta *LayoutElement* umožňuje nastavení velikostí jednotlivých předmětů v seznamu. Komponenta *VerticalLayoutGroup* má na starost zarovnání předmětů do sloupce. Komponenta *ScrollRect* umožňuje listování v seznamu, místy se využívá i komponenta *ContentSizeFitter* umožňující přidat dynamické velikosti například k textovým polím.

· Plant renderer

Nezbytnou součástí hry je zobrazování 3D modelů rostlin. Pro tento účel byl vytvořen znovupoužitelný skript **PlantRenderer** s jedinou veřejnou metodou `SetPlant(PlantBase plant)`. `PlantBase` je záznam, ze které dědí všechny modely rostlin. Obsahuje pouze in-

formace o atributech a částích rostliny. Skript `PlantRenderer` se připojuje k prázdnému objektu v herním prostoru, po nastavení modelu rostliny dojde k jejímu zobrazení.

Skript `PlantRenderer` počítá s existencí tzv. *ModelMapperů* ve scéně, ty mapují hodnoty výčtů na modely, které reprezentují. Dle zobrazované rostliny je vybrán a vytvořen herní objekt stonku, který je připraven pro každý z jeho typů. Ten pod sebou v hierarchii obsahuje prázdné objekty (viz obrázek 5.9, které slouží jako poziční body pro další části rostliny relativně ke stonku. Na pozice těchto bodů se poté postupně vytvářejí 3D modely listů, květů a speciálních detailů.



Obrázek 5.9: Herní objekt **stonku** udávající pozice objektů pro zbytek částí rostliny. Na obrázku lze vidět zvýrazněnou pozici, na které bude vytvořen jeden z květů.

- **NetworkManager**

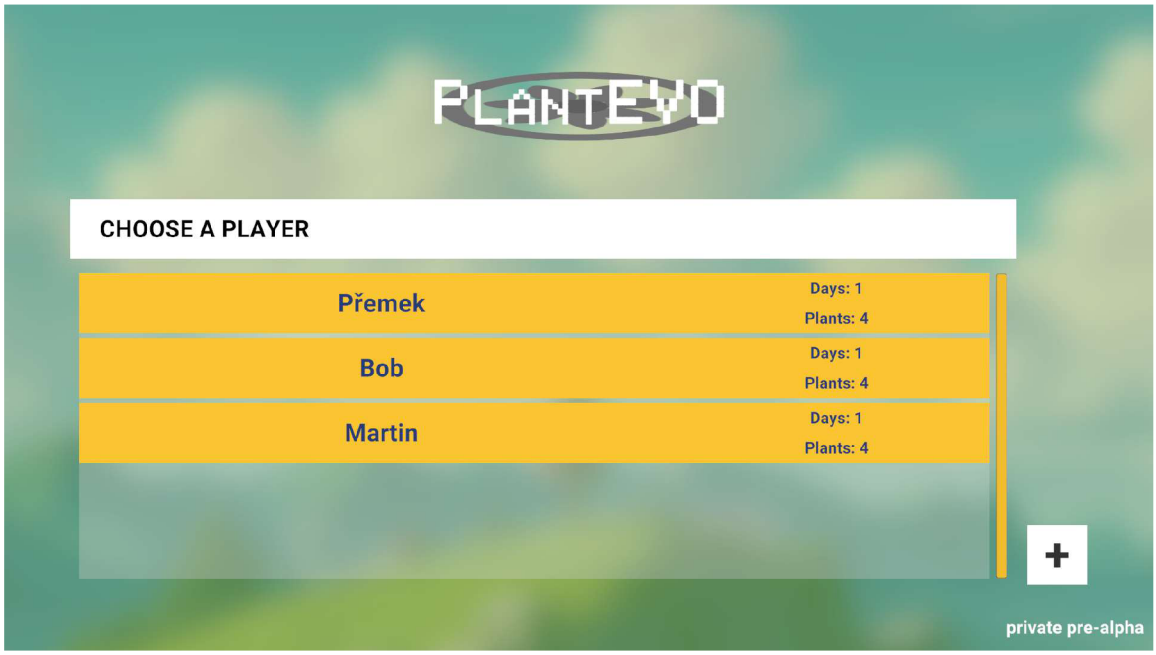
Pro síťovou komunikaci je vytvořen skript **NetworkManager**. Ten je naprogramován podle návrhového vzoru *singleton*. Obsahuje proměnné instancí klientských fasádních tříd, které kontaktují serverové API. Pro každou sekci je definována jedna fasáda obsahující metody korespondující s návrhem aplikačního rozhraní. Odesílání zpráv probíhá pomocí statické třídy **HttpClient** a asynchronních metod.

Před odesláním dat modelů je nutné objekty serializovat do formátu *JSON*, který umožňuje reprezentovat objekt v textovém formátu vhodném k internetovému přenosu. Pro účely serializace dat dotazu a následnou deserializaci odpovědi se využívá knihovna **Newtonsoft.Json**[7]. Pro zobrazení chybových informací je využit systém modalů, který zobrazí kód HTTP odpovědi, v případě, že dotaz skončí neúspěchem.

[7] https://www.newtonsoft.com/json

Přihlašovací obrazovka

PlantEVO uchovává v databázi seznam hrářů, ve hře ovšem aktuálně není přítomna autentizace uživatelů. Místo toho nabízí hra počáteční obrazovku pro výběr hráče. Lze se přihlásit pod identitou kteréhokoliv hráče, kdokoliv také může novou hráčskou identitu vytvořit. Hra samozřejmě do budoucna počítá s integrací standardu *OAuth* nebo jiné služby zajišťující autentizaci. Pro testovací účely ovšem stačí výběr identity ze seznamu hráčů. Vstupní obrazovka je zobrazena na obrázku 5.10.

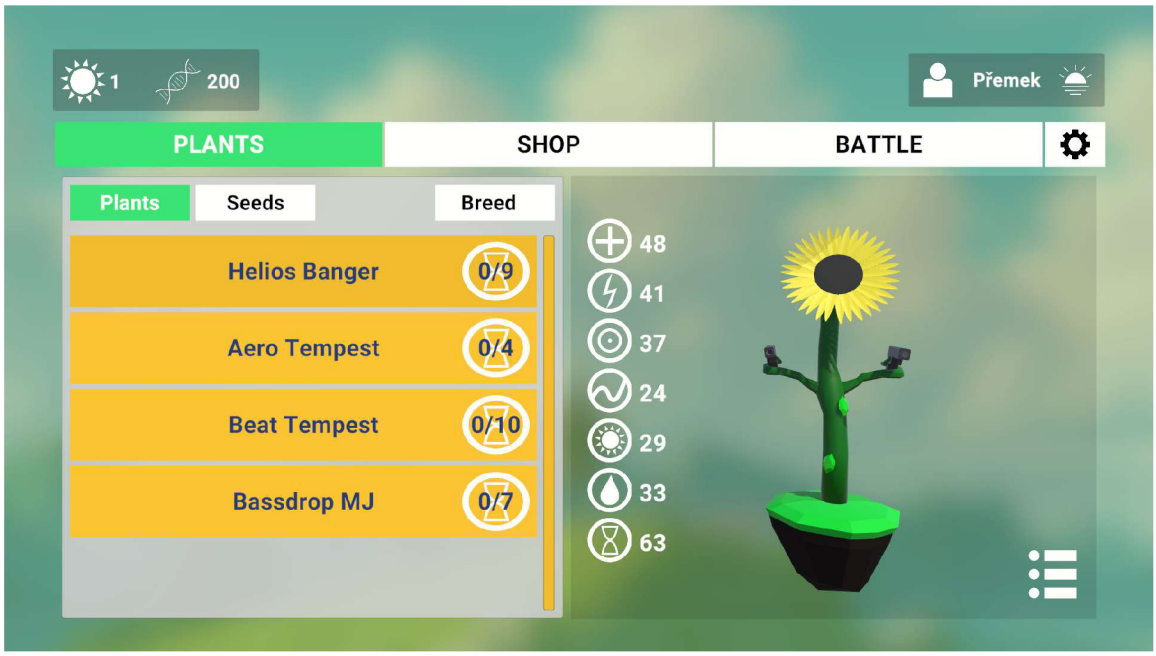


Obrázek 5.10: **Vstupní obrazovka** pro výběr a tvorbu hráčské identity – vstup do hry probíhá po kliknutím na prvek v seznamu. K jednotlivým hráčům jsou zároveň zobrazeny jejich aktuální počty herních dnů a rostlin.

