Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Formálne metódy a strojové učenie vo virtuálnom priestore: interaktívne virtuálne prostredie

Bakalárska práca

2019 **Róbert Rauch**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Formálne metódy a strojové učenie vo virtuálnom priestore: interaktívne virtuálne prostredie

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Ing. Štefan Korečko, PhD.
Konzultant:	

Košice 2019 **Róbert Rauch**

Názov práce:	Formálne metódy a strojové učenie vo virtuálnom priestore: interaktívne virtuálne prostredie
Pracovisko:	Katedra počítačov a informatiky, Technická univerzita v Košiciach
Autor:	Róbert Rauch
Školiteľ:	Ing. Štefan Korečko, PhD.
Konzultant:	
Dátum:	24. 5. 2019
Kľúčové slová:	3D hra, hra pre viacerých hráčov, virtuálne prostredie, autonómna entita, Unity, Unet, MQTT

Abstrakt: Táto práca sa venuje vytvoreniu virtuálneho priestoru, ktorý bude slúžiť ako testovacie prostredie pre autonómnu entitu. Virtuálny priestor bude predstavovať hra pre viacerých hráčov. Hra je jednoduchá strieľačka s cieľom obsadiť vlajku. Hra bola vytvorená pomocou herného rámca Unity. Použité prostriedky sú zadarmo stiahnuteľné v Asset Store, čo je v Unity obchod na prostriedky. Pripojenie hráčov online je riešené pomocou rámca Unet, ktorý poskytuje Unity. Hra sa dá hrať s viacerými hráčmi na jednej inštancií hry pomocou rozdelených obrazoviek a mobilného ovládača. Maximálny počet hráčov na rozdelených obrazovkách je 4. Ovládač je vytvorený ako webová stránka. Na vytvorenie komunikácie medzi hrou a mobilným ovládačom je použitý protokol MQTT.

Thesis title:	Formal Methods and Machine Learning in Virtual Space: Interactive Virtual Environment
Department:	Department of Computers and Informatics, Technical University of Košice
Author:	Róbert Rauch
Supervisor:	Ing. Štefan Korečko, PhD.
Tutor:	
Date:	24. 5. 2019
Keywords:	3D Game, multiplayer, virtual environment, autonomous entity, Unity, Unet, MQTT

Abstract: Focus of this thesis is to create a virtual environment. This environment will be used as a testing ground for an autonomous entity. The virtual environment is a simple online first-person shooter. Players in the game have one goal to capture and hold the objective. This game is created with Unity engine. Assets used in this game are free to download from the Unity's Asset Store. Online connection and synchronization is developed using the Unet framework. Players can play this game on a split screen with up to 4 players using mobile controllers. This controller is developed as a simple web page using MQTT protocol for communication between the game and controller.

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE BAKALÁRSKEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Formálne metódy a strojové učenie vo virtuálnom priestore: interaktívne virtuálne prostredie

Formal Methods and Machine Learning in Virtual Space: Interactive Virtual Environment

Študent: **Róbert Rauch**
Školiteľ: **Ing. Štefan Korečko, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie bakalárskej práce:

1. Analyzovať súčasné softvérové platformy pre vývoj 3D počítačových hier z hľadiska dostupnosti kompatibilných riešení pre strojové učenie a možností integrácie formálne vyvinutého kódu.
2. V analýze sa zamerať na platformy dostupné zdarma.
3. Na základe analýzy pre vybranú platformu navrhnúť a implementovať hru pre viacerých hráčov počas hrania ktorej bude prebiehať učenie sa autonómnej entity, realizovanej s použitím strojového učenia a formálnych metód.
4. V spolupráci s riešiteľom súvisiacej záverečnej práce uvedenú entitu integrovať do vyvinutej hry, experimentálne realizovať proces učenia sa a vyhodnotiť ho.
5. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 24.05.2019
Dátum zadania bakalárskej práce: 31.10.2018

.....
doc. Ing. Jaroslav Porubän, PhD.
vedúci garantujúceho pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -10-

.....
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 24.5.2019 .....

Vlastnoručný podpis

Poďakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce Ing. Štefanovi Korečkovi, PhD. za jeho čas a odborné vedenie počas riešenia mojej záverečnej práce.

Rovnako by som sa rád poďakoval kolegovi a bratrancovi Danielovi Slinčákovi, bez ktorého by tento projekt nebol kompletný ako aj svojim rodičom a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Každý softvér by sa mal pred jeho vydaním dôkladne otestovať. To je hlavne dôležité pre softvér, ktorý bude integrovaný do reálneho sveta vo forme autonómnych strojov. Testovanie takéhoto softvéru v reálnom svete však môže byť časovo a finančne veľmi náročné. Pokiaľ je testovaný softvér napríklad robotom, môže spôsobiť aj rôzne škody na majetku či zdraví. Obísť to vieme nasadením testovaného softvéru do virtuálneho prostredia, ktoré sa bude čo najviac podobať reálnemu svetu. V tomto virtuálnom svete môžeme jednoducho otestovať funkcionalitu nášho softvéru pred jeho nasadením do reálneho sveta.

Tvorba virtuálneho priestoru od základov je však stále časovo a teda aj finančne veľmi náročná. S týmto problémom sa môžeme obrátiť na herný priemysel, ktorý poskytuje rôzne prostriedky pre tvorbu virtuálneho priestoru. Samostatné virtuálne prostredie však bude fungovať pomocou určitých algoritmov, ktoré autonómna entita môže jednoducho predvídať a teda náš softvér by sa neotestoval správne. Preto je ideálne vytvorenie určitých mechaník a cieľov do hry, ktoré by mohli hráči vykonávať počas testovania. Tieto ciele je potrebné vytvárať v okolí autonómnej entity, čím vytvoríme jej okolie viac chaotické a nepredvídateľné, a teda sa toto okolie bude viac približovať skutočnej reprezentácií reálneho sveta.

Táto bakalárska práca je zameraná na tvorbu práve takéhoto virtuálneho sveta. Pri tvorbe sme sa zamysleli aj nad myšlienkou, či vytvorenie jednoduchej hry, v ktorej by jeden hráč chodil a spĺňal určité úlohy, bolo postačujúce pri testovaní softvéru. Pri takejto hre by sa agent počas celého životného cyklu nemusel ani dostať do kontaktu s hráčom. Preto sme začali uvažovať ako vytvoríme hru, v ktorej by bol softvér v neustálom kontakte s nejakým človekom. Z toho vznikol projekt, v ktorom by sme vytvorili hru pre viacerých hráčov, na ktorých by sme otestovali našu autonómnu entitu. V tomto prípade by bola väčšia šanca, že softvér by bol neustále kontrolovaný aspoň jedným hráčom.

1 Formulácia úlohy

Cieľom práce je analyzovať dostupné možnosti pre vývoj počítačových hier ako aj tvorba nového obsahu do už existujúcich hier. Pri tejto analýze sa pozrieme na možnosti pre vývoj autonómnej entity a formálne vyvinutého kódu. Všetky možné platformy musia byť zdarma, ako aj samotný vývoj. Po vybratí platformy vytvoríme hru pre viacerých hráčov, do ktorej sa bude vkladať autonómna entita. V hre vytvoríme cieľ pre hráčov tak, aby sa hráči nachádzali v okolí nášho testovaného agenta.

2 Výber platformy

V tejto kapitole sa zamierame na výber nástroja, v ktorom budeme naše virtuálne prostredie vytvárať. V tomto prostredí sa bude učiť náš agent strojového učenia. V nami vybranom prostredí by malo byť jednoduché ukladanie a načítavanie dát. Tieto dáta budeme totiž využívať na ukladanie naučeného modelu. Samozrejmosťou by malo byť aj jednoduché vloženie cudzieho softvéru, a teda nášho agenta, do virtuálneho prostredia, pričom musíme brať do úvahy fakt, že cudzí softvér môže byť napísaný v inom programovacom jazyku. Posledným dôležitým aspektom sú možnosti publikácie, aby si naše vytvorené prostredie mohlo zahrať čo najviac ľudí. V dnešnej dobe je veľa hier, ktoré poskytujú prostriedky na vytvorenie vlastných módov. Pokiaľ by sme cudzí softvér vložili do už existujúcej hry, to by nám uľahčilo veľa práce, pretože virtuálne prostredie by sme už mali vytvorené. Teda by sme už len vložili cudzí softvér do danej hry a vytvorili rôzne úlohy pre hráča. Preto sa na začiatku pozrieme na pár takýchto populárnych hier. Okrem módov pre hry sa pozrieme aj na herné rámce, v ktorých si vieme zložitejším spôsobom vytvoriť vlastné virtuálne prostredie.

2.1 Dota 2

Dota 2 je hra pre viacerých hráčov, ktorá od roku 2015 prešla do novej fázy[25], v ktorej pridali vývojári vytváranie vlastných módov. Tieto módy fungujú ako samostatné hry, ktoré si hráči po celom svete vedia stiahnuť.

2.1.1 Ukladanie a načítavanie naučeného modelu

Ako už bolo spomínane, ukladanie a načítanie naučeného modelu je jeden z najdôležitejších aspektov nášho projektu. Na ukladanie týchto dát nám postačí jednoduchý textový dokument, do ktorého sa budú ukladať naučené hodnoty. Dota

2 však neposkytuje samostatné servery pre tieto hry. Hranie viacerých hráčov je riešené tak, že prvý hráč, ktorý založil miestnosť do ktorej sa hráči napájajú, slúži aj ako server. A teda všetky serverové časti sa nachádzajú na klientovi, ku ktorým nemáme prístup. To je pre náš projekt problematické, avšak dá sa to obísť vďaka funkcií CreateHTTPRequest[18], pomocou ktorej by sme vedeli posielať požiadavky POST, na zapisovanie naučených dát na náš server po konci každej hry a požiadavky GET, ktorá by získala informácie z nášho serveru na začiatku každej hry.

2.1.2 Vkladanie cudzieho softvéru

Ďalším aspektom nášho projektu je vloženie cudzieho softvéru do nami vytvoreného virtuálneho prostredia. Táto problematika úzko súvisí s programovacím jazykom, ktorý sa používa pri vytváraní daných módov. Na vytvorenie módov sa používajú dva programovacie jazyky. Prvým programovacím jazykom je KeyValues[24], ktorý určuje štruktúru predmetov a iných hodnôt v hre. Pomocou skriptovacieho jazyka Dota Lua sa dá naprogramovať správanie daných predmetov. Skriptovací jazyk Lua je veľmi populárny a preto by nájdenie prostriedku, ktorý by slúžil na komunikáciu respektíve preklad cudzieho softvéru na skriptovací jazyk Lua nemusel byť veľký problém. Ak by bola na vytvorenie správania nášho agenta použitá napríklad B-Metóda, na preklad vieme použiť program BKPI[2], ktorý dokáže B-Metódu preložiť do jazykov Java a C#. Následne na preklad jazyka Java na Lua vieme použiť rôzne prekladače ako napríklad lua-java-tranlator[3] od používateľa Vadim Klevtsov. Problém by mohol nastať v situácií, kedy by si daný prostriedok nevedel poradiť s knižnicami, ktoré sú potrebné pre vytvorenie Dota módu.

2.1.3 Publikácia

Posledným krokom je zverejnenie nášho módu pre hráčov. Na to vývojári hry Dota 2 využívajú prostredie, ktoré sa nazýva Steam Workshop. V tomto prostredí si ľudia vedia prehľadávať všetky módy od ľudí, pričom si tieto módy vedia stiahnuť a následne hrať v prostredí hry Dota 2. Mód je možné stiahnuť okamžite po tom, čo ho zverejníme, nie je teda potrebné čakať na nejaké potvrdenie od vývojárov. Zverejniť mód na túto platformu môže hocikto, čo pre nás môže spôsobiť problémy,

pretože sa náš projekt môže v priebehu pár minút stratiť z prednej stránky Steam Workshopu, a teda by náš mód nepoznalo tak veľa hráčov.

2.1.4 OpenAI

OpenAI[23] je spoločnosť, ktorá sa zaoberá tvorbou umelej inteligencie. V roku 2017 vytvorili agenta[17], ktorý sa naučil hrať hru Dota 2, bez akýchkoľvek vnútorných informácií o hre. Po dlhej dobe učenia agenta sa rozhodli vyzvať najskúsenejších hráčov hry a to počas jej najväčšieho turnaja The International. V týchto zápasoch hráči prehrali, čo bola veľká výhra pre tím OpenAI. Aj keď jeden tím v Dote 2 tvorí päť hráčov, tieto hry boli iba jeden agent proti jednému hráčovi. To zmenili v roku 2018, kedy vytvorili OpenAI Five[13], čo je súbor piatich agentov spolupracujúcich na porazení nepriateľského tímu. V auguste 2018 vyhlásili že týto agenti porazili 99,95% hráčov.

2.2 Minecraft

Minecraft je hra s otvoreným svetom, ktorá nie je striktne pre viacerých hráčov, avšak vývojári podporujú servery, na ktorých ľudia vedia hrať spoločne. Podľa štatistík[11] má Minecraft 91 miliónov hráčov. Veľká popularita hry by jednoznačne pomohla pri testovaní nášho softvéru.

2.2.1 Zakladanie servera

Oproti hre Dota 2, kedy sa hráči napájali na server vytvorený jedným z hráčov, v hre Minecraft je potrebné vytvoriť server, ktorý bude bežať nepretržite. Kvôli tomu musí bežať na nejakom zariadení, na ktoré sa budú pripájať hráči. Za účelom vytvárania a udržiavania serverov existujú rôzne stránky. Tieto stránky sú však často platené, pričom stránky, ktoré túto službu poskytujú zadarmo sú značne obmedzujúce. Vývojári však poskytujú softvér na vytvorenie lokálneho servera. Pri vytvorení malej skupinky účastníkov a vytvorení takéhoto serveru vieme náš softvér otestovať na lokálnej sieti.

2.2.2 Ukladanie a načítavanie naučeného modelu

Keďže server beží nepretržite, náš agent by si svoj naučený model nemusel ukladať a načítavať zo žiadnej stránky. Namiesto toho by si periodicky ukladal model do textového dokumentu, z ktorého by si tieto dáta znova načítal, napríklad pri výpadku servera.

2.2.3 Vkladanie cudzieho softvéru

V hre Minecraft sa dajú vytvárať takzvané pluginy, ktoré dokážu pridávať rozličný obsah do hry, napríklad pridanie nového materiálu či predmetu. Takýto plugin je napríklad Bukkit[15], ktorý má voľne dostupný zdrojový kód. Po jeho stiahnutí si ho vieme jednoducho vložiť do nami zvoleného prostredia, napríklad IntelliJ či Eclipse a následne vieme vytvárať nový obsah do hry. Pluginy sú písane v Jave. Cudzí softvér nie je potrebné prispôsobovať, stačí iba prostriedok, ktorý by cudzí softvér preložil do Javy. Pokiaľ by správanie nášho agenta bolo definované B-Metódou, vieme použiť už spomínaný BKPI prekladač.

