MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Tereza Kotěšovcová

Hra založená na evolučních algoritmech

Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: Mgr. Martin Pilát, Ph.D.

Studijní program: Informatika

Studijní obor: Počítačová grafika a vývoj počítačových her

Praha 2021

Na tomto místě bych ráda poděkovala svému vedoucímu Mgr. Martinu Pilátovi, Ph.D. za jeho rady, ochotu kdykoliv konzultovat a za skvělou psychickou podporu. Poděkování patří také mé rodině, která mne v průběhu práce podporovala a bez které bych to nezvládla.

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 21.7.2021 Tereza Kotěšovcová

Název práce: Hra založená na evolučních algoritmech

Autor: Tereza Kotěšovcová

Katedra / Ústav: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: Mgr. Martin Pilát, Ph.D.

Abstrakt:
Evoluce pohybu virtuálních bytostí je zajímavou aplikací evolučních algoritmů v oblasti umělého života. Cílem této práce bylo prozkoumat možnosti jejího využití v počítačových hrách.
V rámci práce jsem navrhla hru s tématem evoluce pohybu virtuálních bytostí, vytvořila jsem systém pro jejich evoluci na základě poznatků z existujících prací a rozšířila jej o prvky potřebné pro navrhovanou hru.
V průběhu práce se ukázalo, že pro použití ve hře je evoluce příliš výpočetně náročná. Vytvořila jsem proto zjednodušený herní prototyp a v práci jsem se zaměřila spíše na pokusy s různými variantami algoritmů pro evoluci virtuálních bytostí.

Klíčová slova: evoluční algoritmy, virtuální bytosti, počítačové hry

Title: Game Based on Evolutionary Algorithms

Author: Tereza Kotěšovcová

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Martin Pilát, Ph.D.

Abstract:
The evolution of the movement of virtual creatures is an interesting application of evolutionary algorithms in the field of Artificial Life. The goal of this thesis was to explore the possibilities of its use in computer games.
As part of this thesis I proposed a game with the theme of evolution of the movement of virtual creatures, I created a system for their evolution based on information from existing works and expanded it with the features needed for the proposed game.
In the course of the work, it turned out that the evolution is too computationally demanding to use in the game. Therefore, I created a simplified game prototype and in my work I focused more on experimenting with various settings of the algorithms for the evolution of virtual creatures.

Keywords: evolutionary algorithms, virtual creatures, computer games

Obsah

Úvod ... 1
Cíle práce ... 1
Struktura dokumentu ... 2
1 Související algoritmy ... 3
1.1 Genetický algoritmus ... 3
1.2 Novelty search ... 4
1.3 Neuronové sítě ... 5
1.4 Neuroevoluce ... 6
1.4.1 NEAT ... 6
1.4.2 HyperNEAT ... 9
2 Související práce ... 10
2.1 Evoluční algoritmy ve hrách ... 10
2.2 Evoluce virtuálních bytostí ... 11
2.2.1 Základní myšlenky ... 12
2.2.1.1 Reprezentace jedince ... 12
2.2.1.2 Fitness ... 13
2.2.1.3 Parametry evoluce ... 14
2.2.2 Hierarchický NEAT ... 14
3 Návrh hry ... 15
4 Implementace ... 16
4.1 Unity ... 17
4.2 Novelty search ... 18
4.3 Složitější prostředí ... 19
4.4 Interaktivní evoluce ... 21
4.5 Editory ... 22
4.6 Navigování terénem ... 24
5 Výsledky ... 27
5.1 Výsledky evoluce ... 27
5.2 Vyhodnocení využitelnosti ve hrách ... 33
Závěr ... 35
Bibliografie ... 36
Obsah elektronické přílohy ... 38
Příloha A - Uživatelská dokumentace ... 39
Příloha B - Programátorská dokumentace ... 47

Úvod

Existuje celá kategorie algoritmů, které jsou inspirovány přírodními jevy. Napodobují chování neuronů v lidském mozku, inspirují se pohybem hejn ptáků nebo využívají konceptů Darwinistické evoluce. Tyto algoritmy mají svá využití při optimalizaci nebo v oblasti strojového učení. Dají se ale také využít pro vytváření umělého života. Do této oblasti, v angličtině nazývané Artificial Life, patří práce Karla Simse (Sims, 1994), ve které se zabývá evolucí virtuálních bytostí. Vytváří trojrozměrné tvory, jejichž úkolem je naučit se pohybovat ve virtuálním prostředí. Jedná se o první práci, kde se pomocí evolučních algoritmů vyvíjí mozek a tvar těla bytosti zároveň.

Na Simsův článek navázalo mnoho dalších, kteří viděli potenciální využití evoluce virtuálních bytostí zejména v oblasti robotiky - pro automatizovaný design tvaru i řídícího systému robota (Hornby a Pollack, 2001; Cheney et al., 2014). Já bych ale chtěla prozkoumat jiný aspekt virtuálních bytostí - zábavnost procesu jejich tvorby.

Zábavnost je důležitou vlastností počítačových her. Evoluční algoritmy byly použity pro vytváření obsahu do her, nebo pro tvorbu umělé inteligence, která hry hraje (Risi a Togelius, 2017). Většinou jsou tedy využívány pouze jako nástroj, který předem vygeneruje obsah, a hráč je ve výsledné hře nevidí. Chtěla bych proto vyzkoušet evoluci pro hráče odhalit a dát mu ji do rukou jakožto herní mechaniku.

Některé hry se o podobné využití evoluce již pokusily, šlo ale většinou o drobné hry s cílem vysvětlovat fungování evolučních algoritmů (Lind a Nihlén, 2017), nikdy nešlo o evoluci pohybu trojrozměrných bytostí.

Cíle práce

Tato práce si bere za cíl:

1. navrhnout hru, která evoluci bytostí použije jako herní mechaniku
2. vytvořit systém pro evoluci virtuálních bytostí
3. zhodnotit výsledné řešení a celkově využitelnost evoluce virtuálních bytostí ve hrách

Struktura dokumentu

V kapitole 1 představím několik konceptů a algoritmů, které bude evoluce virtuálních bytostí využívat. V kapitole 2 pak předložím přehled her, které obsahují evoluční algoritmy a také přehled prací, které se zabývaly evolucí virtuálních bytostí. V kapitole 3 navrhnu hru, která by evoluci bytostí využívala jakožto hlavní herní mechaniku. V kapitole 4 pak popíšu implementaci vlastního systému pro evoluci bytostí. V kapitole 5 shrnu výsledky práce a zhodnotím možnosti využití evoluce virtuálních bytostí v počítačových hrách.

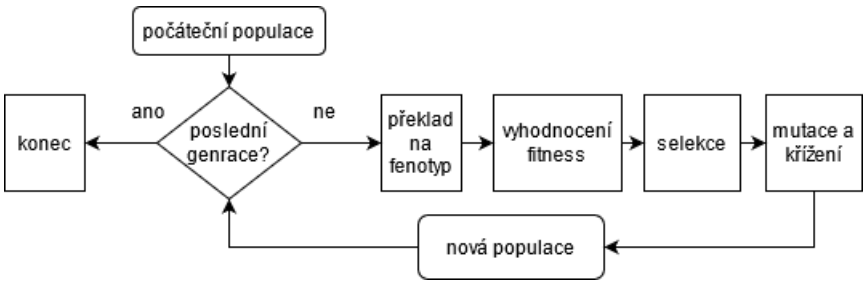
1 Související algoritmy

Tato kapitola popisuje algoritmy, které bývají využívány při evoluci virtuálních bytostí.

1.1 Genetický algoritmus

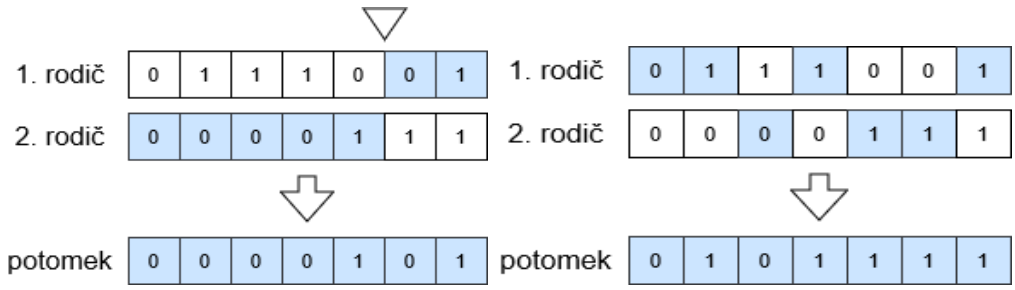
Genetický algoritmus (Eiben a Smith, 2015) hledá optimální řešení nějakého problému a obvykle při tom prohledává rozsáhlý prostor možných řešení. Inspiruje se poznatky z biologie, imituje proces evoluce.

Funguje tak, že si udržuje populaci jedinců, kde každý jedinec reprezentuje jedno možné řešení problému. Toto řešení je navíc zakódováno do podoby tzv. genotypu. Populace tedy neobsahuje přímo samotné jedince, ale vlastně jejich DNA. Aby mohlo dojít k ohodnocení kvality jedinců, provede se překlad genotypu na fenotyp – fyzickou reprezentaci jedince. Tyto fyzické reprezentace poté mezi sebou soutěží v daném úkolu a jsou ohodnoceny fitness funkcí. Úspěšnější jedinci (tj. lepší řešení problému) získají vyšší fitness. Poté následuje fáze selekce – přirozený výběr. Jen jedincům s vysokou fitness je umožněno vytvořit potomky. Potomci vznikají pomocí mutací (náhodných malých změn uvnitř genotypu rodiče) nebo křížením (zkombinováním genetické informace dvou rodičů). Nová generace je pak tvořena těmito potomky, případně do ní může postoupit i několik málo jedinců z generace předchozí (elitismus). Tento proces se opakuje, dokud není dosaženo předem daného počtu proběhlých generací, nebo dokud není dosaženo nějaké požadované hodnoty fitness [Obr. 1].



Obr. 1: Schéma průběhu genetického algoritmu.

Nejjednodušším příkladem genetického algoritmu je SGA (Simple Genetic Algorithm). Jedinec je zde reprezentován řetězcem bitů. Výběr jedinců ke křížení může probíhat více způsoby. Například ruletovou selekcí, která každému jedinci přiřadí pravděpodobnost, že bude vybrán, úměrnou jeho fitness, nebo turnajovou selekcí, kde je z celé populace vybráno N náhodných jedinců a ke křížení z nich bude vybrán ten s nejvyšší fitness. Křížení samotné má také několik variant [Obr. 2]. Jednobodové křížení funguje tak, že se náhodně zvolí bod v bitovém řetězci a potomek pak od prvního rodiče získá bity od začátku do zvoleného bodu a od druhého rodiče bity od zvoleného bodu do konce řetězce. Uniformní křížení zas postupuje od začátku řetězců a do potomka dá každý bit vždy z náhodně zvoleného rodiče. Dalším operátorem jsou mutace. Pro tuto reprezentaci jedinců dává smysl bit-flip mutace které s danou pravděpodobností změní hodnotu bitu. Kdyby geny netvořily jen nuly a jedničky ale třeba reálná čísla, bylo by možné použít mutaci, která k číslu přičte náhodné číslo z normálního rozdělení.



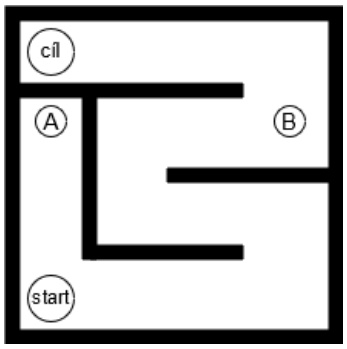
Obr. 2: Druhy křížení. Vlevo 1-bodové křížení, kde byl náhodně zvolen bod mezi pátým a šestým genem. Prvních pět genu proto potomek získá od druhého rodiče a zbylé dva od prvního. Vpravo uniformní křížení, kde si potomek pro každý gen náhodně volí, od kterého rodič jej zdědí.

1.2 Novelty search

Hlavní myšlenkou novelty searche (Lehman a Stanley, 2008) je nahrazení maximalizace fitness maximalizací „novosti“ nalezeného řešení. V průběhu evoluce se vytváří archiv se zástupci již nalezených řešení. Noví jedinci jsou pak hodnoceni podle toho, jak moc se od jedinců v archivu liší.

Odlišnost je určována na úrovni fenotypu, v archivu proto není DNA jedinců, ale jejich chování. V případě pohybu bytostí může být chování definováno jako bod v prostoru, kam bytost dokázala dojít. Novost je pak průměrná vzdálenost tohoto bodu od všech bodů z archivu.

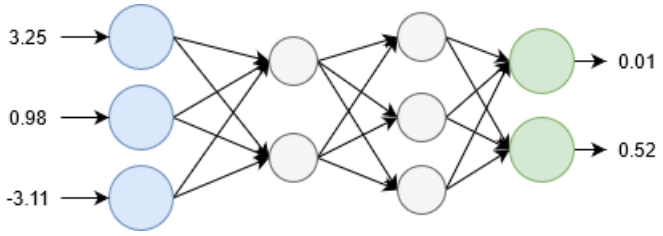
Efektem novelty searche je lepší zachování diverzity v populaci. To pomáhá proti zaseknutí se v lokálním optimu. Hlavně se prokázal jako užitečný v situaci, kdy je fitness zavádějící. Jde o situace, kdy řešení geneticky blízké ideálnímu řešení má nízkou fitness. Dobře je to vidět např. na úkolu hledání cesty v bludišti [Obr. 3], kde jedinec, který by došel do bodu A má vyšší fitness než ten, který by došel do bodu B, přitom cesta do bodu B je blíže k nalezení optimálního řešení. Fitness je zde tedy zavádějící.



Obr. 3: Situace vhodná pro novelty search. Je-li úkolem najít cestu ze startu do cíle a fitness je určována podle blízkosti k cíli vzdušnou čarou, při standardní maximalizaci fitness by jedinec, který dojde do bodu A měl vyšší fitness, než ten, který by došel do bodu B. Přitom cesta do bodu B je lepší - je geneticky blíže k nalezení optimálního řešení. Hodnota fitness je tedy v průběhu evoluce zavádějící.

1.3 Neuronové sítě

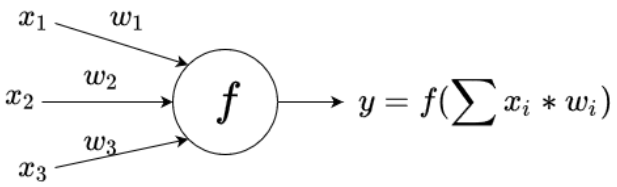
Neuronová síť je výpočetní model, který se částečně inspiruje fungováním biologických neuronů tvořících mozky živočichů. Síť se skládá z množiny různým způsobem pospojovaných uzlů označovaných jako neurony. Je rozdělena na skupinu vstupních neuronů, které své hodnoty na vstupy získávají zvenčí, vnitřních neuronů (také označovaných jako skryté), které počítají, a nakonec skupinu výstupních neuronů, jejichž hodnoty tvoří výsledek výpočtu sítě [Obr. 4].



Obr. 4: Neuronová síť. Modře jsou vyznačeny vstupní neurony, šedě vnitřní a zeleně výstupní neurony. Uspořádání vnitřních neuronů může být různé, závisí na typu sítě. Na obrázku je feedforward síť, kde jsou vnitřní neurony rozděleny do vrstev, přičemž spoje jsou povoleny jen ve směru od předchozí vrstvy do následující.

Neuron má několik vstupních spojů (synapsí), hodnoty na nich jsou přenásobeny vahou daného spoje a poté sečteny. Na získanou hodnotu neuron aplikuje svou

vnitřní funkci. Obvyklými funkcemi jsou sigmoid, tanh nebo ReLU. Je ale možné použít v podstatě jakoukoliv funkci – konkrétně oblast evoluce virtuálních bytostí využívá velmi pestrou paletu vnitřních funkcí, které proberu v kapitole 4. Výsledek funkce pak neuron pošle na svůj výstup. [Obr. 5]



Obr. 5: Neuron. Neuron obsahuje funkci f a vedou do něj synapse s vahami w_i. Hodnoty x_i jsou buď výsledky jiných neuronů, nebo vstupní hodnoty sítě (pokud by se tento neuron nacházel ve vstupní vrstvě sítě). Vstupní hodnoty jsou přenásobeny vahami synapsí a sečteny. Na tuto hodnotu neuron aplikuje svou funkci a výsledek (hodnotu y) pošle na svůj výstup.

1.4 Neuroevoluce

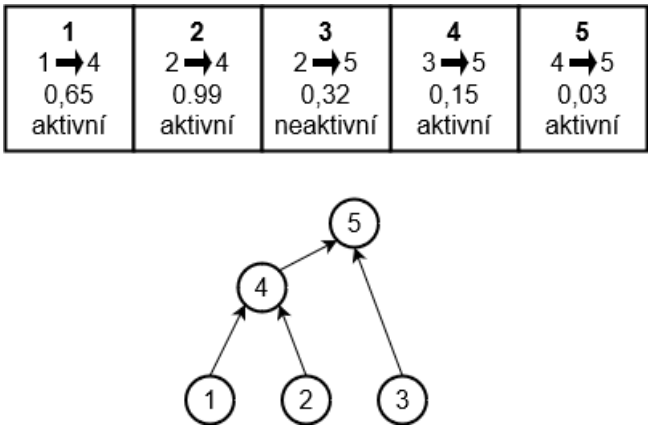
Neuroevoluce se zabývá evolucí, kde jedinci jsou neuronové sítě. Síť je při ní postupně upravována, aby dávala co nejlepší výsledky. Upravují se hodnoty vah na synapsích mezi neurony, je ale možné měnit i strukturu celé sítě (přidávat a ubírat neurony, přepojovat synapse). V případě, že máme pevně danou strukturu sítě a evolvovat chceme pouze hodnoty vah, stačí použít obyčejný genetický algoritmus. Pro evolvování struktury sítě je potřeba složitější algoritmus, který představím v následující kapitole.

