Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Bot pro hru Rocket League

2023

Lukáš Kopačka

Vedoucí práce:
Mgr. Petr Osička, Ph.D.

Studijní program: Informační technologie,
kombinovaná forma

Bibliografické údaje

Autor:	Lukáš Kopačka
Název práce:	Bot pro hru Rocket League
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní program:	Informační technologie, kombinovaná forma
Vedoucí práce:	Mgr. Petr Osička, Ph.D.
Počet stran:	29
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Lukáš Kopačka
Title:	Bot for Rocket League
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study program:	Information Technologies, combined form
Supervisor:	Mgr. Petr Osička, Ph.D.
Page count:	29
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Tato bakalářská práce se zabývá prozkoumáním knihovny RLBot, která slouží k tvorbě botů pro hru Rocket League. Práce se zaměřuje na různé implementace botů a popisuje obecný postup tvorby vlastního bota.

Hlavním cílem práce je zhodnocení limitací a možností knihovny RLBot. Práce by měla poskytnout užitečný náhled na proces tvorby botů a přispět k lepšímu porozumění knihovny RLBot a jejího použití pro tvorbu botů. Implementace botů bude provedena v jazyce Python.

Synopsis

This bachelor's thesis explores the RLBot library, which is used to create bots for the game Rocket League. The work focuses on various bot implementations and describes the general process of creating your own bot.

The main goal of the work is to evaluate the limitations and possibilities of the RLBot framework and evaluate its suitability for creating bots for the game Rocket League. The work provides useful insight into the bot creation process and contributes to a better understanding of the RLBot library and its use for bot creation. The bots will be implemented in Python.

Klíčová slova: Rocket League, RLBot, Bot, RLGym, Umělá inteligence

Keywords: Rocket League, RLBot, GUI, RLGym, Artificial Intelligence

Děkuji Mgr. Petr Osička, Ph.D. za odborné vedení při zpracování bakalářské práce.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

Seznam obrázků

1 Úvod

V této bakalářské práci se zabýváme problematikou tvorby botů pro hru Rocket League. Konkrétně se zaměřujeme na prozkoumání knihovny RLBot, která slouží právě k tomuto účelu. Hlavním cílem práce je poskytnout užitečný náhled na proces tvorby botů a zhodnotit limitace a možnosti knihovny RLBot.

V úvodu se nejprve budeme věnovat stručnému představení hry Rocket League. Následně představíme samotnou knihovnu RLBot. Dále se budeme zabývat implementací již existujících botů vytvořených pomocí této knihovny, na základě kterých navrhneme postup pro implementaci bota.

Důležitou částí práce bude zhodnocení limitací a možností knihovny RLBot. Tyto limitace získáme díky zkoumání cizích botů společne s návrhem a implementací vlastního bota.

Na závěr práce poskytneme ucelený náhled na proces tvorby botů pro hru Rocket League a zhodnocení vhodnosti knihovny RLBot pro tento účel. Práce se snaží přispět k lepšímu porozumění knihovny RLBot, jejího použití pro tvorbu botů v hře Rocket League a být užitečným průvodcem pro všechny, kdo se chtějí do této problematiky ponořit.

2 Rocket League

Rocket League [1]je oblíbená videohra žánru sportovní arkády, která byla poprvé představena veřejnosti 7. července 2015. Hra byla vytvořena a vyvinuta společností Psyonix, nyní známou jako Psyonix Studios, která je dceřinou společností Epic Games. Rocket League kombinuje prvky fotbalu a závodních her, kde hráči ovládají futuristická vozidla s raketovými motory a snaží se skórovat góly tím, že do soupeřovy branky navedou velký nafukovací balón.

Hra se stala rychle populární díky své jednoduchosti a zábavné hratelnosti. Rocket League podporuje singleplayer i multiplayer a je dostupná na různých platformách, včetně PlayStation, Xbox, Nintendo Switch a PC.

Kromě základní hry, která je nyní zdarma ke stažení, Rocket League nabízí pravidelné aktualizace, sezónní události a prémiový obsah ve formě kosmetických předmětů. Hra rozšířila svou přítomnost do e-sportového prostředí s oficiálními turnaji a soutěžemi, které se konají po celém světě a přitahují tisíce hráčů a fanoušků.

2.1 Herní prvky

V této sekci popíšeme všechny herní prvky, které budeme potřebovat pro implmentaci námi navrženého bota. Prvky jako: trunaje, vizualní změny aut, hráčský chat a mnoho dalších proto popisovat nebudeme.

1. Auta: Jsou nejdůležitějším prvkem. Každé auto je ovládáno hráčem nebo programem. Každé auto disponuje následujícími schopnostmi: změna směru,

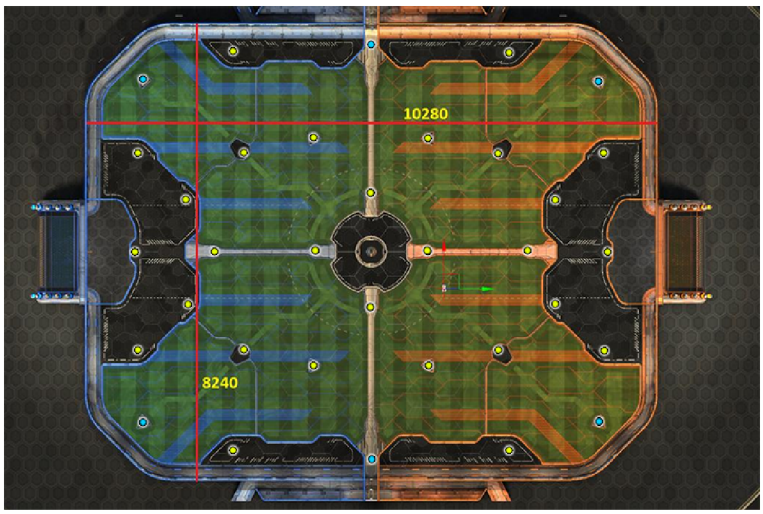
zrychlení vpřed nebo vzad, skok, dvojitý skok, ruční brzda a využití raketového pohonu.

2. Balón: Balón je navržen tak, aby co nejvíce připomínal skutečný míč, což zajišťuje sami realistické odrazy míče při nárazech. Tato vlastnost však ztěžuje určení přesného místa, kam by měl hráč míč zasáhnout.

3. Brány: Branková zóna je definována jako prostor, ve kterém se musí balón zcela nacházet, aby byl považován za gól. Ve hře Rocket League existují módy, které obsahují různá umístění brankových zón, avšak pro naše účely se budeme zabývat pouze variantou z klasického fotbalu.