2.2.4 Malmo

Malmo[21] je projekt zameraný na tvorbu umelej inteligencie. Celý projekt je dostupný pre verejnosť a má voľne stiahnuteľný zdrojový kód. Tento softvér nám otvára možnosti, medzi ktoré patrí vytvorenie agenta, ktorý by voľne komunikoval s ostatnými agentmi alebo hráčmi. Táto funkcionalita komunikácie už je jeho súčasťou. Malmo je postavený na dvoch častiach. Prvá časť sa stará o agentove vnímanie sveta a dovoľuje mu v tomto svete vykonávať rôzne operácie. Druhou časťou je mód, pomocou ktorého sa zapína samotný agent, ktorý môže byť vytvorený pomocou rôznych programovacích jazykov.

2.3 Hry s otvoreným zdrojovým kódom

Dota 2 a Minecraft poskytuje prostriedky, pomocou ktorých vedia vývojári dopĺňať obsah do hry. Okrem takéhoto obmedzujúceho vytvárania obsahu do hry, existujú hry, ktoré majú voľne dostupný celý zdrojový kód, a tak vývojári po celom svete vedia voľne dopĺňať obsah do hry.

2.3.1 PlaneShift

PlaneShift[5] je hra zo žánru RPG, ktorá má voľne dostupný zdrojový kód ako pre klienta, tak aj pre server. Práve dostupnosť zdrojového kódu pre server nám umožňuje zapnúť server na našom stroji a nepretržite učiť nášho agenta, respektíve periodicky zapisovať naučený model do textového dokumentu. Aj napriek veku hry(14 rokov) je stále podporovaná a vylepšovaná vývojármi[19]. V januári roku 2017 prešli na nový herný rámec Unreal Engine. Unreal Engine je herný rámec, v ktorom sa hry vytvárajú pomocou jazyka C++, a teda vytvorenie nového obsahu do hry nie je vôbec problematické. Ak by bol cudzí softvér vytvorený v B-Metóde, postupovali by sme rovnako ako to bolo pri hre Dota 2, no na preklad do jazyka C++ nemusíme sťahovať žiaden prekladač. To je z dôvodu, že AtelierB používaný na tvorbu B-Metódy už obsahuje prekladač [1] do jazyka C++. Tento prekladač je zatiaľ iba experimentálny.

2.3.2 Worldforge

Worldforge[8] je ďalšou hrou, ktorá má voľne dostupný zdrojový kód. Táto hra využíva prostredie nazývané Ember. V tejto hre si každý hráč vie vytvoriť svoj vlastný server, ktorý slúži ako vlastný svet pre daného hráča. Hráči samozrejme vedia pozvať na svoj server aj ostatných hráčov. Prostredie Ember slúži na stavbu celého sveta. Hra poskytuje predinštalované prostriedky, respektíve modely, ktoré môže každý používať. To však nie je obmedzenie, keďže si takéto prostriedky vedia vývojári pridávať aj vlastné. Po vytvorení sveta, sa akcie vo svete vytvárajú pomocou kódovania v jazyku Python. Na preklad z programovacieho jazyka ako je Java vieme použiť prostriedok java2python[4] od používateľa Troy Melhase.

2.4 Roblox

Roblox je platforma, ktorá je veľmi podobná herným rámcom, jej základ je postavený na vytváraní hier v prostredí nazvanom Roblox Studio.

2.4.1 Ukladanie a načítavanie naučeného modelu

Pri Robloxe si nemusíme zapisovať naučený model do textového dokumentu. Namiesto nich budeme používať niečo, čo Roblox vývojári nazývajú dátové úložiská[14]. Tieto úložiská slúžia pre načítanie a uloženie údajov počas hrania.

2.4.2 Vkladanie cudzieho softvéru

Programovací jazyk[26] používaný na naprogramovanie správania je Roblox Lua, ktorý je zjednodušenou verziou jazyka Lua. Je to teda ten istý prípad ako pri Dote 2, kedy by knižnice, ktoré Roblox používa, mohli spôsobovať problém pri implementácií cudzieho softvéru. Pri preklade cudzieho softvéru by sme postupovali rovnakým spôsobom ako to bolo pri hre Dota 2.

2.4.3 Publikácia

Po vytvorení hry môžeme jednoducho zverejniť našu hru. A to tak, že v nastaveniach hry jednoducho zmeníme viditeľnosť hry na verejnú. Po zverejnení je naša hra dostupná na stiahnutie pre veľké množstvo hráčov.

2.5 Unity

Unity je jeden z herných rámcov, ktoré herný priemysel poskytuje. Okrem klienta na vytvorenie virtuálneho prostredia poskytuje taktiež veľké množstvo prostriedkov, ktoré sú zadarmo k stiahnutiu. Okrem prostriedkov ako sú modely či skripty, ktoré by nám uľahčili vývoj hry, poskytujú taktiež prostriedky na vytvorenie hier pre viacerých hráčov.

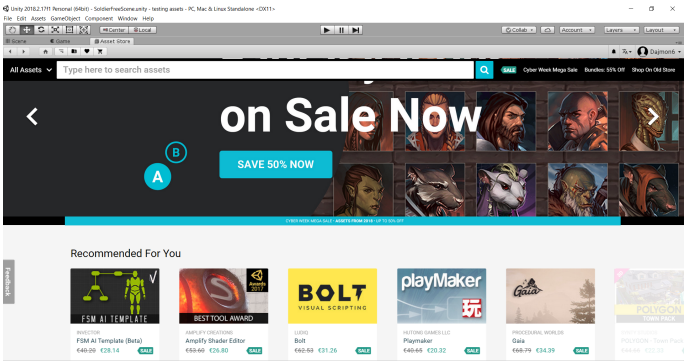
2.5.1 Publikácia

Ak chceme správne testovať náš softvér musíme ho vystaviť do styku s čo najviac hráčmi. Pri Unity to môže byť náročné, keďže nami vyvíjaná hra na začiatku nebude známa. Riešením tohto problému je publikácia našej hry na rôznych platformách ako je Steam či Origin. Avšak ani to nezaručuje, že si našu hru vyskúša čo najväčší počet ľudí. Preto je riskantné testovať softvér týmto spôsobom, avšak

stále je možnosťou nájsť si testovaciu skupinku ľudí, na ktorej by sa daný softvér otestoval.

2.5.2 Financovanie

V tejto sekcii sa pozrieme na potrebné financovanie pri používaní tejto platformy. Ako prvé je potrebné pozrieť na financovanie samotnej platformy. Na to Unity využíva mesačné plány[7]. Základným plánom je osobný, ktorý je úplne zadarmo. Unity má pre tento plán obmedzenie v zmysle, že osoba vlastniaca danú licenciu nesmie zarábať alebo mať rozpočet na projekty viac ako sto tisíc dolárov na rok. Stredný plán nazvaný plus mení spomínanú podmienku na dvesto tisíc na rok. Okrem toho Unity pri tomto pláne poskytuje zľavy v ich obchode na prostriedky Asset Store. Tento plán však už nie je zadarmo, stojí 25 dolárov na mesiac. Posledným plánom je profesionál, ktorý spomínanú podmienku zase mení, a to tak, že ročný zárobok, respektíve rozpočet môže dosahovať akejkoľvek výšky. Samozrejmosťou sú spomínane výhody z plus verzie. Cena tohto mesačného plánu je 125 dolárov na mesiac.



Obr. 2.1: Obchod prostriedkov

Pri vývoji virtuálneho prostredia nastávajú finančné náklady aj na textúry, modely, zvuky či rôzne skripty. Za týmto účelom Unity používa už spomínaný obchod prostriedkov, kde sa dajú tieto textúry, modely, zvuky či skripty stiahnuť a jednoducho importovať do projektu. Na obrázku 2.1 môžeme vidieť ukážku hlavnej stránky tohto obchodu. Okrem toho môžeme vidieť aj ceny niektorých prostriedkov. Tieto prostriedky sú vytvárane používateľmi a oni si stanovujú ceny svojich prostriedkov. Preto sa v tomto obchode nachádza aj nespočetné množstvo prostriedkov zadarmo.

2.5.3 Výskum strojového učenia v Unity

Unity od septembra roku 2017 poskytuje voľne dostupný doplnok, pomocou ktorého vývojári môžu používať algoritmy pre strojové učenie[6] vo svojich projektoch. Jedna z hier, ktorá využíva tento doplnok je Metal Warfare[20]. Táto hra je zameraná na trénovanie AI pomocou odmien. Po natrénovaní AI hráči vedia vyzvať do súboja AI ostatných hráčov. Ďalším projektom pre strojové učenie je robot na varenie palaciniek[10]. Pri tejto hre sa robot učí hádzať palacinku na tanier. Pričom robot dostáva odmeny za správne hodenie palacinky na tanier. Nakoniec bol vytvorený robot, ktorý taktiež pomocou odmien prenesie maslo popri prekážkach k palacinkám. Úlohou poslednej hry[16], na ktorú sa pozrieme, je skrývanie sa pred nehrateľnou postavou, ktorá mala preddefinované určité správanie. Zo začiatku mala byť táto hra prispôsobená tak, aby sa agent strojového učenia stále vyhýbal nehrateľnej postave. Tá ho nakoniec stále našla a preto autor zmenil cieľ hry tak, aby agent strojového učenia vedel hru vyhrať.

2.5.4 Ukladanie a načítavanie naučeného modelu

Pri predošlých platformách by sme museli ukladanie a načítanie naučeného modelu riešiť my. Pri použití prostriedku od Unity na tvorbu agentov nám túto funkcionalitu zabezpečuje už tento prostriedok. My teda musíme iba stanoviť cestu, do ktorej sa bude naučený model ukladať, respektíve z ktorého sa bude načítavať.

2.5.5 Online pripojenie

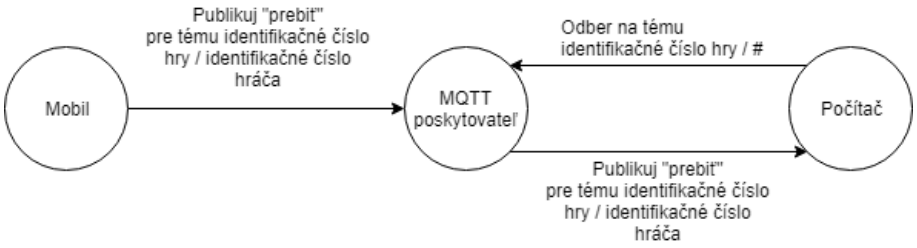
Keďže vytvárame hru pre viacerých hráčov, je potrebne nejakým spôsobom vytvoriť server, na ktorý by sa hráči vedeli pripojiť. Na výber máme viacero možností, najprívetivejšie však sú Unet[12] a Photon[9]. Unet je rámec, ktorý je vstavaný už do Unity a vďaka tomu nie je potrebné sťahovať žiadne knižnice. Unet pripojenie hráčov je zamerané na vytváranie servera na klientovej časti. A teda hráč, ktorý zakladá hru sa stáva serverom. Nevýhodou Unet rámca je, že Unity s týmto rámcom končí, a teda bude v budúcnosti zrušený. Unity úplne zruší Unet až v roku 2022, čo nám dáva dostatok času na náš výskum. Photon je druhý rámec pre pripojenie hráčov online. V tomto prípade sa server nevytvára na strane klienta ale na Photon serveroch. To zaručuje lepšie pripojenie a rýchlejšiu odozvu k serveru pre

hráčoch v rôznych krajinách. Taktiež klienti nebudú odpojení, po odpojení hráča, ktorý vytvoril server, ako by to bolo pri Unet rámci. Photon však vyžaduje stiahnutie knižníc a je zložitejší na tvorbu online hry, preto je pre náš relatívne malý projekt vhodnejšie použitie Unet rámca.

2.5.6 OpenLab

OpenLab je interaktívny priestor, ktorý využíva rôzne technológie. Nachádza sa v otvorenom vestibule na piatom poschodí Technickej univerzity v Košiciach. Skladá sa z viacerých častí, kde pre nás je najatraktívnejšia časť skladajúca sa z deviatich obrazoviek. Na týchto obrazovkách by sme vedeli spustiť našu hru, kde by ju žiaci mohli testovať. Pri predošlých platformách táto možnosť nebola veľmi prívetivá, pretože tieto hry potrebovali veľký súbor na stiahnutie, ktorý by sa ťažko dával na OpenLab. Druhým nedostatkom pre predošlé platformy je grafická náročnosť, ktorú nevieme veľmi znížiť a teda hra by nemusela bežať plynule.

Na komunikáciu s týmto počítačom by sme potrebovali ovládač. Najlepšie riešenie je vytvorenie ovládača pre mobilné zariadenie. Na vytvorenie ovládača máme dve možnosti. Jeho vytvorenie priamo pomocou Unity. Tento ovládač by používal Unet, ktorý by posielal údaje do hry. Pri tomto spôsobe je však potrebné aby si užívateľ nainštaloval náš mobilný ovládač.



Obr. 2.2: Ukážka komunikácie hry a mobilného ovládača

Druhý spôsob ako vytvoriť mobilný ovládač je pomocou MQTT protokola[22]. Náš ovládač by bola teda jednoduchá stránka, na ktorej by bežal náš ovládač, ktorého funkcionalitu môžeme vidieť na obrázku 2.2. V tomto prípade pripojené mobilné zariadenie posiela jednoduchú správu MQTT serveru s určitou témou a tento MQTT server posiela správu hre, ktorá sa pri zapnutí hry prihlásila k odberu tej istej témy.

2.6 Vyhodnotenie

V tejto časti si povieme jednoznačné výhody a nevýhody spomínaných platforiem a určíme platformu, ktorá najviac vyhovuje naším požiadavkám.