1.4.1 NEAT

Neuroevolution of Augmented Topologies (NEAT) (Stanley a Miikkulainen, 2002) je příkladem algoritmu, který umožňuje upravovat váhy synapsí i strukturu neuronové sítě. Používá tři hlavní koncepty, kterými jsou historické značky, rozdělení do druhů a inkrementální růst z minimální struktury.

Jedinec je zde reprezentován seznamem uzlů (neuronů) a seznamem spojů (synapsí). Každá synapse má informace o tom, jestli je aktivní, jakou má váhu, z kterého neuronu vede a kam. Navíc má ještě historickou značku, která je každé synapsi přidělena při jejím vzniku a v průběhu algoritmu se nemění [Obr. 6].

Na začátku evoluce je struktura minimální, tj. vstupní neurony jsou napojeny přímo na neurony výstupní.

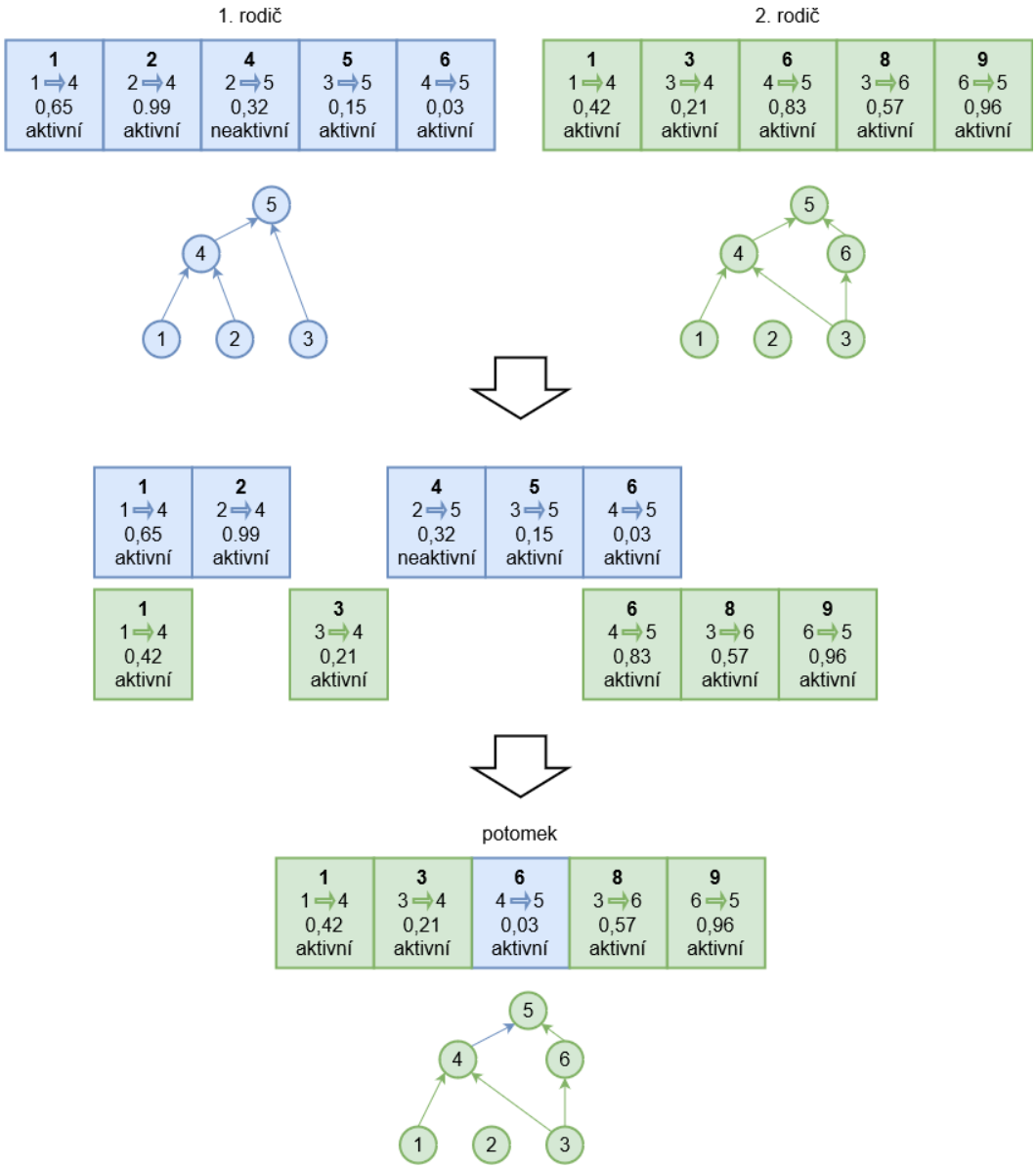


Obr. 6: Reprezentace sítě v NEATu. Síť je reprezentována seznamem synapsí. Každá synapse má historickou značku, označení odkud kam vede, hodnotu váhy a informaci o tom, jestli je aktivní. Obrázek vytvořen na základě původního článku o NEATu (Stanley a Miikkulainen, 2002).

Mutace probíhají jednoduše náhodnou změnou váhy, přidáním nové synapse nebo přidáním nového neuronu. Historická značka mutaci nepodléhá.

Křížení dvou sítí s odlišnou strukturou je komplikovanější záležitost. NEAT zde využije zmiňované historické značky a rozdělí podle nich synapse. Ze synapsí, které se nacházejí v obou rodičích, získá potomek vždy jednu z náhodně vybraného rodiče. Zbylé synapse podědí od rodiče s vyšší fitness [Obr. 7].

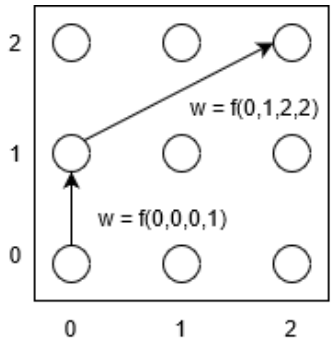
Co se rozdělení do druhů týče, pro každou dvojici genotypů lze spočítat míru podobnosti (využijí se k tomu opět historické značky), na jejímž základně bude populace rozdělena do několika druhů. V rámci jednoho druhu je pak explicitní sdílení fitness – fitness jedince je vydělena velikostí druhu, do kterého patří. Účelem rozdělení do druhů je zachování diverzity, což částečně pomáhá proti zaseknutí se v lokálním optimu.



Obr. 7: Křížení v NEATu. Nahoře jsou dva rodiče, jejichž geny jsou pak uprostřed obrázku zobrazeny tak, aby bylo vidět, které geny spolu oba sdílejí (podle historických značek). Dole je výsledný potomek, který gen se značkou 1 získal náhodně z druhého rodiče, gen číslo 6 náhodně z prvního rodiče a zbylé geny podědil od rodiče s vyšší fitness (na obrázku předpokládáme, že vyšší fitness má druhý rodič). Obrázek vytvořen na základě původního článku o NEATu (Stanley a Miikkulainen, 2002).

1.4.2 HyperNEAT

Algoritmus hyperNEAT (Stanley et al., 2009) je rozšířením NEATu. Struktura sítě je na začátku zafixovaná a neurony jsou umístěny v systému souřadnic. Tato síť je nazývána substrát. Váha synapse mezi dvěma uzly je definována funkcí jejich souřadnic [Obr. 8]. Pro kódování této funkce se v HyperNEATu používá další neuronová síť, které se vytváří pomocí standardního algoritmu NEAT. Tato síť je CPPN (Compositional Pattern Producing Network), což znamená, že povoluje použití velkého množství různých funkcí v neuronech. Tím je možné dosáhnout opakování a symetrií.



Obr. 8: Substrát v HyperNEATu. Neurony jsou umístěny v souřadnicovém systému. Váha synapsí mezi dvěma uzly je funkcí jejich souřadnic. Např. váha synapse na obrázku mezi neurony [0,1] a [2,2] se spočte jako w = f(0,1,2,2).

2 Související práce

V této kapitole představím několik her, které obsahují nějakou formu evoluce, a také představím různé přístupy k evoluci virtuálních bytostí.

2.1 Evoluční algoritmy ve hrách

Existují hry založené na evolučních algoritmech, většinou ale jde o menší hry, které slouží hlavně k vysvětlení fungování těchto algoritmů.

Například Darwin's Avatars je minihra demonstrující výsledky jednoho článku o evoluci virtuálních bytostí (Lessin a Risi, 2015). Dva hráči zde dostanou každý jednu z jejich algoritmem předem vyvinutých bytostí. Jde ale pouze o tělo bez příslušného nervového ovládacího systému. Hráči pak ovládají stahovaní jednotlivých svalů pomocí klávesnice a závodí mezi sebou ve sprintu po rovince.

Hra Evolution[1] se také věnuje evoluci virtuálních bytostí, vyvíjí ale pouze neuronovou síť. Tělo je 2D a hráč ho předem nadesignuje sám, a pak jen sleduje, jak se příšerky postupně vyvíjejí. Právě tato hra má výukové ambice, neboť umožňuje hráči nastavovat parametry algoritmu jako druh selekce, četnost mutací nebo velikost neuronové sítě, vždy s vysvětlením, k čemu daný parametr slouží. Her tohoto typu existuje větší množství, tuto jsem vybrala jako nejreprezentativnější. [Obr. 9]

O využití interaktivní evoluce jakožto hlavní herní mechaniky se pokouší hra Genetic Athletics (Lind a Nihlén, 2017). Hráč zde ručně provádí selekci atletů, nejde tu však o evoluci pohybu ani neuronových sítí, ale jen několika málo parametrů jako výška a váha atleta. Cílem hry je také hlavně seznámit hráče s fungováním jednoduchého genetického algoritmu.

Komplexnější hrou, která využívá evoluce neuronových sítí, je hra Creatures[2]. Hráč zde učí příšerky rozpoznávat předměty, může je učit plnit povely, dávat jim odměny či tresty a nebo křížit příšerky mezi sebou.

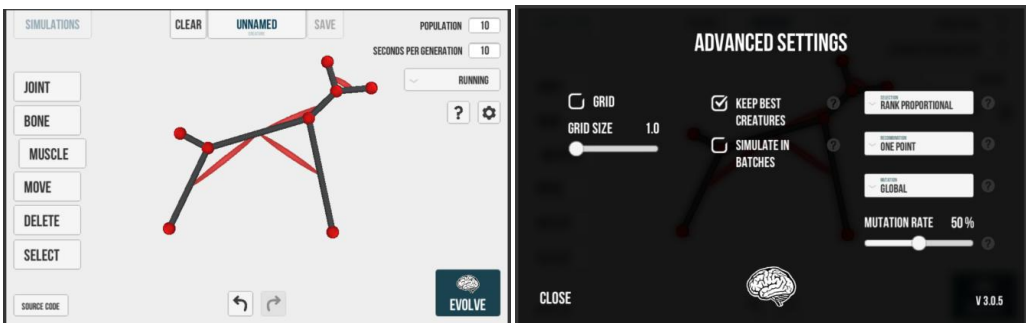
Zajímavým využitím evoluce je hra Project Hastur[3], kde není potřeba nepříjemně dlouho čekat, než se evoluce dopočítá, protože každý level hry odpovídá časově jedné iteraci algoritmu. Jde o hru typu tower defence, kde se nepřátelé po každém dokončeném levelu evolvují. Adaptují se tak na případný hráčův převažující herní

[1] https://keiwando.com/evolution

[2] https://en.wikipedia.org/wiki/Creatures_(video_game_series)

[3] https://polymorphicgames.com

styl. Důsledkem je postupné zvyšování obtížnosti s rostoucím počtem odehraných levelů.



Obr. 9: Hra Evolution. Vlevo je vidět uživatelské rozhraní pro tvorbu těla bytosti. Vpravo obrazovka s možností nastavovat parametry evolučního algoritmu, kliknutím na ikonku otazníku se zobrazí vysvětlení, co daný parametr znamená.

2.2 Evoluce virtuálních bytostí

Za evoluci virtuálních bytostí je považováno využití genetického algoritmu pro vytváření 2D nebo 3D tvorů za účelem evoluce pohybu, přičemž evoluce pro ně hledá tvar těla a nervový ovládací systém zároveň. Průkopníkem v této oblasti byl Karl Sims (Sims, 1994), jehož práci se budu detailněji věnovat v kapitole 2.2.1.

Na jeho práci navázalo mnoho dalších, jako například Krčah, který místo základního genetického algoritmu použil Hierarchický NEAT (Krčah, 2008), nebo také zkoumal možnosti použití novelty searche v oblasti evoluce virtuálních bytostí (Krčah, 2010). Chaumont a kolektiv vyvinuli bytosti s novou schopností, kterou bylo házení bloků do dálky (Chaumont et al., 2007). Nový druh chování přidali také Shim a Kim, kteří se věnovali schopnosti letu (Shim a Kim, 2003). Lessin se zaměřil na vytvoření systému, který by umožňoval kumulaci většího množství různých chování v jedné bytosti (Lessin et al., 2013).

Když se píše o praktickém využitím evoluce virtuálních bytostí, nejčastěji je zmiňována oblast robotiky. Použitelná by tedy byla pro automatický design tvaru robotů, a to jak těch složených z pevných stavebních bloků, čemuž se věnoval například Hornby (Hornby a Pollack, 2001), tak třeba i soft-body robotů, jejichž těla by byla reprezentována voxelovou krychlí jako v práci Chenneyho a kolektivu (Cheney et al., 2014). Mým cílem je prozkoumat možnosti využití evoluce virtuálních bytostí v oblasti počítačových her.

Většina prací zabývajících se evolucí virtuálních bytostí navazuje na původní článek Karla Simse a různými způsoby jeho myšlenky rozšiřuje. V následující kapitole tedy popisuji, jak tato práce, která i mně sloužila jako základ, funguje.

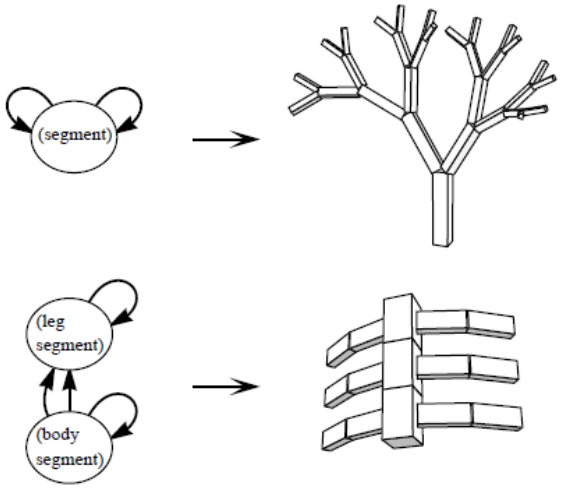
Následně zmiňuji ještě práci Petera Krčaha, ze které jsem využila jeho myšlenku Hierarchického NEATu.

2.2.1 Základní myšlenky

Nejprve popíši reprezentaci jedince tak, jak ji používá Sims (Sims, 1994). Začínám reprezentací tvaru těla, a pak popisuji vnořenou reprezentaci nervového systému. Poté popíši úkoly, které mají bytosti plnit, a s tím související fitness funkce. Na závěr ukáži, jaké parametry evolučního algoritmu Sims používal včetně používaných genetických operátorů.

2.2.1.1 Reprezentace jedince

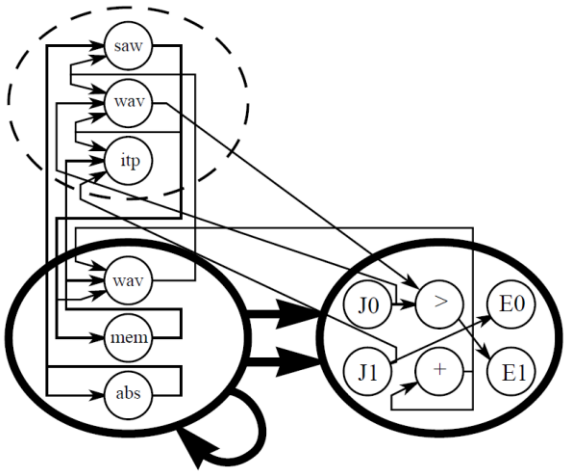
Genotyp popisující tělo bytosti je orientovaný graf. Může osahovat cykly a násobné hrany. Jedním z parametrů uložených v uzlech je rekurzivní limit. Díky němu je pak možné získat z této reprezentace fenotyp ve formě stromu – průchodem grafu od kořenového uzlu do hloubky, přičemž počet návštěv každého uzlu je omezen jeho rekurzivním limitem [Obr. 10]. Vznikne tak fyzická reprezentace jedince tvořená kostkami propojenými pomocí kloubů. Dalšími parametry uzlu jsou rozměry kostky, kterou reprezentuje, kloub a jeho limity a lokální nervový systém. Hrany mají také parametry, které určují konkrétní místo připojení následujícího uzlu.



Obr. 10: Překlad genotypu na fenotyp. Vlevo genotyp, vpravo příslušný fenotyp. Na spodním obrázku začíná průchod grafem v body segmentu, rekurzivní limit body segmentu je 3 a rekurzivní limit leg segmentu 2. (obrázek převzat z původního Simsova článku (Sims, 1994))

Lokální nervový systém se skládá ze senzorů, efektorů a vnitřních neuronů, které jsou mezi sebou propojeny opět do podoby orientovaného grafu. Senzory měří aktuální natočení kloubů, dále jsou k dispozici dotykové senzory a fotosenzory

(slouží k detekci polohy cíle, který je v Simsově práci prezentován jako zdroj světla). Vnitřní neurony mají několik vstupních spojů, počítají různé funkce (sum, max, sin, sigmoid, oscillate-wave,…) a výsledek pošlou na své výstupní spoje. Použité funkce detailněji popíši v kapitole 4. Spoje mají jako parametr váhu, kterou je každá hodnota, která tudy projde, přenásobena. Efektory ze sítě získají číselnou hodnotu, kterou aplikují na kloub, čímž změní jeho natočení. Spoje nemusí existovat jen v rámci jednoho lokálního nervového systému, mohou propojovat i neurony ze dvou sousedních částí těla. Navíc mohou existovat neurony, které nejsou součástí žádné konkrétní části těla – globální část mozku. Genotypem mozku bytosti je tedy graf vnořený do grafu reprezentujícího její tělo [Obr. 11].