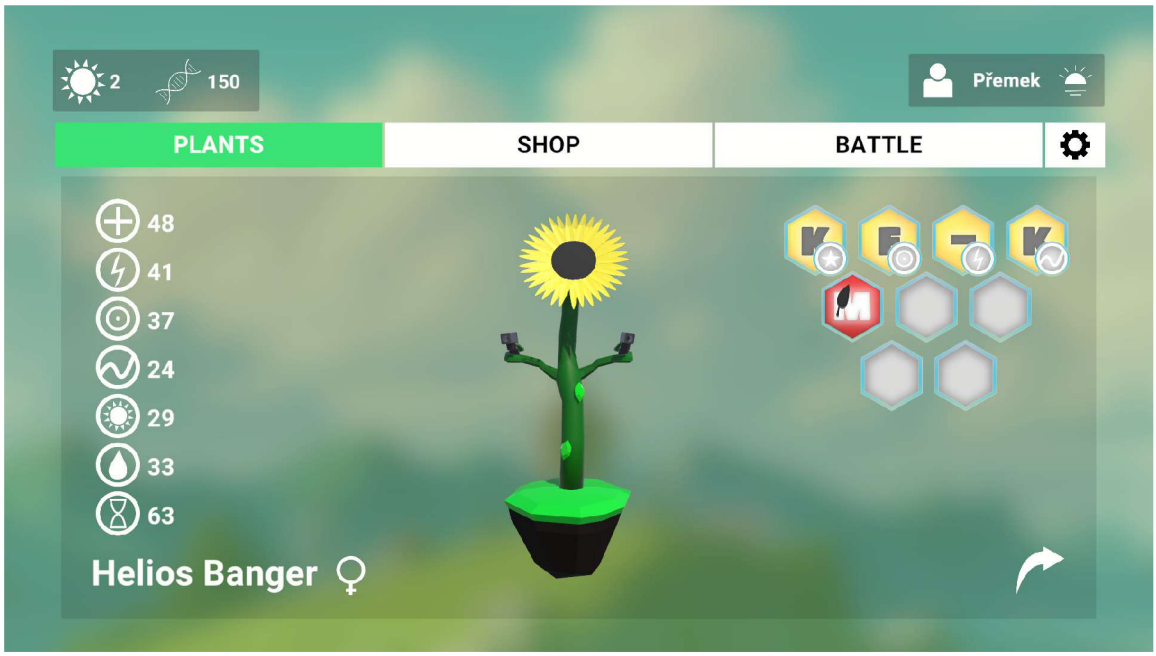
Herní menu

Hra se z velké části odehrává v herním menu, které slouží jako prostředek ke správě sbírky hráčových rostlin a tvoří rozcestník k dalším akcím hry. Herní menu se skládá ze tří sekcí: *Plants*, *Shop* a *Battle*. V sekci *Plants* si lze prohlížet rostliny, dále je možné rostliny šlechtit nebo zasadit. Sekce *Shop* zatím není naprogramována a je součástí dalšího vývoje. V sekci *Battle* lze vybrat rostlinu do souboje a vyhledávat protivníka.

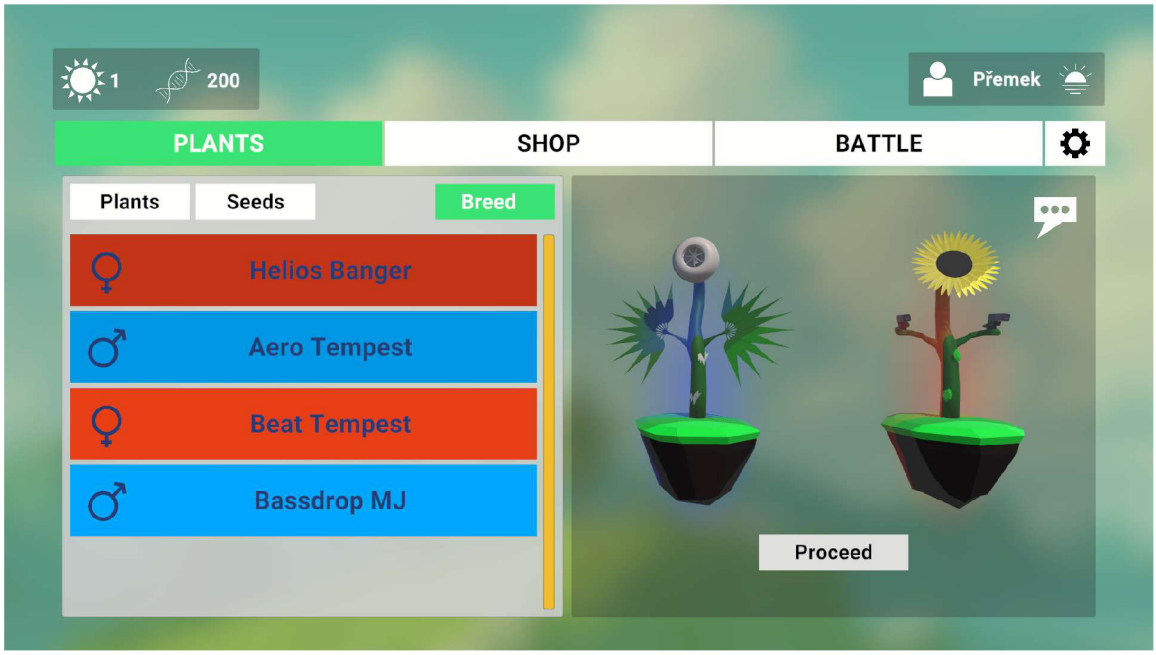
Jednotlivé sekce herního menu jsou popsány na obrázcích. Obrázek 5.11 představuje hlavní prvky herního menu a základní seznam sloužící k prohlížení rostlin. Obrázek 5.12 dále zobrazuje detailní pohled na rostlinu. Obrázek 5.13 představuje seznam šlechtění a obrázek 5.15 pohled na semínka hráče. Obrázek 5.14 zobrazuje detailní pohled na rostliny před šlechtěním.