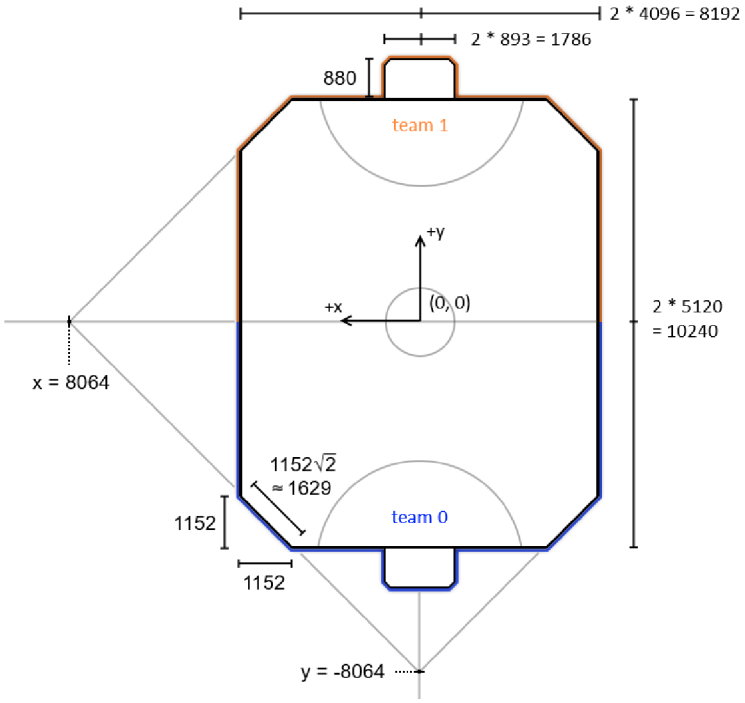
4. Raketový pohon: Raketový pohon využívá raketové palivo, které jsme předem nashromáždili. Jeho použití rapidně zvyšuje akceleraci vozidla a umožňuje také dosáhnout přibližně o 60 % vyšší maximální rychlost. Mimo jiné umožňuje provádět změny směru letu.

5. Raketové palivo, známé jako boost (dále označováno jako boost), je rozmístěno v aréně ve formě levitujících zlatých koulí nad podložkami. Celkový počet boostů na mapě je 34. Existují dva typy boostů: úplný a částečný. Na mapě je 6 úplných boostů, které obnoví 100 % paliva. Těchto 6 boostů se regeneruje za 10 sekund po použití. Na mapě je také 28 částečných boostů, které přidávají 12 % paliva a obnovují se za 5 sekund po použití. Hráči začínají každé kolo s 33 % boostu. Umístění viz obrázek 1. [2] [3]



Obrázek 1: Rozmístění boostů na hřišti

6. Aréna: Arény se ve hře liší svým designem, velikostí a atmosférou. Některé arény mohou mít zvláštní pravidla nebo vlastnosti. V našem případě budeme používat pouze zakladní verzi arény, viz obrázek 2.



Obrázek 2: Rozměry hřiště

7. Startovní pozice: Základní mapy mají pět startovních pozic, které se dají rozdělit do tří skupin: brankář, obránci a křídla. Startovní pozice by měla určit funkci hráče v týmu, viz obrázek 3.

Obrázek 3: Startovní pozice

8. Týmy: Hráči se mohou účastnit hry jako členové modrého či oranžového týmu, přičemž každý tým může mít až 4 členy.

9. Časomíra: Většinou trvají zápasy 5 minut, což je časový limit, a pokud je skóre nerozhodné po uplynutí časového limitu, hra pokračuje do pro-

dloužení. Prodloužení trvá, dokud jeden z týmů nedá vítězný gól. Zápasy jsou rozděleny do kol. Kolo začíná třísekundovým odpočtem, po kterém následuje výkop a standardní hra až do padnutí gólu.

10. Boti: Boty budeme prozkoumávat hlouběji v následující sekci.

2.2 Boti v Rocket League

V této sekci se budeme zabývat boty v Rocket League. Prozkoumáme jejich funkci, druhy botů, jak schopní jsou a proč má smysl implementovat vlastního bota.

Boti nahrazují hráče v ovládání aut a nasazují se ve chvíli, kdy je nedostatek hráčů, kteří by tvořili plnohodnotný tým. Mohou být použiti v offline zápasech, jako je sezóna nebo trénink, a také v online zápasech, když je některý z hráčů odpojen a nevrátí se zpět do hry. Boti se obvykle automaticky přidávají do zápasu, pokud je to možné.

Boti ve hře Rocket League se dělí do úrovní dle obtížnosti, které určují jejich schopnosti a výkon během zápasu. Tyto úrovně jsou:

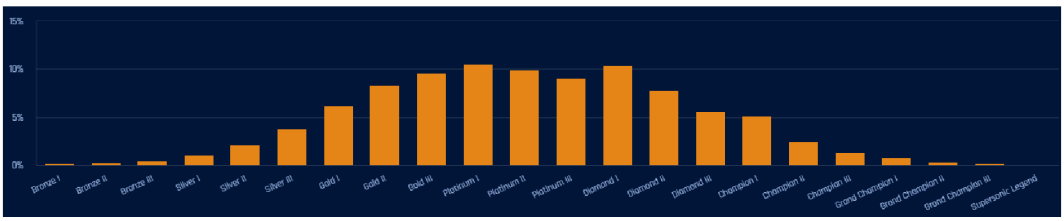
1. Nováček (Rookie) – Nejnižší úroveň obtížnosti, boti na této úrovni mají základní dovednosti a reakce.

2. Prostřední (Pro) – Střední úroveň obtížnosti, boti na této úrovni mají lepší dovednosti a rychlejší reakce než nováčci.

3. Veterán (Allstar) – Nejvyšší úroveň obtížnosti, boti na této úrovni mají pokročilé dovednosti a rychlé reakce srovnatelné s pokročilými hráči.

Pro porovnání síly botů s hráči nejprve vysvětlíme, jak hodnotíme schopnosti hráčů a zobrazíme možné rozložení jednotlivých hráčů.

Hráči Rocket League mají možnost hrát hodnocené zápasy, díky kterým jsou následně rozděleni do kategorií. Na obrázku 4 je přehled rozložení hráčů do kategorií od nejslabších po nejsilnější.



Obrázek 4: Rozložení hráčů do kategorií dle schopností

Určení síly botů od Psyonixu (nováček, prostřední, veterán) není možné jinak než odhadem, ale komunita se shoduje, že dosahují schopností mezi bronzovým a zlatým rankem. To znamená, že v případě nasazení bota v zápasech na úrovni

platiny je tým znevýhodněn. Navíc hráči na vyšších úrovních už nemají možnost hrát offline proti protivníkovi, který by byl na jejich úrovni.

Tento nedostatek dal vzniknout komunitě lidí, kteří se rozhodli vytvořit své vlastní boty, které by předčily implementaci botů od Psyonixu. Produktem této skupiny je knihovna Rlbot.

3 RLBot

3.1 Základní informace

RLBot [4] [5] je open source projekt zaměřený na vytváření botů pro hru Rocket League. Projekt byl založen komunitou nadšenců a vývojářů, kteří chtějí rozšířit zážitek ze hry. Díky této knihovně mají hráči možnost hrát proti schopnějším botům, vytvářet své vlastní boty a také pořádat turnaje mezi vytvořenými boty.

RLBot knihovna byla vytvořena v jazyce Python, který je často používán v oblasti umělé inteligence a strojového učení.

Komunita poskytuje zdroje, dokumentaci, návody a podporu pro ty, kteří se chtějí zapojit do vývoje botů a učit se o umělé inteligenci a strojovém učení v kontextu s videohrami.