Vyhodnotenie platforiem						
Názov platformy	Programovací jazyk	Napájanie hráčov	Zber dát	prostredie	Publikácia	Virtuálne prostredie
Dota 2	Dota Lua	na klienta	zložitý	Dota 2 Workshop Tools	Steam Workshop	predvytvorené
Minecraft	Java	na server	jednoduchý	IntelliJ, Eclipse a iné	Žiadna	predvytvorené
PlaneShift	C++	na server	jednoduchý	Unreal Engine	žiadna	predvytvorené možné modifikácie
Worldforge	Python	na server	jednoduchý	Ember	žiadna	predvytvorené možné modifikácie
Roblox	Roblox Lua	na server	jednoduchý	Roblox Studio	Roblox	nami vytvorené
Unity	C#	neobmedzujúce	mierne zložitý	Unity	žiadna	nami vytvorené

Tabuľka 2.1: Vyhodnotenie platforiem

V tabuľke 2.1 môžeme vidieť stručný prehľad jednotlivých platforiem, z ktorých si vieme určiť výhody a nevýhody daných platfórm. Pre Dota 2 sú tieto ne-

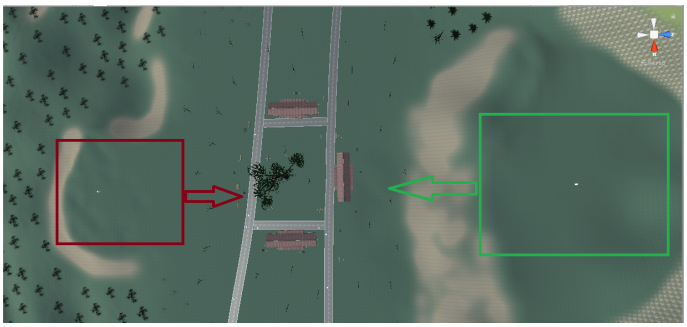
výhody ako programovací jazyk, tak aj napájanie hráčov na klienta, z čoho vzniká zložitý prístup k dátam. Jednoznačnou výhodou je však Steam Workshop, ktorý je veľmi populárny a tak by bol náš mód dostupný pre veľké množstvo hráčov. Minecraft má viaceré výhody, od prostredia, ktoré je pre Java vývojárov dosť známe, tak aj napájanie hráčov na server, a teda aj jednoduchý zber dát. Nevýhodou je však to, že náš server nebude na žiadnom veľkom zozname serverov, kvôli čomu sa náš server nemusí stať populárnym. Roblox má nevýhodu programovací jazyk a vytváranie celého virtuálneho prostredia. Výhodou však je jednoduchý zber dát a jednoduchý prístup hráčov pre našu hru. Ďalšou výhodou je, že Roblox má už vytvorenú kostru, ktorá zahŕňa pripájanie viacerých hráčov. Vďaka tomu by sme nemuseli riešiť správanie aktérov v hre počas ich interakcie. Hlavnou nevýhodou Unity je to, že celé virtuálne prostredie bude vytvorené nami. To však nemusí byť zápor, keďže vďaka tomu nie sme ničím obmedzovaní pri vytváraní úloh pre hráča alebo vkladaní cudzieho softvéru. Okrem toho musíme stále brať do úvahy hru pre viacerých hráčov a aj keď nie sme obmedzení vzájomnou komunikáciou hráčov, je to jednoznačne náročnejší proces. Preto je Unity jednou z časovo najnáročnejších variant. Okrem toho je nevýhoda aj publikácia pretože nevieme zaručiť, že naša hra bude u hráčov populárna. Výhodou Unity je jednoznačne fakt, že podporuje komplexný programovací jazyk C#. Ďalšou výhodou, ktorá sa v tabuľke nespomína, je OpenLab, vďaka ktorému by sme rýchlo našli skupinku žiakov, ktorá by náš softvér otestovala. Práve pre tento neobmedzujúci fakt je Unity najlepšou možnosťou pre naše virtuálne prostredie.

3 Návrh riešenia

Virtuálne prostredie pre nášho agenta bude predstavovať hra, ktorá bude vytváraná už v spomínanom hernom rámci Unity.

Hra bude strieľačka z pohľadu prvej osoby. Hráči budú rozdelení do dvoch tímov, pričom cieľ tímov bude získať určitý počet bodov na výhru kola. Tím, ktorý vyhrá tri kolá vyhráva hru. Tímy budeme rozlišovať pomocou farieb, kde jeden bude zelený a druhý červený. Na získanie potrebného počtu bodov budú musieť hráči získať vlajku. Vlajka bude pridávať tímu jeden bod v priebehu jednej sekundy. Okrem vlajky budú hráči získavať body aj zabitím člena nepriateľského tímu, čím tiež získajú jeden bod. Po svete sa budú náhodne vytvárať objekty zbraní, liečivých predmetov a nábojov, čo im pomôže v naplnení ich cieľa. Okrem predmetov sa po svete budú náhodne vytvárať aj barikády, čím sa stane mapa sveta viac dynamická pre hráčov a ťažšie naučiteľná pre agenta. Okrem statických prekážok a barikád si hráči a náš agent budú musieť dať pozor aj na dynamickú prekážku, ktorú bude reprezentovať auto prechádzajúce z jednej strany mapy na druhú. Hra bude hrateľná pomocou myšky a klávesnice, no v hre sa bude nachádzať aj špeciálny mód, ktorý sa bude volať prezentačný mód. Tento mód umožní hráčom hrať hru na rozdelených obrazovkách, na ktoré sa budú vedieť napojiť maximálne štyria hráči pomocou mobilných zariadení a hrať na jednej inštancii hry.

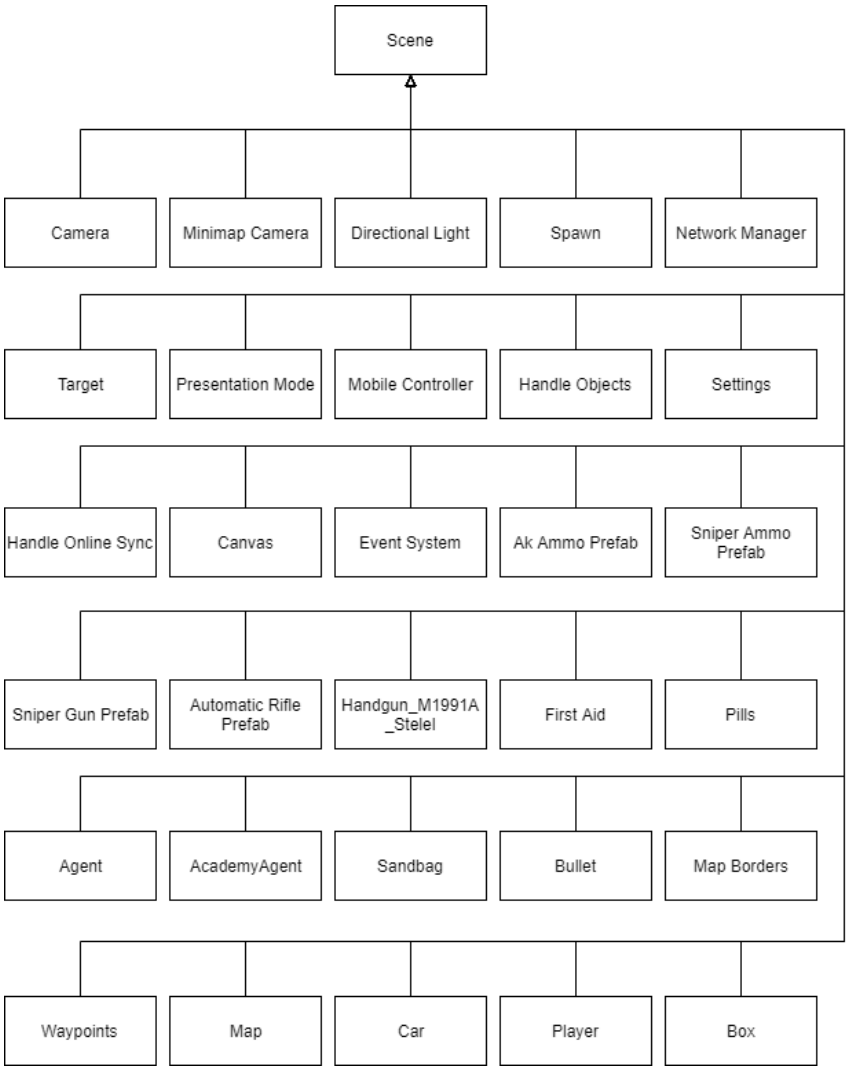
Každý hráč bude mať inventár. Tento inventár bude môcť nosiť štyri predmety. Do inventára si hráči budú dávať už spomínané zbrane a liečivé predmety, ktoré nájdu po svete.



Obr. 3.1: Mapa herného sveta

Na obrázku 3.1 môžeme vidieť mapu hry, zobranú už z implementovanej hry, kde môžeme vidieť označenie začiatočných pozícií tímov pomocou štvorcov. Farba týchto štvorcov reprezentuje tím, ktorý má na danej pozícií začiatočnú pozíciu. Šípky od týchto pozícií naznačujú smer, ktorým sa hráč musí hýbať, aby sa dostal k vlajke. Mapa bude ohraničená hranicami. Hráč po prekročení týchto hraníc dostane upozornenie na vrátenie do hracej plochy. Po uplynutí určitého času hráč zomrie.

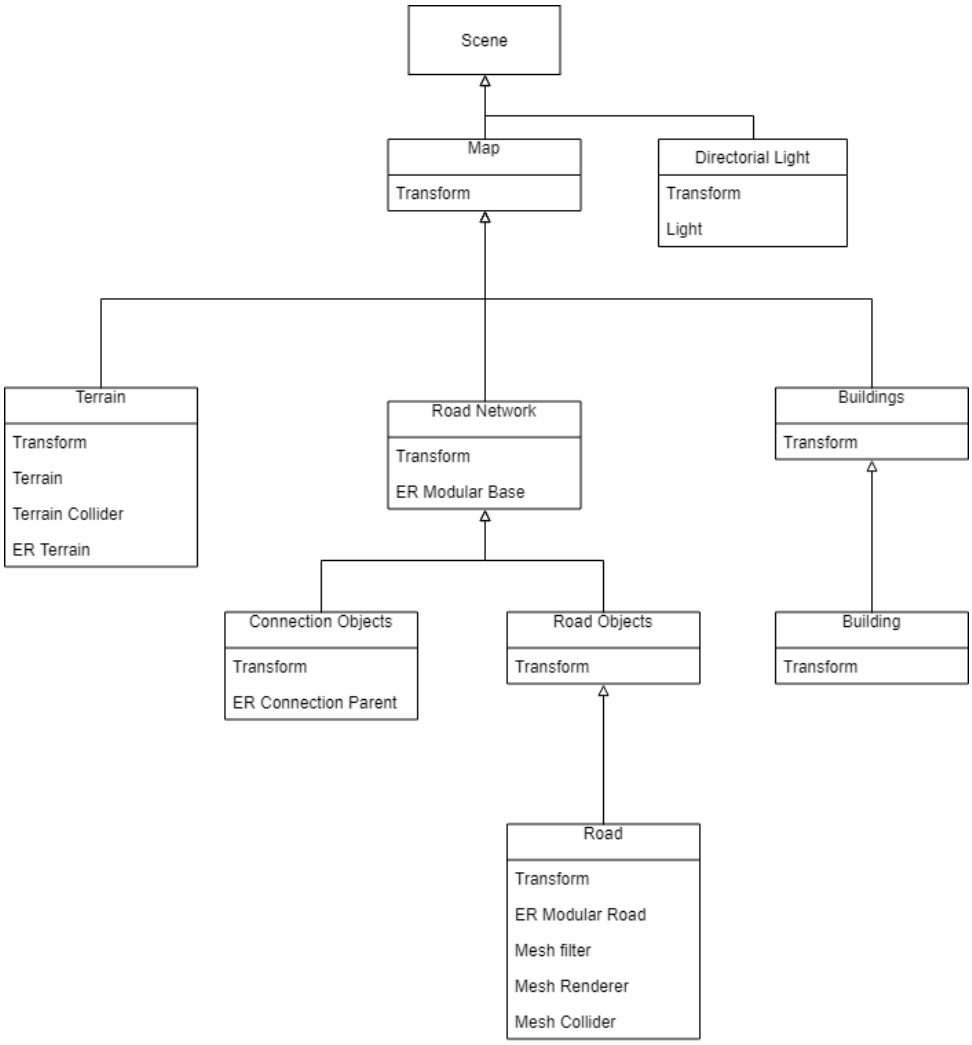
V tejto kapitole si povieme o všetkých objektoch, ktoré budeme potrebovať na vytvorenie takejto hry. Ukážeme si diagramy dedičnosti všetkých objektov vo svete a ich vzťah vzhľadom k scéne, respektíve ich dedičnosť. Tieto objekty sú herné objekty v Unity reprezentované ako GameObject.



Obr. 3.2: Diagram dedičnosti objektov v scéne

Na obrázku 3.2 môžeme vidieť dedičnosť všetkých objektov, s ktorými sa hráč bude môcť dostať do styku počas hrania. Tieto objekty si rozdelíme do desiatich kategórií, ktoré sú mapa a osvetlenie, online pripojenie hráčov, vytváranie objektov, scenár hry, vozidlo, nastavenia, hráč, agent, prezentačný mód a používateľské rozhranie. V nasledujúcich podkapitolách sa pozrieme na tieto jednotlivé kategórie, ktoré objekty sa v nich nachádzajú, aké majú komponenty a aký bude ich účel.

3.1 Mapa a osvetlenie

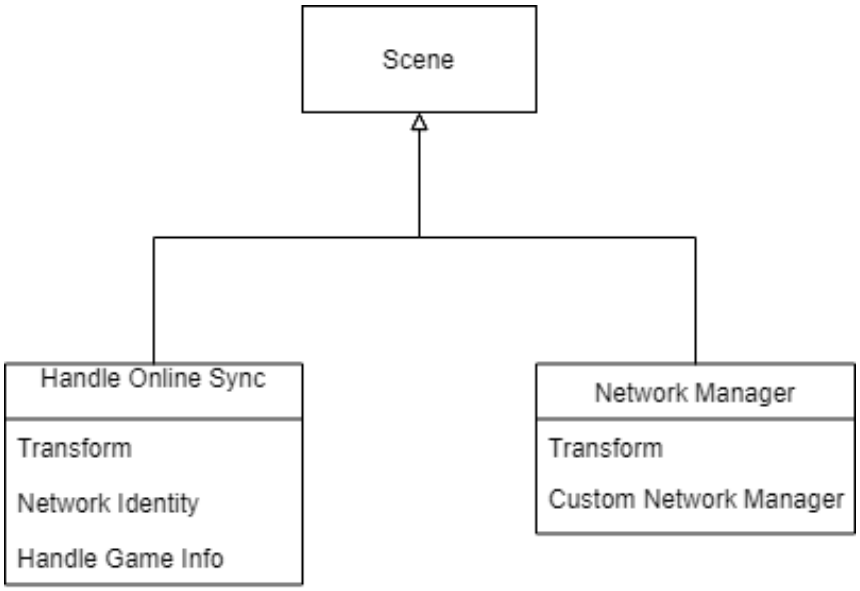


Obr. 3.3: Diagram dedičnosti pre mapu a osvetlenie

Ako môžeme vidieť na obrázku 3.3, táto kategória sa skladá z dvoch hlavných objektov. Prvým objektom je Directional Light, ktorý sa bude starať o osvetlenie našich objektov vo svete. Directional Light obsahuje dva komponenty. Prvým komponentom je Transform predstavujúci pozíciu, na ktorej sa náš objekt bude nachádzať vo svete. Tento komponent je povinným komponentom pre každý objekt, ktorý nie je súčasťou používateľského rozhrania. Druhým komponentom je Light. Tento komponent je vytvorený Unity rámcom. Pomocou tohto komponentu budeme vedieť meniť osvetlenie v celej scéne. Objekt Directional Light je vytvorený

spolu s vytvorením scény. Druhým hlavným objektom v tejto kategórií je objekt Map. Tento objekt je iba prázdnym objektom, ktorý má troch potomkov. Prvým potomkom je objekt Terrain, pomocou ktorého vytvoríme zem, po ktorej budú hráči chodiť, stromy ako aj trávu či kvety. K vytvoreniu prostredia nám pomôže Unity komponent Terrain. Collider je komponent, ktorý zabraňuje iným objektom obsahujúcim komponent RigidBody v prejdení cez tento objekt. Je niekoľko druhov Collider komponentov, ktoré majú rôzne tvary ako napríklad Collider v tvare kocky, guli či kapsule. Objekt Terrain využíva Collider typu Terrain Collider, ktorý sa dynamicky mení podľa tvaru terénu. Posledným komponentom objektu Terrain je ER Terrain. Tento objekt je dôležitý pre ďalší objekt Road Network napomáhajúci pri vytváraní ciest na mape. Konkrétne nám vytváranie ciest zabezpečí komponent ER Modular Base a vytvárané cesty sa budú nachádzať ako potomkovia prázdneho objektu Road Objects. Posledným potomkom objektu Map je Buildings. Tento objekt bude prázdny objekt rodiča pre každý objekt predstavujúci budovu na mape.