Obr. 11 Genotyp jedince. Tělo jedince je reprezentováno grafem, který je vyznačen tlustou čarou. Uzel vlevo je kořenový. Tenkou čarou je zobrazen vnořený nervový systém. Obsahuje lokální senzory (na obrázku J0 a J1), efektory (E0, E1), vnitřní neurony, které jsou součástí lokálního mozku nějaké části těla (wav, mem, abs, >, +), a neurony globální části mozku označené přerušovanou čarou (saw, wav, itp). Kořenový uzel nemá senzory ani efektory. Synapse mohou vést i mezi dvěma sousedními lokálními mozky. Obrázek převzat z původního Simsova článku (Sims, 1994).

2.2.1.2 Fitness

Bytost je simulována 10 sekund virtuálního času v nějakém prostředí a je jí přiřazena fitness. Prostředí a fitness závisí na typu požadovaného chování. Simsova práce řeší pohyb po rovině, pohyb ve vodě a pohyb v jednom z těchto prostředí za polohu měnícím cílem. V případě pohybu za cílem v jedné generaci průměruje výsledky několika běhů, pokaždé s cílem na jiném náhodném místě.

Ještě před simulací je provedeno několik kontrol korektnosti, aby se neplýtvalo výpočetními prostředky na simulaci bytostí, které porušují pravidla – jsou příliš velké, jejich části těla se navzájem protínají nebo zneužívají nějaké chyby ve fyzikální simulaci. Takovéto bytosti jsou zmraženy na startu a je jim přidělena nulová fitness.

2.2.1.3 Parametry evoluce

Co se průběhu evolučního algoritmu týče, Sims začíná s populací náhodně generovaných bytostí. V každé generaci vybere 1/5 nejlepších, kterým dovolí se reprodukovat. Jejich potomci vyplní zbývající 4/5 nové generace. Způsoby tvorby potomků zahrnují mutace grafů – je možné s různými pravděpodobnostmi přičíst k číselným parametrům nějakého uzlu nebo hrany náhodné číslo z normálního rozdělení, přidat a zapojit nový uzel, přepojit hranu jinam, přidat nebo ubrat hranu. Dalším způsobem je křížení grafů – Sims zde navrhuje dvoubodové křížení a grafting. Ty zde ale nebudu detailněji popisovat, protože ve své implementaci použiji křížení navržené v kapitole o Hierarchickém NEATu.

2.2.2 Hierarchický NEAT

Peter Krčah (Krčah, 2008) používá ve své práci stejnou reprezentaci genotypu i stejná prostředí. Pro evoluci však používá Hierarchický NEAT – algoritmus založený na NEATu upravený pro práci na hierarchické struktuře, jakou jsou dva vnořené grafy reprezentující genotyp virtuálních bytostí.

Cílem NEATu je evolvovat strukturu neuronové sítě za použití tří hlavních konceptů, kterými jsou historické značky, rozdělení do druhů a inkrementální růst z minimální struktury. Hierarchický NEAT to využije pro evoluci neurálního i morfologického grafu.

Křížení proběhne nejprve na úrovni morfologického grafu, kde jsou porovnány jednotlivé uzly podle historické značky. Na parametrech dvojice uzlů se stejnou značkou se provede uniformní křížení – potomek získá kopii tohoto uzlu, číselné parametry budou nakombinovány každý z náhodně vybraného rodiče. Toto uniformní křížení se ale netýká parametru lokální mozek, který není číslo. Lokální mozek je neuronová síť a na ní bude použit opět NEAT, potomek si tedy do své kopie uzlu dá jako lokální mozek výsledek křížení těchto dvou neuronových sítí.

Tento způsob křížení se prokázal jako výhodnější, než Simsovo dvoubodové křížení či grafting, které působily spíše jako velké mutace a často vytvořili jedince, který nebyl korektní. Navíc na rozdíl od Simse, který začíná s populací náhodně vygenerovaných genotypů, Krčah využívá poznatky NEATu a začíná s minimální strukturou, aby na začátku nepřekážely neopodstatněné prvky, které evoluci spíše škodí.

3 Návrh hry

Mým cílem je využít evoluci virtuálních bytostí a jejich pohybu jako hlavní herní mechaniku. Měla by tedy být ve hře jasně viditelná a hráč by ji měl mít nějakým způsobem pod kontrolou. Půjde proto o hru o evoluci – podobně jako v Genetic Athletics nebo Evolution bude úkolem hráče vyvinout bytost tak, aby měla určité schopnosti, v tomto případě schopnosti pohybové.

Jak ukázaly předchozí práce, je možné vyvinout bytosti tak, aby uměly následovat polohu měnící cíl. Tím se otevírá možnost, jak hráč může vytvořenou bytost ovládat – kliknutím na nějaké místo na mapě se toto místo stane novým cílem a hráč takto může bytost navigovat terénem. Ve hře proto budou jakési závody v orientačním běhu. Jeden level bude obsahovat mapu s různými překážkami a několika checkpointy. Checkpointy bude potřeba všechny navštívit, ale půjde to provést v libovolném pořadí. Úkolem hráče bude vymyslet cestu, která bude co nejkratší, ale zároveň schůdná pro jeho bytost. Rozšířením, které by přidalo na zábavnosti, by bylo přidání multiplayeru, kde by mohlo více hráčů společně závodit, každý se svou vlastní bytostí.

Aby však levely mohly být zajímavé, je potřeba, aby bytosti uměly překonávat různé druhy překážek. Předchozí práce se zabývaly pouze pohybem po dokonalé rovině. Bude proto potřeba systém rozšířit o evoluci pohybu ve složitějších prostředích. Toto rozšíření proberu v kapitole 4.3.

Aby nešlo pouze o využití výsledků evoluce (jako ve hře Darwin's Avatars), ale opravdu o použití evoluce jako herní mechaniky, hodilo by se, aby hráč mohl evoluci nějak ovlivňovat. Před jejím začátkem proto bude moct vytvořit či vybrat prostředí, ve kterém se bytosti budou učit chodit. K tomu bude potřeba implementovat editor map, který je popsán v kapitole 4.5. Hráč také bude mít možnost ručně vytvořit tělo bytosti a evolvovat pak jen její mozek. Implementuji tedy ještě editor těl popsaný v kapitole 4.5. Sama evoluce by pak mohla být interaktivní – hráč by mohl hodnotit výkony bytostí, vybírat bytosti ke křížení nebo je mutovat. Implementace interaktivní evoluce je problematická, konkrétní překážky probereme v kapitole 4.4. Výslednou bytost pak bude možné pojmenovat, nějak vizuálně upravit, ale hlavně použít v závodech, kde se ukáže, jak dobře vyvinuté jsou její schopnosti pohybu.

4 Implementace

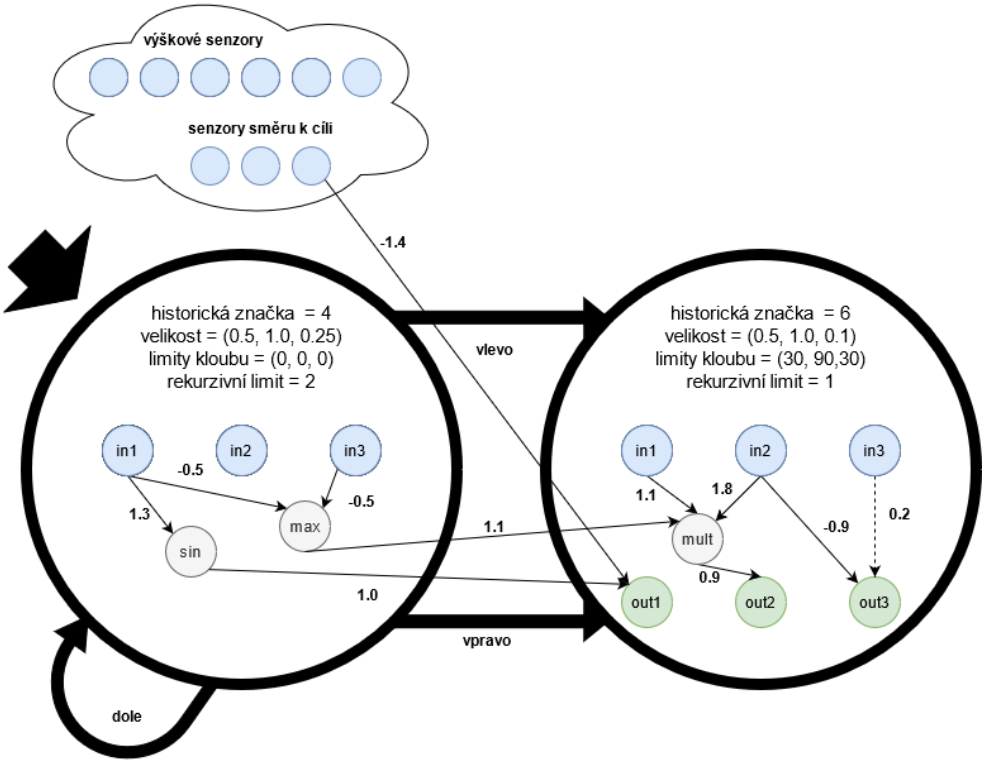
Moje implementace systému pro evoluci virtuálních bytostí je založena na již zmíněných pracích. Používám stejnou reprezentaci genotypu, jako je ta pospaná v kapitole 2.2.1, jedinci jsou simulováni pomocí physics enginu uvnitř Unity a pro účely evoluce používám výše popsaný Hierarchický NEAT.

Rozdílem je, že povoluji připojení maximálně jednoho kloubu na každou stěnu kostky. Tím se zjednoduší kontroly korektnosti, které hledají vzájemně se protínající části těl. Trochu to omezí prostor možných řešení, protože některá řešení s více spoji na jedné stěně by byla korektní a možná i úspěšná. Převážná část by ale vytvořila protínající se části těla, které způsobují problémy v průběhu fyzikální simulace. Také globální část mozku používám výhradně pro globální senzory, synapse tedy mohou vést pouze směrem z globálního mozku ven [Obr. 12]. Jako funkce vnitřních neuronů jsem implementovala atan, sin, cos, sigmoid, abs, sign, memory, min, max, oscillate-saw, oscillate-wave, sum, multiply a difference [Tabulka 1].

Po dokončení základní implementace jsem systém rozšířila pro účely pohybu ve složitějších terénech a pro urychlení startu evoluce jsem přidala možnost použít novelty search, jak je detailněji popsáno v následujících kapitolách.

abs(x)	$\|x\|$	cos(x)	$cos(2\pi x)$
sum(x,y)	$x + y$	sin(x)	$sin(2\pi x)$
multiply(x,y)	$x * y$	atan(x)	$atan(x)$
sign(x)	$sgn(x)$	oscillate-wave(x)	$time \mathrel{+}= 1$ $return\ cos(2\pi * time * 0.04 + x)$
sigmoid(x)	$\frac{1}{1+e^{-x}}$	oscillate-saw(x)	$time \mathrel{+}= 1$ $h = time * 0.04 + x$ $return\ h - \lfloor h \rfloor$
max(x,y)	$max(x, y)$	memory(x)	x_{t-30}
min(x,y)	$min(x, y)$	difference(x)	$x_t - x_{t-1}$

Tabulka 1: Přehled neuronových funkcí. $\boldsymbol{x}$ či $\boldsymbol{x_t}$ značí hodnotu na vstupu v aktuálním čase, $\boldsymbol{x_{t-1}}$ hodnotu v předchozím kroku simulace. Memory si udržuje posledních 30 těchto hodnot a vrací tu nejstarší. Oba oscilátory si nasčítávají počet uběhlých kroků simulace do proměnné ***time***, hodnotu na vstupu používají jako offset generované vlny.



Obr. 12: Příklad genotypu jedince v mé implementaci. Tělo jedince je reprezentované grafem, který je vyznačen tlustou čarou. Tenkou čarou je zobrazen vnořený nervový systém. Levý uzel je kořenový, proto nemá kloub a tudíž ani výstupní neurony. Senzory odpovídají vstupním neuronům a jsou zobrazeny modře, efektory zeleně, vnitřní neurony šedě. Každá nekořenová část těla má tři vstupy i výstupy, protože odpovídají stupňům volnosti kloubu. Vnitřní neurony mají vlastní historické značky a také nějakou funkci (na obrázku sin, max, *). Synapse mají váhu z intervalu -2 až 2 a mohou být neaktivní (přerušovaná čára). Od Simsovy reprezentace se liší tím, že globální část mozku obsahuje pouze globální senzory a parametry spojů jsou omezeny pouze na hodnoty vlevo, vpravo, nahoře, dole, vepředu, vzadu, s tím že ke každé stěně smí být připojena pouze jedna kostka.

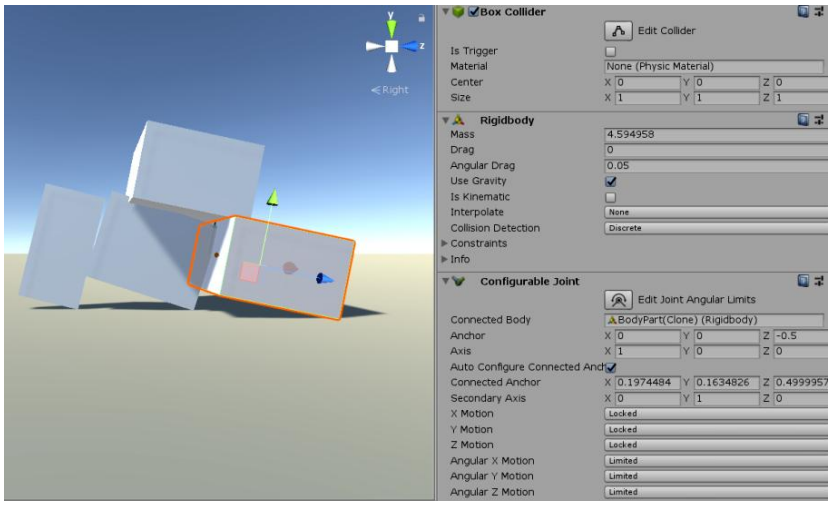
4.1 Unity

Pro implementaci jsem se rozhodla použít herní engine Unity[4]. Jednak protože cílem bylo vytvořit počítačovou hru, a také protože součástí Unity je Nvidia PhysX engine, který mi usnadní provádění fyzikální simulace nutné k ohodnocování výkonů virtuálních bytostí.

Části těla ve fenotypu jsou tedy kostky s komponentou BoxCollider, která detekuje kolize, a dále komponentou Rigidbody, která umožňuje ovlivňovat pohyb objektu působením fyzikálních sil. Jednotlivé kostky jsou pak propojeny pomocí komponenty ConfigurableJoint, která reprezentuje kloub a umožňuje libovolně

[4] www.unity.com

nastavovat jeho parametry. Kolize mezi přímo sousedícími kostkami jsou zakázány, místo toho se aplikují limity příslušného kloubu [Obr. 13].



Obr. 13: Reprezentace fenotypu v Unity. Tělo jedince je složeno z kostek s komponentami BoxCollider, Rigidbody a ConfigurableJoint. Velikost kostky a parametry komponenty ConfigurableJoint jsou dány genotypem bytosti.

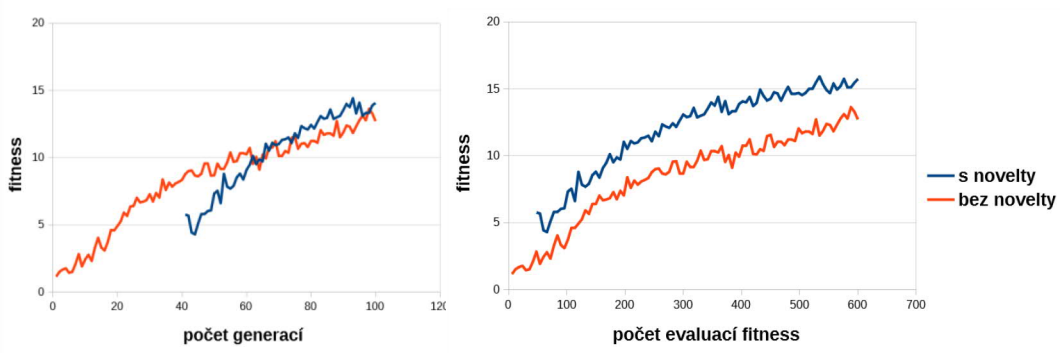
4.2 Novelty search

Při vyvíjení chůze za cílem jsem narazila na zajímavý problém. Stávalo se, že prvních několik generací bytosti jen ležely na startu a měly víceméně nulovou průměrnou fitness. V některých případech se nakonec našel jedinec, který to „odstartoval“, ale jindy se fitness prostě dál nezlepšovala a bylo potřeba evoluci restartovat. Jedná se o takzvaný bootstrap problem – úkol je příliš náročný na to, aby fitness funkce sloužila jako dobrý selekční tlak v raných generacích (Gomez a Miikkulainen, 1997).

Je vhodné proti tomuto problému bojovat metodami zachování diverzity (Silva et al., 2016). Částečně tedy pomáhá rozdělení populace do druhů, které je součástí Hierarchického NEATu. Další známou metodou je novelty search. Jeho hlavní myšlenkou je nahrazení maximalizace fitness maximalizací „novosti“ nalezeného řešení vůči nějakému archivu již prozkoumaných řešení.