Jedním z úspěchů RLBot komunity je organizování turnajů a soutěží, kde boti vytvoření různými vývojáři zápasí proti sobě. Tyto turnaje nejenže zvyšují povědomí o projektu, ale také umožňují vývojářům porovnat své boty a zjistit, jaké strategie a techniky jsou nejefektivnější.

Vývoj RLBot knihovny je stále aktivní a komunita neustále pracuje na zdokonalování a rozšiřování možností botů ve hře Rocket League. To zahrnuje testovaní nových algoritmů, technik a integraci nových herních funkcí a vylepšení uživatelského rozhraní a nástrojů pro vývoj a testování.

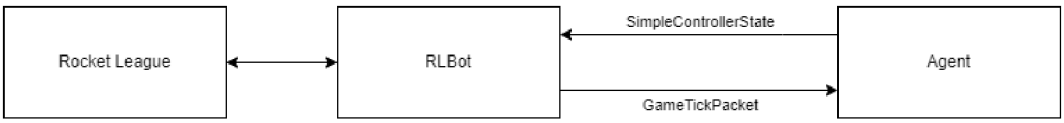
3.2 Komponenty RLBot knihovny

Prozkoumání komponent nám umožní lépe porozumět jak správně s knihovnou komunikovat, jaké datové typy můžeme používat a jaké pomocné funkce můžeme využít. Komponenty které jsme blíže zkoumaly: Renderer, Bot Manager, Agent, GameTickPacket, SimpleControllerState, FieldInfoPacket, BallPrediction, GameState.

- Renderer [6] je nástroj, který umožňuje vývojářům vykreslovat 2D a 3D objekty přímo v herním okně Rocket League. To je užitečné pro zobrazování informací o botovi, debuggování nebo vizualizaci dat, jako je trajektorie míče nebo pozice hráčů.

- Bot Manager: Zajišťuje správu a komunikaci mezi boty a herním prostředím. To zahrnuje načítání botů do hry, jejich synchronizaci s hrou a zpracování příkazů od botů.

- Agent je základní třída, která je při implementaci přepisována k vytvoření vlastního bota. Agent obsahuje metody jako „___init___“, „get_output“, „get_field_info“, „get_ball_prediction_struct“, které lze dle potřeby přepsat. Komunikaci mezi agentem a RLBot knihovnou můžeme vidět na obrázku 5.



Obrázek 5: Proudění dat

- GameTickPacket (dále jako tick) je datová struktura, která obsahuje aktuální stav herního prostředí, včetně pozic hráčů, míče, boostů a dalších relevantních informací. Data v této struktuře jsou aktualizována v každém herním ticku. V základním nastavení by měl bot obdržet 60 ticků za sekundu.
- SimpleControllerState je datová struktura, která umožňuje botům ovládat své pohyby a akce ve hře. Zde je přehled možností nastavení:
  - Zatáčení: (-1.0, 1.0) = vlevo/vpravo
  - Akcelerace: (-1.0, 1.0) = zpět/vpřed
  - Pitch: (-1.0, 1.0) = naklonění dopředu/naklonění dozadu
  - Yaw: (-1.0, 1.0) = naklonění doleva/naklonění doprava
  - Roll: (-1.0, 1.0) = otáčení proti/otáčení po směru hodinových ručiček
  - Jump: True = skok
  - Boost: True = raketový pohon
  - Handbrake: True = ruční brzda
- FieldInfoPacket poskytuje informace o herním poli, jako jsou rozměry a umístění branek nebo podložek s boosty.
- BallPrediction je funkce, která poskytuje předpovědi trajektorie míče v průběhu času. Maximální předpověd míče je přibližně 6 vteřin dopředu. Funkce je schopná předpovídat odrazy míče od stěn.
- GameState je třída, která umožňuje vývojářům měnit stav herního prostředí, jako je pozice míče nebo hráčů. To může být užitečné pro tréninkové účely nebo pro vytváření specifických scénářů pro testování a ladění botů.

3.3 Boti vytvořeni pomocí RLBot knihovny

Před vlastní implementací bota jsme prozkoumali zdrojové kódy některých volně dostupných botů implementovaných v Pythonu. Mnoho z nich používá pokročilé techniky, které vyžadují dobré znalosti jak hry, tak analytické geometrie v prostoru.

1. ReliefBot: ReliefBot je jeden z prvních botů vytvořených pomocí RLBot knihovny. Byl vyvinut programátorem jménem Terhart, který byl také jedním z prvních členů komunity RLBot. ReliefBot je schopen provádět základní akce, jako je střelba na branku, bránění a sběr boostů.

2. BeastBot: BeastBot je další populární bot vytvořený pomocí RLBot knihovny. Byl navržen tak, aby byl schopen provádět akrobatické manévry a pokročilé taktiky, jako je aerial control (akrobace ve vzduchu) a dribling.

3. Self-driving Car: Tento bot byl navržen tak, aby simuloval chování samořízeného automobilu. Používá pokročilé algoritmy pro zpracování obrazu a plánování trajektorií, což mu umožňuje provádět složité manévry a reagovat na změny ve hře.

4. SkyBot: SkyBot je bot zaměřený na aerial hru a byl navržen tak, aby měl vysokou úroveň kontroly ve vzduchu. Tento bot je schopen provádět pokročilé aerial manévry, jako je freestyling a driblink ve vzduchu.

5. GoslingBot: GoslingBot [7] byl navržen s cílem demonstrovat schopnosti a možnosti RLBot knihovny a poskytnout komunitě dobrý základ pro další vývoj botů.

6. VirxERLU: VirxERLU [8] je bot vytvořený v rámci RLBot knihovny, který se zaměřuje na vývoj pokročilých technik a strategií ve hře Rocket League. VirxERLU byl navržen tak, aby byl schopen provádět různé akce a manévry, jako je driblink, vzdušná akrobace, hraní o zeď. Inspiruje se GoslingBotem.

3.4 Zhodnocení

Kódy těchto botů nejsou zcela vhodné jako základ pro vytvoření vlastního návrhu, jelikož často chybí dostatečné komentáře a kontext ohledně zvolených hodnot nebo stavů, ve kterých se bot musí nacházet pro provádění určitých operací. Abychom lépe porozuměli těmto kódům, je nutné vkládat do nich vlastní pomocné funkce, které nám umožní rozpoznat autorský záměr. I přesto je složité udržet si přehled o všech hodnotách potřebných pro výpočty. Z tohoto důvodu použijeme pouze jejich obecnou strukturu a některé pomocné funkce.

4 RLGym

RLGym [9] je alternativní framework pro vytváření a trénování botů pro hru Rocket League. Na rozdíl od RLBot knihovny, který se zaměřuje na ruční implementaci strategií a pohybových operací, RLGym využívá reinforcement learning pro trénování botů, čímž dosahuje lepších výsledků a vyššího výkonu.

