LANDING FALCON 9 ROCKET USING NN AND GA

SEMESTRÁLNÍ PRÁCE PRO PŘEDMĚT ZÁKLADY UMĚLÉ INTELIGENCE (BI-ZUM) NA FIT ČVUT.
ZDROJOVÝ KÓD: HTTPS://GITHUB.COM/MSKL/FIT-BI-ZUM-SPACEX

POPIS PROJEKTU

Implementoval jsem 2D simulaci pro přistávání prvního stupně rakety Falcon9 firmy SpaceX v herním engine Unity. Využívám neurální síť, která se vyvíjí pomocí genetického algoritmu. Hlavním cílem prvního stupně rakety je přistát s omezeným množstvím paliva a s omezeným časem na cílovou přistávací platformu.

ZPŮSOB ŘEŠENÍ

V následující kapitole se pokusím vysvětlit princip řešení mého programu a popsat základní principy simulace. V každé generaci se vygenerují přistávací moduly na náhodné pozici na obrazovce. Buď z náhodných neuronů, nebo na základě předchozí generace. Později probíhá simulace a po daném časovém úseku skončí. Následně dojde ke křížení a ke generování neurálních sítí pro příští generaci.

NEURÁLNÍ SÍŤ

Veškeré vstupy do neurální sítě jsou normalizovány na hodnoty v rozmezí <-1, 1>. Experimentálním přístupem jsem dospěl k síti o 6 vrstvách s 10 neurony. Tento počet by byl v ideálním případě vyšší, narážel jsem ovšem na výkonnostní omezení počítače, na kterém jsem model trénoval.

Neurální síť je popsána v souboru *Brain.cs*, neuron je popsán v souboru *Neuron.cs*.

- **Vstupní neurony**: 6 vstupů
  - Vzdálenost x
  - Vzdálenost y
  - Rychlost x
  - Rychlost y
  - Rotace kolem osy z
  - Rychlost rotace kolem osy z
- **Skryté vrstvy**: 6 vrstev po 10 neuronech
- **Výstupní neurony**: 3 výstupy
  - Výkon hlavního motoru
  - Náklon hlavního motoru
  - Výkon (a směr) doplňkové trysky v horní části rakety

GENETICKÝ ALGORITMUS

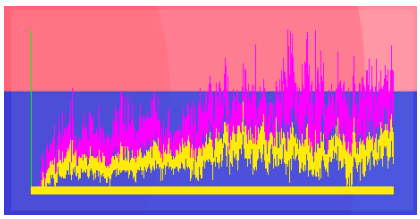
Genetický algoritmus je popsán v souboru *Genetic.cs*. Výpočet fitness probíhá přímo ve skriptu handling, který je odpovědný za ovládání jednotlivých instancí přistávacích modulů.

Rekombinace: Pro kombinaci dvou mozků využívám roulette wheel selection na základě výsledku fitness funkce z průběhu simulace. Nejprve vyberu dvě neurální sítě ke spáření, jejichž neurony následně od náhodně zvoleného crossover bodu začnu rekombinovat a tímto způsobem vytvořím novou neurální síť.

Mutace: Šance na mutaci jednotlivého neuronu je nastavena na hodnotu 3%. Pokud dojde k mutaci, nastaví se neuronu náhodná váha z Gaussovy křivky v okolí současné hodnoty.

Fitness funkce: Nejobtížnějším úkolem bylo vytvořit fitness funkci, která by nejlépe motivovala jednotlivé instance raket k správnému vývoji. Fitness funkce je popsána ve skriptu který se stará o ovládání přistávacího modulu *Handling.cs*. V implementaci jsem se snažil zohlednit několik faktorů:

- **V průběhu letu**:
  - Směr letu (bonus za let směrem k platformě)
  - Správná rotace (velká penalizace za přetočení kolem osy z)
  - Bonus za délku letu (rakety které letí déle jsou lepší)
- **V případě neúspěchu** (pád do moře, silný náraz, náraz do hranice světa)
  - Vzdálenost od cílové platformy
  - Síla nárazu
- **V případě úspěchu**
  - Množství benzínu v nádrži (čím více, tím vyšší bonus)
  - Bonus za přistání