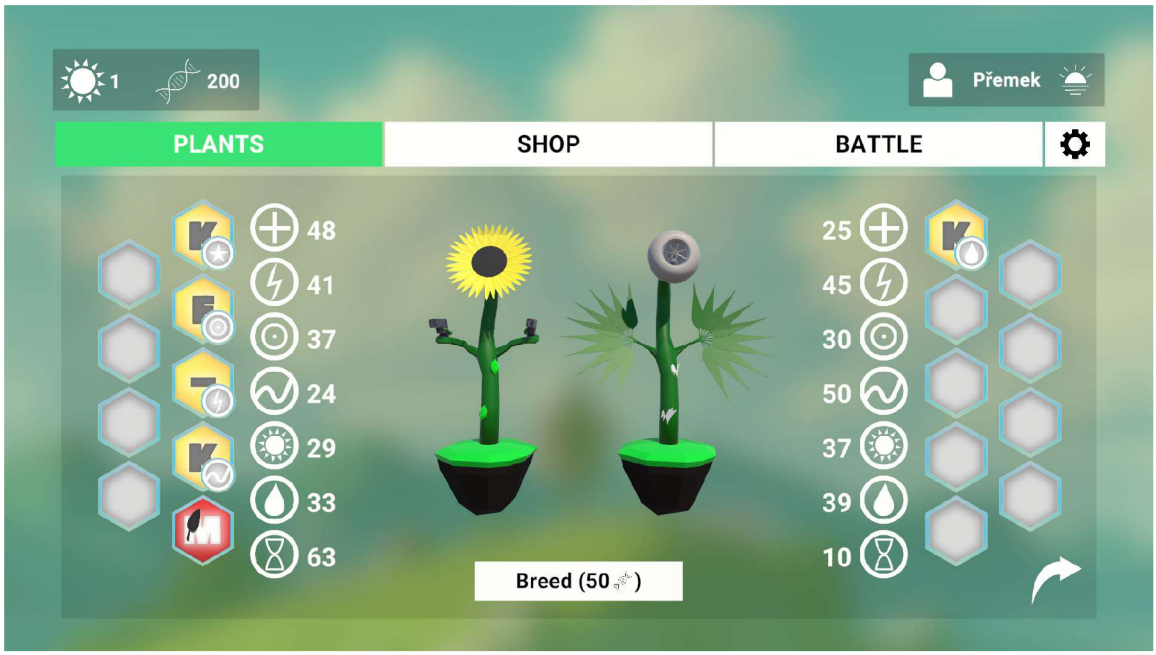
Obrázek 5.11: **Herní menu a seznam hráčových rostlin** – pod počítadlem herních dnů a měny se nachází navigační tlačítka, pomocí kterých lze přepínat mezi jednotlivými sekcemi. V levé polovině se nachází seznam rostlin. Každý řádek seznamu má kromě jména rostiny zobrazen také počet dnů, které je rostlina naživu z celkového počtu dnů, které je schopna přežít. Po kliknutí na prvek v seznamu se lze přesunout do detailního pohledu na rostlinu, ten je ukázán na obrázku 5.12. Důležitá je i sekce s identitou hráče v pravém horním rohu herního menu, která zobrazuje jméno přihlášeného hráče a tlačítko na přeskakování herních dnů.



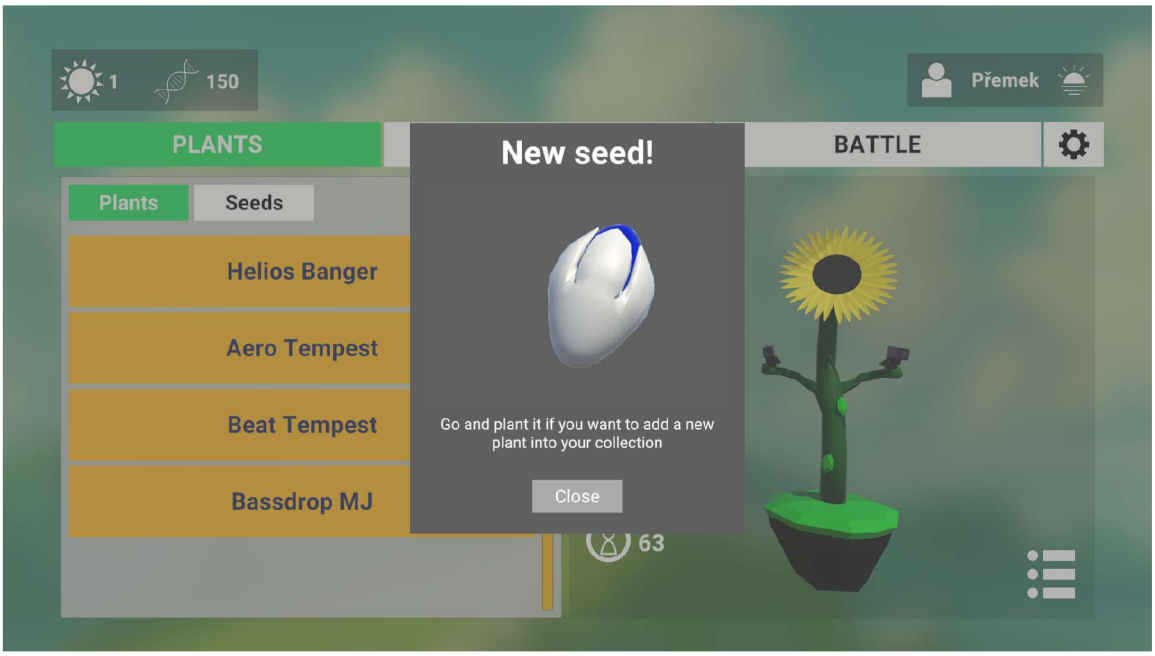
Obrázek 5.12: **Detailní pohled na rostlinu** – kromě atributů rostliny si lze prohlédnout také její geny, které jsou zobrazovány pomocí třídy `GeneRenderer`. Rostlina během času stráveném v detailním pohledu rotuje, hráč si ji tedy může prohlédnout z více úhlů.



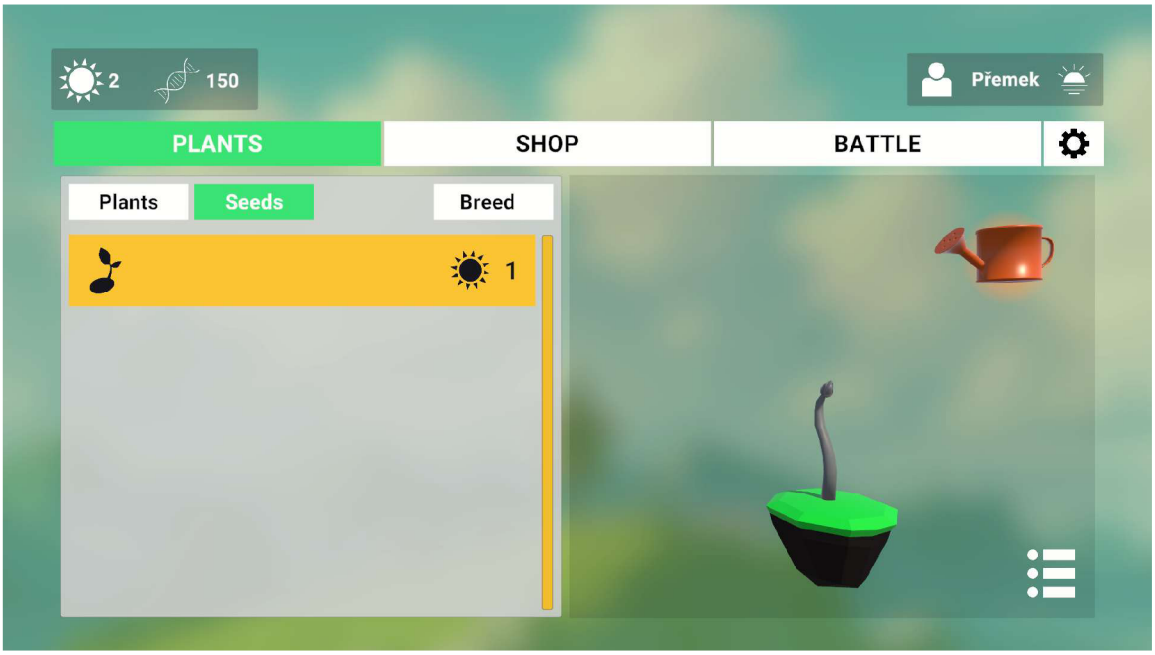
Obrázek 5.13: **Seznam šlechtění** – zobrazuje pouze rostliny, které lze šlechtit. Oproti hlavnímu seznamu rostlin umožňuje tento seznam táhnutí elementů, rostliny lze vybrat zmáčknutím levého tlačítka myši, tahem a následným puštěním nad ostrůvkem. Barva prvku seznamu určuje pohlaví rostliny.