Díky posilovanému učení se boti v RLGym učí řešit různé situace ve hře tím, že jsou odměňováni za správné rozhodnutí a chování. Tento přístup umožňuje botům zlepšovat své dovednosti a stávat se silnějšími a efektivnějšími, což je důvod, proč boti vytvoření v RLGym často předčí boty vytvořené pomocí RLBot knihovny. To je potvrzeno skutečností, kdy v turnajích bot Nexto (nejsilnější veřejně dostupný bot vytvořený pomocí RLGym) jasně dominoval ostatním. Tento úspěch ukazuje potenciál RLGym jako knihovny pro vytváření vysoce konkurenceschopných botů pro hru Rocket League.

Nicméně, vytváření botů v prostředí RLGym může představovat problém, protože někteří lidé mohou tyto boty zneužít k podvodům v hodnocených zápasech. V minulosti byl takto zneužit například bot Nexto.

5 Matematické operace

K implementaci bota pro hru Rocket League jsou operace s vektory klíčové, protože botovi umožňují pohyb po hřišti a lépe interagovat s herním prostředím. Díky těmto výpočtům může bot lépe určit svou pozici, směr a rychlost vůči ostatním objektům, jako je míč, branky nebo ostatní vozidla. Výsledkem je lepší koordinace, rozhodování a celková úroveň hry bota.

5.1 Směrový vektor

Funkce „direction“ vypočítá směrový vektor mezi dvěma body v prostoru. Směrový vektor je matematická reprezentace, která popisuje směr a délku mezi dvěma body. Směrový vektor mezi body A a B se vypočítá jako:

$$\text{direction}(A, B) = B - A = (B_x - A_x, B_y - A_y, B_z - A_z)$$

5.2 Skalární součin

Funkce „dot“ provádí skalární součin dvou vektorů. Skalární součin je matematická operace, která vezme dva vektory a vrátí skalární (číselnou) hodnotu. Skalární součin dvou vektorů A a B se vypočítá jako:

$$\text{dot}(A, B) = A_x \cdot B_x + A_y \cdot B_y + A_z \cdot B_z$$

Skalární součin se často používá k zjištění úhlu mezi dvěma vektory nebo k projekci jednoho vektoru na druhý. [10]

5.3 Vektorový součin

Funkce „cross“ provádí vektorový součin dvou vektorů. Vektorový součin je matematická operace, která vezme dva vektory a vrátí nový vektor kolmý na oba původní vektory. Vektorový součin dvou vektorů A a B se vypočítá jako:

$$\text{cross}(A, B) = (A_y \cdot B_z - A_z \cdot B_y, A_z \cdot B_x - A_x \cdot B_z, A_x \cdot B_y - A_y \cdot B_x)$$

Vektorový součin se často používá k nalezení normálového vektoru pro plochu nebo k výpočtu momentu síly.[11]

5.4 Normalizace

Normalizuje vektor, což znamená, že upraví jeho délku (magnitudu) na jedničku, zatímco zachová jeho směr. Normalizace vektoru A se provádí takto:

$$\text{normalized}(A) = \frac{A}{\|A\|}$$

Kde magnituda(A) je délka vektoru A. Normalizované vektory se často používají pro reprezentaci směru. [12]

5.5 Délka

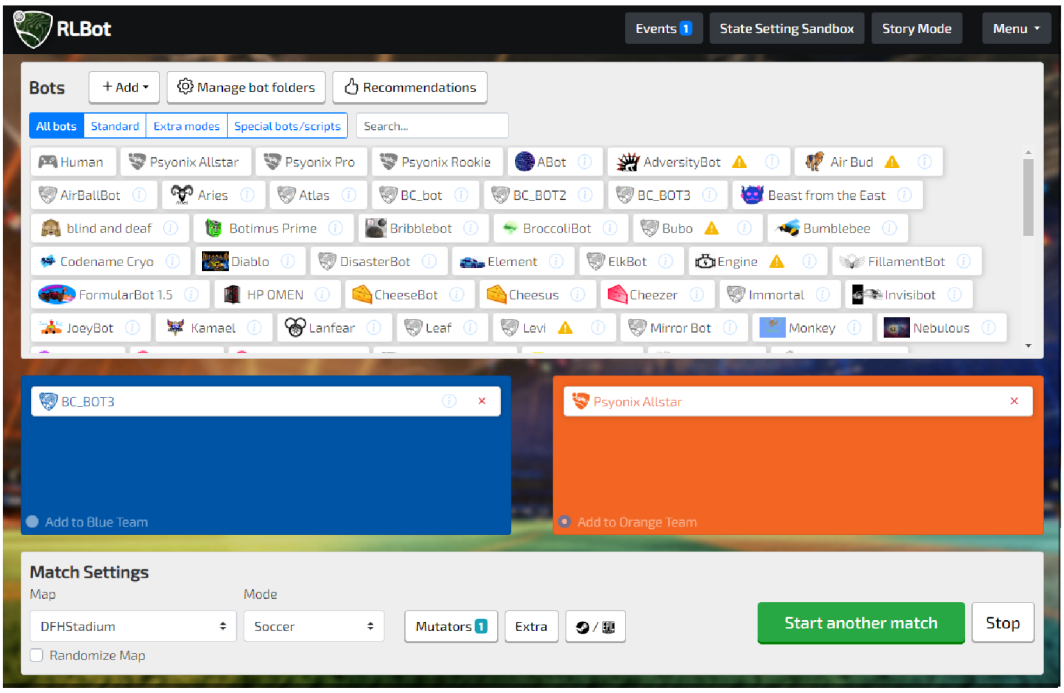
Funkce „magnitude“ vypočítá délku vektoru. Délka vektoru A se vypočítá jako:

$$\|A\| = \sqrt{A_x^2 + A_y^2 + A_z^2}$$

Délka vektoru je důležitá pro určení vzdálenosti mezi dvěma body nebo pro normalizaci vektoru. [13]

6 Testovaní

Během vývoje bota lze jeho implementaci testovat v herní simulaci prostřednictvím RLBot GUI. Viz obrázek 6.