Použila jsem tedy novelty search s tím, že jako míru novosti beru průměrnou vzdálenost bytosti na konci simulace od všech ostatních bodů v prostoru, kam kdy nějaká bytost došla. Vzhledem k tomu, že cílem je překonat bootstrap problem, používám novelty search pouze po dobu několika generací na začátku evoluce, a poté přepnu na standardní maximalizování fitness. Důvodem proč používat novelty search po celou dobu evoluce by byla situace, kdy by fitness funkce byla zavádějící (Krčah, 2010), to však není případ úkolu následování cíle.

Na grafech na obrázku je vidět efekt využití novelty searche [Obr. 14]. Verze, která jej používá v průběhu prvních 40 generací, dosáhne ve sté generaci výsledků srovnatelných s verzí, která všech sto generací maximalizovala fitness [Obr. 14 vlevo]. Rozdíl ale spočívá v tom, že jedné generaci novelty searche stačí jedno vyhodnocení fitness, kdežto maximalizace fitness potřebuje v jedné generaci fitness vyhodnotit šestkrát, protože průměruje výsledky z šesti běhů pokaždé s jinak umístěným cílem. Fyzikální simulace nutná k vyhodnocení fitness je velmi časově náročná. Novelty search tedy umožní dosáhnout dobrých výsledků v kratším čase [Obr. 14 vpravo].

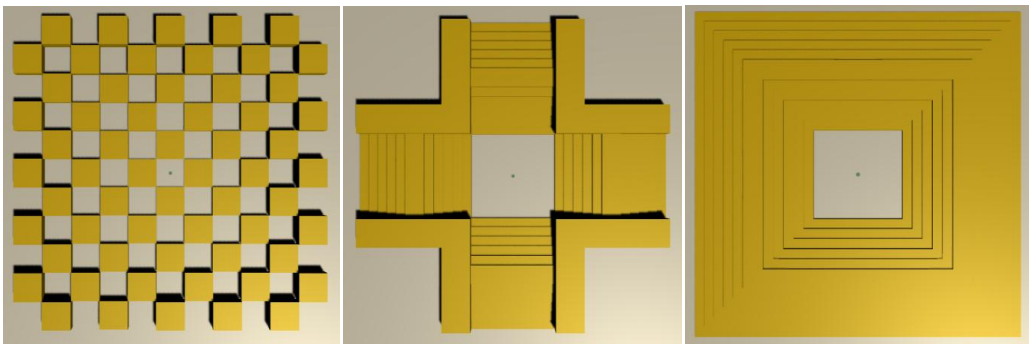


Obr. 14: Výhody novelty searche. Oba grafy ukazují průběh 100 generací algoritmu. Oranžová křivka popisuje variantu, kdy se po celou dobu maximalizovala fitness. Modrá varianta prvních 40 generací používala místo maximalizace fitness novelty search. Z grafu vlevo je vidět, že po 100 generacích dosahují obě varianty srovnatelných výsledků. Graf vpravo ukazuje, že novelty search těchto výsledků dosáhne v kratším reálném čase.

4.3 Složitější prostředí

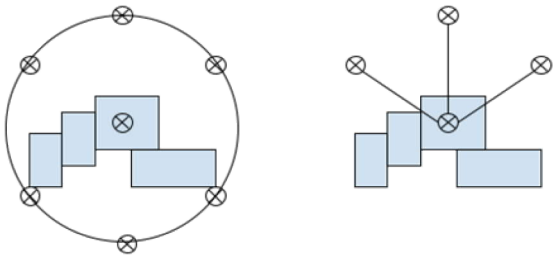
Původní systém evoluce virtuálních bytostí obsahuje jakožto nejsložitější úkol pohyb za cílem po dokonalé rovině. Pro účely využití výsledné bytosti ve hře by se hodilo, kdyby byla schopna naučit se chodit ve složitějších prostředích.

Vyzkoušela jsem tři různá prostředí: rovinu, schodiště (nahoru i dolu) a bloky různých výšek rozmístěné jako šachovnice. Vzhledem k tomu, že při evoluci následování cíle se v každé generaci průměrují výsledky několika pokusů pokaždé s jinak umístěným cílem, je potřeba, aby bylo prostředí co nejvíce symetrické, aby nezvýhodňovalo některé směry víc než jiné. [Obr. 15]



Obr. 15: Náročnější terény. Vlevo bloky různých výšek rozestavěné do šachovnice. Uprostřed první varianta schodů, která byla později nahrazena variantou vpravo kvůli znevýhodňování některých směrů.

Bylo potřeba přidat bytostem nové senzory, dosud totiž vůbec nepotřebovaly vnímat své okolí, stačilo jim pozorovat jen natočení vlastních kloubů. Jednou z možností bylo přidat dotykový senzor do každé končetiny, avšak pro dostatečnou přesnost by bylo potřeba mít na každé stěně každé kostky jeden senzor, což by extrémně zvýšilo komplikovanost sítě a zpomalilo vývoj bytostí (zejména v situaci pohybu po rovině, kdy vnímání překážek vůbec není potřeba). Nakonec jsem se rozhodla přidat šest globálních senzorů, které měří výšku terénu v bezprostředním okolí bytosti [Obr. 16 vlevo]. Původně jsem chtěla použít jen senzor umístěný vepředu a případně k tomu ještě dva po stranách [Obr. 16 vpravo]. Z toho ale sešlo, protože předem není možné určit, který směr si bytost zvolí jako své „dopředu“, navíc ani není nutné, aby jeden takový permanentní směr dopředu existoval. Většina bytostí sice fungovala tak, že se nejprve otočila směrem k cíli a pak se k němu vydala, vznikaly ale i bytosti, které se příliš neotáčely a místo toho se pohybovaly do stran trochu jako krab. U nich by takovýto senzor nedával smysl.



Obr. 16: Rozmístění senzorů. Vlevo varianta s rovnoměrně rozmístěnými senzory. Vpravo varianta s hlavním senzorem vepředu a dvěma dalšími po stranách.

Bytosti jsou nyní schopny chodit po různých terénech. Problémem je, že bychom chtěli, aby jedna bytost zvládala více různých terénů zároveň. Chceme, aby měla k dispozici chůzi po rovině, do schodů, přes překážky…, a aby mezi těmito různými

chováními byla schopna smysluplně přepínat. Nabízí se použít výslednou bytost z úkolu chození po rovině pro inicializaci úkolu chození do schodů a takto postupně kumulovat všechna potřebná chování. Tím by však docházelo k neustálému přepisování existující struktury a tedy vlastně k zapomínání již naučených úkolů. Rozhodla jsem se proto každý úkol trénovat zvlášť a ukládat do separátní DNA, přičemž po dokončení prvního úkolu zafixuji tvar těla a v dalších úkolech se evolvuje pouze neuronová síť. Všechny neuronové sítě takto vzniklé jsou pak kompatibilní s původním tělem bytosti (mají stejné počty senzorů a efektorů), je tedy možné mít jedno tělo a měnit, který mozek (neuronovou síť) aktuálně používá.

4.4 Interaktivní evoluce

Interaktivní evoluce spočívá v tom, že místo automatického výpočtu fitness hodnotí výkon jednotlivých jedinců člověk. Problémem je v takovém případě velikost populace. Ve hře Genetic Athletics stačilo mít 10 jedinců, ale pro náročný úkol, jako je evoluce pohybu, je potřeba populace až stovek jedinců. Není možné, aby hráč v každé generaci prohlédl stovky výsledků, natož pak aby ještě přiřadil fitness každému z nich. Řešením může být umělé omezení počtu jedinců, ze kterých hráč vybírá. Například je možné spočítat jako obvykle fitness a na jejím základě vyřadit z výběru několik procent nejhorších a pro zachování diverzity zajistit ve výběru alespoň jednoho jedince z každého druhu.

Druhým problémem je virtuální doba trvání evoluce, daná počtem generací. Podle experimentů z kapitoly 5.1 je vidět, že k nalezení funkčních řešení jsou potřeba desítky až stovky generací. Hráč by tedy selekci musel provádět třeba stokrát. Je to velmi repetitivní úkol, takže by rychle přicházela ztráta pozornosti a hlavně by to nebylo zábavné. Je tedy nutné omezit počet hráčových zásahů – např. nechat ho provádět ruční selekci jen jednou za deset generací.

Navíc hodnocení člověka je subjektivní, nekonzistentní a může mít tendenci vybírat spíše vizuálně hezká než funkční řešení. Proto se s interaktivní evolucí setkáme spíše v oblasti evolučního umění (např. Picbreeder (Secretan et al., 2008)). Je náročné najít rovnováhu takovou, aby hráčova subjektivní selekce příliš nenarušila funkčnost řešení a zároveň na něj měla dostatečný vliv. Hráč totiž musí vnímat, že jeho akce mají jasné důsledky, jinak by taková mechanika byla zbytečná - hráč by měl pocit, že je vlastně úplně jedno, na co kliká. V tomto ohledu bych navrhovala

opustit myšlenku interaktivní selekce za účelem tvorby výkonného řešení a ponechala bych ruční selekci pouze pro estetické účely. Nepůjde tím evoluci „rozbít“ a přitom bude mít hráč jasnou vizuální odezvu svých činů. Umožnila bych tedy hráči po dokončení evoluce vybrat z množiny „dobrých“ řešení z poslední generace konkrétní jedince, jejichž DNA bude chtít uložit, případně dovolit inicializovat následující evoluci nějakým z existujících jedinců.

Nyní se dostáváme k tomu nejzávažnějšímu, což je reálná doba běhu evolučního algoritmu. Vysoká výpočetní náročnost simulace totiž způsobuje, že na běžném osobním počítači, na kterém by hra byla spuštěna, může trvat jeden běh evoluce desítky minut až hodiny. To je pro účely využití ve hře zásadní problém. Také proto nejspíš hry jako Darwin’s Avatars vlastně vůbec neobsahují evoluci samotnou, pouze její předem vytvořené výsledky. Hra Evolution se zas omezuje na 2D prostředí, neobsahuje interaktivní evoluci a i tak je pro její dohrání potřeba desítky minut pouze pasivně sledovat dění na monitoru. Z tohoto důvodu se cíl této práce v průběhu změnil z implementace hry spíše na tvorbu prostředí pro pokusy s evolucí virtuálních bytostí.

4.5 Editory

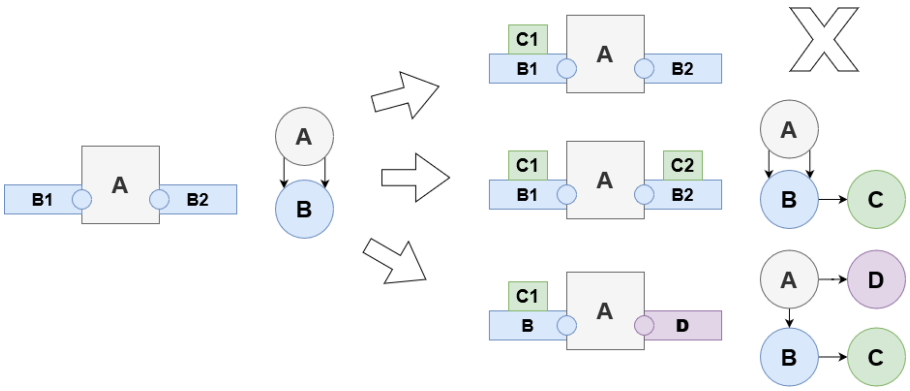
Tvorba počátečních těl bytostí a map terénu probíhá celá před začátkem evoluce, není proto nijak omezena její časovou náročností. Součástí programu je tedy editor map [Obr. 17 vlevo]. Člověk má k dispozici mřížku pevných rozměrů, do které může umisťovat různé překážky (bloky, schodiště, hrbolatý terén). Výsledek lze uložit do souboru (přípona .map). Editorem lze vytvořit terény zmiňované v této práci (rovinu, schodiště do všech směrů, bloky rozmístěné do šachovnice), ale potenciálně mnoho dalších. Do budoucna by šlo přidat další objekty umístitelné do mapy.

Také jsem vytvořila editor těl [Obr. 17 vpravo]. Ten umožňuje nahrát a upravit nějakou bytost (přípona .dna), případně vytvořit novou bytost od nuly. Jde o editor tvaru těla – člověk přidává či ubírá kostky a mění jejich velikosti, neuronová síť nově přidaných kostek bude inicializována jako prázdná. Drobným problémem je, že hráč zde tvoří přímo fenotyp bytosti, ale výsledek jeho práce je potřeba uložit ve formě genotypu. Je zde potřeba překlad v opačném směru, než je obvyklé, a to není vždy možné. Rozhodla jsem se pro řešení zobrazené na obrázku vpravo uprostřed [Obr. 18]. Když člověk umístí kurzor myši na nějaké místo na těle bytosti, zobrazí se

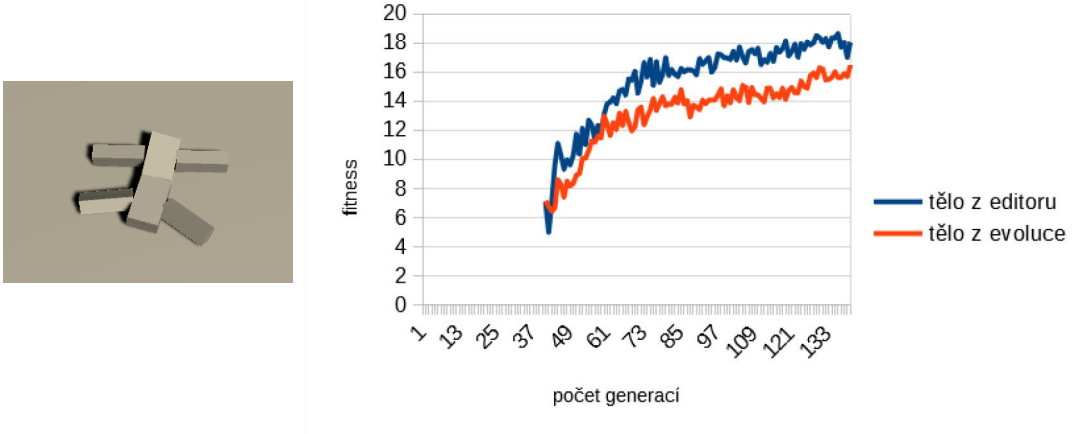
průsvitná kopie kostky na všech místech, kam by se kostka musela nakopírovat. Kliknutím člověk potvrdí, že se na všechna tato místa má kostka skutečně umístit.

Tvorbou těla v editoru je možné evoluci usnadnit, pokud je tvar navržen chytře [Obr. 19]. Při špatném návrhu těla [Obr. 20] je naopak evoluce znesnadněna, protože nezačíná s minimální strukturou, jak doporučuje NEAT, ale s neopodstatněnými prvky, které při pohybu ve výsledku spíš překáží.

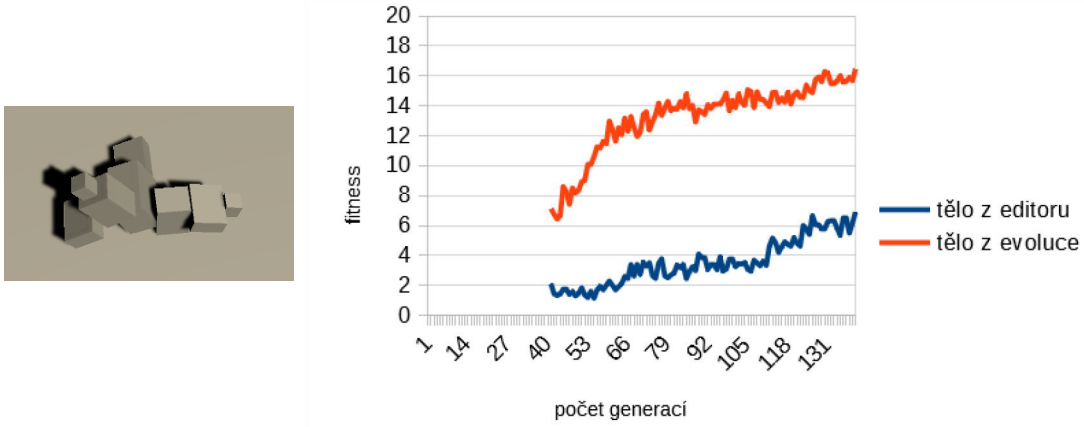
Obr. 17: Editory. Vlevo editor map, vpravo editor těl.



Obr. 18: Překlad fenotypu na genotyp. Fenotyp je označen obdélníky, genotyp kruhy. Obrázek popisuje situaci, kdy chce hráč k tělu vlevo přidat kostku C1. V případě překladu v opačném směru než obvykle, tj. z fenotypu na genotyp, nastává problém. Existují fenotypy, pro které neexistuje genotyp, který by je kódoval (vpravo nahoře). Jedním možným řešením je tvorbu takových fenotypů v editoru zakázat (vynutit nakopírování na všechna potřebná místa jako na obrázku vpravo uprostřed). Nebo lze DNA „rozbalit“ jako na obrázku vpravo dole, čímž získáme možnost tvorby libovolného fenotypu, ale za cenu ztráty opakování a symetrií.



Obr. 19: Úspěšný tvar těla. Při dobrém návrhu těla byla evoluce mírně úspěšnější. Šlo o evoluci pohybu po rovině, 40 generací novelty search a 100 generací maximalizace fitness. Pro tělo z editoru se vyvíjela jen neuronová síť, proto byly zakázány všechny mutace těla kromě mutace limitů kloubů.



Obr. 20: Neúspěšný tvar těla. Příliš komplikovaný návrh těla může průběh evoluce výrazně zhoršit. Šlo o evoluci pohybu po rovině, 40 generací novelty search a 100 generací maximalizace fitness. Pro tělo z editoru se vyvíjela jen neuronová síť, proto byly zakázány všechny mutace těla kromě mutace limitů kloubů.