Obrázek 5.14: **Detailní pohled na rodiče** před šlechtěním – tento pohled ukazuje atributy a geny rodičů. Umožňuje provést analýzu šlechtění a vyžaduje hráčovu úvahu.



(a) Oznámení o nutnosti zasazení semínka

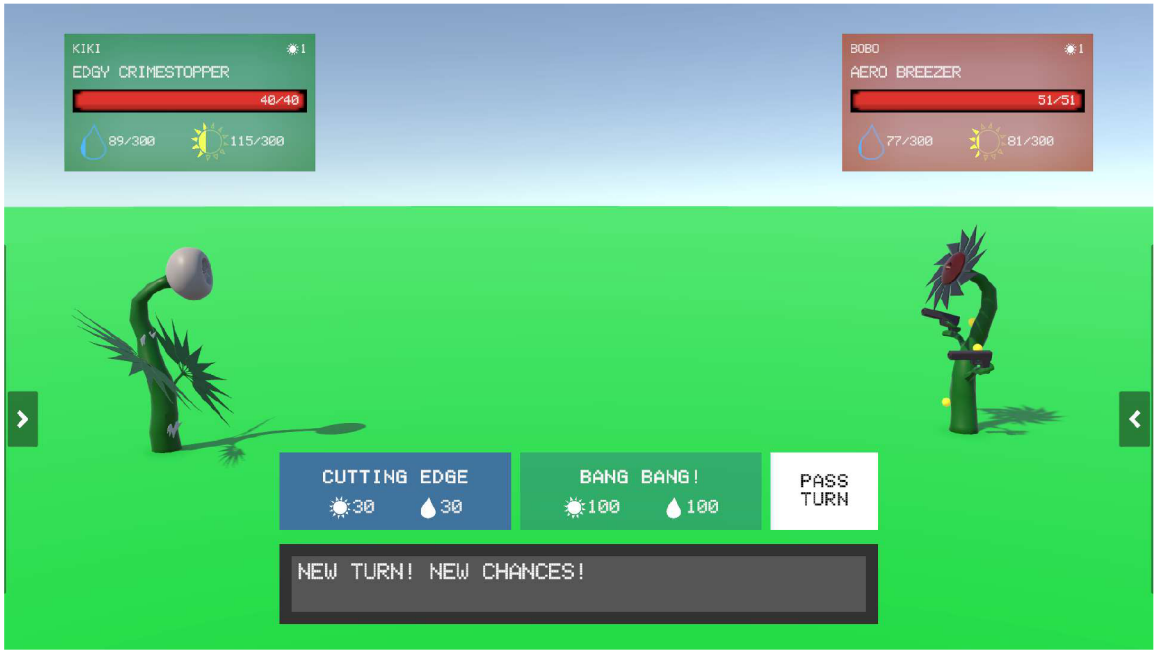


(b) Pohled na semínka

Obrázek 5.15: **Správa semínek** – po úspěšném šlechtění je hráči oznámeno, že musí nové semínko zasadit. Ve hře existují 3 různě barevné modely semínek, které se odvíjejí od typů povrchů rostlin. V seznamu semínek lze vidět počet dní, které semínko roste. Po uplynutí 4 dní rostlina vyroste a ze seznamu semínek je přesunuta do seznamu rostlin. Ve hře je také přidán model konvičky, která zatím nemá žádný efekt a na jeho implementaci se pracuje.

5.5 Soubojový systém

Do soubojů se přechází ze sekce *Battle* v herním menu. Lze bojovat až 3krát za herní den, poté je nutné přeskočit na nový den pomocí tlačítka se západem slunce v pravém horním rohu herního menu. Po vybrání rosliny (pomocí kliknutí na tlačítko s mečem) se zobrazí modalové okno, které oznamuje, že je vyhledáván protivník pro souboj. Jakmile jiný hráč začne vyhledávat zápas, hráči jsou automaticky přesunuti do duelu, kde se snaží jeden druhého porazit. Tato sekce do hloubky popíše logiku a algoritmy za soubojovým systémem pro klienta i server. Logika obou aplikací je totiž neoddělitelně provázaná.



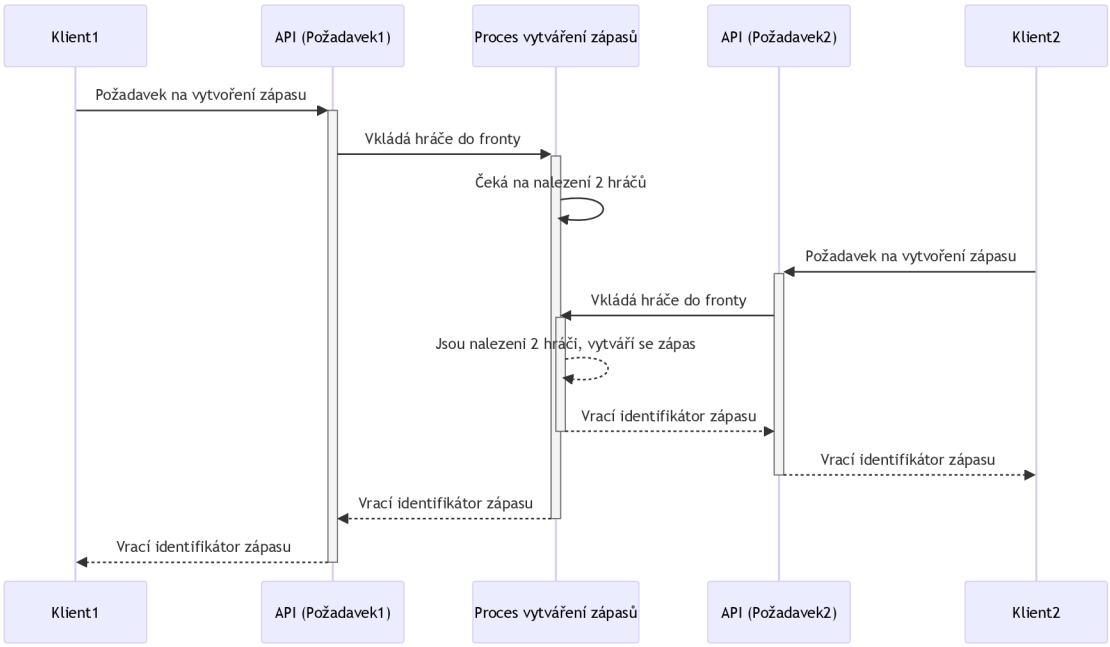
Obrázek 5.16: **Souboj v průběhu** – rostliny obou hráčů jsou zobrazeny ve scéně s rotující kamerou. V horních rozích jsou zobrazeny statusy rostlin v souboji včetně informací o hráči vlastnící danou rostlinu. Ve spodní středové části se nachází komponenta **BattleText**, která vypisuje soubojové informace. Nad ní jsou zobrazeny akční tlačítka útoků, pomocí kterých hráč vybírá svůj následující útok. Po bocích obrazovky lze vidět rozbalovací tlačítka, které umožňují zobrazit atributy rostlin v souboji.