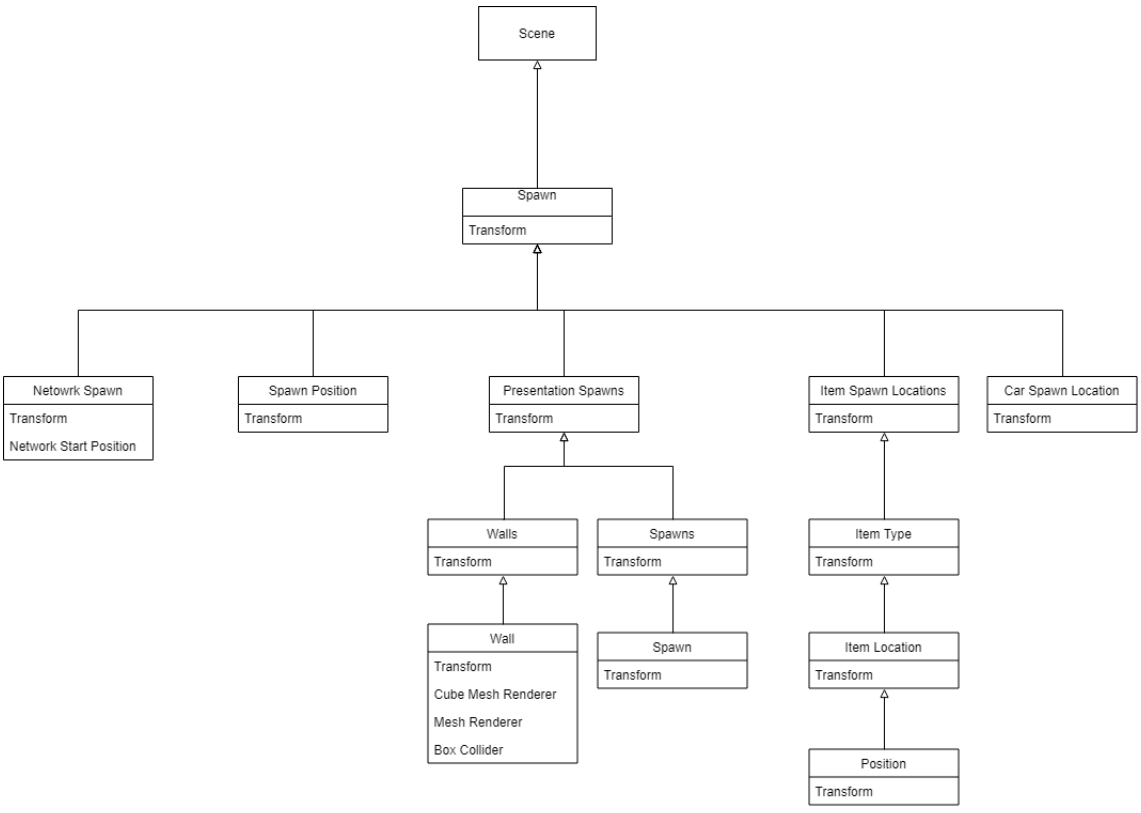
3.2 Online pripojenie hráčov



Obr. 3.4: Diagram dedičnosti online objektov potrebných pre online pripojenie

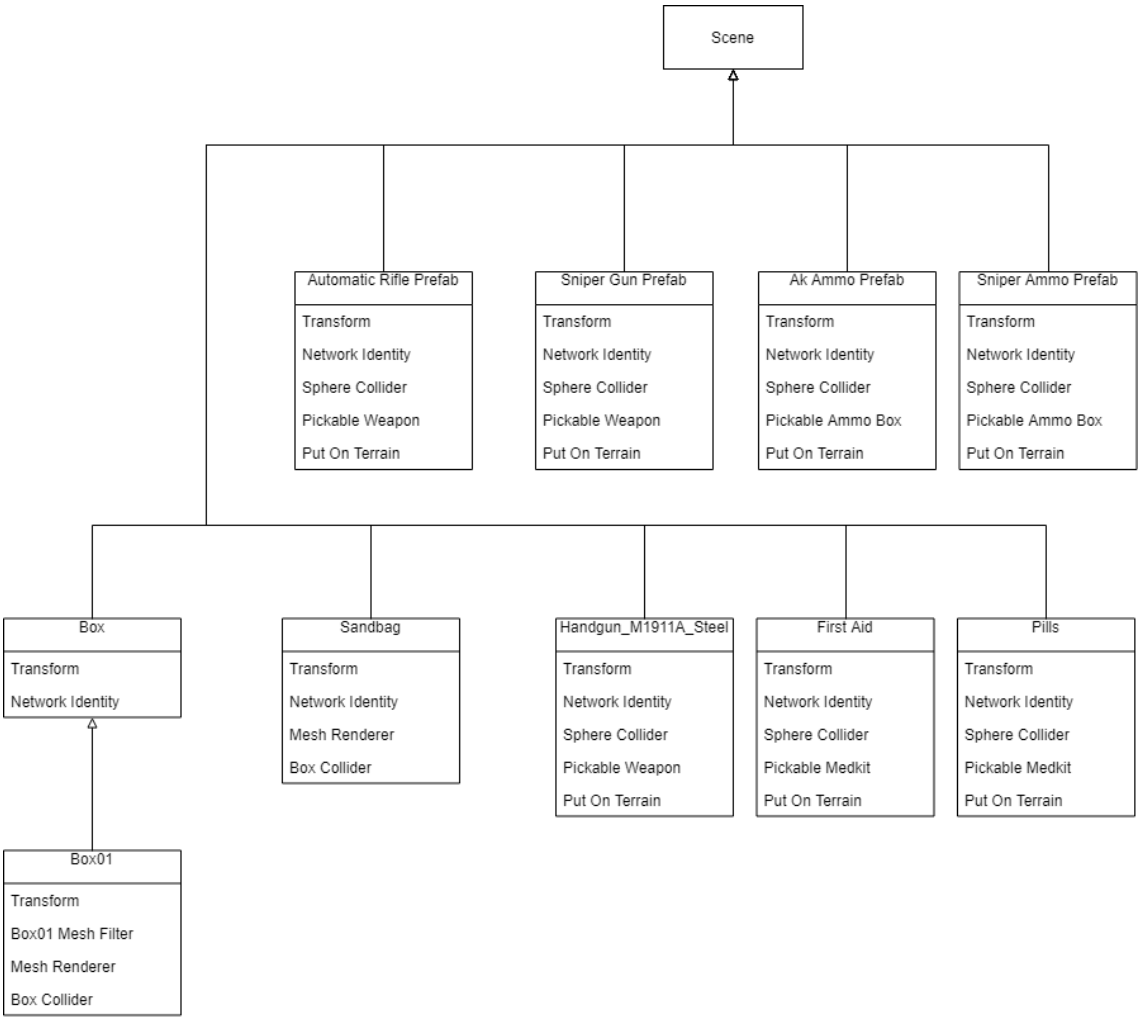
Obrázok 3.4 zobrazuje dva základné objekty, ktoré budeme používať na pripojenie hráčov online. Objekt Network Manager bude pomocou komponentu Custom Network Manager pripájať ľudí, vytvárať ich postavy a každý iný objekt, ktorý musia mať hráči zosynchronizovaný, respektíve aby mal každý hráč rovnaké objekty v scéne. Ako už napovedá názov, Handle Online Sync bude slúžiť na synchronizáciu, avšak v tomto prípade bude slúžiť na synchronizáciu údajov. Komponent Handle Game Info bude teda synchronizovať údaje ohľadom aktuálnej hry. Novo pripojení hráči dostanú zo servera najdôležitejšie informácie o hre, aby sa im podľa toho upravilo ich používateľské rozhranie. Network Identity v Handle Online Sync je dôležitá a povinná súčasť každého objektu, ktorý má úlohu v online synchronizácií.

3.3 Vytváranie objektov



Obr. 3.5: Diagram dedičnosti pomocných objektov

Spawn na obrázku 3.5 je rodič pre všetky objekty napomáhajúce vo vytváraní a presúvaní iných objektov. Ako prvý potomok objektu Spawn je Network Spawn. Tento objekt obsahuje komponent Network Start Position, ktorý bude hovoriť komponentu Custom Network Manager z kapitoly 3.2, na akú pozíciu bude vytvárať objekt hráča po pripojení nového klienta. Spawn Position sú začiatočné pozície pre jednotlivé tímy. Na túto pozíciu budeme teleportovať hráča keď sa pripojí do hry, po jeho smrti a skončení kola či hry. Presentation Spawn má dvoch potomkov. Prvým potomkom je objekt Walls, ktorý je rodičom všetkých objektov Wall. Tieto objekty budú tvoriť jednoduchý tvar kocky. Táto kocka bude tvoriť miestnosť, v ktorej budú objekty hráčov čakajúcich na pripojenie a ovládanie pomocou mobilného ovládača. Presné pozície v tejto miestnosti budú definované pomocou potomkov objektu Spawns. Viac informácií ohľadom prezentačného módu bude v kapitole 3.9. Item Spawn Positions je objekt, ktorý nám pomôže pri vytváraní rôznych objektov v scéne. Obsahuje jedného potomka Item Type reprezentujúceho rôzne typy objektov, ktoré budeme vytvárať pred začiatkom hry. Na tieto objekty sa pozrieme v druhej časti tejto kapitoly. Potomkom Item Type je Item Location určujúci rôzne lokácie, v ktorých budeme vytvárať objekty. Ak máme tri budovy, môžeme mať tri Item Location predstavujúce rôzne lokácie pre vytvorenie objektov. Nakoniec potomok pre Item Location je Position, ktorý určuje presnú pozíciu pre vytvorenie daného objektu v danej lokácií. Takýto systém nám pomôže v balancovaní hry a lepšej prehľadnosti, v ktorých častiach mapy sa môžu vytvoriť aké typy objektov. Auto na mape je preberané v kapitole 3.6. Lokácia, kde sa bude toto auto vytvárať, je reprezentované pomocou posledného objektu v našom diagrame - Car Spawn Location.

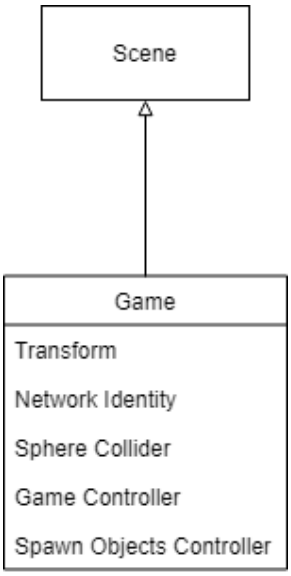


Obr. 3.6: Diagram dedičnosti vytváraných objektov

V prvej časti tejto kapitoly sme si povedali o objektoch v scéne, ktoré budú slúžiť ako referencia na pozíciu, aby sme nemuseli udržiavať presné súradnice v scéne. V tejto časti si povieme o objektoch, ktoré sa budú vytvárať na pozíciách udržiavaných objektom Item Spawn Location. Tieto objekty sú zobrazené pomocou diagramu na obrázku 3.6. Keďže chceme, aby každý hráč mal rovnaké objekty v scéne, musíme tieto objekty synchronizovať online. Preto je potrebné, aby každý objekt v našom diagrame mal komponent Network Identity. Objekty v diagrame môžeme rozdeliť do dvoch kategórií. Prvou kategóriou sú objekty, ktoré hráč bude vedieť zobrať, respektíve si ich bude môcť vložiť do inventára a druhá kategória sú barikády. Objekty pre hráčov inventár sú Automatic Rifle Prefab, Handgun_M1911A_Steel, First Aid, Pills a aj keď náboje nebudú priamo v hráčovom inventári, vieme tam zaradiť aj Ak Ammo Prefab a Sniper Ammo Prefab.

Tieto komponenty majú Sphere Collider. Tieto Collider komponenty však nebudú slúžiť na to, aby tento objekt bol solídny, ale aby sme vedeli či sa v danom Collider komponente nachádza iný objekt. V tomto prípade budeme pozerať či objekt nachádzajúci sa v Collider komponente je hráč. Hráč nachádzajúci sa v tomto komponente si bude vedieť pridať tento predmet do inventára. Všetky informácie ohľadom objektu sa budú nachádzať v skriptoch, ktoré sa začínajú so slovom Pickable. A teda Pickable Weapon bude mať informácie ohľadom toho aká zbraň daný objekt reprezentuje, koľko má nábojov a iné informácie dôležité pre hráčov. Pickable Medkit bude obsahovať informácie o type predmetu na vyliečenie, jeho efektívnosť a iné. Okrem informácií dôležitých pre hráča, ako sú napríklad počet nábojov a typ nábojov, bude Pickable Ammo Box fungovať aj ako kontrolór sledujúci, či sa v ňom nachádza hráč, Ak sa hráč nachádza, náboje sa mu automaticky pridajú. Sandbag a Box v našom diagrame budú objekty barikád. Tieto barikády budú solídne objekty, ktoré budú slúžiť na krytie a ako dynamická zmena mapy medzi hrami.