4.6 Navigování terénem

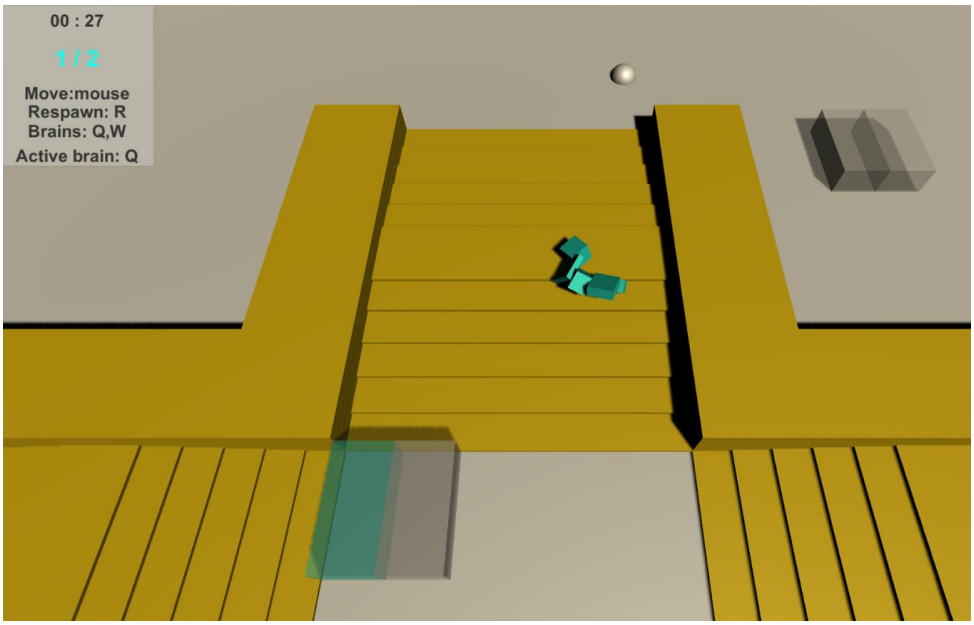
Vytvořila jsem scénu, která kromě možnosti otestovat výsledky evoluce slouží také jako zjednodušený prototyp navrhovaných závodů v orientačním běhu. Lze zde nahrát ze souboru nějakou mapu prostředí a poté nahrát pro každého hráče jednu bytost a k ní (volitelně) jeden mozek navíc [Obr. 21]. Umožňuji zde pro demonstraci přepínat mezi dvěma mozky, bytost jich ale interně může mít libovolně mnoho, pro jejich využití by tedy stačilo přidat další prvky do UI. Nyní lze mezi mozky přepínat klávesami Q a W (pro druhého hráče O a P).

Při nahrání mapy, která obsahuje alespoň jednu překážku typu checkpoint, se po startu zapne časomíra a počítadlo navštívených checkpointů. Checkpointy jsou

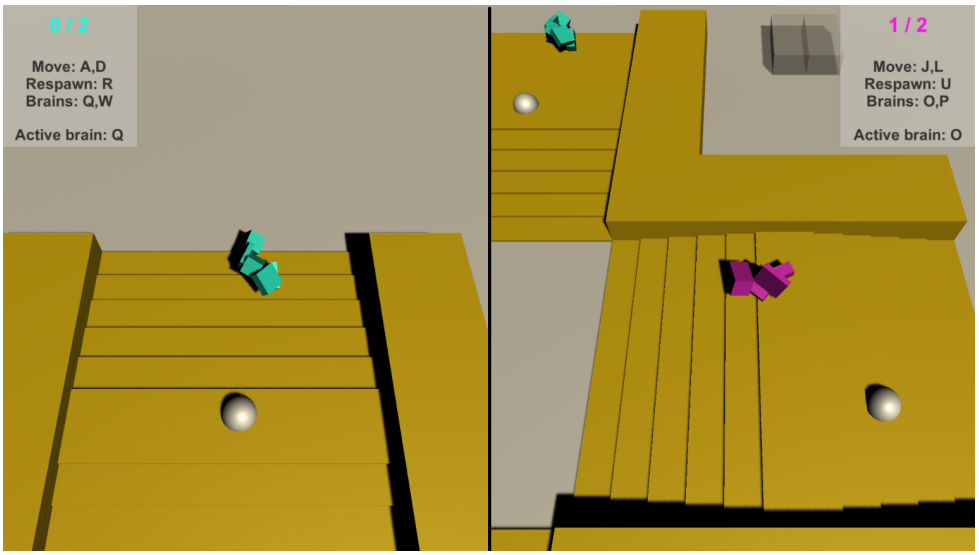
triggery, které detekují kolize s objekty typu bytost. Po doteku se všemi checkpointy v mapě se časomíra zastaví a je konec hry.

Původním plánem bylo ovládat bytost pomocí myši - kliknutím na libovolné místo na terénu se toto místo nastaví jako nový cíl, ke které se bytost bude snažit dojít. Tento způsob navigování je použit v singleplyer módu [Obr. 22]. Chtěla jsem ale umožnit i soupeřit s druhým hráčem ve split screen módu [Obr. 23]. Hráči nemohou mít každý svou myš, proto bylo potřeba vytvořit jiný způsob ovládání. Indikátor cíle se nachází na kružnici okolo bytosti a hráč jej klávesami A a D (pro druhého hráče J a L) otáčí proti či po směru hodinových ručiček. Oba způsoby ovládání jsou si podobné a nezdá se, že by jeden byl jasně lepší, než druhý. Ve split screen módu pro jistotu oba hráči používají k ovládání klávesnici, aby měli stejné podmínky.

Obr. 21: Menu hry. Umožňuje načíst mapu, dvě bytosti a pro každou z nich ještě jeden sekundární mozek.



Obr. 22: Singleplayer mód. V této scéně lze navigovat bytost klikáním myší na místa na mapě a přepínat mezi jejími chováními pomocí kláves Q a W. Cílem je navštívit všechny checkpointy na mapě. Dotykem bytosti se barva checkpointu změní z šedé na barvu hráče.



Obr. 23: Split screen mód. Zde lze navigovat bytosti pomocí klávesnice. Klávesami A a D případně J a L se cíl otáčí proti či po směru hodinových ručiček okolo bytosti.

5 Výsledky

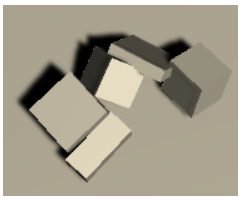
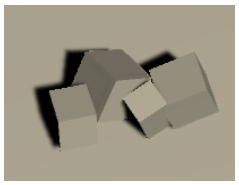
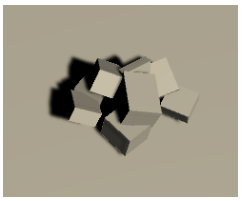
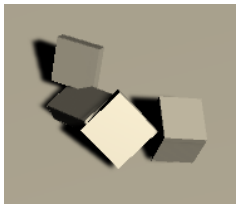
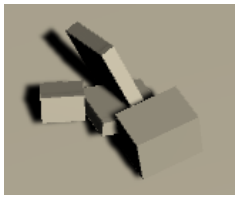
V této kapitole představím výsledky, které vytváří můj systém pro evoluci bytostí, a zhodnotím využitelnost evoluce virtuálních bytostí v navržené hře a v počítačových hrách obecně.

5.1 Výsledky evoluce

Cílem bylo vyvíjet bytosti, které se budou umět pohybovat v různých terénech, a to tak, že budou vždy následovat nějaký cíl – hráčem zvolený bod v prostoru. Popíši zde provedené experimenty, které ukazují schopnosti pohybu bytostí po rovině a srovnám je s výsledky článku Karla Simse. Dále popíši experimenty, které porovnávají výkony v různých složitějších terénech a také demonstrují rozdíly mezi různými nastaveními evoluce.

Všechny experimenty byly provedeny s populací velikosti 100. Cílový bod v prostoru je vždy šestkrát náhodně generován ve vzdálenosti 20 jednotek od startu a fitness je průměr ušlé vzdálenosti správným směrem z těchto šesti pokusů. Maximální dosažitelná fitness je proto 20. Každý pokus je simulován po dobu 20 sekund. Jako složitější terény jsem použila schodiště a šachovnici s překážkami (soubory schody.map a sachy.map v příloze ve složce Mapy). Překážky v šachovnici mají výšku zvětšující se se vzdáleností od startu od 0.1 do 0.3 jednotek. Jeden schod má výšku 0.1 a první schod se nachází 5 jednotek od startu. Pravděpodobnosti jednotlivých druhů mutací a křížení byly nastaveny na hodnoty, které jsou v přiloženém programu nastaveny jako výchozí [Tabulka 3]. Určeny byly ručním laděním na základě předběžných experimentů.

Příklady vyvinutých bytostí jsou uvedeny v následující tabulce [Tabulka 2]. Další tabulky obsahují finální hodnoty fitness v různých nastaveních evoluce [Tabulka 4] [Tabulka 5].

ID	obrázek	Fitness	popis
012		19.0 14.5 13.8	Po stranách kořenové kostky dvě dvoučlánkové končetiny používá k odrážení. Po rovině chodí rychle, do schodů je trochu pomalejší, ale úspěšná. Na šachovnici zvládá překážky do výšky 0.2, na vyšších se zasekne nebo převrátí.
016		16.9 12.7 13.9	Dvě kostky po stranách používá k odrážení. Vpředu má útvar, kvůli kterému se většinou po chvíli převrhne a už nevstane. To vadí při chůzi po schodech. U šachovnice má výhodu velkého těla a zvládá i nejvyšší překážky, bohužel se opět snadno převrhne.
017		9.0 7.0 6.6	Dokáže sledovat cíl. Nízkou fitness má, protože se pohybuje velmi pomalu. Dělá to pomocí drobné oscilace všech svých kostek. Po schodech ani po šachovnici tímto způsobem chodit nezvládá.
018		19.1 17.8 12.9	Třínohá bytost se pohybuje velmi rychle, dosahuje proto vysoké fitness, přestože za cílem většinou neběží nejkratší možnou cestou. Zvládá překážky do výšky 0.2 ale ne moc elegantně.
019		16.8 14.5 13.0	Má jednu krychlovou vpravo a jednu dvoučlánkovou končetinu vlevo, kterými se odráží. Rychlá, ale má problémy s některými relativními natočeními. Jde o velmi nízké tělo, proto je pro ni náročný pohyb na šachovnici.
020		19.0 17.4 17.1	Má zadní nohu jako pohon, krátkou levou nohu pro zatáčení a zdánlivě zbytečnou vrchní ploutev. Otáčí se většinou jen doprava. Překvapivě dobře zvládá chůzi po šachovnici.

Tabulka 2: Příklady vyvinutých bytostí. Fitness uváděna v pořadí chůze po rovině, do schodů, přes šachovnici. Maximální dosažitelná fitness je 20. Výška schodu je 0.1, výška bloků v šachovnici se se vzdáleností od startu postupně zvyšuje z 0.1 na 0.3. Po dobu druhých dvou úkolů měly bytosti zafixovaný tvar těla, aby u nich šlo používat přepínání mozků. Kolonka popis obsahuje subjektivní hodnocení pohybu bytostí na základě testování jejich ovládání pomocí myši. Videa s pohybem těchto bytostí lze podle jejich ID najít ve složce Bytosti.

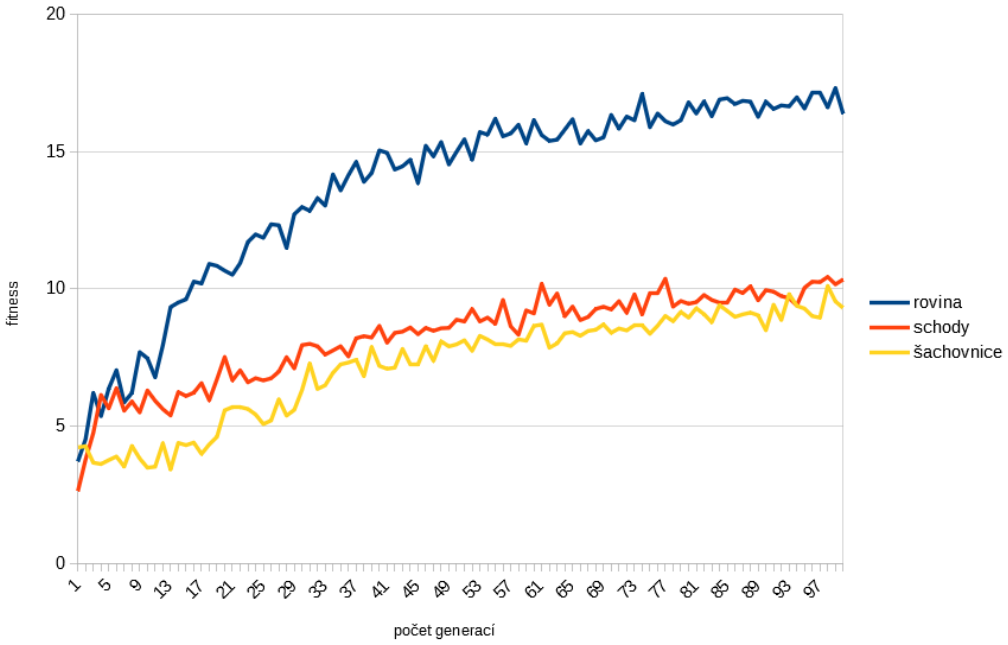
Adjust cube size	0.08	Add synapse	0.20	Crossover probability	0.33
Adjust anchor offset	0.13	Swap function	0.10	Stabilization time	3
Adjust joint limit	0.08	Adjust weight	0.30	Evaluation time	20
Adjust recursive limit	0.25	Add neuron	0.10	Evaluation count	6
Add body node	0.02	Flip enabled	0.05		

Tabulka 3: Nastavení parametrů. Pro všechny experimenty popisované v této kapitole platí nastavení parametrů popsané touto tabulkou. První sloupeček popisuje pravděpodobnosti jednotlivých mutací těla, druhý sloupeček pravděpodobnosti mutací mozku. V posledním sloupečku je nastavitelná pravděpodobnost křížení, čas na stabilizaci před každým hodnocením fitness, délka tohoto hodnocení (v sekundách) a počet pokusů v rámci jedné generace, ze kterých se dělá průměr. Pouze pro test kumulování schopností byly zakázány všechny mutace těla, tj. všechny hodnoty v levém sloupečku tabulky byly nastaveny na nulu. Jména parametrů jsou uvedena tak, jak jsou zobrazena v UI přiloženého programu.

Pro následování cíle po rovině výsledné bytosti dosahují kvalit chování srovnatelných s výsledky původního článku od Simse. Vzniklo mnoho velmi rozdílných bytostí, které různými způsoby zvládají následovat cíl. Při použití novelty searche se všechny bytosti nakonec dokáží pohybovat a velká většina úspěšně sleduje cíl. Stejně jako v Simsově práci má občas některá bytost problém s cílem umístěným v nějakém konkrétním úhlu od jejího aktuálního směru chůze. Špatně například snáší otočky o 180 stupňů nebo některé bytosti preferují raději otočku o 270 stupňů, než aby se otočily o 90 stupňů pro ně problematickým směrem.

Navíc jsem systém rozšířila o evoluci pohybu ve složitějších prostředích. Na následujícím grafu [Obr. 24] je porovnán průměrný výkon v jednotlivých terénech. Ve všech případech začala evoluce 40 generacemi novelty searche, neboť ten se ukázal jako užitečný, viz kapitola 4.2. Na grafu je pak znázorněno následných 100 generací maximalizace fitness.

Pohyb po schodech byl často úspěšný, nízkou fitness má, protože chůze do schodů byla pomalejší než pohyb po rovině. Bytosti by do cíle vzdáleného 20 jednotek došly, nestíhaly to ale v časovém limitu simulace. Úkol chůze po šachovnici byl nejtěžší, překážky výšky 0.3 umístěné ve vzdálenosti přibližně 15 jednotek od startu byly pro bytosti příliš vysoké.



Obr. 24: Průběh evoluce pohybu v různých terénech. 100 generacím znázorněným v tomto grafu předcházelo 40 generací novelty searche.

Bytosti z Tabulka 2 se vyvíjely pro pohyb po rovině a výsledný nejlepší jedinec pak byl použit pro inicializaci vývoje pohybu po schodech či šachovnici. To umožnilo, aby jedna bytost měla více mozků, mezi kterými může přepínat. Také to ale vedlo ke zvýšení průměrné fitness v náročnějších terénech, jak je vidět z porovnání v tabulce [Tabulka 4]. Úkol chůze po rovině je lehčí, tudíž nevyprodukuje jedince, kteří by byli nepoužitelní (průměrně jen jeden případ z deseti nezvládal následování cíle na rovině dostatečně dobře). Bytost tedy má nějaké základní schopnosti pohybu a otáčení se směrem k cíli, následná evoluce to pak může využít a tyto schopnosti už jen přizpůsobovat konkrétnímu terénu. Při startu od nuly v náročnějším terénu se bytost musí učit nejen chůzi po schodech, ale i hledání cíle a otáčení se za ním. To je možná příliš úkolů najednou, dle tabulky byly bytosti úspěšné pouze v polovině případů.

Efekt kumulování schopností je dobře vidět např. na řádce 6 v Tabulka 4. Výsledek vývoje chůze po schodech od nuly zde má fitness 5.6, tedy bytost pouze došla před schodiště. Navázání na chůzi po rovině zde dokázalo využít toho, že na rovině měla tato bytost fitness vysokou (19.1) a dostala se tak na fitness 12.9, která již pro schody značí průměrně dobrý výkon.

	rovina	schody	šachovnice	rovina	rovina +schody	rovina +schody +šachovnice
1	17.6	6.1	5.7	17.6	10.6	10.3
2	12.6	5.4	9.0	12.6	5.2	9.6
3	15.8	6.5	7.8	15.8	13.4	10.2
4	9.6	7.7	4.1	9.6	7.5	9.2
5	16.5	13.3	11.5	16.5	12.4	16.9
6	19.1	5.6	9.1	19.1	12.9	10.5
7	17.2	10.6	10.2	17.2	13.9	11.3
8	18.2	10.3	16.4	18.2	19.3	16.3
9	18.4	17.0	5.3	18.4	14.6	12.6
10	18.8	18.1	13.9	18.8	16.2	12.3
průměr	**16.4**	**10.1**	**9.3**	**16.4**	**12.6**	**11.9**

Tabulka 4: Porovnání terénů. Levá část tabulky ukazuje odděleně vyvíjené schopnosti. S náhodně inicializovanou první generací proběhlo 40 generací novelty searche a 100 maximalizace fitness. Pravá část tabulky popisuje kumulování schopností, kde chůze po rovině byla vyvinuta stejným způsobem jako v prvním případě, výsledek pak byl použit pro inicializaci evoluce pohybu po schodech a její výsledek zas pro inicializaci pohybu po šachovnici, pokaždé proběhlo 100 generací.