Vytváření zápasů

Pro funkci vytváření zápasů složí na straně serveru 3 koncové body. První z nich je koncový bod typu *GET*, který vyžaduje zahrnutí identifikátorů rostliny a hráče v *URL* cestě dotazu. Tento koncový bod umožňuje začít vyhledávat zápas. Na straně serveru je hráč, který zasílá dotaz, zařazen do fronty. API zpracovávající klientův dotaz následně kontroluje sdílený seznam připravených zápasů a čeká do chvíle, dokud neobsahuje zápas přiřazený vyhledávajícímu hráči. Ve stejné chvíli paralelní proces serveru kontroluje frontu pro výskyt alespoň dvou čekajících hráčů. V případě, že hráče nachází, je vytvořen proces zápasu a jeho identifikátor vložen do seznamu připravených zápasů. Ten je následně vrácen jako odpověď klientovi. Klient si po doručení odpovědi uloží identifikátor zápasu do paměti a načítá scénu duelu. Tento postup je vizualizován pomocí sekvenčního diagramu na obrázku 5.17.

Druhý koncový bod slouží k zastavení vyhledávání zápasu. Po odeslání je hráč odebrán z fronty a je ukončen HTTP dotaz vyhledávající jeho zápas. Tento požadavek se volá při zavření modalového okna, které oznamuje probíhající vyhledávání zápasu.

Poslední koncový bod slouží k znovupřipojení do zápasu. Server ukládá seznam probíhajících zápasů. Pokud se hráč pokusí vyhledat nový zápas, je nejprve nahlédnuto, zda už náhodou neprobíhá zápas, ve kterém hráč figuruje. V případě, že je takový zápas nalezen, je hráči vrácen identifikátor probíhajícího zápasu namísto nového a hráč se je schopen znovu připojit do svého zápasu.



Obrázek 5.17: Sekvenční diagram popisující proces vyhledání zápasu.

Datové modely

Sdílená knihovna Common (viz sekce 5.2) obsahuje jmenný prostor **BattleProtocol**. Ten obsahuje zejména modely, které udávají formát a strukturu dat zasílaných v průběhu souboje.

Soubojová data rostlin se odesílají skrz záznam `PlantBattleModel`, který se skládá ze záznamů `PlantInit` a `PlantState`. `PlantInit` nese informace o atributech a částích rostliny, dále také obsahuje seznam útoků, které má rostlina k dispozici. `PlantState` obsahuje informace o životech a energiích rostliny, dále ukládá speciální efekty, kterými je rostlina zasažena.

Každá rostlina v souboji je reprezentována jedním záznamem `PlantBattleModel`. Stav duelového souboje jako celek kompletně shrnuje záznam typu **DuelState**, který sdružuje 2 záznamy typu `PlantBattleModel`. Zobrazení informací z `DuelState` umožňuje projekci aktuálního stavu rostlin, dodatečné statické data souboje, jako jsou například jména hráčů, jsou doplněny pomocí záznamu typu **DuelInit**. `DuelState` je zasílán na začátku nového tahu v záznamu `TurnInfo`, dále je odesílán po provedení útoků.

Poslední sadou modelů jsou záznamy **AttackAction**, **AttackActionResult**, **AttackInfo**, **AttackResult** a **TurnResult**. Ty společně tvoří systém předávání informací o používání útoků. Statické informace o útoku, jako je jeho cena nebo popis, se hráč dozvídá pomocí záznamu `AttackInfo`. Zasílání útoku, který chce hráč provést, probíhá pomocí záznamu `AttackAction`, který obsahuje informaci o typu útoku a číslu tahu, kterého se akce týká. Odpověď ze serveru přichází v záznamu `TurnResult`, ten obsahuje dva záznamy `AttackActionResult` a identifikátor rostliny, jejíž útok byl rychlejší. Uvnitř záznamu `AttackActionResult` je zapsané, kdo útok používá a na koho je mířený. Dále je zde uveden jeho typ a `DuelState` obsahující stav rostlin po provedení útoku.

V poslední řadě obsahuje záznam `AttackActionResult` seznam speciálních efektů, které útok způsobil a seznam zásahů, které jsou uloženy v záznamu `AttackResult`. Útoky mohou způsobovat vícero zásahů, například útok *Bang Bang*, který je propojen s typem listů *Glock Leaves* způsobí 2 zásahy. Každý zásah buď může minout nebo být kritický, více o významu jednotlivých zásahů je napsáno v následující podsekci s názvem *Útoky*.

Systém soubojových modelů je komplexní, ale robustní a bere v úvahu budoucí rozšiřitelnost, přidávání nových útoků i možnost souboje více párů rostlin zároveň. Praktické využití datových modelů je popsáno v podsekci 5.5 popisující soubojovou smyčku.

Útoky

Souboje v PlantEVO fungují na bázi tahového systému, kdy během tahu každý hráč zvolí útok, který jeho rostlina využije. Systém implementující logiku útoků je poměrně komplexní. Hráč má k dispozici útoky, které se vážou na jeho typy částí květů a listů. V průběhu souboje v každém tahu zasílá serveru informaci, který útok využívá. Ten klienta nechá čekat do chvíle, kdy útok vybere i hráčův protivník. Server následně vyhodnotí, který z útoků je rychlejší a upraví `DuelState` souboje dle implementace útoku. Poté je vyhodnocen pomalejší útok stejným principem. Informace o zásazích útoku ukládá do záznamu `AttackResult`. Klienti po získání odpovědi následně výsledky útoků zobrazí.

Pro vyhodnocení efektů útoků server implementuje třídu **AttackBase**. Ta obsahuje metodu `Execute`, která modifikuje hodnoty modelů rostlin. Metodu `Execute` lze v dědících třídách finálních útoků kompletně přepsat a definovat nové chování. V základu metoda zaplatí za cenu útoku pomocí sluneční a vodní energie. Následně vyhodnotí zásahy útoku pomocí konstanty `AttacksCount`. Pro každý zásah se vyhodnocuje, zda je kritický nebo zda zásah protivníka minul.

Poslední oblastí ve vyhodnocování útoků je aplikace speciálních efektů. Některé útoky způsobují speciální efekty. Například útok *Power of Music* zvýší *Ohebnost* protivníkovi rostliny za cenu snížení jejího atributu *Přesnosti*. Efekty také mohou trvat omezený počet tahů, po jejich vypršení se stav rostliny vrátí do stavu před aplikací efektu. Pro účel aplikace efektů při použití útoku slouží virtuální metoda `AdditionalEffects` třídy `AttackBase` a třída `BaseStatusEffect`. Ta obsahuje metodu `Activate`, která způsobí změnu v modelu rostliny cíle efektu.