3.4 Scenár hry

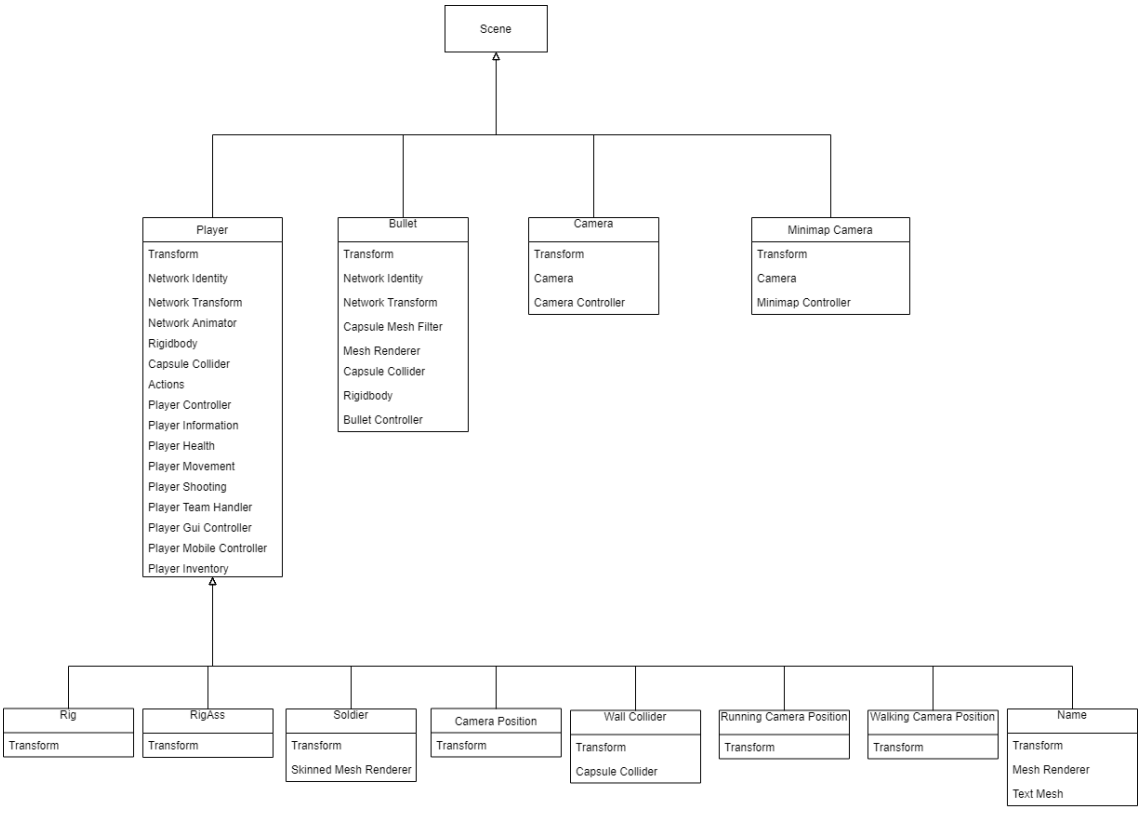


Obr. 3.7: Diagram dedičnosti objektov starajúcich sa o scenár hry

Už bolo spomenuté, ako bude vyzerať priebeh hry, no stále nevieme, ktorý objekt sa o tento priebeh hry bude starať. Na obrázku 3.7 je zobrazený objekt Game

obsahujúci komponent Game Controller, ktorý sa nám o tento priebeh hry bude starať. Okrem riadenia hry bude objekt Game reprezentovať aj vlajku, ktorú sa hráči budú snažiť prebrať pre svoj tím. A teda tento objekt bude mať špecifickú pozíciu na mape a Sphere Collider, pomocou ktorého budeme vedieť koľko hráčov sa nachádza vo vlajke. V každej hre sa bude nachádzať presne osem hráčov a teda hra bude štyria proti štyrom. Koľko hráčov bude vedieť byť v hre budeme riadiť počas vytvárania servera. O to sa bude starať spomínaný Custom Network Manager komponent. Posledný komponent objektu Game v diagrame je Spawn Objects Controller. Tento objekt bude mať za úlohu na začiatku hry vytvoriť všetky barikády a všetky objekty, ktoré vie hráč zobrať.

3.5 Hráč

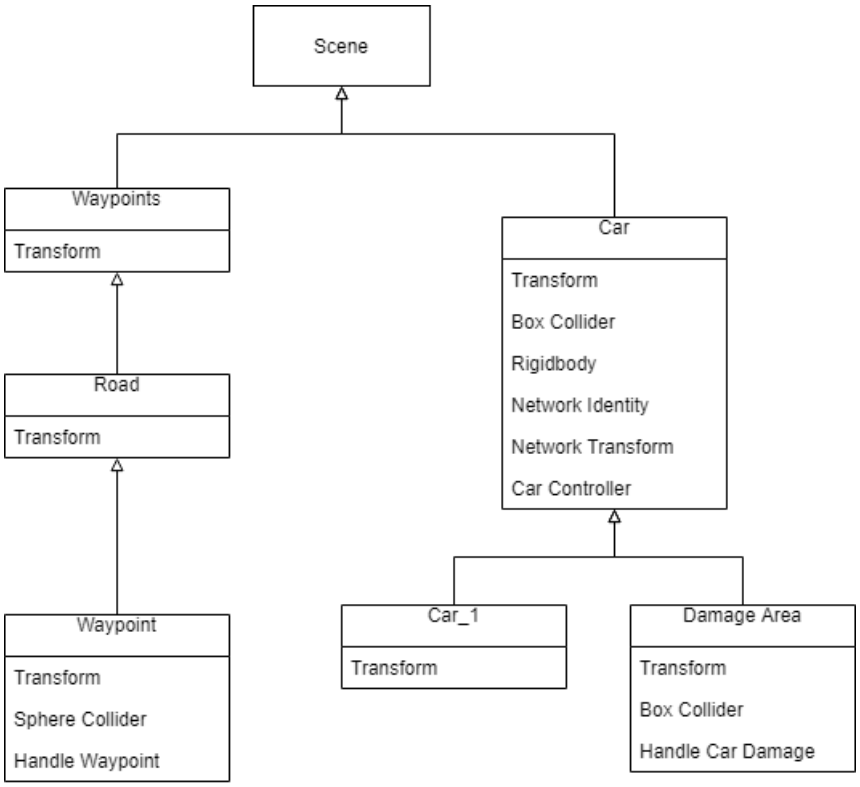


Obr. 3.8: Diagram dedičnosti objektov súvisiacich s hráčom

Diagram na obrázku 3.8 zobrazuje dedičnosť pre všetky objekty potrebné pre hráčovu funkcionalitu. Samozrejmosťou je samotný objekt Player, ktorý hráč bude

ovládať pomocou vstupov na klávesnici a myške alebo mobilnom zariadení. Potomkovia objektu Player, Rig a RigAss definujú spodok objektu a miesta, na ktorých sa budú nachádzať zbrane. Soldier objekt je model samotného hráča, pod ktorým ho budú vidieť ostatní hráči. Camera Position je prázdny objekt základnej pozície pre kameru, Running Camera Position označuje miesto, kde sa bude nachádzať kamera počas behania, pričom Walking Camera Position bude označovať miesto, kde sa bude kamera nachádzať počas chodenia. Wall Collider bude pomocný objekt obsahujúci komponent Capsule Collider. Tento komponent zabráni hráčovi prechádzať cez steny. Posledný potomok objektu Player je Name, ktorý bude zobrazovať 3D meno hráča nachádzajúceho sa nad jeho objektom. Network Identity a Network Transform komponenty v objekte Player budú synchronizovať pohyb online, pričom Network Animator bude synchronizovať animácie. Actions a Player Controller budú predvytvorené s prostriedkom hráča, pričom tieto komponenty budú držať a meniť zbrane používané hráčom. Player Information bude udržiavať všetky základne informácie o hráčovi. Player Health bude udržiavať a meniť život hráča. Player Movement zabezpečí ovládanie pomocou klávesnice a myšky. Player Shooting bude zabezpečovať streľbu pre hráča a teda bude vytvárať objekt Bullet, ktorého správanie bude definované v komponente Bullet Controller. Player Team Handler bude udržiavať a meniť hráčov tím. Player Gui Controller bude meniť grafické používateľské rozhranie pre lokálneho hráča. Player Mobile Controller zabezpečí mobilný pohyb, streľbu a iné hráčove akcie cez mobilný ovládač. Posledný komponent je Player Inventory, ktorý bude zabezpečovať hráčov inventár. Camera a Minimap Camera objekty obsahujú komponenty Camera Controller a Minimap Controller, ktoré budú definovať správanie týchto kamier.

3.6 Auto

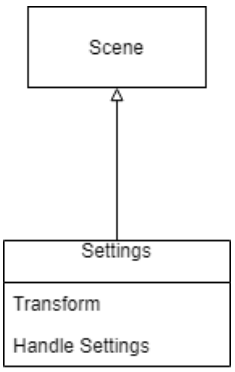


Obr. 3.9: Diagram dedičnosti objektov potrebných pre auto

Obrázok 3.9 reprezentuje diagram objektov a ich komponentov potrebných pre fungovanie auta. Samozrejmosťou je samotné auto, ktoré je reprezentované objektom Car. Rigidbody mu pridá jednoduchú fyziku. Keďže chceme aby bol tento objekt jednotný pre každého hráča, musí obsahovať komponent Network Identity. V tomto prípade hráčovi nestačí iba vediet o jeho existencií. Potreujú si synchronizovať aj jeho pozíciu na čo nám poslúži Network Transform, ako to bolo aj u hráčovho objektu. Komponent Car Controller bude slúžiť na riadenie auta, a teda jeho pohybu. Aby tento pohyb nebol úplne chaotický budeme používať objekty Waypoint, ktoré budú neviditeľné pre hráča a auto vďaka nim bude vedieť, ku ktorej pozícií sa má pomaly pohybovať. Objekt Car má dvoch potomkov. Prvým potomkom je Car_1, čo je objekt, ktorý bude reprezentovať auto. To bude jeho model, pričom všetky ďalšie dedenia a komponenty budú definované prostriedkom, ktorý bude použitý pre auto. Druhým potomkom je Damage Area. Jeho kompo-

nent Box Collider bude slúžiť ako priestor pred autom, do ktorého pokiaľ vojde hráč tak zomrie, pretože auto prešlo daného hráča. Zabitie hráča bude riadiť komponent Handle Car Damage.

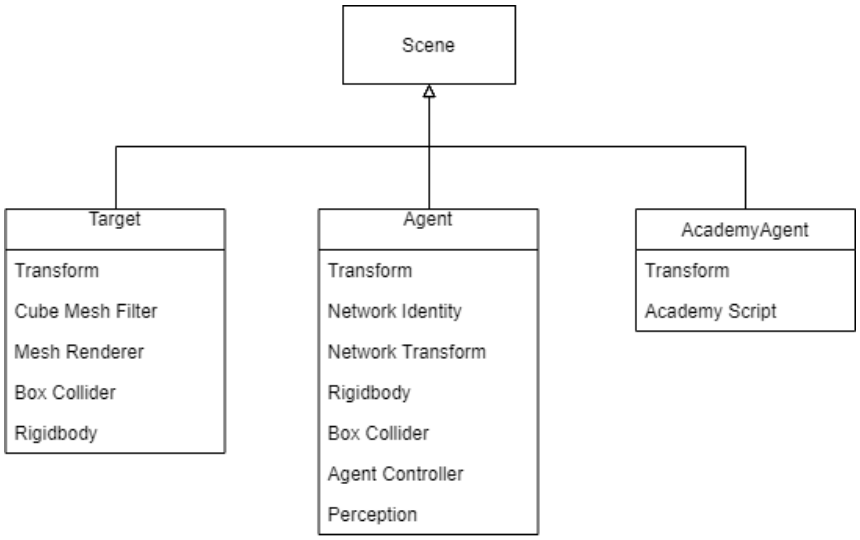
3.7 Nastavenia



Obr. 3.10: Diagram dedičnosti objektov potrebných pre nastavenia

Objekt Settings na obrázku 3.10 bude riadiť ukladanie nastavení, ktoré si hráč bude vedieť zvoliť v používateľskom rozhraní počas hrania. Tieto nastavenia budú hráčove meno a informácia o tom, či je zapnutý prezentačný mód. Tieto nastavenia sa budú dynamicky ukladať do textového dokumentu a načítavať pri zapnutí hry.

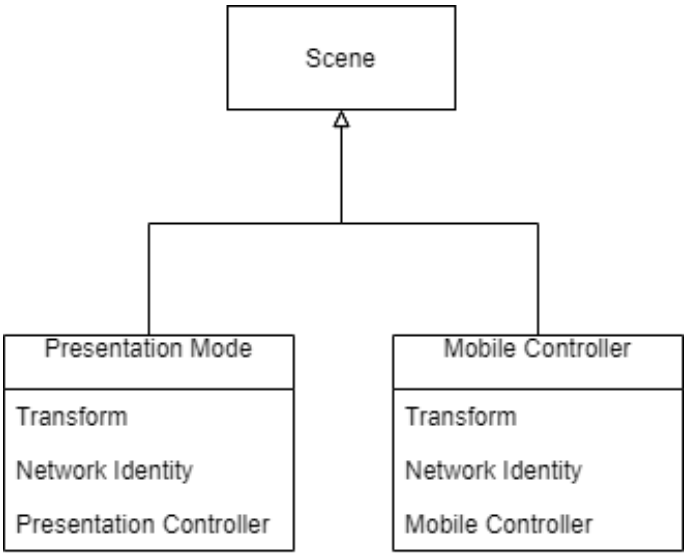
3.8 Agent



Obr. 3.11: Diagram dedičnosti objektov potrebných pre agenta

Objekty reprezentujúce agenta a objekty potrebné pre jeho správanie sú zobrazené v diagrame na obrázku 3.11. Najdôležitejším objektom je objekt Agent reprezentujúci samotného agenta. Okrem zabráneniu prechádzania cez steny, bude mať Box Collider aj iný účel. Tento účel je priestor, do ktorého hráč bude musieť streliť, aby zabil agenta. Agent Controller dodá agentovi zdravie a vytváranie nábojov, zbraní a liečivých objektov po jeho smrti. Agentov cieľ bude dostať sa k objektu Target. Tento objekt obsahuje Rigidbody, aby stále spadol na terén. Posledný objekt AcademyAgent bude riadiť učenie nášho agenta.

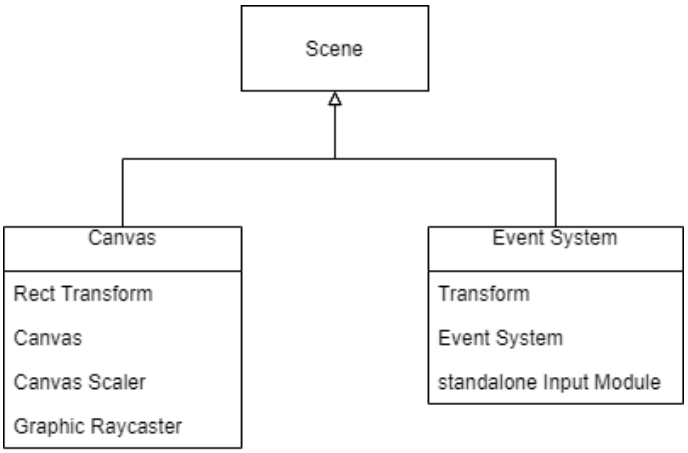
3.9 Prezentačný mód



Obr. 3.12: Diagram dedičnosti objektov potrebných pre prezentačný mód

Prezentačný mód bude špeciálny mód, ktorý po zapnutí a reštartovaní hry automaticky zapne server, a teda hráč so zapnutým prezentačným módom nebude vedieť byť klient a musí byť serverom. Hra so zapnutým prezentačným módom bude rozdelená na deväť obrazoviek. Horné tri obrazovky budú zobrazovať stav hry. Ľavý stredný monitor bude zobrazovať mena všetkych hráčov v hre, ich zabitia a smrti. Na ľavej dolnej obrazovke budú QR kódy, vďaka ktorým sa hráči budú vedieť pripojiť pomocou mobilných zariadení do hry. Po načítaní QR kódu sa hráčovi otvorí stránka s ovládačom, ktorý bude využívať MQTT protokol na posielanie správ do hry. Všetky potrebné objekty na túto funkcionalitu môžeme vidieť v diagrame na obrázku 3.12. Prvý objekt, ktorý budeme potrebovať je Presentation Mode, ktorý nám vďaka komponentu Presentation Controller zabezpečí rozdelenie obrazovky a pridelí kamery hráčom. Mobile Controller je druhý potrebný objekt, ktorý nám vďaka komponentu Mobile Controller zabezpečí pripojenie na MQTT server, tak aj spracovanie prijatých správ z MQTT servera.