Již v kapitole 4.2 jsem ukázala užitečnost novelty searche pro překonání bootstrap problemu. V případě pohybu po rovině jeho použití způsobilo urychlení evoluce, a tudíž zvýšení průměrné fitness, jak ostatně ukazuje i Tabulka 5 v porovnání s průměrnou fitness z levé části předchozí tabulky [Tabulka 4]. Následně jsem zkoušela porovnat jeho efekt i ve složitějších prostředích. Při chůzi do schodů byly úspěšné jen tři bytosti z deseti (fitness okolo hodnoty 6 totiž značí, že bytost pouze došla k prvnímu schodu). Pro šachovnici také mírně přibylo zcela nepoužitelných řešení (viz řádek 1 a 6, Tabulka 5). Dobré běhy bez novelty searche vytvořily obdobně úspěšné bytosti jako bez něj, rozdíl spočívá v tom, že bez novelty searche se zvýšil počet běhů neúspěšných.

	rovina bez novelty	schody bez novelty	šachovnice bez novelty
1	6.7	5.7	2.3
2	18.3	6.1	8.7
3	14.0	15.7	7.4
4	2.7	6.4	7.0
5	18.8	10.4	13.1
6	16.8	4.3	1.8
7	16.5	5.5	9.4
8	15.6	15.6	10.0
9	17.3	7.1	11.9
10	10.3	5.3	13.4
průměr	**13.7**	**8.2**	**7.9**

Tabulka 5: Porovnání terénů s a bez použití novelty searche. Levá část tabulky používá na začátku evoluce 40 generací novelty searche. Pravá část začíná rovnou maximalizací fitness.

Kromě porovnávání fitness je potřeba vzít v úvahu i subjektivní hodnocení vhodnosti výsledků pro použití v navrhované hře. Celkově bylo dosaženo dostatečné kvality následování cíle, aby bylo možné navigovat bytost pomocí myši. Jediným drobným nedostatkem je, že v průběhu evoluce nebyl kladen důraz na to, aby otáčení se za cílem bylo provedeno na místě, některé bytosti tak pro otočku potřebují relativně velký prostor, což může být nepříjemné v prostředí s mnoha překážkami. Jinak ale bytosti reagují celkem rozumně a při jejich ovládání má člověk pocit, že bytost se opravdu „snaží“ jeho příkazy plnit.

5.2 Vyhodnocení využitelnosti ve hrách

Aby evoluce virtuálních bytostí dosáhla dobrých výsledků (obzvlášť ve složitějším prostředí), potřebuje desítky až stovky generací a vyhodnocování fitness spočívá ve výpočetně náročné fyzikální simulaci. Trvá tudíž tak dlouho, že je ve hře těžko použitelná. Přestože je až překvapivě uspokojivé pasivně sledovat vývoj bytostí, rozhodně to člověka nebude bavit desítky minut. Navíc evoluce obsahuje prvky náhody a není vyloučené, že se zasekne v lokálním optimu. Tedy i po dlouhém čekání nemusí být výsledná bytost pro závody zcela použitelná.

Aby hráč mohl zasahovat do průběhu evoluce, musí být populace dostatečně malá, aby si o ní dokázal udržovat přehled, musí mít málo generací, aby hráče nezačala takto repetetivní činnost nudit a zároveň potřebuje i relativně malou reprezentaci jedince, aby změna jednoho parametru měla pro hráče jasně viditelný vliv na výsledek. Interaktivní evoluce se tedy pro náročný úkol, jako je evoluce pohybu virtuálních bytostí, příliš nehodí.

Nechat hráče předem nadesignovat tvar těla a pak sledovat průběh evoluce (jako ve hře Evolution) je možné, problémem se tu ale stává výpočetní náročnost fyzikální simulace pohybu ve 3D prostředí, která trvá příliš dlouho na to, aby hráč celou dobu jen pasivně sledoval dění na obrazovce.

Závody samotné tvoří potenciálně nejzábavnější část hry. Člověk bytost ovlivňuje jen nepřímo - říká jí, kam má jít, ale způsob provedení pohybu závisí na vývoji bytosti. Je zábavné sledovat různá selhání bytosti, kdy se snaží přelézt příliš vysokou překážku, nevytočí zatáčku nebo se převrhne (v tu chvíli působí všechny její pokusy o pohyb obzvlášť směšně). O to větší uspokojení pak pro hráče přijde, když se bytost úspěšně doplazí do cíle. Se špatně vytrénovanými bytostmi to ale může rychle sklouznout spíše do pocitu frustrace. Proto je potřeba nezačínat ve hře s evolucí od nuly, ale buď mít čistě jen předem vytvořené jedince (jako v Darwin's Avatars), nebo alespoň předtrénované jedince, kterými lze další evoluci inicializovat.

Má představa o možnostech využití evoluce virtuálních bytostí ve hře se v průběhu práce postupně měnila a přizpůsobovala nalezeným limitům. Výsledkem práce je nakonec spíše nástroj pro pokusy s evolucí virtuálních bytostí a k tomu zjednodušený prototyp navrhované hry. To ale neznamená, že hry založené na evolučních algoritmech by neměly budoucnost. Většina problémů by odpadla, kdyby bylo možné urychlit průběh evoluce, aby na běžném osobním počítači trvala

maximálně jednotky minut. Přínosem by také mohlo být nalezení způsobu, jak udělat bytosti vizuálně atraktivnější – např. krychlovitou kostru potáhnout elastickou kůží. Pak by nevadilo mít pouze předtrénované kostry a hráč by mohl interaktivně evolvovat estetické prvky, které k vyhodnocení své fitness nevyžadují fyzikální simulaci.

Závěr

Prvním cílem této práce bylo navrhnout počítačovou hru, která by používala evoluci virtuálních bytostí jako herní mechaniku. Návrh takové hry jsem podala v kapitole 3.

Pro jeho implementaci bylo potřeba nejen vytvořit systém pro evoluci virtuálních bytostí, ale také jej rozšířit o prvky, které navrhovaná hra potřebuje - zejména o vývoj pohybu v komplikovanějších terénech. Výsledné bytosti vytvořené implementovaným systémem jsou prezentovány v kapitole 5.1, v téže kapitole jsem také porovnala jejich výkony v různých terénech a při různých nastaveních evolučního algoritmu. Zejména využití novelty searche se ukázalo jako dobrý způsob, jak urychlit začátek evoluce. Vyvinuté bytosti dosahují velmi dobrých výsledků při pohybu po rovině a zvládají i náročnější terény.

Za účelem kumulování více schopností do jedné bytosti jsem bytosti po absolvování prvního úkolu zafixovala tvar těla (viz kapitola 4.3). Je to umělý zásah, který omezuje možnosti evoluce, a proto by bylo zajímavé do budoucna zkusit najít jiné řešení.

Hlavním problém ale je, že evoluce (konkrétně fyzikální simulace nutná pro výpočet fitness) je výpočetně náročná a k dosažení těchto dobrých výsledků je potřeba, aby se bytosti vyvíjely desítky minut až hodiny. To je hlavním důvodem, proč zařazení evoluce pohybu virtuálních bytostí do počítačové hry není zcela vhodné. V kapitole 5.2 jsem detailněji zhodnotila, které aspekty evoluce pohybu virtuálních bytostí jsou pro účely hry použitelné. Vytvořila jsem zjednodušený herní prototyp jakožto jednu ze součástí vytvořeného nástroje pro pokusy s evolucí virtuálních bytostí.

Interaktivní evoluci pohybu ve složitém prostředí bude možné mít jako součást nějaké hry pouze pokud by se v budoucnu podařilo její průběh několikanásobně zrychlit. Šlo by se ale zaměřit na oblast, která už teď funguje dobře a má potenciál být zábavná, a to na využití již hotových výsledků evoluce. Přínosem v této oblasti by mohlo být nalezení způsobu, jak udělat bytosti vizuálně atraktivnější. Pak by hráč mohl interaktivně evolvovat estetické prvky, které k vyhodnocení své fitness nevyžadují fyzikální simulaci.

Bibliografie

EIBEN, A.E. a J.E. SMITH. *Introduction to Evolutionary Computing.* 2. vyd. Heidelberg: Springer, 2015. ISBN 978-3-662-44873-1.

GOMEZ, Faustino a Risto MIIKKULAINEN. Incremental Evolution of Complex General Behavior. *Adaptive Behavior.* 1997, **5**(3-4), 317-324. ISSN 1059-7123.

HORNBY, Gregory S. a Jordan B. POLLACK. Evolving L-systems to generate virtual creatures. *Computers & Graphics.* 2001, **25**(6), 1041-1048. ISSN 00978493.

CHAUMONT, Nicolas, Richard EGLI a Christoph ADAMI. Evolving Virtual Creatures and Catapults. *Artificial Life.* 2007, **13**(2), 139-157. ISSN 1064-5462.

CHENEY, Nicolas, Jeff CLUNE a Hod LIPSON. Evolved Electrophysiological Soft Robots. In: *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. MIT Press, 2014, 222-229. ISBN 9780262326216.

KRČAH, Peter. Toward Efficient Evolutionary Design of Autonomous Robots. In: HORNBY, Gregory S., Lukáš SEKANINA a Pauline C. HADDOW, ed. *Evolvable Systems: From Biology to Hardware*. Heidelberg: Springer, 2008, 153-164. Lecture Notes in Computer Science. ISBN 978-3-540-85856-0.

KRČAH, Peter. Solving deceptive tasks in robot body-brain co-evolution by searching for behavioral novelty. In: *10th International Conference on Intelligent Systems Design and Applications.* IEEE, 2010, 284-289. ISBN 978-1-4244-8134-7.

LEHMAN, Joel a Kenneth O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. *Proceedings of the Eleventh International Conference on Artificial Life.* MIT Press, 2008.

LESSIN, Dan a Sebastian RISI. Darwins Avatars. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation.* New York, USA: ACM, 2015, 329-336. ISBN 9781450334723.

LESSIN, Dan, Don FUSSEL a Risto MIIKKULAINEN. Open-ended behavioral complexity for evolved virtual creatures. *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference - GECCO 13.* New York, USA: ACM Press, 2013. ISBN 9781450319638.

LIND, Sebastian a Karl NIHLÉN. *Interactive evolution as a game mechanic.* Malmö, 2017. Bakalářská práce. Malmö högskola. Vedoucí práce Steve Dahlskog.

RISI, Sebastian a Julian TOGELIUS. Neuroevolution in Games: State of the Art and Open Challenges. *IEEE Transactions on Computational Intelligence and AI in Games.* 2017, **9**(1), 25-41. ISSN 1943-068X.

SECRETAN, Jimmy et al. Picbreeder: Collaborative Interactive Evolution of Images. *Leonardo.* 2008, **41**(1), 98-99. ISSN 0024-094X.

SHIM, Yoon-Sik a KIM Chang-Hun. Generating flying creatures using body-brain co-evolution. In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* Eurographics Association, 2003, 276-285.

SILVA, Fernando et al. Open Issues in Evolutionary Robotics. *Evolutionary Computation.* 2016, **24**(2), 205-236. ISSN 1063-6560.

SIMS, Karl. Evolving virtual creatures. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH 94.* New York, USA: ACM Press, 1994, 15-22. ISBN 0897916670.

STANLEY, Kenneth O. a Risto MIIKKULAINEN. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation.* 2002, **10**(2), 99-127. ISSN 1063-6560.

STANLEY, Kenneth O., David B. D'AMBROSIO a Jason GAUCI. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life.* 2009, **15**(2), 185-212. ISSN 1064-5462.

Obsah elektronické přílohy

Přiložený archiv obsahuje:

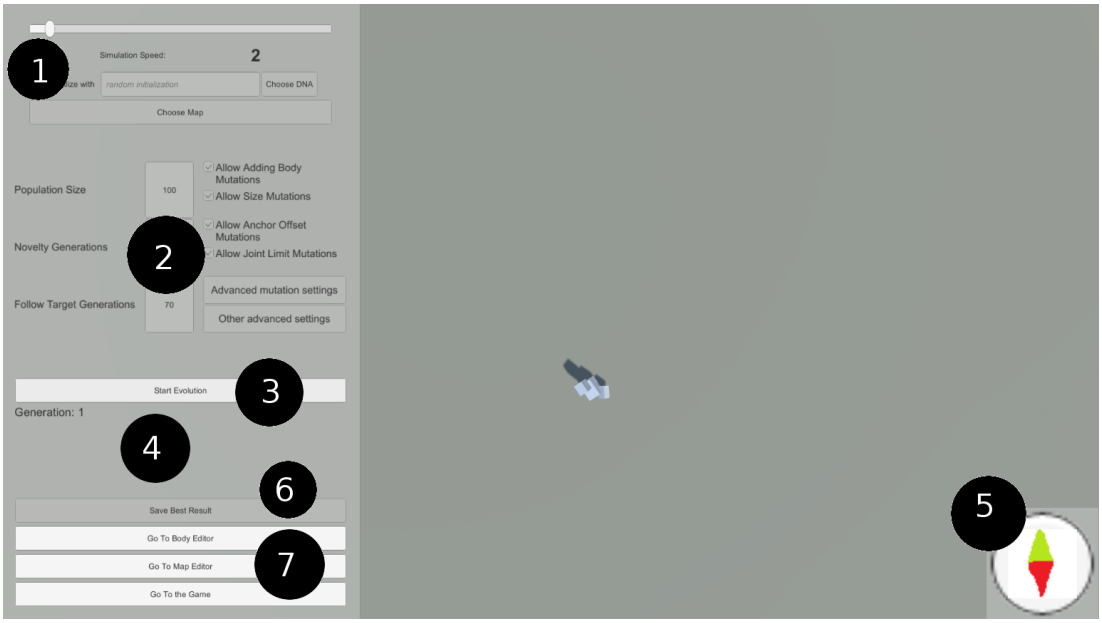
- Složku **Bytosti** obsahující příklady vytvořených bytostí a krátká videa zaznamenávající jejich způsoby pohybu.
- Složku **Dokumentace** s vygenerovanou dokumentací.
- Složku **EVC** se souborem EVC.exe sloužícím ke spuštění aplikace a složkou EVC_Data, která je nastavena jako defaultní adresář pro uživatelem uložené soubory s genotypem bytostí (.dna), uložené soubory s terénem (.map) a ukládají se do ní informace o naměřených hodnotách fitness (.csv).
- Složku **Mapy** obsahující terény zmiňované v textu práce (schody.map, sachy.map) a také několik map pro hru (TheFields.map, ThePit.map, TheTemple.map).
- Složku **Projekt** obsahující Unity projekt, který lze otevřít v Unity editoru (projekt byl vytvářen ve verzi 2019.2.12f1).
- Složku **Texty** obsahující uživatelskou dokumentaci, programátorskou dokumentaci a elektronickou verzi tohoto textu.
- Soubor README.TXT s těmito informacemi o jednotlivých složkách.

Příloha A - Uživatelská dokumentace

Program se spouští pomocí EVC.exe, který se nachází ve složce EVC. Ukončí se stisknutím klávesy Esc v hlavní obrazovce.

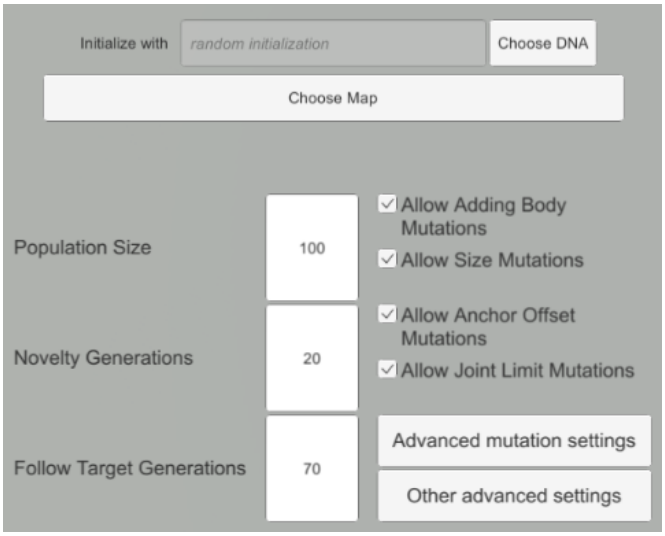
Hlavní obrazovka

Hlavní obrazovka [Obr. 25] slouží ke spouštění evoluce bytostí. Lze předem nastavit různé parametry, v průběhu evoluce je zaznamenávána průběžná fitness do .csv souboru a DNA nejlepšího výsledného jedince lze po skončení evoluce uložit do souboru .dna.



Obr. 25 Hlavní obrazovka. UI obsahuje slider pro určení rychlosti simulace (1), sekci nastavitelných parametrů (2), tlačítko pro spuštění evoluce (3), prostor pro zobrazování informací o průběhu evoluce (4), kompas ukazující k aktuálnímu cíli bytostí (5), tlačítko pro uložení výsledku (6) a možnosti přechodu do dalších scén (7).