Komponenta BattleText

Základním prvkem scény duelu je skript **BattleText**, který umožňuje zobrazení soubojových akcí pomocí postupného vypisování do textového pole. Text se vypisuje písmeno po písmenu. `BattleText` obsahuje 3 varianty pro výpis informací. Nastavení hodnoty `Text`, metoda `FlushAndWrite` a metoda `DisplayAttack`. Přiřazení do hodnoty `Text` vkládá text do fronty, která se vyprazdňuje postupným vypisováním. Metoda `FlushAndWrite` vyprázdní

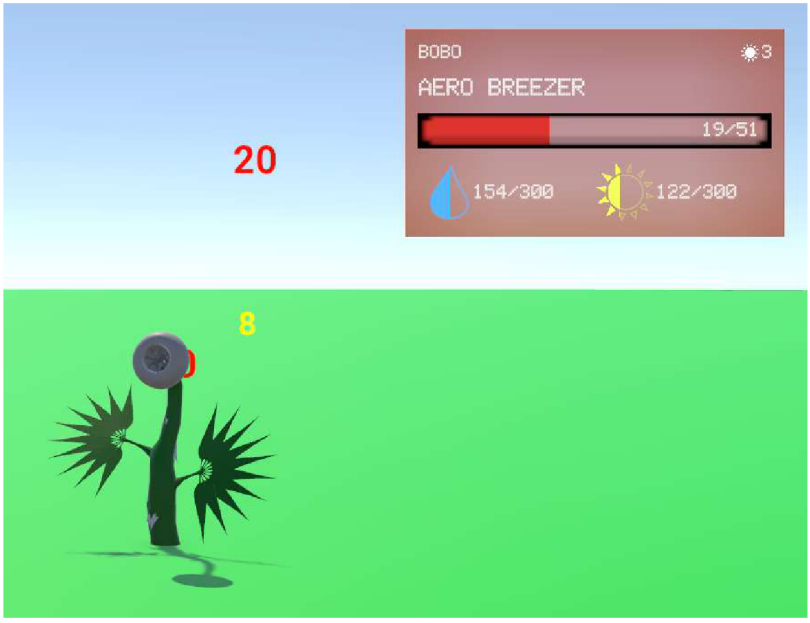
frontu a ihned vypíše text předaný v argumentu. Metoda `DisplayAttack` přijímá za argument model výsledku útoku. Jejím výsledkem je výpis celkového poškození, léčení a efektů útoku. `BattleText` lze vidět na obrázku 5.16 ve spodní části obrazovky.

Status rostliny

Během souboje dochází k řadě událostí. Ať už jde o klesání životů, nabíjení sluneční a vodní energie nebo o změnu v atributech rostliny pomocí speciálních efektů útoků, je nutné umožnit zobrazit status rostliny v průběhu souboje. Pro jeho zobrazení jsou využity skripty `PlantStatusControl` a `PlantBattleStats`. `PlantStatusControl` po nastavení rostliny zobrazí maximální a aktuální hodnoty životů rostliny, podobně zobrazí také hodnoty vodní a sluneční energie, toto zobrazení probíhá pomocí plnění ukazatelů. Nastavení modelu rostliny probíhá vždy na začátku tahu a po vyhodnocení efektu útoku. Komponentu statusu rostliny lze vidět na obrázku 5.16, je zobrazena pro každou rostlinu v horních rozích obrázku.

Indikátory poškození

Útoky mohou mít vícero efektů na životy rostlin. Kromě udělení poškození mohou také léčit, zároveň mohou dát vícero zásahů. Celkové poškození nebo léčení útoků se poté vypočítá jako suma efektů jednotlivých zásahů. Zásah může také minout, nebo udělit kritické poškození. Jednotlivé zásahy je nutné vizualizovat, k tomuto slouží skript **DamagePopper**. Ten generuje čísla vylétávající z rostliny, která symbolizují jednotlivé zásahy. Čísla mají různé barvy podle jejich významu a využívají skriptu **Billboard**, který směruje čísla kolmo na kameru scény. Efekt skriptu `DammagePopper` lze vidět na obrázku 5.18.



Obrázek 5.18: Číselné indikátory zásahů – žlutá barva udává klasický zásah, červená poté zásah kritický. Zelená barva čísla znamená léčení, při minutí zásahu je zobrazen šedý text *Miss*. Tyto dva efekty nejsou na obrázku zobrazeny.

Soubojová smyčka

Každý souboj má na straně serveru dedikovaný jeden proces. V tomto procesu figuruje soubojová smyčka, jejíž jedna iterace znamená jeden tah ve hře. Serverová logika této smyčky je znázorněna v algoritmu 2. Svou smyčku implementuje také klient (viz algoritmus 3), její implementace komplementuje smyčku serveru.

Algoritmus 2 Soubojová smyčka na straně serveru

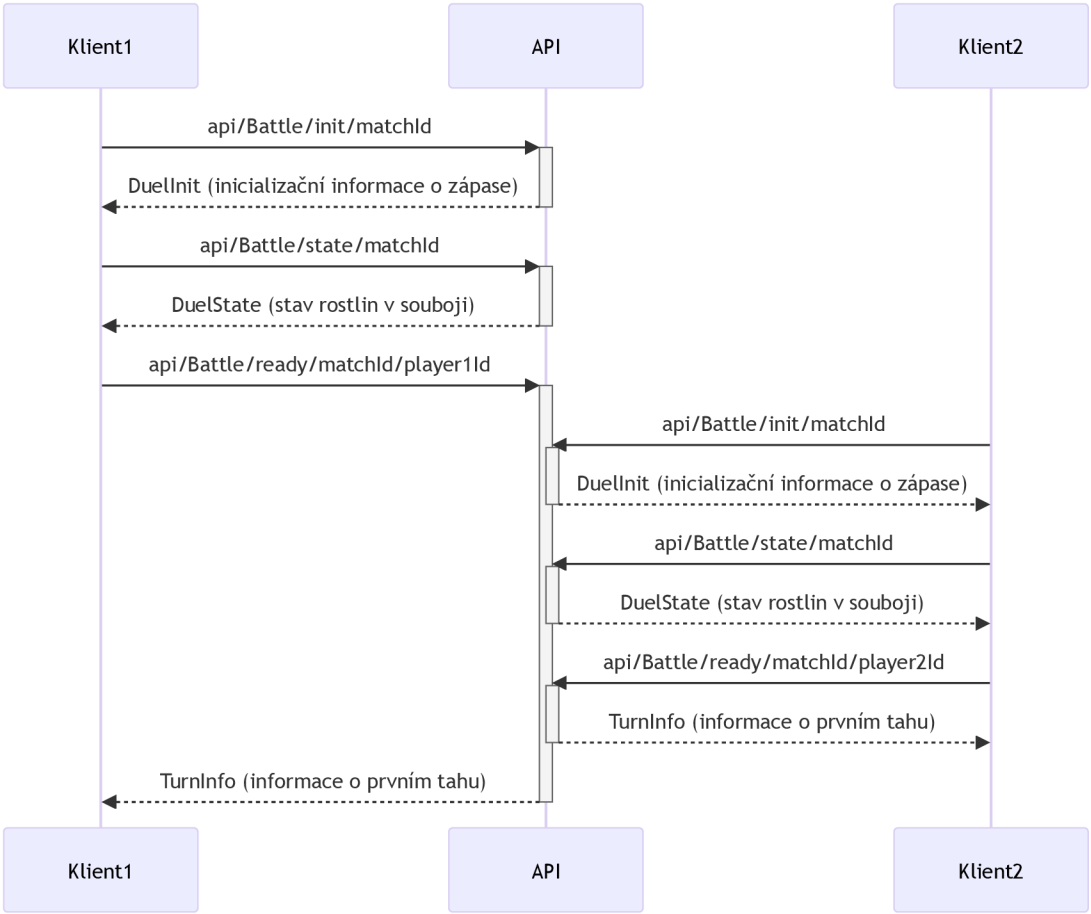
```
1:  procedure ServerLoop
2:      while State = Ongoing do
3:          wait until not TurnReady                    ▷ hráči vybírají své útoky
4:          attack1 ← zvolený útok první rostliny
5:          attack2 ← zvolený útok druhé rostliny
6:          TurnResult ← vyhodnocení útoků, předání klientům
7:          if ShouldTerminate then                     ▷ jedna z rostlin byla poražena
8:              State ← Finished
9:          end if
10:     end while
11:     if Plant1.Fainted then
12:         OnWin(Player2)
13:     else
14:         OnWin(Player1)
15:     end if
16:     TerminateAfterDelay(this)     ▷ proces souboje je ukončen po krátké prodlevě
17: end procedure
```