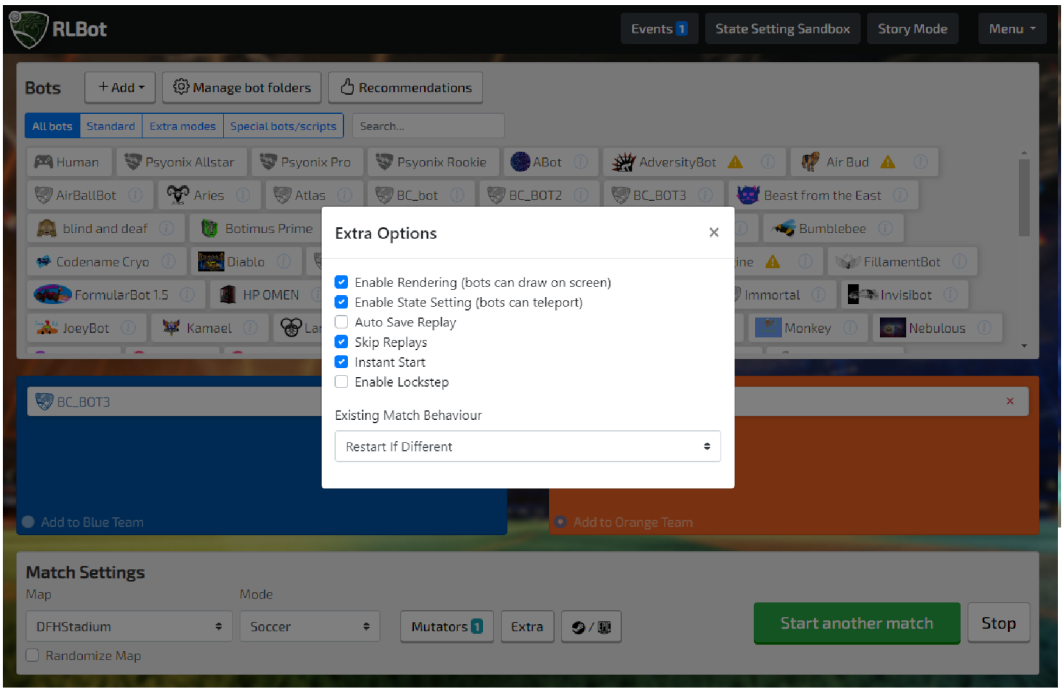


Obrázek 6: RLBot GUI

Toto grafické rozhraní umožňuje výběr botů, jež mají být otestováni. V rámci práce byli boti testováni proti botu Psyonix Allstar.

Pro vývoj je duležité v sekci „extra options“ povolit možnost „Enable Rendering (bots can draw on screen)“ pro zobrazení pomocných prvků či křivek v herním prostoru. Viz obrázek 7.

Pro efektivní testování je vhodné v menu „mutators“ nastavit nekonečnou dobu trvání hry, neboť testování často přesahuje pětiminutový limit. GUI obsahuje také užitečnou sekci „State Settings Sandbox“, která umožňuje snadno měnit pozici míče či pozastavit hru a analyzovat hodnoty zobrazené ve hře nebo v konzoli. Tímto způsobem lze posoudit, zda bot provádí předpokládané operace a zda jsou hodnoty zobrazeny na správných místech. V této sekci je navíc možné zrychlit herní proces.



Obrázek 7: „Extra“ možnosti v RLBot GUI

6.1 Testovací scénáře

RLBot knihovna přináší propracované prostředí pro tvorbu vlastních testovacích scénářů, což je klíčové pro detailní testování a ladění různých aspektů chování bota. Během vývoje můžeme definovat různé situace, které si přejeme otestovat, jako jsou výkopy, obrana brány, střelba na bránu nebo provádění různých triků. V případě výkopů lze například nastavit specifické pozice botů a simulovat různé typy výkopů. Pokud jde o obranu brány, lze vytvořit různé útočné scénáře a analyzovat, jak bot reaguje na různé pozice a rychlosti, ve kterých se nachází nejen on, ale i míč nebo soupeř. Stejně tak u střelby na bránu může být testována s různými úhly, rychlostmi a časováním, aby se simulovaly reálné herní situace. Dokonce lze také vytvořit simulace složitějších manévrů a triků (vzdušné driblinky, stěnové střely), což přispívá k pochopení schopností a omezení vašeho bota v různých strategických situacích.

7 Tvorba vlastního bota

Nyní, když známe komponenty hry, funkcionalitu knihovny, máme znalost vektorových funkcí a prozkoumali jsme jiné boty, tak jsme schopni začít s implementací bota.

7.1 Příprava vývojového prostředí

Jako první si postupně zprovozníme vývojové a testovací prostředí pro úpravu kódu základního bota v RLBot:

- Python: RLBot je postaven na Pythonu, takže se nejprve ujistíme, že máme nainstalovaný Python, ideálně verzi 3.7 nebo novější. Můžeme si jej stáhnout z oficiálních stránek Pythonu.
- Nainstalujeme Rocket League: Pro otestování bota musíme mít nainstalovanou hru. Hru je možné získat přes Epic Games Launcher, který si můžeme stáhnout a nainstalovat z oficiálního webu.
- Stáhneme RLBotGUI: RLBotGUI zahrnuje RLBot knihovnu, takže není nutné ji instalovat samostatně. Stáhneme si nejnovější verzi RLBotGUI z officialních stránek RLBot nebo z RLBot repozitáře a nainstalujeme ji.
- Vytvoříme nového bota: V RLBotGUI klikneme na tlačítko "+ Add"a zvolíme "Start Your Own Bot!". Uvedeme jméno našeho nového bota (například BC_Bot3). Zvolíme jazyk (v našem případě Python). RLBotGUI automaticky vytvoří novou složku s přednastaveným kódem a konfigurací pro našeho bota.
- Nastavení RLBotGUI: Pro testování bota vybereme našeho bota a vložíme ho do jednoho týmu (například do modrého) a protože bota budeme testovat proti Psyonix All-Star, tohoto bota vložíme do opačného týmu (oranžového). Dále pro testování nastavíme v mutatorech, aby zápas netrval 5 minut, ale byl neomezený. Poslední změnou bude povolení, aby boti mohli vykreslovat pomocné čáry nebo hodnoty do hry. To uděláme v menu Extra, kde zvolíme "Enable Rendering (bots can draw on screen)". V tuto chvíli už jen stačí spustit zápas, kde můžeme vidět našeho nově vzniklého bota proti botu od Psyonix.
- Úprava kódu bota: V RLBotGUI, při otevření informací o našem botovi, můžeme otevřít složku s botem. V této složce se nachází soubor bot.py, který implementuje našeho bota a několik dalších konfiguračních souborů. Ve složce se také nachází podadresář util, kde jsou uloženy pomocné funkce.

7.2 Specifikace implementace

Implementace je určena pro typ hry 1v1, což znamená, že nebude brát v úvahu přítomnost spojenců a předpokládáme, že soupeř je rovněž pouze jeden. Díky tomuto přístupu nebudeme muset implementovat komunikaci mezi boty, což zjednoduší celý proces.

7.3 Metodika

Metodika vývoje je založena na iterativním přístupu. Po spuštění hry se zaměříme na sledování bota až do okamžiku, kdy narazíme na situaci, kde se bot nechová dle našich očekávání. Hru zastavíme stisknutím klávesy "Escape"nebo prostřednictvím RLBot GUI v menu "state setting sandbox", kde zvolíme možnost "Freeze Game". Následně navrhneme vhodné řešení a implementujeme ho. Díky automatické aktualizaci RLBotu se změny projeví během krátké doby, což nám umožní zhodnotit, zda jsou implementace a návrh uspokojivé, nebo zda je třeba provedení dalších úprav.