Význam jednotlivých nastavitelný parametrů je detailněji popsán v hlavním textu práce a jejich kompletní seznam je uveden v poslední kapitole tohoto textu. Mezi základní nastavení patří velikost populace, počet generací novelty searche a počet následných generací standardního maximalizování fitness. Lze začít s náhodně generovanými jedinci, nebo načíst DNA, kterou bude evoluce inicializována. Dále je možné povolit či zakázat různé druhy mutací a samozřejmě je zde možnost nahrát terén ze souboru .map [Obr. 26].



Obr. 26 Nastavitelné parametry. Kromě nastavení velikosti populace a počtu generací lze povolit či zakázat různé druhy mutací - Adding Body Mutations přidávají nové kostky a mění rekurzivní limity, Size Mutations mění velikosti kostek, Anchor Offset Mutations mění konkrétní polohu připojení kloubů a Joint Limit Mutations mění limity kloubů. Dále lze nahrát terén (Choose Map) a DNA pro inicializaci evoluce (Choose DNA).

Kliknutím na tlačítko Start Evolution se spustí evoluce se zvoleným nastavením, v průběhu evoluce už nastavení nejde měnit, lze pouze průběžně měnit rychlost simulace. Bytosti jsou rozmístěny do čtvercové mřížky, kameru lze posouvat pohybem myši ke kraji obrazovky.

Po skončení evoluce se zpřístupní tlačítko Save Best Result, které uloží DNA nejlepšího jedince z poslední generace a také pod stejným názvem uloží do složky EVC_Data csv soubor s daty o fitness v průběhu generací.

Tlačítka v dolní části obrazovky umožňují přechod do jiných obrazovek, při jejich použití bude jakákoliv probíhající evoluce zrušena. Stisknutím klávesy Esc lze ukončit celý program.

Hra

Zde je možné otestovat schopnosti bytostí v drobné hře, která spočívá v závodech v orientačním běhu. Nejprve je v menu [Obr. 27] potřeba zvolit mapu, na které se bude závodit. Poté je potřeba nahrát jednu bytost pro každého hráče. Také je možné volitelně nahrát další DNA reprezentující sekundární ovládací systém bytosti (například mozek natrénovaný pro nějaké složitější chování jako třeba chůze po schodech). Je nutné, aby vybrané dva mozky byly kompatibilní – tj. aby se opravdu jednalo o mozek pro daný tvar těla. Pokud bude mozek nekompatibilní, nahrávání selže.

Hru lze spustit v singleplayer nebo multiplayer módu. Po kliknutí na příslušné tlačítko Start se na obrazovce objeví nahrané bytosti. Pár vteřin stráví dopadem na podlahu a stabilizací, poté je bude možné navigovat.

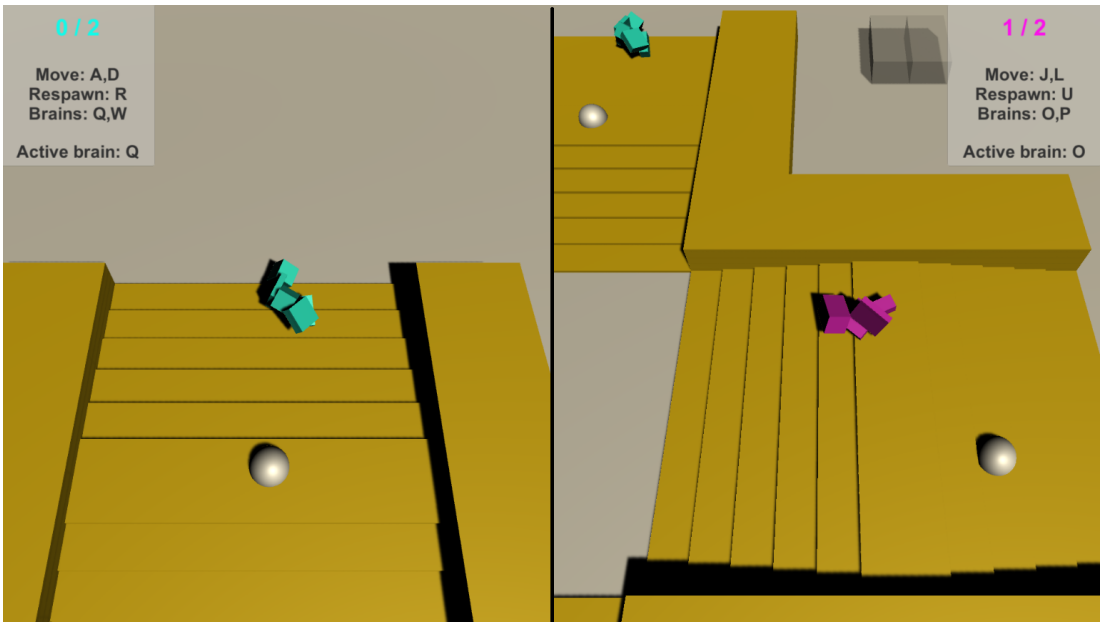
Úkolem bytosti je navštívit všechny checkpointy na mapě (v libovolném pořadí). Na obrazovce je vidět, kolik z celkového počtu checkpointů již bylo navštíveno, a v případě singleplayer módu také kolik uplynulo času od startu. Navštívený checkpoint má barvu hráčovy bytosti, nenavštívený je šedý.

V singleplayer módu se kliknutím na libovolné místo na mapě dané místo nastaví jako nový cíl bytosti, pro hráče se také označí šedou koulí. Mezi mozky bytosti lze přepínat klávesami Q a W. Klávesou R je možné respawnovat bytost zpět na start za cenu časové penalizace 30 sekund. Klávesa Esc pozastaví hru a zobrazí opět menu. Kamera se v této scéně pohybuje automaticky – sleduje pohyb bytosti.

Multiplayer mód je ve formě split screenu – tedy polovina obrazovky pro každého z hráčů [Obr. 28]. Liší se tím, že první hráč ovládá směr chůze klávesami A a D, přepíná mozky klávesami Q a W a respawnuje se pomocí R. Druhý hráč pro ovládání pohybu používá J a L, pro přepínání mozků O a P a pro respawn U. V tomto módu se neměří čas, ale vyhraje ten hráč, který první navštíví všechny checkpointy.

Lze zde například využít přiložené příklady vyvinutých bytostí ze složky Bytosti na terénech ze složky Mapy.

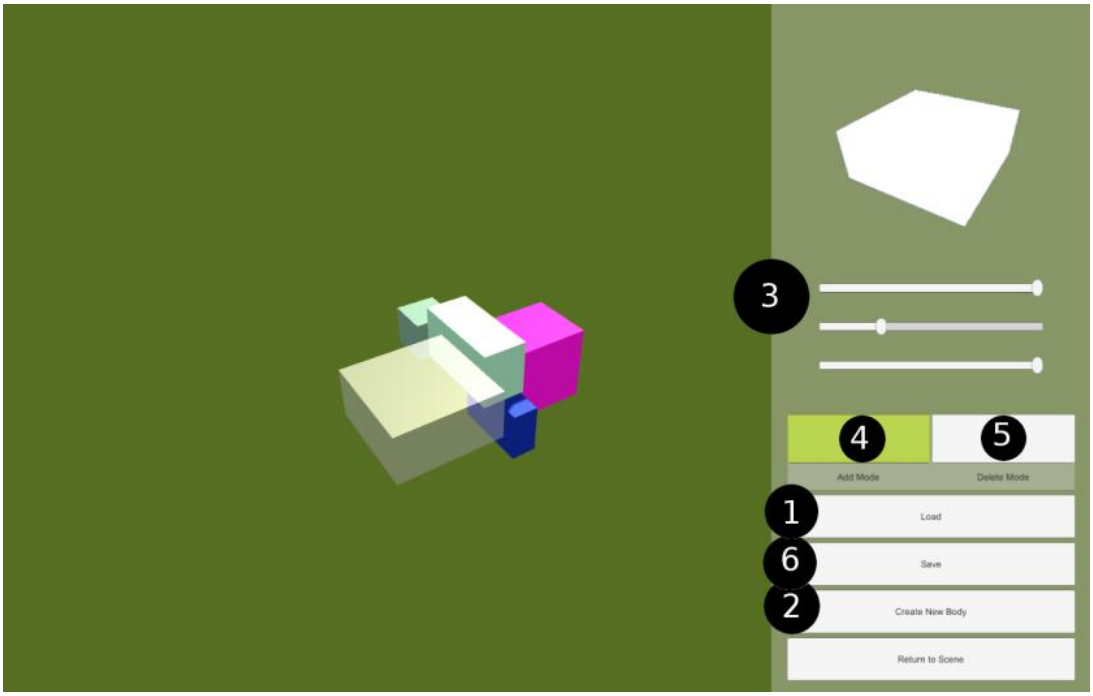
Obr. 27: Herní menu. Zde je možné zvolit mapu, pro každého hráče jednu bytost a k ní volitelně jeden další mozek. Hru pak lze spustit v režimu pro jednoho nebo dva hráče.



Obr. 28: Split screen. Na obrázku je zobrazen průběh hry pro dva hráče. Každý hráč má polovinu obrazovky s kamerou vycentrovanou na jeho bytost. Modrý hráč používá klávesy A, D, R, Q a W, fialový hráč J, L, U, O a P. V rohu obrazovky je vidět, kolik již který hráč navštívil checkpointů (a kolik existuje checkpointů celkem).

Editor těl

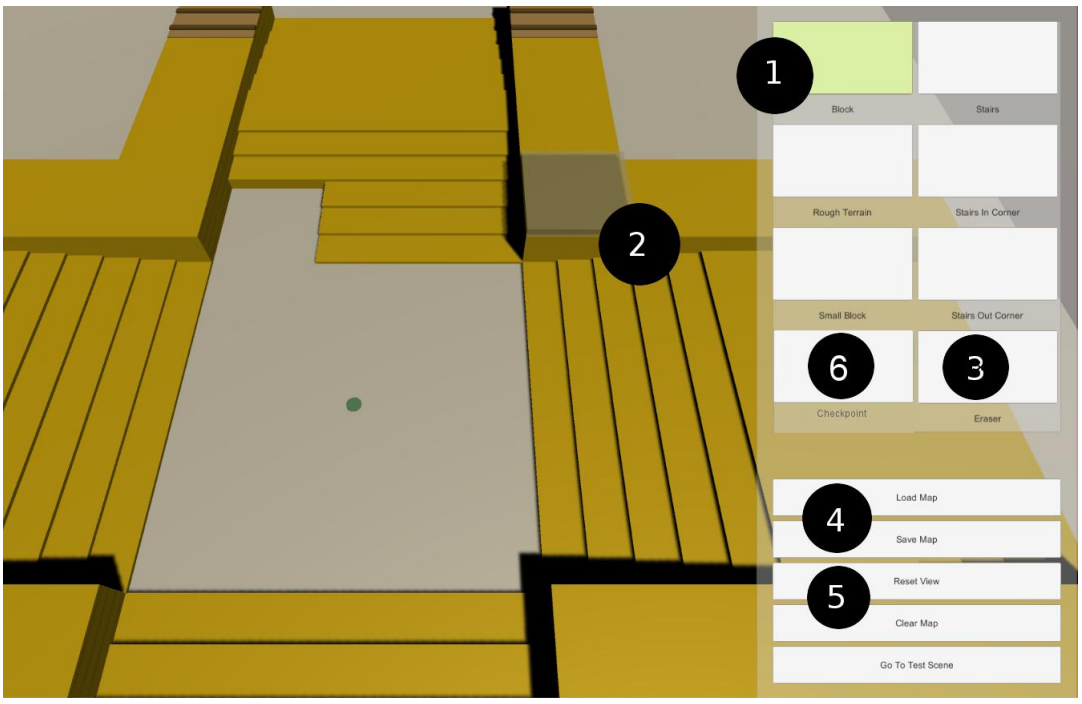
V editoru těl je možné nahrát a upravovat existující bytost [Obr. 29 – 1], nebo začít tvořit tvar těla od nuly [Obr. 29 – 2]. Při začátku od nuly je automaticky přidána první kostka, její rozměry odpovídají aktuálnímu nastavení sliderů [Obr. 29 – 3]. Ke každé stěně lze přidat jednu novou kostku. Nejprve je potřeba zvolit rozměry nové kostky pomocí sliderů [Obr. 29 – 3] a kliknutím ji pak připojit na konkrétní místo na těle bytosti. Kostky lze takto přidávat, pokud zeleně svítí tlačítko Add Mode [Obr. 29 – 4], lze ale přepnout i do Delete Modu [Obr. 29 – 5], ve kterém se kostky ubírají. Kostka k odebrání (a všechny její identické kopie) se označí černě, klinutím budou smazány. Je možné mazat pouze kostky na okraji, ke kterým není nic dalšího připojeno, aby bytost zůstala vždy v jednom kuse. Výsledné tělo je možné uložit do souboru [Obr. 29 – 6]. Soubory s DNA bytosti mají příponu .dna. Maximální velikost těla bytosti je 8 kostek. Pohybem myši při držení jejího pravého tlačítka se otáčí kamera okolo bytosti. Kolečkem myši lze zoomovat.



Obr. 29: Editor těl. Umožňuje vytvořit nové nebo upravovat existující tělo bytosti.

Editor map

V editoru map lze ze základních bloků skládat terén, kterým bude bytost později chodit. V pravém horní části obrazovky [Obr. 30 – 1] se nachází nabídka těchto bloků. K dispozici jsou čtvercové bloky dvou různých výšek, schodiště (včetně dvou schodišť rohových) a hrbolatý terén. Aktuálně vybraný blok svítí zeleně a při pohybu kurzoru se zobrazuje šedý poloprůhledný blok [Obr. 30 – 2] na místě, kam bude blok v případě kliknutí levým tlačítkem myši umístěn. Klikáním pravým tlačítkem myši lze šedý blok otáčet pokaždé o 90 stupňů. Speciálním druhem bloku je checkpoint [Obr. 30 - 6], který není používán při evoluci v hlavní scéně, ale ve hře (v testovací scéně) slouží pro označení míst na mapě, kterými má bytost za úkol projít. Bloky lze i odstraňovat, když se vybere možnost Eraser [Obr. 30 – 3]. Blok k vymazání se označí černě, po kliknutí bude vymazán. Je možné stavět mapu od nuly, nebo nějakou nahrát a upravovat ji, výsledná mapa jde opět uložit do souboru [Obr. 30 – 4]. Soubory s mapami mají příponu .map. Kamera se hýbe posunutím kurzoru myši k okraji obrazovky. Pro vycentrování zpět je zde tlačítko Reset View [Obr. 30 – 5], a pro smazání všech bloků najednou je tlačítko Clear Map [Obr. 30 – 5].



Obr. 30: Editor map. V tomto editoru je možné vytvářet a upravovat terény, ve kterých se bytosti budou pohybovat.

Pokročilá nastavení

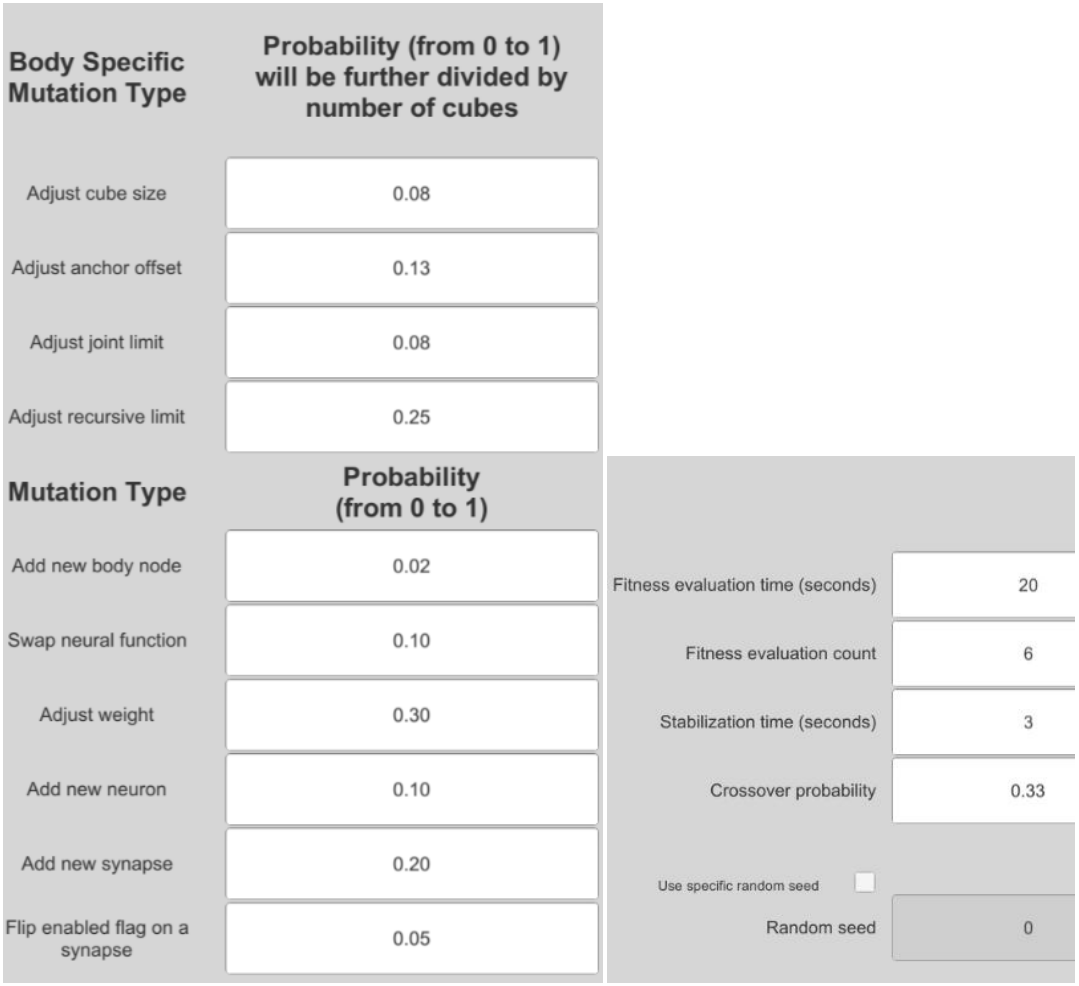
V hlavní obrazovce jsou dvě tlačítka pro zobrazení pokročilých nastavení. Při spuštění programu jsou nastaveny na základní hodnoty používané při experimentech popisovaných v hlavním textu práce. Tyto hodnoty jsou viditelné na následujícím obrázku [Obr. 31].