Algoritmus 3 Sobojová smyčka na straně klienta

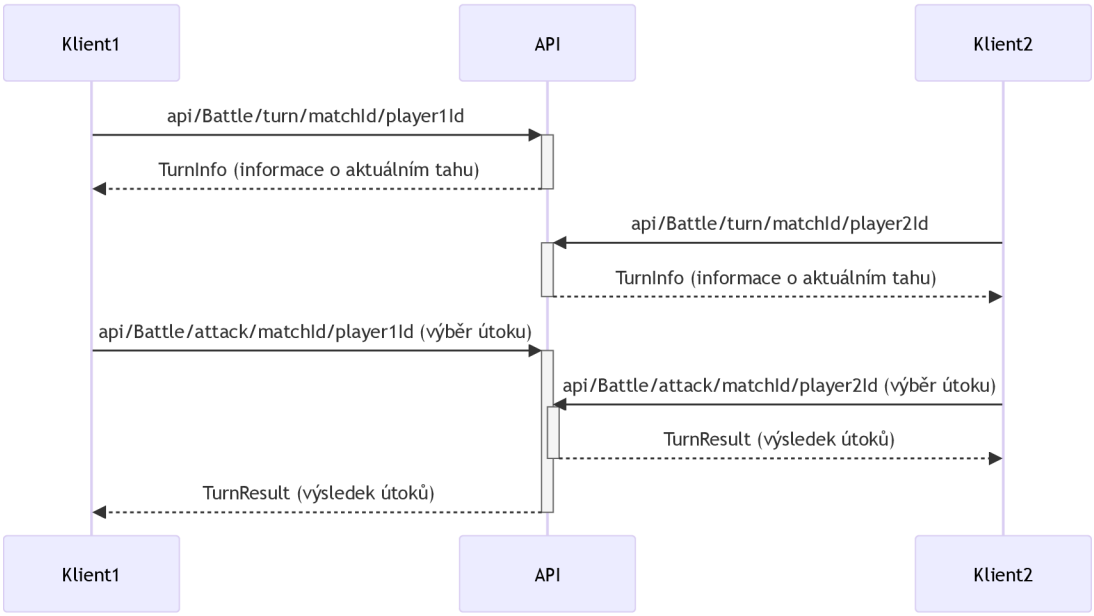
```
1:  procedure ClientLoop
2:      while true do
3:          TurnInfo ← api.GetTurnInfo          ▷ získá stav souboje v aktuálním tahu
4:          if not plant.CanAttack or plant.ForcedAttack ≠ null then
5:              SelectedAttack ← AttackType.None
6:          else
7:              wait until SelectedAttack ≠ null          ▷ vstup z uživatelského rozhraní
8:          end if
9:          attackAction ← new AttackAction
10:         attackAction.AttackType ← SelectedAttack.Value
11:         attackAction.Turn ← TurnInfo.TurnNumber
12:         attackResult ← api.UseAttack(attackAction)
13:         DisplayResult(attackResult)                  ▷ zobrazení efektů útoků
14:         if ShouldTerminate then                     ▷ jedna z rostlin byla poražena
15:             break
16:         end if
17:         SelectedAttack ← null
18:     end while
19: end procedure
```

Volání API probíhá ve dvou procedurách. První procedura je odstartována před začátkem souboje po načtení scény duelu. Jejím cílem je ujištění, že se oba hráči byli schopni úspěšně připojit do souboje. Ihned poté, co klienti oznámí serveru, že jsou připraveni, server spouští paralelní proces soubojové smyčky. Po jeho spuštění oznámí klientům, že hra je připravena a souboj může začít. Tato procedura je popsána sekvenčním diagramem na obrázku 5.19. Druhá procedura popisuje sled volání metod API, které se opakují každý tah. Sekvenčním diagramem je vyobrazena na obrázku 5.20, soubojová smyčka je zde popsána z hlediska síťové komunikace.

Součástí budoucího rozšíření je přidání časového limitu na tah. Server bude vlastnit odpočet, podle kterého se budou synchronizovat jednotliví klienti. V aktuální vývojové fázi hry by však prozatím představoval spíše zbytečnou překážku pro testování.



Obrázek 5.19: Sekvenční diagram volání serverového API při proceduře inicializace souboje – klient získá všechny důležité informace o souboji, následně čeká na to, než je získá i protivník. Poté, co jsou oba hráči připraveni, server spouští souboj a hráči vybírají svůj první útok.



Obrázek 5.20: Sekvenční diagram volání serverového API během jedné iterace soubojové smyčky – klienti zasílají požadavky na provedení útoku. Poté co server vlastní požadavek od obou klientů, vyhodnotí výsledek útoků, ten následně odesílá klientům.

5.6 Testování a dosažené výsledky

Hra byla po celou dobu vývoje ukládána do vzdáleného *GIT* repositáře. Po dokončení vývoje byla vytvořena *CICD Pipeline* pomocí *Github Actions*, která umožňuje automatické nahrání nové verze programu serveru na vzdálené produkční prostředí při změně zdrojových kódů. Pro tento účel byla vybrána služba *Azure*.

Následně proběhlo krátké uživatelské testování, kterého se zúčastnilo 5 testerů ve věku od 20 do 25 let. Výsledkem testování byla oprava některých chyb, hra je aktuálně schopna běhu bez většího množství chyb. Zpětná vazba testerů se ve větší míře shoduje na faktu, že hra není dokončená a není připravena na veřejný přístup. Nejčastější poznámkou byla absence animací herních modelů, dále malý počet útoků, které má rostlina k dispozici při soubojích. Jeden tester zdůraznil, že je potřeba větší vhled do fungování atributů pomocí obohacení *tooltipů* o již převedené vlastnosti rostliny (například přesný počet získaného počtu sluneční energie za tah). K tomuto faktu bude přihlédnuto v průběhu následujícího vývoje. Celková zpětná vazba je spíše pozitivní.

Další poznatek z testování je fakt, že nikdo moc nevěnoval pozornost genům rostlin. Tato mechanika byla přidána, aby udělala proces šlechtění méně přímočarý, je ovšem možné, že předkládá před hráče až příliš mnoho pravidel a informací. Jednou z možností modifikace genového systému je nechat volbu genového argumentu na hráči. Tím bude vyžadováno jeho seznámení s principem fungování konkrétního genu a zároveň přidá další možnost interakce s procesem šlechtění.

Plány dalšího vývoje

Hra plánuje přidání animací ke statickému stavu rostlin, aby působily živěji. Dále je v procesu herní soundtrack. V soubojovém systému je zapotřebí vytvořit animace útoků a k nim i zvukové efekty. Je také možné do soubojů zakomponovat počasí a jiné prostředí souboje.

Systém atributů rostlin se poměrně osvědčil a nebude ho nutné měnit. Částím rostlin bude časem přidáno více útoků, je také v plánu přidat více typů částí. Jednou z nich by mohly být kořeny rostliny, ty prezentují bojovou mechaniku jejich růstu, kdy se hráč v průběhu souboje snaží nalézt podzemní vodu. Mimo jiné je třeba přidat logiku ke speciálním detailům rostlin.

Dále je potřeba přidat oznámení o dokončení růstu semínka na začátku nového herního dne ideálně ve formě animovaného modalováho okna prezentující detailní přehled působení genetických operátorů, pomocí kterých vznikla nová rostlina. Podobným způsobem by bylo vhodné oznámit i úmrtí rostlin.

Další nedílný prvek, který musí být do hry přidán, je autentizační služba. Hra nemůže v produkčním prostředí prezentovat systém pro výběr identity bez zadání hesla. Tato část bude muset být v budoucnu více prozkoumána a dokončena.