Obrázek 8: Přehled informací při pozorování bota

7.4 Postup při implementaci vlastního bota

Dále budou postupně uvedeny kroky jak můžeme postupovat při implementaci vlastního bota:

1. Datové struktury:
    - Analýza: Základní projekt přistupuje k většině informací pomocí struktury zvané "GameTickPacket". Tento přístup však způsobuje určitou neprůhlednost při získávání potřebných proměnných a komplikuje manipulaci s daty.
    - Návrh: Abychom dosáhli efektivnější práce s daty, plánujeme vytvořit specializované datové struktury pro balón, auta a branky. To nám umožní jednodušší manipulaci s klíčovými hodnotami potřebnými pro řízení a navigaci vozidla.
    - Implementace: Struktury pro auto a balón budou obsahovat inicializační metodu a metodu pro aktualizaci hodnot, která bude spouštěna v každém tiku. Každý z těchto objektů bude obsahovat tři typy hodnot: numerické, boolean a objektové hodnoty typu Vec3, který zahrnuje hodnoty pro tři osy a příslušné operace.
2. Výkop:
    - Analýza: Po spuštění rychle narazíme na první problém, kdy náš bot prohrává při výkopu. Tento problém je způsoben tím, že náš bot v daný okamžik nepoužívá boost a vždy, když dosáhne rychlosti mezi 750j a 800j (hodnoty uvadíme v "jednotkách"- vzdálenost mezi dvěma sousedními body), udělá flip, který nám v tu chvíli neposkytuje žádnou výhodu.
    - Návrh: Jelikož se výkop opakuje pravidelně, můžeme pro něj vytvořit více specifický plán. Nejprve musíme detekovat, že se jedná o výkop. Poté upravíme bod, ke kterému bot bude směřovat tak, aby v případě, že začínáme na stranách, poslal míč směrem do brány a ne směrem do zdi. Nakonec budeme muset odhadnout vzdálenost, na které bot musí provést flip, aby balón zasáhl ve středu a tím ho nevymrštil příliš do vzduchu.
    - Implementace: Výkop poznáme z dvou dat, která nám přicházejí v "packet.game_info", a to, "is_round_active"a "is_kickoff_pause". Obě musí být nastaveny na "true". Bod, ke kterému budeme od začátku směřovat, posuneme na y-ose o 180j. Stranu určíme podle hodnoty našeho týmu (1, pokud je team == 1, jinak -1). Vzdálenost auta od míče zjistíme odečtením pozice auta od míče. Během výkopu nastavíme, aby se používal veškerý dostupný boost. Po testování nastavíme, aby se flip prováděl při vzdálenosti 600j od pozice, kam směřujeme. Při této implementaci se již nestane, že bychom z výkopu dostávali góly.

3. Bránění/Útočení:

    - Analýza: Poté, co se dostaneme za výkop, se bot chová tak, že v případě, že míč je od hráče více jak 1500j, tak bot jede na pozici, kam se míč dostane za jednu vteřinu, a pokud je blíže, tak jede přímo do míče. Obě dvě chování v tuto chvíli způsobují, že bot nerozlišuje mezi svou a nepřátelskou stranou a vůbec se nesnaží bránit svou bránu nebo střílet na protihráčovu bránu.
    - Návrh: Nikdy není výhodné být blíže k nepřátelské bráně než míč, a proto vždy, když se nacházíme dále od naší brány než míč, směřujeme auto do středu naší brány. Nicméně se musíme vyhnout situaci, kdy bychom míč tlačili před sebou, což by mohlo vést k vlastnímu gólu. Tento problém řešíme tak, že míříme na bod míče nejbližší k ose x, takže v případě, že se k míči dostaneme, ho odstrkujeme směrem mimo naši bránu.
    - Implementace: Nejprve spočítáme vzdálenost míče od naší brány a vzdálenost našeho auta od brány. Pokud je vzdálenost auta od brány větší než vzdálenost míče od brány, zvolíme střed brány jako cíl a směřujeme k němu. Zda se míč nachází před autem a můžeme ho tedy vést směrem od brány, zjistíme pomocí porovnání směrů pohybů.

      Určíme směr pohybu míče a auta tím, že vezmeme jejich rychlost a vytvoříme z nich normály. Nad těmito dvěma normalizovanými vektory provedeme vektorový součin, čímž získáme jejich podobnost v hodnotě od -1 do 1, přičemž 1 znamená, že jsou ve stejném směru. Tuto hodnotu převedeme na radiány pomocí arccosinu a následně na stupně.

      Pokud bude směr pohybu míče a auta podobný do rozmezí 15 stupňů, zaměříme se na vnitřní stranu míče. Vnitřní stranu míče určíme podle x-ové souřadnice míče. Vybereme stranu, která je blíže k hodnotě 0, protože osa x je uprostřed hřiště.

4. Předpověď balónu:

    - Analýza: Dalším problémem bude zlepšit naše útočné schopnosti. Začneme predikcí míče. Tuto předpověď umožňuje vestavěná funkce get_ball_prediction_struct. V tuto chvíli předpovídáme, kde se bude míč nacházet za vteřinu. Tato funkce nám vrací list objektů obsahující informace o pozici míče v určitém čase. Což v případě, že se míč pohybuje velkou rychlostí nebo když jsme od míče daleko, způsobuje získání špatného směru.
    - Návrh: Lepší bude, když se nám podaří nalézt společnou budoucí pozici. V ideálním případě bychom měli zohlednit dobu nasměrování na míč, nicméně k tomu v tuto chvíli nemáme dostatek informací. Proto musíme určit dobu, za kterou je auto schopné dorazit k nejbližší předpokládané pozici míče.

- Implementace: Nejprve získáme data o budoucnosti míče z funkce get_ball_prediction_struct, kde pro každý "Slice"(pozici míče v budoucnosti) budeme počítat, jak dlouho trvá se dostat z pozice, kde auto je, do místa, kde míč bude, rychlostí kterou v tuto chvíli auto jede. V případě, že doba příjezdu bude nižší než doba daného "Slice", vybereme pozici a ukončíme cyklus.

5. Bod zásahu:

   - Analýza: Útočná schopnost je velmi nízká. V současné chvíli se nám nedaří změnit směr míče jinam, než ve směru, kterým k míči přijíždíme.
   - Návrh: Nalezneme bod na míči, který je optimální pro zásah, což způsobí, že náš směr nebude přímo do středu míče, To způsobí mírnou úpravu trajektorie míče směrem k nepřátelské bráně. Ideální bod stanovíme jako takový bod, který prochází směrovým vektorem od brány k míči. V této implementaci budeme ignorovat směr, rychlost a výšku míče, které mají vliv na ideální bod.
   - Implementace: Nejprve získáme směrový vektor od míče do středu nepřátelské brány, tento vektor normalizujeme, abychom získali pouze směr. Poté tento směr invertujeme, aby ukazoval na opačnou stranu, než je nepřátelská brána, a nakonec ho vynásobíme poloměrem míče. Výsledný vektor přičteme k poloze míče a získáme tak ideální střelecký bod, ke kterému budeme směrovat auto.