3.10 používateľské rozhranie



Obr. 3.13: Diagram dedičnosti objektov potrebných pre používateľské rozhranie

Obrázok 3.13 zobrazuje dva hlavné objekty potrebné na vytvorenie UI v našej hre. Prvým objektom je Canvas, ktorý bude držať všetky potrebné objekty zobrazujúce sa na obrazovke. Bude rodičom každého objektu v používateľskom rozhraní. Druhý objekt Event System sa vytvára spolu s objektom Canvas. Tento objekt slúži na riadenie jednoduchej manipulácie s používateľským rozhraním, respektíve s jej používateľskou interakciou.

4 Implementácia

Keď už máme predstavu aké objekty potrebujeme na vytvorenie nami špecifikovanej hry, môžeme sa pozrieť na implementáciu jednotlivých elementov. Vytvorené diagramy tried v tejto kapitole budú kvôli priestorovej náročnosti tvorené iba triedami, premennými a metódami potrebnými pre danú podkapitolu. Zoznam všetkých tried ich funkcií a premenných je možne nájsť v systémovej príručke, ktorá je z priestorovej náročnosti iba v elektronickej forme na priloženom CD médiu.

4.1 Online Komponent

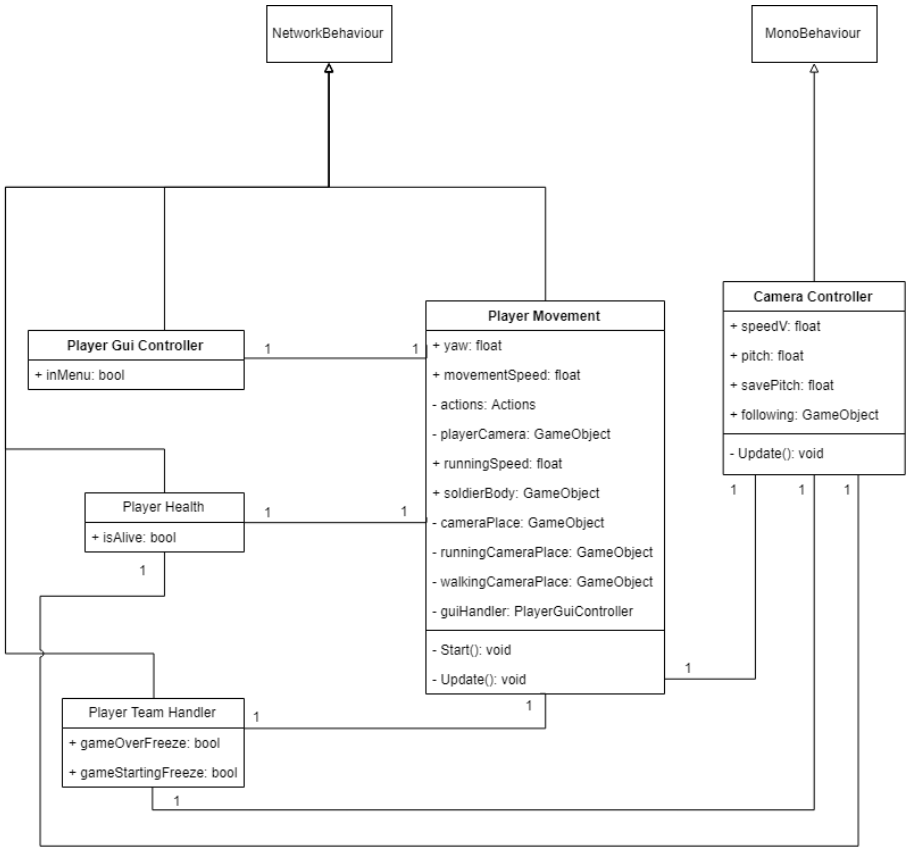
Na prípravu hry pre online hranie je potrebné najprv vytvoriť skript Network Manager zabezpečujúci pripájanie hráčov a vytváranie serverov. Ako prvé je potrebné nastaviť v Spawn Info premennú Player Prefab na objekt hráča. Vďaka tejto premennej bude tento komponent vedieť, ktorý objekt má vytvoriť pri pripojení nového hráča, ktorý bude predstavovať objekt, ktorý hráč bude ovládať. Všetky ostatné objekty, ktoré nie sú hráči ale majú sa vytvárať pre všetkých hráčov na serveri, musíme priradiť do poľa Registered Spawnable Prefabs. Keď už budú hráči pripojení na serveri budeme vedieť posielať príkazy serveru pomocou anotácie Command. Príkazy posielané zo servera hráčom budú mať priradenú anotáciu Rpc.

4.2 Terén

V tejto kapitole sa pozrieme podrobnejšie na objekt Terrain a jeho komponent Terrain. V tomto komponente sú pre nás najdôležitejšie možnosti zmeny výšky terénu a pridávanie trávy, kvetov či stromov pomocou kefky, ktorej vieme meniť veľkosť a hustotu. Ďalšou dôležitou súčasťou tohto komponentu sú nastavenia te-

rénu, kde vieme hru optimalizovať. Keďže hra bude bežať v móde, kedy budú aktívne 4 kamery, potrebujeme hru čo najlepšie optimalizovať, a tak zmeníme nastavenie Billboard Start určujúce v akej diaľke sa 3D textúry menia na 2D. Ďalšie nastavenie je Tree Distance, ktoré určuje pri akej vzdialenosti sa stromy prestanú renderovať. Posledné nastavenie, ktoré nám pomôže hru optimalizovať, je Detail Distance určujúce pri akej vzdialenosti ma textúra najhoršiu kvalitu.

4.3 Pohľad a pohyb



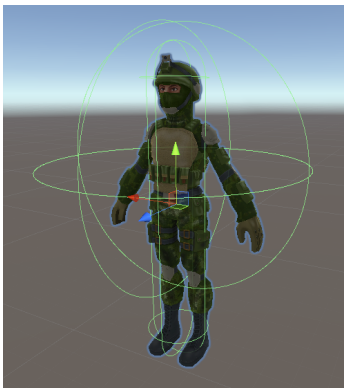
Obr. 4.1: Diagram tried potrebných na pohľad a pohyb hráčov

Čo sa zobrazuje pre hráčov je definované v Unity vďaka kamerám renderujúcim svet pred nimi. Keďže každý hráč po sieti bude mať zapnutú vlastnú inštanciu hry nie je potrebné, aby sa kamera generovala pre každého hráča zvlášť ale stačí, aby kamera bola objekt vo svete. Na obrázku 4.1 môžeme vidieť diagram tried, respektíve skriptov potrebných pre hráčov pohľad a pohyb. Camera Controller je

skript, meniaci pozíciu kamery pri každom volaní metódy *Update*(). Táto metóda sa volá pri každej novej snímke hry. Pozícia, na ktorú sa kamera neustále presúva, je v objekte hráča, konkrétne na mieste jeho hlavy. Táto pozícia sa však mení s animáciou preto aj my túto pozíciu musíme meniť podľa toho, či hráč beží. Skript Player Movement, ktorý slúži na riadenie hráčovho pohybu, bude neustále pozerať na to, či hráč drží stlačené určité klávesy a podľa toho bude meniť aktuálny stav hráča. Pokiaľ hráč drží stlačenú napríklad klávesu W posunieme objekt hráča o jednu jednotku dopredu. Rýchlosť akou sa hráč pohybuje záleží od toho, či hráč drží stlačenú klávesu SHIFT, pričom ak ju má stlačenú pohybuje sa dvojnásobnou rýchlosťou. Pohyb hráča bude zamrznutý a klávesy nebudú mať žiaden vplyv v situácií, keď je hráč mŕtvy, nachádza sa v menu alebo je začiatok, respektíve koniec hry. Na pozeranie okolo seba meníme rotáciu objektov. Objekt hráča nemá žiadnu animáciu, respektíve rotáciu vertikálne. Tento nedostatok obídeme rotáciou kamery. Horizontálnu rotáciu už môžeme vykonávať iba na objekte hráča, keďže túto rotáciu bude objekt hráča zdieľať s kamerou. Týmto spôsobom teda ostatní hráči nebudú vidieť do akej výšky sa hráč pozerá, no budú aspoň vedieť, do ktorého smeru. Rotácia bude pozostávať na pozorovaní virtuálnej osi myšky a pri zmene pozície myšky v tejto osi zmeníme rotáciu hráča.

4.4 Fyzika hráča

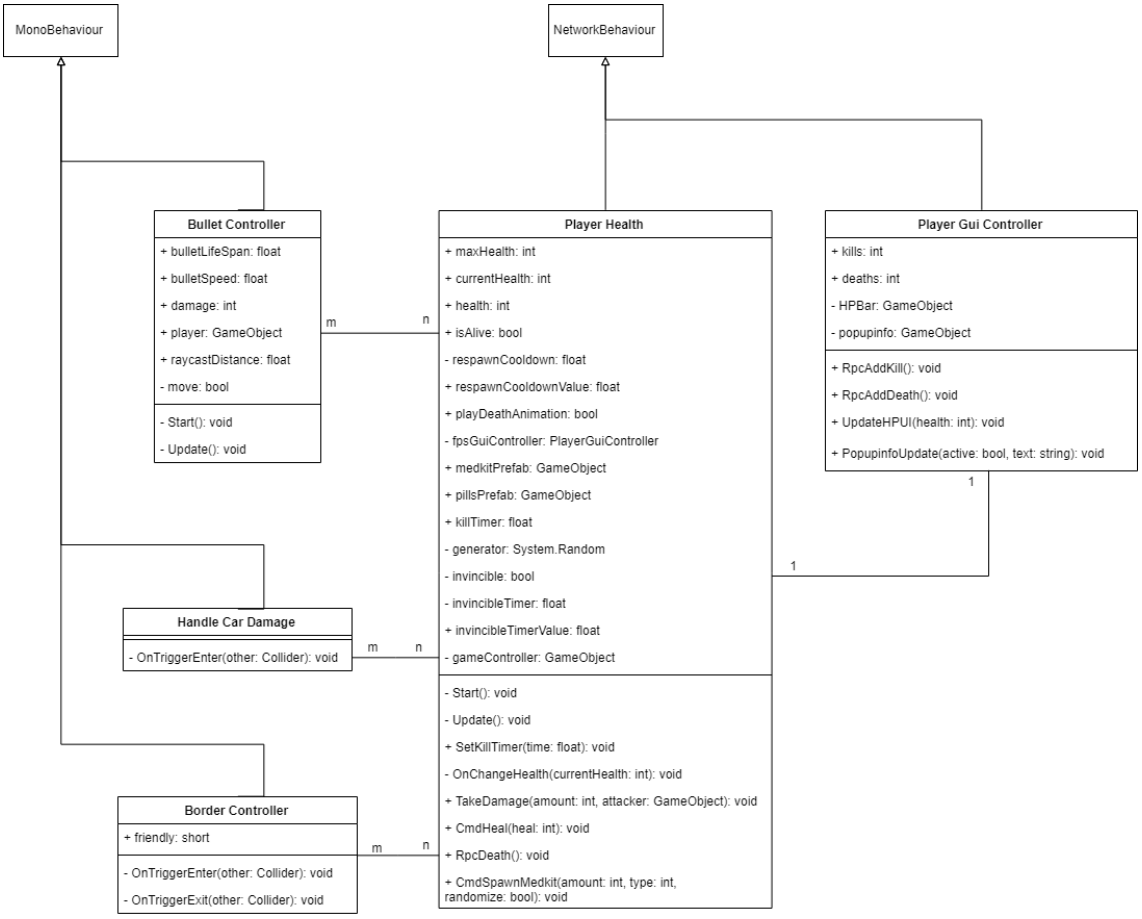
Aktuálne sa do hry vedia napájať hráči, ktorý sa vedia pohybovať po mape. Problémom však je, že títo hráči môžu prechádzať cez stromy ako aj vzájomne cez seba. Hráči nie sú ani nijak položený na terén a teda iba lietajú vo vzduchu. Pokiaľ by sa v mape nachádzal kopec, hráč by jednoducho prešiel cez neho. To nám vyriešia dva už spomínané komponenty. Prvým komponentom je Rigid Body pridávajúci objektu fyziku. Tento komponent taktiež hovorí, že daný objekt má podliehať gravitácií. Druhým komponentom je Collider, vďaka ktorému hráč prestane prechádzať cez ostatné objekty obsahujúce Collider komponent.



Obr. 4.2: Zobrazenie hráčových Collider komponentov

Obrázku 4.2 zobrazuje objekt hráča obsahujúci komponenty Capsule Collider a Circle Collider. Tieto komponenty vytvárajú okolo hráča neviditeľnú stenu, ktorá zabraňuje prechádzaniu cez ostatné objekty s Collider komponentom, a teda každý objekt, cez ktorý nechceme aby náš hráč vedel prejisť musí obsahovať Collider komponent, pričom tento komponent nesmie byť nastavený ako trigger. Dôvodom pre dva tieto komponenty je, že Capsule Collider slúži na kolíziu s terénom aby po ňom vedel hráč chodiť a zároveň je to miesto, ktoré po zasiahnutí uberie hráčovi životy. Circle Collider slúži na kolíziu s ostatnými objektami vo svete. Pokiaľby sme nechali iba prvý Collider komponent, hráč by kvôli nedostatku v Unity fyzike dokázal v určitých momentoch prechádzať cez steny medzi jednotlivými snímkami. Okrem občasného prechádzania cez steny by hráč dokázal pozerať cez steny. Posledným nedostatkom je, že pokiaľ by hráč stál na miernom kopci padal by jednoducho na zem. To sa dá vyriešiť v komponente Rigid Body, kde vieme zapnúť obmedzenie na zamrznutie rotáciie objektu v jednotlivých smeroch osi. Pre nášho hráča môžeme zamrznúť všetky smery, keďže nie je potrebné, aby Rigid Body akýmkoľvek spôsobom menilo rotácie hráča.