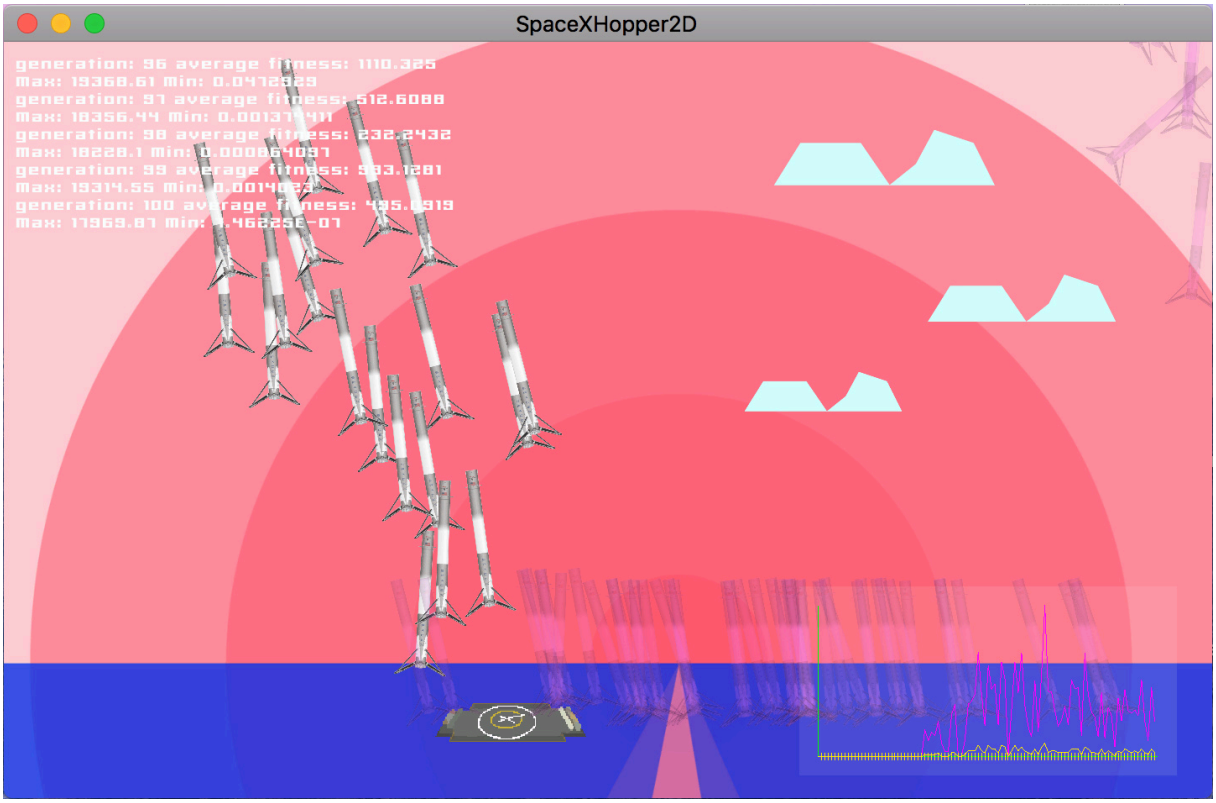
IMPLEMENTACE

Projekt je vytvořený v herním engine Unity3D. Engine Unity nabízí jednoduchý interface jak pro simulaci fyzikálních těles, tak pro aplikaci síly a sledování kolizí objektů s okolím. Aplikace se dá navíc vyexportovat na libovolnou platformu. PC počínaje, Androidem a WebGL konče.

OVLÁDÁNÍ APLIKACE

Pro vygenerování první generace je nutné stisknout tlačítko **F**. Pomocí něj se dá vygenerovat nová generace s náhodnými geny i po delší době běhu programu. Tlačítkem **N** lze vynutit ukončení současné generace a vygenerování následující.

PŘÍKLAD VÝSTUPU



Na obrázku je vyobrazena aplikace po generaci číslo 100. Konzole obsahující informace o současné generaci a několika předchozích se nachází v levém horním rohu. Graf v pravém dolním rohu zachycuje fialovou barvou nejvyšší fitness v generaci. Žlutou barvou je znázorněna průměrná fitness v generaci.

ZÁVĚR

Neurální síť použitou v tomto případě jsem vyvinul původně pro projekt, ve kterém jsem učil auta projíždět bludištěm a vyhýbat se objektům. V takovém případě fungovalo spojení neurální sítě s genetickým algoritmem naprosto bezchybně a auta se skutečně naučila po několika generacích vyhýbat se objektům. Pro přistávání raket je ovšem potřeba zohlednit mnohem větší množství proměnných, které vstupují do neurální sítě. Taktéž je více výstupů, tudíž je třeba využít rozsáhlejší neurální síť a trénink je tak obtížnejší. Každá malá odchylka může způsobit změnu v chování přistávacího modulu. Křížení pomocí genetického algoritmu tak často rozhodí přesnou souhru správných genomů což způsobí nefunkční modul i po generacích, ve kterých některé rakety již dokázaly přistát. Ve většině případů po tisíci generacích více než polovina přistávacích modulů úspěšně přistane.

V reálném prostředí se neurální síť nepoužívá. Hlavím důvodem je zřejmě fakt, že v případě nehody by bylo v podstatě nemožné zjistit a opravit její příčinu.