V první části pokročilého nastavení lze určovat pravděpodobnosti jednotlivých typů mutací. Zde je stručný popis toho, co tyto mutace dělají:

- Adjust cube size – mění velikost kostek
- Adjust anchor offset – mění konkrétní polohu připojení kloubu k dané stěně
- Adjust joint limit – mění horní i dolní limit kloubu v jeho stupních volnosti
- Adjust recursive limit – mění rekurzivní limit kostky
- Add new body node – přidá nový typ kostky
- Swap neural function – změní funkci neuronu na náhodnou jinou
- Adjust weight – upraví váhu synapse
- Add new neuron – přidá nový neuron do náhodného lokálního mozku
- Add new synapse – přidá novou synapsi (spoj mezi 2 neurony)
- Flip enabled flag – změní, jestli je synapse aktivní, nebo ne

V druhé části jsou další obecná nastavení:

- Fitness evaluation time – čas na chůzi za jedním cílem, v sekundách
- Fitness evaluation count – počet cílů v rámci jedné generace
- Stabilization time – čas na uklidnění rigidbodies před začátkem chůze
- Crossover probability – pravděpodobnost že kromě mutace bude i křížení
- Use random seed – je možné použít konkrétní random seed



Obr. 31: Podrobné nastavení. Umožňuje nastavovat pravděpodobnosti jednotlivých mutací, a další pokročilá nastavení. Hodnoty jsou na začátku nastaveny na hodnoty použité v experimentech popisovaných v hlavním textu práce.

Příloha B - Programátorská dokumentace

Program je vytvořen pomocí herního enginu Unity, verze 2019.2.12f1. Po otevření v Unity editoru je potřeba stáhnout asset Runtime File Browser verze 1.4.7 (https://assetstore.unity.com/packages/tools/gui/runtime-file-browser-113006) a jeho složku **SimpleFileBrowser** umístit do složky **Plugin**). Vygenerovanou dokumentaci kódu naleznete ve složce Dokumentace.

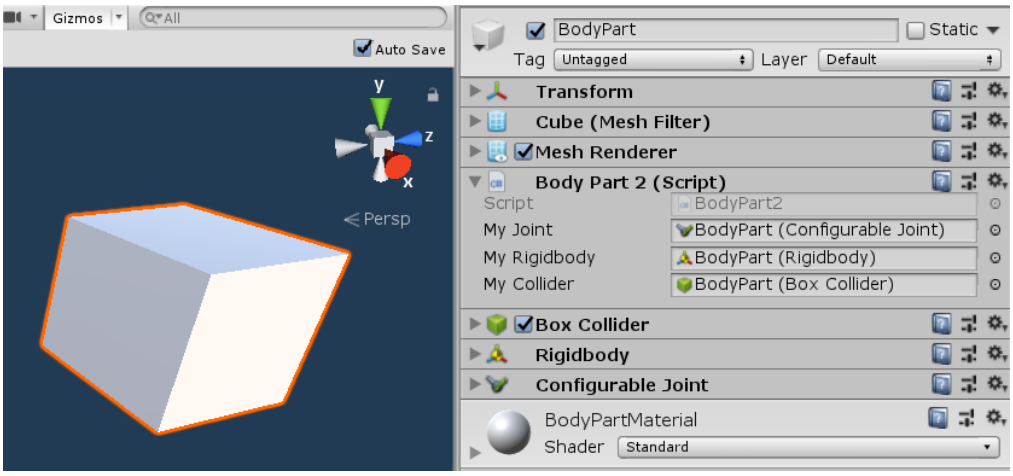
Struktura Unity projektu

Složka **Assets** obsahuje následující podsložky:

- **Materials** – osahuje materiály určující barvu objektů.
- **Plugin** – sem je potřeba umístit **SimpleFileBrowser**.
- **Prefabs** – obsahuje prefab reprezentující jednu fyzickou část těla bytosti (`BodyPart`), prefab pro vyznačení místa, kam hráč kliknul (`Indicator`), prefaby pro využití v edioru těl (`EditorCubePrefab` a `EditorGhostCubePrefab`) a složku **Obstacles**, která pro každý existující typ překážky obsahuje její model v několika variantách a template objekt.
- **Scripts** – obsahuje všechny skripty.
- **Scenes** – obsahuje scény `MainScene`, `TestingScene`, `BodyEditorScene` a `MapEditorScene`.

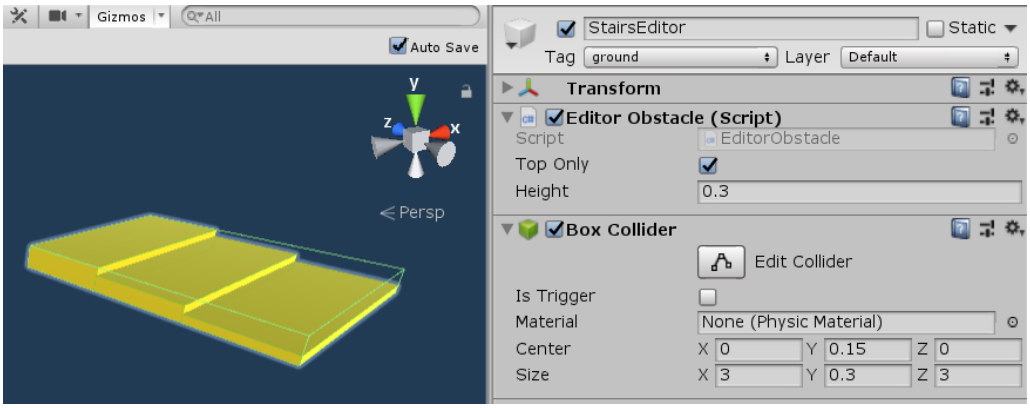
Prefaby

Nejdůležitějším prefabem je `BodyPart`, který reprezentuje jednu část těla [Obr. 32]. Je to kostka s komponentami BoxCollider, Rigidbody a Configurable Joint (kloub). Navíc má na sobě skript `BodyPart`, který tyto komponenty využívá (detekuje aktuální natočení kloubu a nastavuje nové cílové natočení kloubu). Parametry těchto komponent jsou nastaveny na výchozí hodnoty, konkrétní hodnoty pak udává genotyp bytosti.

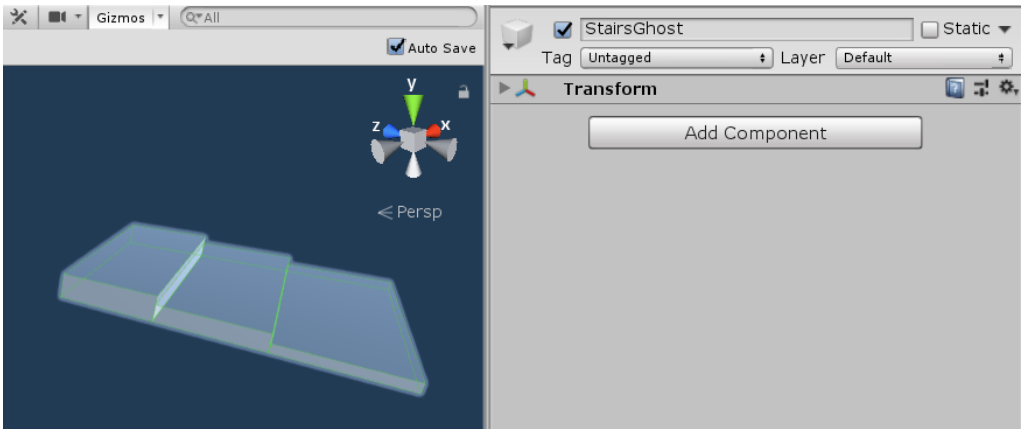


Obr. 32 Prefab BodyPart.

Každá překážka potřebuje čtyři prefaby. Prvním je její model s BoxColliderem, dalším je verze modelu se skriptem `EditorObstacle` a colliderem, který pro účely editoru nekopíruje přesný tvar překážky, ale obaluje ji do kvádru [Obr. 33]. Dále je to jeho průsvitná varianta bez collideru [Obr. 34] a template, což je prázdný objekt se skriptem `EditorObstacleTemplate` a obsahuje odkazy na předchozí prefaby.



Obr. 33 Příklad modelu překážky pro použití v editoru map.



Obr. 34 Příklad průsvitného modelu překážky.

Speciálním druhem překážky je `Checkpoint`, který je při načítání mapy za účelem evoluce ignorován a využívá se jen ve scéně `TestingScene`, kde je úkolem bytosti navštívit všechny checkpointy. Místo collideru má trigger collider, kterým detekuje, jestli skrze něj prošla nějaká bytost.

Scény

Program se skládá ze čtyř scén: `MainScene` sloužící ke spouštění evoluce, `TestingScene`, která slouží k otestování vytvořených bytostí ve hře, `BodyEditorScene` pro editor těl a `MapEditorScene` pro editor terénů. Každá scéna obsahuje kameru, jedno světlo, podlahu, a příslušný manager skript. `TestingScene` má pro účely multiplayeru kamery dvě. UI se nachází v hierarchii pod objektem Canvas a jeho funkčnost je podrobněji popsána v uživatelské dokumentaci.

Skripty

Skripty jsou napsané v jazyce C#. Nejpodstatnější je skupina skriptů, které řeší průběh evoluce. V hlavní scéně je `EvoManager`, který z UI načte parametry a spustí evoluci. O její průběh se pak stará třída `Population`. Ta si udržuje seznam bytostí (`Creature`) v aktuální generaci a opakovaně používá coroutine `RunOneGeneration(List<Vector2> targets)` [Obr. 35]. Coroutine je metoda, která vrací `IEnumerator` a může pomocí `yield return` rozložit svůj výpočet mezi více framů.

```
Pro každý cíl:
      spawnuje se populace
      čeká se 3s
      po daný počet kroků simulace:
            každá bytost simuluje jeden krok
            čeká se na update fyziky
      spočte se fitness
Spočte se průměrná fitness
Provede se selekce, mutace a křížení
```

Obr. 35 Obsah metody RunOneGeneration.

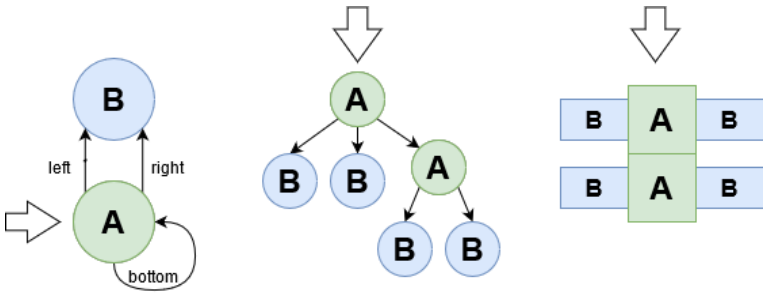
Pro úkol následování cíle je potřeba získat průměrnou fitness z několika pokusů s cílem vždy na jiném náhodném místě. Cíl je generován pro všechny bytosti stejný a vždy ve vzdálenosti 20 jednotek od startovní pozice dané bytosti. Spawnování

spočívá v překladu na fenotyp [Obr. 36]. Jeden krok simulace znamená načtení hodnot ze senzorů, propagování signálu neuronovou sítí a aplikování jejích výsledků na klouby.

Třída `Creature` reprezentuje jednu bytost, má odkaz na svůj genotyp ve třídě `DNA` a fenotyp ve třídě `Phenotype`. Umí se spawnovat, provést jeden krok simulace, spočítat svou fitness a metodou `ChangeTarget` je možné nastavit její aktuální cíl.

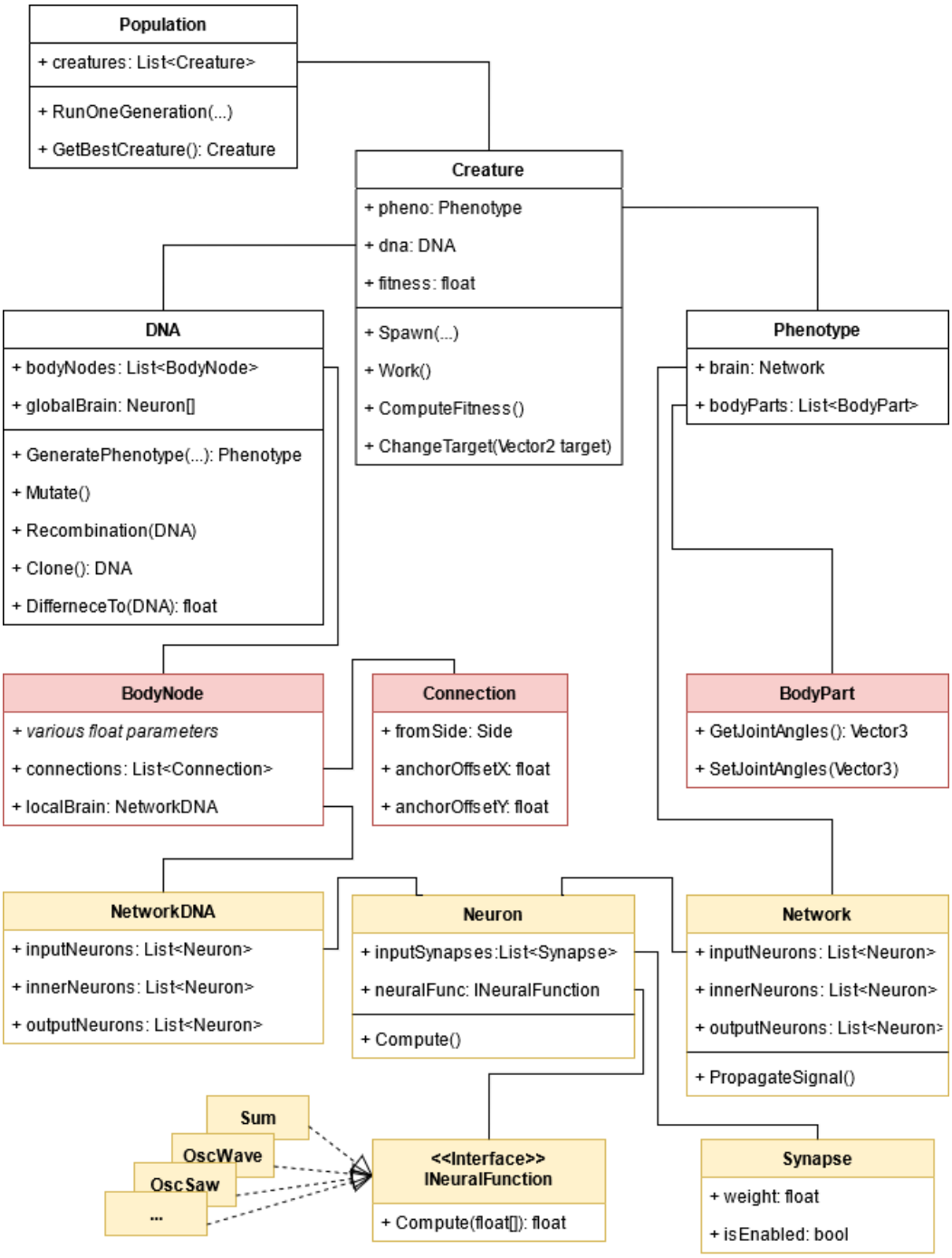
`DNA` obsahuje graf, jehož uzly jsou typu `BodyNode` a jsou propojené pomocí `Connection`. `BodyNode` představuje jeden typ části těla, jeho parametry jsou velikost kostky, limity kloubů, rekurzivní limit, seznam `Connection`, jejichž parametry určují místo připojení kloubu, a lokální mozek. Ten je reprezentován třídou `NetworkDNA`, která má vstupní, vnitřní a výstupní neurony, kde každý `Neuron` má svou vnitřní funkci a seznam synapsí. `Synapse` má svou váhu a informaci o tom, jestli je aktivní. Funkce neuronu musí implementovat rozhraní `INeuralFunction`.

`Phenotype` je výsledkem překladu `DNA` [Obr. 36]. Obsahuje seznam částí těla reprezentovaných třídou `BodyPart`, která je připojena k prefabu `BodyPart` a umožňuje pracovat s kloubem, neboli komponentou ConfigurableJoint. Druhou částí fenotypu je mozek typu `Network`. `Network` se skládá opět ze vstupních, vnitřních a výstupních neuronů a umí propagovat signál od vstupů po výstupy.



Obr. 36 Příklad překladu z genotypu na fenotyp. Vlevo genotyp, uprostřed je zobrazen průchod genotypem při jeho překladu, vpravo výsledný fenotyp. Průchod grafem je do hloubky s tím, že v každé větvi smí být jen tolik kopií uzlu, jaký je jeho rekurzivní limit. Na obrázku má uzel A rekurzivní limit 2.

Následující diagram [Obr. 37] popisuje vztahy výše zmiňovaných tříd. V levé části od třídy `Creature` se nachází třídy popisující genotyp bytosti a vpravo fenotyp. Červené části se týkají těla bytosti a žluté jejího nervového ovládacího systému.



Obr. 37 Diagram tříd.

Ostatní třídy jsou managery jednotlivých scén a jejich UI, ovladače kamery pro různé scény a `SaveLoader`, který s pomocí assetu Runtime File Browser řeší ukládání a nahrávání map vytvořených v editoru a genotypů ručně vytvořených či evolucí vyvinutých bytostí. Také obsahuje seznam všech existujících typů překážek. Navíc je zde ještě statická třída `Helper`, která udržuje konstanty a obsahuje různé pomocné metody, například přiděluje historické značky.