4.5 Zdravie



Obr. 4.3: Diagram tried potrebných pre hráčove zdravie, jeho zobrazovanie a manipuláciu

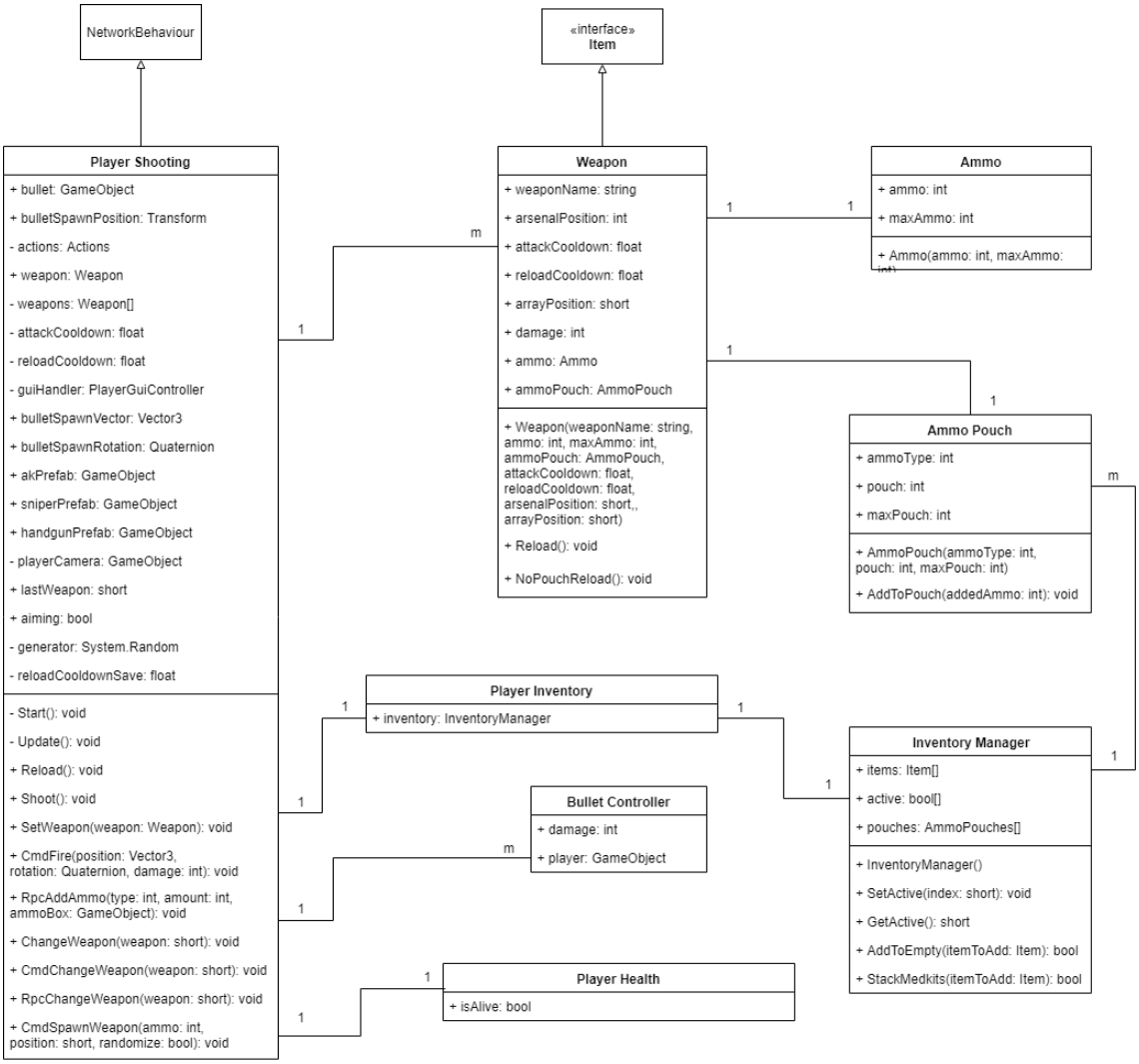
Na pridanie hráčovho zdravia je potrebný už spomínaný skript Player Health zobrazený na obrázku 4.3. Tento skript slúži na udržiavanie a riadenie hráčovho zdravia. Hodnota zdravia je uložená iba na serveri, pričom server posiela informácie, ohľadom zmeny tohto zdravia hráčom, aby si ho mohli zobraziť vo svojom používateľskom rozhraní. Týmto spôsobom zabránime podvádzaniu, pretože iba server bude mať skutočné informácie o hráčovom zdraví a lokálne si hráči toto zdravie nebudú môcť nijakým spôsobom zmeniť. Tri skripty dokážu znížiť hráčove zdravie. Prvé dva skripty a teda Bullet Controller a Handle Car Damage znížia hráčove zdravie po zrážke s hráčovým objektom. Skript Border Controller nastaví hráčovi časovač, ktorý po dosiahnutí hodnoty nula zabije daného hráča.

Tento skript zaručí aby ľudia neopúšťali hrateľnú zónu. Po hráčovej smrti mu nastavíme časovač, pričom tento časovač hovorí o tom kedy sa po smrti hráč obnoví na počiatočnú pozíciu hráčovho tímu. Po jeho obnovení na počiatočnú pozíciu nechceme, aby bol hráč znova zraniteľný, keďže to by mohlo viesť k zabíjaniu hráčov na ich počiatočnej pozícií. Preto sa hráčom po smrti nastaví časovač, počas ktorého sú títo hráči nesmrteľní. Každá smrť hráča sa pripočítava v skripte Player Gui Handler. Pokiaľ hráč zomrel nábojom, ktorý nesie informáciu, ktorý objekt hráča daný náboj vytvoril, pripočíta sa tomuto objektu zabitie.

Obr. 4.4: Používateľské rozhranie hráčovho zdravia

Na obrázku 4.4 môžeme vidieť ako sa bude hráčom počas hrania zobrazovať ich aktuálny stav zdravia. Tento objekt bude zobrazený pomocou objektu Slider. Tento objekt má nastavené červené pozadie a zelený predok, pričom má odstránenú páčku na ovládanie tohto komponentu. Následne pri zmene premennej Value v komponente Slider sa zmení veľkosť prednej časti a zobrazí sa červené pozadie čím sa indikuje aktuálne zranenie hráča.

4.6 Streľba



Obr. 4.5: Diagram tried potrebných pre streľbu

Keďže vytvárame akčnú hru z pohľadu prvej osoby, streľba je jednou z najdôležitejších mechaník v hre. Na obrázku 4.5 je zobrazený diagram tried potrebných na správnu funkčnosť tejto mechaniky. Pre streľbu budeme pozerať na to, či hráč drží stlačené ľavé tlačidlo na myške. Ak ho drží stlačené, vytvárame nový objekt náboja na mieste hráčovej zbrane. To však vytvorí náboj iba na strane hráča, ktorý tento náboj vystrelil. Aby sa náboj objavil aj pre ostatných hráčov je potrebne zavolať metódu s atribútom Command, v ktorom mu pošleme informácie o mieste, kde treba daný náboj vytvoriť, akú má mať rotáciu a inštanciu objektu hráča, ktorý

daný náboj vystrelil. V tejto metóde sa zavolá funkcia *NetworkServer.Spawn*() s parametrom novo vytvoreného náboja. Táto funkcia vytvorí daný objekt pre každú inštanciu hry pripojenú na server, a teda v hre každého klienta.

4.6.1 Náboje

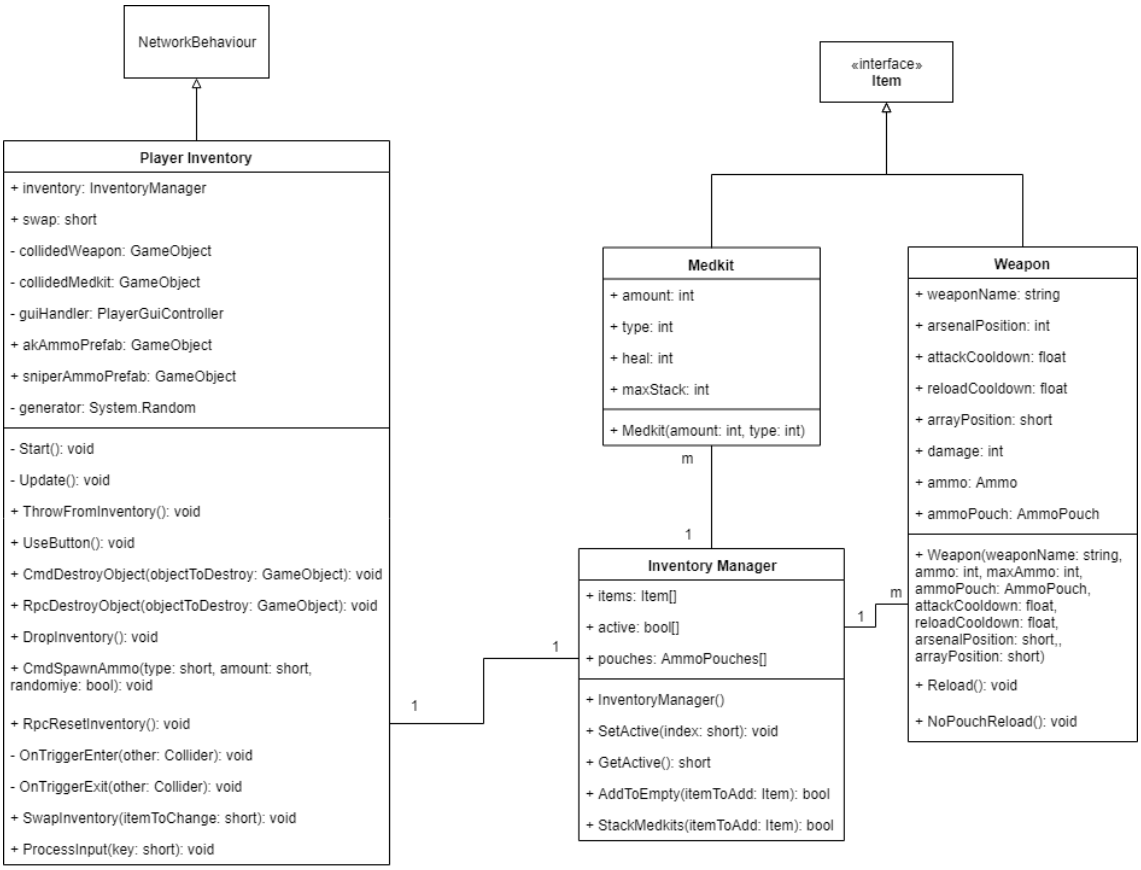
Už sme si povedali ako náboj ovplyvňuje zdravie hráča. V tejto časti si priblížime vzhľad a správanie náboja. Náboj je čiernej farby a je jednoduchého podlhovastého tvaru. Tento objekt sa každú snímku posunie dopredu podľa predurčenej rýchlosti. Ak narazí na hocijaký objekt, respektíve jeho Collider komponent, pozrie sa či daný objekt obsahuje skript zdravia, pokiaľ tento skript obsahuje zníži v tomto skripte zdravie hráča. Detekciu s iným objektom môžeme vytvárať viacerými spôsobmi. Príkladom je pridanie náboju komponent Collider, ktorý nastavíme ako trigger. Po náraze sa spustí dedikovaná metóda v Unity, v ktorej by sme náboj zrušili a pokiaľ by bol narazený objekt hráčom, znížili by sme mu životy. Tento postup má však nedostatok, ktorý už bol spomínaný v kapitole 4.4, kde sme si povedali, že pokiaľ je objekt príliš rýchly, môže prejsť cez objekty. Najlepším riešením je teda využitie Raycast funkcie z knižnice fyziky, ktorú poskytuje Unity. Táto funkcia vyšle lúč od určitej pozície, do nami zvoleného smeru a vzdialenosti. V našom prípade ho posielame od pozície objektu náboja, v smere pohybujúceho sa náboja, do vzdialenosti päť metrov. Vďaka týmto lúčom vieme nastaviť rýchlosť náboja aj na vyššie hodnoty, napríklad sto metrov za sekundu. Na záver je potrebné pridať tento objekt náboja do skriptu streľby cez verejnú premennú herného objektu a aj pridanie tohto objektu do už spomínaného Network Managera medzi vytvárané objekty serverom, respektíve Spawnable Prefabs, pričom prefab je objekt, ktorý sa nachádza v projekte a je ho možné hocikedy počas hrania vytvoriť.

Obr. 4.6: Používateľské rozhranie nábojov

Obrázok 4.6 zobrazuje používateľské rozhranie s aktuálnym počtom nábojov a všetkých nábojov v hráčovom vaku, respektíve nábojov dostupných na znovu nabitie zbrane. V tomto prípade má hráč vo vaku nekonečno nábojov. Dôvodom je,

že pištoľ nemá vak pre náboje, ale má neobmedzený počet nábojov na znovu nabitie. Tieto náboje sa v používateľskom rozhraní zobrazujú pomocou jednoduchého textu, ktorý prepisujeme pri strieľaní a prebíjaní zbrane.

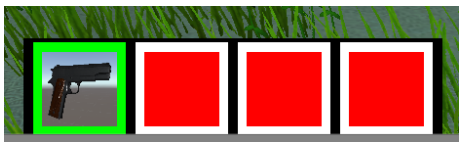
4.7 Inventár



Obr. 4.7: Diagram tried potrebných pre inventár

Obrázok 4.7 reprezentuje diagram tried potrebných pre správnu funkcionalitu inventára. Pomocou skriptu Player Inventory bude hráč ovládať svoj inventár, reprezentovaný triedou Inventory Manager. Hráč môže mať v inventári dva typy predmetov, definované rozhraním Item. Tieto typy sú zbrane a liečivé predmety. Na ovládanie inventára sa budeme pozerať, či hráč stlačil klávesy 1, 2, 3 alebo 4. Po stlačení sa spracuje daný vstup a predmet na danej pozícií sa stáva aktívnym. Aktívna zbraň sa stáva hráčova aktuálna zbraň, no aktívny liečivý predmet nemá zatiaľ žiaden efekt. Liečivé predmety sa použijú až po ich opätovnom aktivovaní. Na

vyliečenie je potrebné dva krát stlačiť klávesu reprezentujúcu pozíciu v inventári, kde sa nachádza tento predmet. Po stlačení klávesy T sa aktívny predmet vyhodí z inventára, jeho inštancia v Inventory Manager sa vynuluje a vo svete sa vytvorí nový objekt, ktorý reprezentuje predmet na danej pozícií inventára. Na výmenu predmetov v inventári je potrebné mať jeden predmet ako aktívny a stlačiť klávesu F, ktorá aktivuje mód na výmenu. Po následnom stlačení jednej z kláves 1, 2, 3 alebo 4 sa vymenia predmety na tejto a aktívnej pozícií. Na zdvihnutie predmetu zo sveta sa pozeráme, či sa objekt hráča nachádza na objekte predmetu, pričom po stlačení klávesy E sa daný objekt vo svete zničí a na prvej voľnej pozícií v hráčovom inventári sa vytvorí reprezentácia predmetu zo sveta. Po smrti hráča sa celý obsah inventára vyhodí do herného sveta.