6. Směr příjezdu k balónu:

   - Analýza: Útočná schopnost je stále velmi nízká. Směr, ze kterého přijíždíme k bodu, je většinou mimo směr nepřátelské brány.
   - Návrh: Pro zlepšení úhlu, pod kterým přijíždíme na balón, můžeme zvolit pomocný bod ve výhodném směru a vzdálenosti od míče, a v určitou chvíli ho změnit na míč a tím docílit lepšího směru pro najíždění na míč. V potaz je potřeba brát předpověd balónu a meze hřiště. Tato implementace nedosáhne ideálního směru při nárazu na míč, nicméně ho zlepší.
   - Implementace: Implementujeme pomocný bod ve směru od ideálního bodu zásahu. V našem případě je tento bod vzdálený 800 jednotek od předpokládané pozice střetnutí s balónem. Díky posunutí bodu o 800 jednotek se může stát, že nastavíme cílový bod mimo hřiště, což by mohlo způsobit, že naše auto bude často najíždět na stěnu. Tomuto problému předcházíme tím, že omezíme cílový bod na obou osách. Na ose X nesmí hodnota přesáhnout 5000 jednotek a na ose Y 4000 jednotek. Při útoku vždy míříme na tento pomocný bod, dokud nebudeme k míči blíže než 800 jednotek. Jakmile je tato podmínka splněna, zaměříme se na předpokládaný bod pro střelu na bránu.

7. Střelba

    - Analýza: V tuto chvíli auto věčinou balón pouze tlačí před sebou, protože nemá dostatečnou rychlost.
    - Návrh: Při jízdě k míči bychom měli zjistit, zda auto směřuje k bodu, kam chceme, aby míč zasáhlo. Pokud ano, auto by mělo zrychlit, aby mělo větší vliv na změnu směru míče nebo aby míč putoval rychleji do nepřátelské branky. Auto by mělo skočit, čímž ještě zvýší svoji rychlost a zasáhne míč tak, aby ho zasáhl blíže ke středu.
    - Implementace: Nejdříve potřebujeme získat vektor od auta k bodu, na který míří, a vektor směru auta. Oba vektory normalizujeme a budeme ignorovat výšku. Poté tyto vektory porovnáme. Pokud se liší pouze o velikost stanovené odchylky. Pokud splňujeme podmínky a jsme od míče vzdáleni pouze 400 jednotek, tak vyskočíme. Při skoku se nesmí používat boost jinak bychom míč mohli přeletět.

8. Změna směru:

    - Analýza: Při náhlé změně směru a při velkých rychlostech se auto otáčí ve velkých obloucích, i když je balón kousek od nás.
    - Návrh: V případě, že auto bude mít podstatně jiný směr než je směr k cíli a auto se bude pohybovat velkou rychlostí, auto zpomalíme, aby se bylo schopné se otočit na menším prostoru.
    - Implementace: Vypočítáme podobnost směru jízdy auta společně se směrem od auta k cíli, podobně jako u zjišťování, jestli se balón nepohybuje stejným směrem. V případě, že rozdíl těchto směrů bude větší než 45 stupňů a rychlost auta bude vyšší než 1000j, změníme směr akcelerace směrem dozadu, což způsobí náhlé zpomalení auta a zlepšení manévrovatelnosti.

9. Používání boostu:

    - Analýza: Bot používá boost pouze při výkopu, ale boost, který nasbírá během hraní, nevyužívá.
    - Návrh: Díky tomu, že bot nepoužívá boost, se pomalu vrací ke své bráně, aby zamezil protihráči v dání gólu, a stejně tak při střelbě na nepřátelskou bránu se pohybuje tak pomalu, že balón sotva odrazí. Bot by měl v případě, že se vrací k bráně, použít boost, aby se pokusil dosáhnout maximální rychlosti, která je 2300j jednotek. Díky tomu by se k danému místu měl dostal co nejrychleji. Stejně tak v případě, kdy se bude snažit střílet na bránu, by měl použít boost, aby zvýšil svou rychlost při zásahu a tím zvětšil vzdálenost, na kterou míč odrazí.
    - Implementace: Pokud se nachazí bot v módu obrana, aktivujeme boost. Boost také aktivujeme v případě, že se bot připravuje na střelu, což znamená že míří na cílový bot na míči.

10. Sběr boostu:

    - Analýza: Bot používá pouze boost, na který najede během jízdy k cíli. Nesbírá boosty, které jsou v jeho blízkosti, což způsobuje, že je často bez boostu. To má za následek, že při návratu k bráně často nemáme dostatek boostu na to, abychom se k ní dostali dříve než soupeř.
    - Návrh: Z hry jsme schopni získat informace o všech boostech ve hře a o tom, zda jsou aktivní nebo ne. Jelikož hrajeme 1vs1, většinou není moc času na to sbírat boosty po mapě, protože bychom tak dávali hodně prostoru soupeři. Proto budeme sbírat velké boosty jen pokud budou poblíž auta.
    - Implementace: Ze seznamu všech boostů vyfiltrujeme pouze velké boosty. Z těch vyfiltrujeme boosty, které nejsou aktivní a potom mezi nimi nalezneme nejbližší z nich. Pokud je míč dále než 1000 jednotek od nás a nejbližší boost je k botovi blíže než 1000 jednotek, zamíříme pro něj. Pro boost však nepojedeme, pokud náš bot má více než 40 % paliva.

7.5 Další možnosti vylepšení

Náš bot by v tuto chvíli měl zvládat základní operace na hřišti. Nicméně všechny tyto operace lze dále rozšířit a vylepšit, aby byl bot efektivnější a chytřejší v jeho rozhodování a pohybu.

Například použitím Bézierovy křivky pro nájezd do výhodné polohy pro střelbu by se urychlil jeho následný pohyb k míči a zvýšila se rychlost, s níž do něj bude schopný narazit. Toto řešení by ovšem potřebovalo další podpůrné úpravy, jako například: lepší znalost zatáčení bota v různých rychlostech, abychom byli schopni sestrojit křivku, po které je bot skutečně schopný jet. Také bychom museli být schopni určit, jak dlouho by mu daný manévr trval, aby byl schopen brát v potaz, kde se bude nacházet míč. Další vylepšení by mohlo být používání dvojtého skoku pro změnu směku jízdy nebo optimalizování pohybu po stěnách hracího pole.

Závěr

RLBot framework představuje vynikající nástroj pro vytváření botů do hry Rocket League. Nejenže nabízí uživatelsky přívětivé vývojové prostředí, ale také poskytuje celou řadu nástrojů pro testování a aktivní komunitu, která je ochotná sdílet potřebné informace a návody.

Vytvoření silného bota však nebylo snadným úkolem. Ukázalo se, že to vyžaduje pokročilé znalosti herní strategie a analytické geometrie. Komplexní porozumění mechanikám hry, kombinované s matematickou přesností, je klíčem k úspěchu v této oblasti. Bohužel se nám nepodařilo aby náš bot byl schopen porazit bota od samotných vývojářů hry.

Mezi největší nedostatky našeho bota patří schopnost zasažení míče ve vzduchu a pohyb po křivce, která by ho pozicovala do ideálního úhlu pro střelbu. Tyto chybějící dovednosti se ukázaly jako klíčové v mnoha situacích a vedly k našemu omezenému úspěchu v porovnání s boty vývojářů.

Nicméně, můžeme s důvěrou tvrdit, že během procesu tvorby bota se nám podařilo podrobně prozkoumat framework RLBot, pochopit základy implementace bota a prohloubit naše znalosti řešení operací v prostoru. Tato zkušenost nám poskytla cenné lekce.