Kapitola 6

Závěr

Cílem práce bylo vytvoření hry, ve které figuruje genetický algoritmus jako jeden z klíčových prvků. Cíl byl splněn.

Herní vývoj v Unity byl představen v sekci 3.1, porozumění technikám herního vývoje bylo prokázáno v sekci 5.4. Představení genetických algoritmů bylo provedeno v kapitole 2, která dopodrobna popisuje jednotlivé varianty genetických operátorů a definuje důležité pojmy a principy, které se v oblasti genetických algoritmů vyskytují. Návrh hry byl prezentován v kapitole 4, ve které jsou popsány jednotlivé herní mechaniky a klíčové prvky hry. Proces implementace klíčových částí hry a popis důležitých algoritmů byl zpracován v kapitole 5. Součástí souborů přiložených k bakalářské práci je demonstrační video ze hry.

Práce oproti základnímu zadání obsahuje rozšíření v podobě *online* hratelnosti a klient-server architektury. Výsledkem práce je robustní hra schopná dalších rozšíření s originálními herními mechanikami. Hra se dá označit za ucelenou a hratelnou demo verzi, která implementuje většinu navržených herních mechanik. Nedokončenost hry však souvisí s komplexností hry jako celku a je vyžadován další vývoj. Herní genetický algoritmus dokáže provádět šlechtění rostlin s rozšířením nad kanonickou formou genetického algoritmu v podobě speciálních genů ovlivňujících váhy genetických operátorů. Cennou částí hry je její herní jádro, které pomáhá implementovat základní prvky uživatelského rozhraní.

Tato práce mi umožnila zkusit vyvinout svou vlastní hru, což je něco, o čemž alespoň jednou v životě snila většina programátorů. Na začátku jsem měl nulové zkušennosti a byl řádně překvapen poté, co jsem odhalil, kolik znalostí musí člověk získat a kolik času obětovat pro tvorbu jednoduché hry. Mým osobním cílem pro tuto bakalářskou práci bylo ověřit, zda herní mechaniky, které mi přišly při návrhu hry zábavné, budou zábavné i při skutečném hraní. Především z osobního testování mám ponětí o tom, co ve hře funguje a co naopak potřebuje vylepšit, proto osobně hodnotím výsledek práce jako úspěch. V neposlední řadě si vysoce cením získaných znalostí z oblasti genetických algoritmů a herní historie.

Na hře je nutné ještě hodně pracovat. Statické modely a uživatelské rozhraní nestačí pro plné vtažení hráče do hry. Stejně tak je nutné přehodnotit a upravit genetický algoritmus, aby měl hráč větší kontrolu nad svým osudem ve hře. Hra má potenciál přitáhnout komunitu hráčů a spojovat lidi díky svým *online* prvkům. Se základním konceptem soubojů lze dále experimentovat a přidat kooperativní režim, kdy hráči budou bojovat společně proti silnému nepříteli. Dále je možné do hry přidat příběh, který se bude odkrývat v průběhu herních dnů, a tím docílit většího vtažení hráče do světa hry.

Literatura

[1] Alam, T., Qamar, S., Dixit, A. a Benaida, M. *Genetic Algorithm: Reviews, Implementations, and Applications.* 2020. Dostupné z: https://arxiv.org/abs/2007.12673.

[2] Bellis, M. *The History of Spacewar: The First Computer Game.* 2019. Dostupné z: https://www.thoughtco.com/history-of-spacewar-1992412.

[3] Bellis, M. *The History of Early Computer and Video Games.* 2021. Dostupné z: https://www.thoughtco.com/history-of-computer-and-video-games-4066246.

[4] Bidlo, M. *Aplikované evoluční algoritmy* [Přednáška pro předmět EVO - Aplikované evoluční algoritmy]. Brno, Česká republika: [b.n.], 28. únor 2023. Fakulta Informačních Technologií, Vysoké Učení Technické v Brně. Dostupné z: www.fit.vut.cz.

[5] Doerr, B., Le, H. P., Makhmara, R. a Nguyen, T. D. Fast genetic algorithms. In: *Proceedings of the Genetic and Evolutionary Computation Conference.* New York, NY, USA: Association for Computing Machinery, 2017, s. 777–784. GECCO '17. DOI: 10.1145/3071178.3071301. ISBN 9781450349208. Dostupné z: https://doi.org/10.1145/3071178.3071301.

[6] Eiben, A. E., Raué, P. E. a Ruttkay, Z. Genetic algorithms with multi-parent recombination. In: Davidor, Y., Schwefel, H.-P. a Männer, R., ed. *Parallel Problem Solving from Nature — PPSN III.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, s. 78–87. ISBN 978-3-540-49001-2. Dostupné z: https://www.researchgate.net/publication/220701605_Genetic_algorithms_with_multi-parent_recombination.

[7] Grefenstette, J. J. Genetic algorithms and machine learning. In: *Proceedings of the sixth annual conference on Computational learning theory.* 1993, s. 3–4. Dostupné z: https://dl.acm.org/doi/pdf/10.1145/168304.168305.

[8] Holland, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* 1. vyd. The MIT Press, duben 1992. ISBN 9780262275552. Dostupné z: https://doi.org/10.7551/mitpress/1090.001.0001.

[9] KANNAN, D. *Partially Mapped Crossover in Genetic Algorithm by Deeba Kannan.* 2022. Dostupné z: https://www.youtube.com/watch?v=EZg-l2FF-JM.

[10] Kent, S. *The Ultimate History of Video Games, Volume 1: From Pong to Pokemon and Beyond . . . the Story Behind the Craze That Touched Our Lives and Changed*

the World. 1. vyd. Crown, 2001. Ultimate History of Video Games. ISBN 9780761536437. Dostupné z: https://books.google.cz/books?id=Y7KfAAAAMAAJ.

[11] Kora, P. a Yadlapalli, P. Crossover Operators in Genetic Algorithms: A Review. *International Journal of Computer Applications*. Březen 2017, sv. 162, s. 34–36. DOI: 10.5120/ijca2017913370. Dostupné z: https://www.researchgate.net/publication/315175882_Crossover_Operators_in_Genetic_Algorithms_A_Review.

[12] Masse, M. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. 1. vyd. O'Reilly Media, 2011. ISBN 9781449319908. Dostupné z: https://books.google.cz/books?id=eABpzyTcJNIC.

[13] Nielsen, H., Mogul, J., Masinter, L. M., Fielding, R. T., Gettys, J. et al. *Hypertext Transfer Protocol – HTTP/1.1* [RFC 2616]. RFC Editor, červen 1999. DOI: 10.17487/RFC2616. Dostupné z: https://www.rfc-editor.org/info/rfc2616.

[14] Paredaens, J., De Bra, P., Gyssens, M. a Van Gucht, D. *The structure of the relational database model*. 1. vyd. Springer Science & Business Media, 2012. ISBN 978-3-642-69956-6.

[15] Schmitt, L. M. Theory of genetic algorithms. *Theoretical Computer Science*. 2001, sv. 259, č. 1, s. 1–61. DOI: https://doi.org/10.1016/S0304-3975(00)00406-0. ISSN 0304-3975. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0304397500004060.

Příloha A

Obsah přiloženého paměťového média

Přiložené paměťové médium je tvořeno následujícími složkami:

- **src** – zdrojové soubory a kódy aplikace
- **build** – zkompilované zdrojové kódy pro platformu Windows
- **thesis-src** – zdrojové kódy sloužící k vysázení textu závěrečné práce.

Dále paměťové médium obsahuje tyto soubory:

- **thesis.pdf** – vysázený text závěrečné práce
- **demonstrační video.mp4** – video ze hry představující herní mechaniky
- **readme.md** – manuál obsahující pokyny ke spuštění hry.