Přestože RLBot poskytuje skvělou možnost pro zlepšení našich znalostí a rozvoj dovedností v oblasti umělé inteligence, není nejefektivnější volbou pro tvorbu vysoce konkurenceschopného bota. Tato platforma vyžaduje specifický přístup k řadě situací, ve kterých je obtížné určit správný manévr. Pro tvorbu silnějšího bota bychom proto doporučili využít alternativní RLGym framework.

RLGym framework využívá posilované učení (Reinforcement learning), což ho činí výjimečným nástrojem pro vývoj pokročilých botů. Sice vyžaduje delší dobu pro učení, ale díky své přesnosti a schopnosti učit se z vlastních chyb dominují boti vyvinutí pomocí tohoto frameworku současným turnajům mezi boty.

Celkově lze říci, že proces vytváření bota je velmi zajímavý. Tato zkušenost nám nejen otevřela dveře do světa umělé inteligence a herního vývoje, ale také posílila naše schopnosti analytického myšlení a problémového řešení. Všem nadšencům do Rocket League nebo těm, kdo by chtěli prohloubit své znalosti analytické geometrie nebo strojového učení, můžeme RLBot framework doporučit.

Conclusions

The RLBot framework represents an excellent tool for creating bots for the game Rocket League. It not only offers a user-friendly development environment but also provides a wide range of tools for testing and an active community that is willing to share necessary information and guides.

Creating a strong bot, was not an easy task. It turned out that it requires advanced knowledge of game strategy and analytic geometry. Comprehensive understanding of the game's mechanics, combined with mathematical precision, is the key to success in this area. Unfortunately, we were unable to make our bot capable of defeating a bot created by the game's developers themselves. Among the biggest shortcomings of our bot are the ability to hit the ball in the air and movement along a curve that would position it at the ideal angle for shooting. These missing skills proved to be crucial in many situations and led to our limited success compared to the developers' bots.

Nevertheless, we can confidently say that during the process of creating the bot, we managed to thoroughly explore the RLBot framework, understand the basics of bot implementation, and deepen our knowledge of spatial operations. This experience has provided us with valuable lessons.

Although the RLBot framework offers a great opportunity to enhance our knowledge and develop skills in the field of artificial intelligence, it is not the most efficient choice for creating a highly competitive bot. This platform requires a specific approach to many situations where it is difficult to determine the correct maneuver. For creating a stronger bot, we would, therefore, recommend using the alternative RLGym framework.

The RLGym framework employs reinforcement learning, making it an exceptional tool for the development of advanced bots. While it requires a longer learning period, thanks to its accuracy and ability to learn from its mistakes, bots developed using this framework dominate current tournaments between bots.

Overall, the process of creating a bot is very interesting and enriches the deepening of our knowledge in the respective areas. This experience has not only opened the doors to the world of artificial intelligence and game development but also strengthened our analytical thinking and problem-solving abilities. To all enthusiasts of Rocket League or those who would like to deepen their knowledge of analytical geometry or machine learning, we can recommend the RLBot framework.

A Obsah elektronických dat

Elektronická data obsahují soubory s implementací bota, text bakalářské práce a soubor popisující, jak bota spustit.

Složky s implementací botu obsahují konfigurační soubory společně s adresářem src, kde se nachází soubor bot.py s implementací bota. Dále se v něm nacházejí pomocné struktury, funkce a metadata.

`text/`
: Adresář s textem práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech (textových) příloh, a všechny soubory potřebné pro bezproblémové vytvoření PDF dokumentu textu (případně v ZIP archivu), tj. zdrojový text textu a příloh, vložené obrázky, apod.

`README.txt`
: Soubor obsahuje popis postupu pro spuštění bota. Obsahuje informace o stažení RLBot knihovny, Epic Games Launcheru, stažení Rocket League, kam je potřeba vložit kód bota a jak ho v RLBot GUI spustit.

`MyFirstPythonBot`
: Adresář s implementací prvního zkušebního bota.

`BC_BOT2/src`
: Adresář s implementací druhého zkušebního bota.

`BC_BOT3/src`
: Adresář s implementací třetího zkušebního bota.

`BC_BOT4/src`
: Adresář s implementací čtvrtého zkušebního bota.

`BC_BOT_v1`
: Adresář s implementací prvního kroku při implementaci nového bota (datové struktury).

`BC_BOT_v2`
: Adresář s implementací druhého kroku při implementaci nového bota (vykop).

`BC_BOT_v3`
: Adresář s implementací třetího kroku při implementaci nového bota (útok/obrana).

`BC_BOT_v4`
: Adresář s implementací čtvrtého kroku při implementaci nového bota (předpověď pohybu míče).

BC_BOT_v5
: Adresář s implementací pátého kroku při implementaci nového bota (určení bodu zásahu).

BC_BOT_v6
: Adresář s implementací šestého kroku při implementaci nového bota (zlepšení nájezdního úhlu).

BC_BOT_v7
: Adresář s implementací sedmého kroku při implementaci nového bota (střela na bránu).

BC_BOT_v8
: Adresář s implementací osmého kroku při implementaci nového bota.

BC_BOT_v9
: Adresář s implementací devátého kroku při implementaci nového bota (boost management).

Literatura

[1] STAFF, WIKI (comp.). *Rocket League Wiki*. Dostupný z: ⟨https://rocketleague.fandom.com/wiki/Rocket_League_Wiki⟩.

[2] STAFF, WIKI (comp.). *Rocket League Wiki Boosts*. Dostupný z: ⟨https://rocketleague.fandom.com/wiki/Boost⟩.

[3] IJ (comp.). *Guide to Boost Management*. Dostupný z: ⟨https://steamcommunity.com/sharedfiles/filedetails/?id=544526380⟩.

[4] RLBot, collective (comp.). *RLBot*. Dostupný z: ⟨https://rlbot.org/⟩.

[5] RLBot, collective (comp.). *RLBot Wiki*. Dostupný z: ⟨https://github.com/RLBot/RLBot/wiki⟩.

[6] RLBot, collective (comp.). *RLBot repozitář*. Dostupný z: ⟨https://github.com/RLBot/RLBot⟩.

[7] ddthj (comp.). *ddthj Gosling implementace*. Dostupný z: ⟨https://github.com/ddthj/Gosling⟩.

[8] virxerlu (comp.). *virxerlu github dokumentace*. Dostupný z: ⟨https://virxerlu.virxcase.dev/⟩.

[9] RLGym, collective (comp.). *RLGym*. Dostupný z: ⟨https://rlgym.org/⟩.

[10] Wikipedie (sest.). *Skalární součin*. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Skal%C3%A1rn%C3%AD_sou%C4%8Din⟩.

[11] Wikipedie (sest.). *Analytická geometrie* [online]. [cit. 2023-4-22]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Vektorov%C3%BD_sou%C4%8Din⟩.

[12] Wikipedie (comp.). *Normalizovaný vektor*. Dostupný z: ⟨https://en.wikipedia.org/wiki/Unit_vector⟩.

[13] Wikipedie (comp.). *Délka vektoru*. Dostupný z: ⟨https://en.wikipedia.org/wiki/Magnitude_(mathematics)⟩.

[14] Psyonix (comp.). *Trackovatelna data ze hry Rocket league*. Dostupný z: ⟨https://rocketleague.tracker.network/rocket-league/distribution⟩.