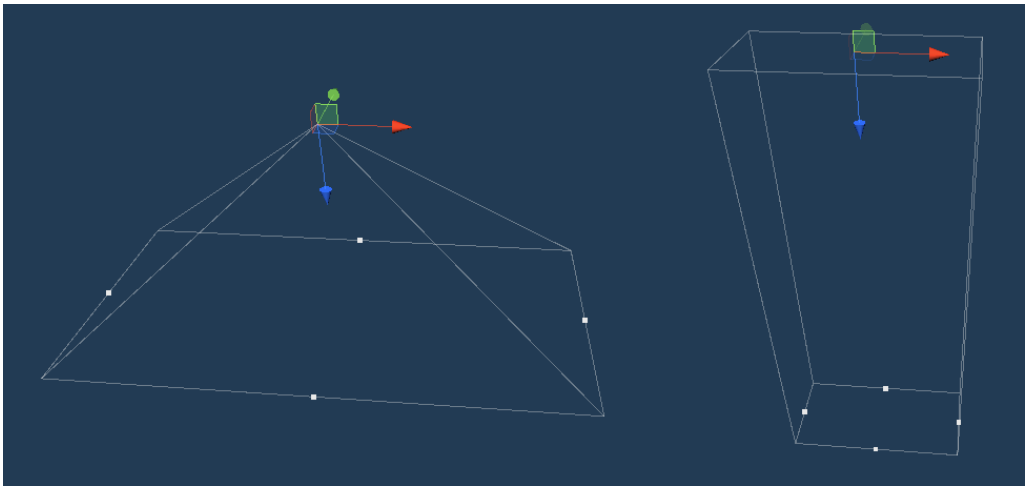
Obr. 4.8: Používateľské rozhranie inventára

Používateľské rozhranie inventára môžeme vidieť na obrázku 4.8. Máme štyri štvorce obrázkov, na pozíciu ktorých vkladáme obrázky podľa toho, aký predmet sa nachádza na danej pozícií v inventári. Pokiaľ sa na danej pozícií nenachádza žiaden predmet, obrázok je iba červená plocha. Okolo týchto obrázkov sú ďalšie obrázky, ktoré sú postavené ako pozadie. Ak je predmet v inventári aktívny, zmeníme farbu obrázka reprezentejúceho pozadie na zelenú. Ak predmet na danej pozícií aktívny nie je, farba pozadia sa zmení na bielu.

4.8 Minimapa

Najjednoduchším spôsobom ako vytvoriť minimapu pre hráča je vytvorenie druhej kamery, ktorá sa bude zobrazovať nad hlavnou kamerou, vďaka premennej Depth v komponente Camera. Túto Depth premennú nastavíme na väčšiu hodnotu ako Depth hlavnej kamery. Na veľkosť a umiestnenie kamery musíme zmeniť Viewport Rect hodnoty X, Y, W a H. Hodnota X určuje pozíciu na osi X a Y určuje pozíciu na osi Y, kde sa bude hráčovi na obrazovke zobrazovať kamera. Hodnoty W a H určujú, akú šírku a výšku bude mať táto kamera. Poslednú úpravu pre kom-

ponent Camera, ktorú musíme spraviť, je zmena hodnoty premennej Projection z Prespective na Orthographic.



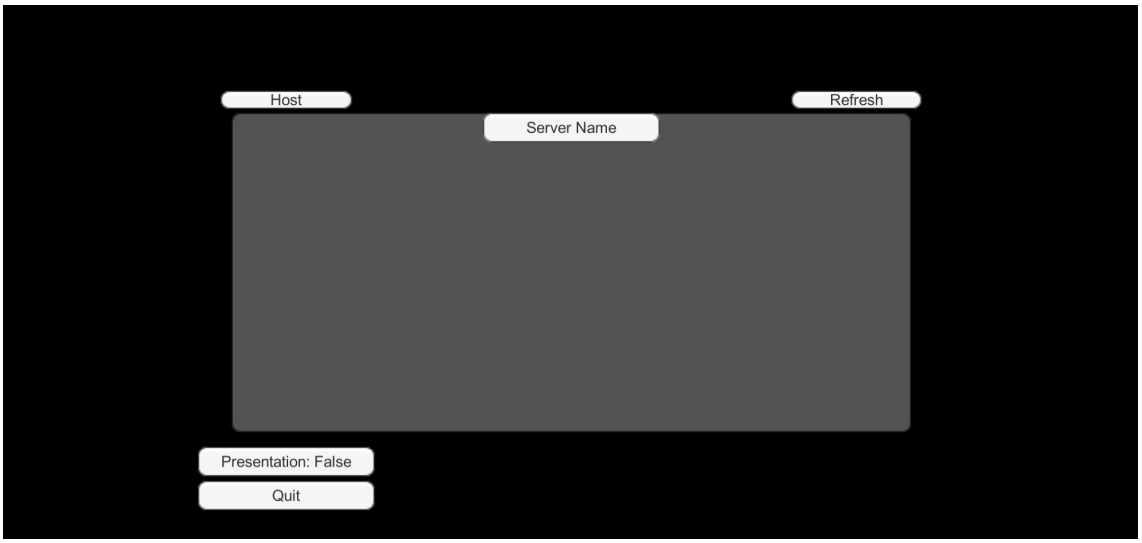
Obr. 4.9: Rozdielné typy kamier

Na obrázku 4.9 môžeme vidieť pôsobenie premennej Projection, pričom naľavo je kamera, ktorá má Projection nastavené na Prespective a napravo je kamera s Projection nastaveným na Orthographic. Biele čiary zobrazujú zorné pole kamery. Kamera v prespective móde teda zobrazuje aj hĺbku objektov a kamera v Orthographic móde zobrazuje svet pred sebou v dvojrozmernom priestore.

4.9 Menu

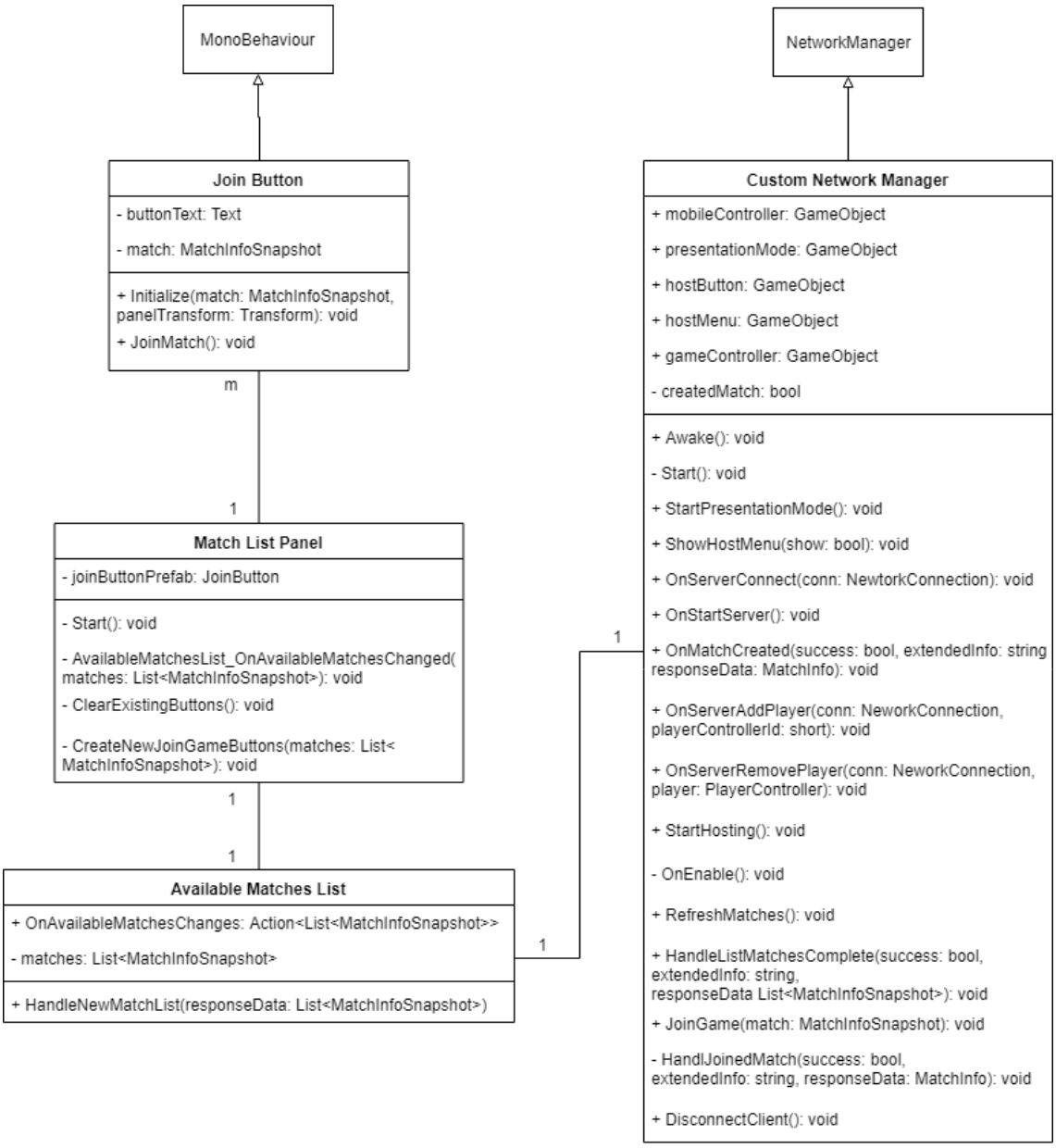
Naša hra bude pozostávať zo štyroch menu. V tejto kapitole si tieto menu prejdeme a aj všetky možnosti, ktoré v nich hráči budú mať.

4.9.1 Menu pre online pripojenie



Obr. 4.10: Používateľské rozhranie pre menu online pripojenia

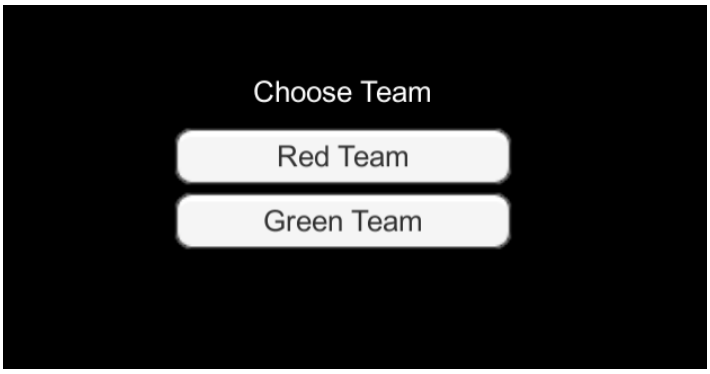
Ako prvé menu, ktoré hráč uvidí po zapnutí hry, je menu na obrázku 4.10. V tomto menu si hráč bude vedieť vytvoriť vlastný server, ktorý si bude vedieť pomenovať. V strede obrazovky sa budú vypisovať všetky dostupné servery, ktoré vytvorili iní hráči. Na obnovenie zoznamu všetkých dostupných serverov nám bude slúžiť tlačidlo Refresh. Posledné dve tlačidlá na spodku obrazovky sú tlačidlá na zapnutie prezentačného módu, aby mal hráč aj v tomto menu možnosť zapnúť si prezentačný mód a nemusel kvôli tomu vytvárať, respektíve sa pripájať na server. Posledné tlačidlo slúži na vypnutie samotnej hry.



Obr. 4.11: Diagram tried potrebných pre menu online pripojenia

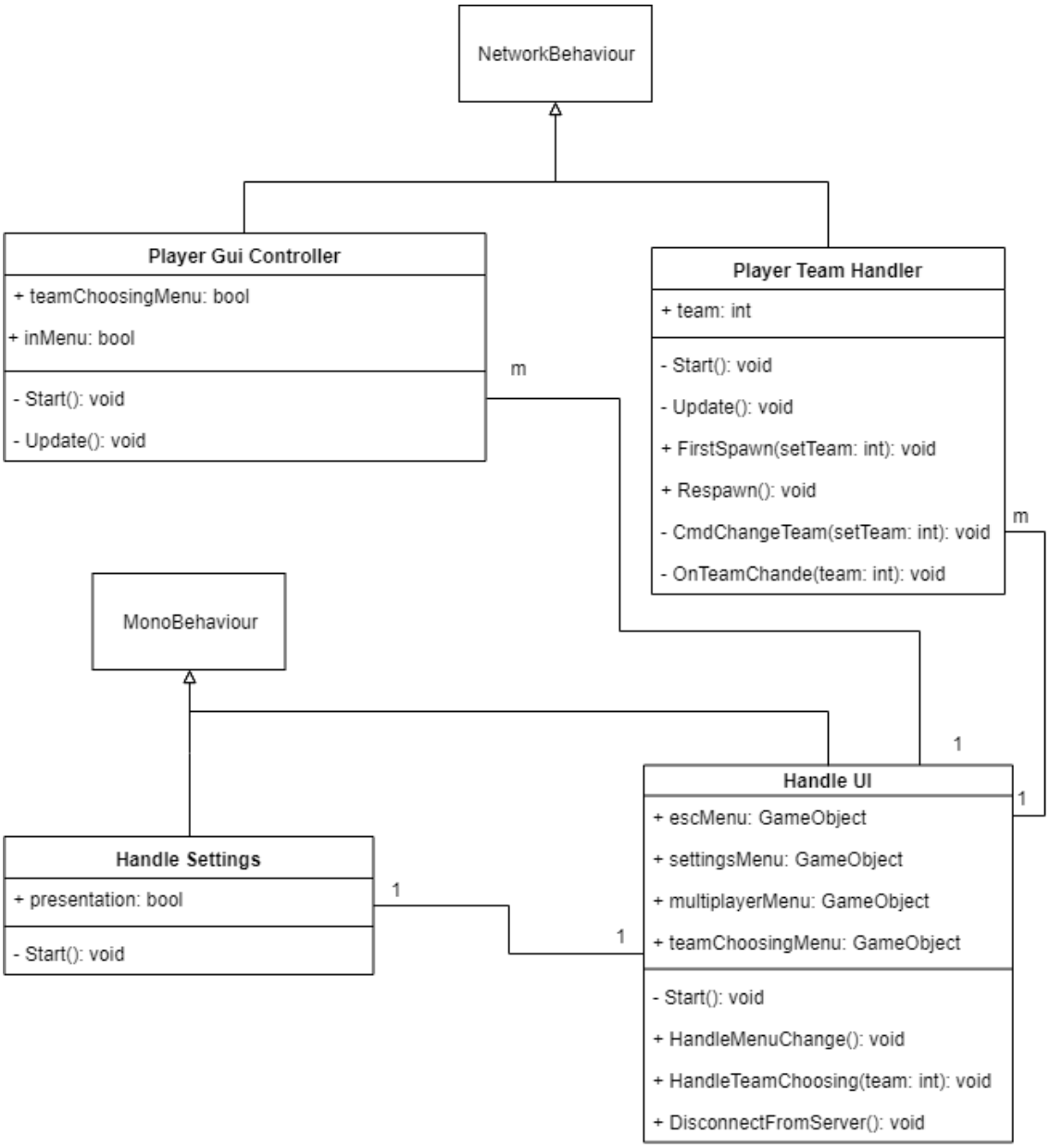
O celú túto interakciu s hráčom v menu sa starajú skripty, ktoré môžeme vidieť na obrázku 4.11. Custom Network Manager je skript starajúci sa o pripojenie hráčov a získanie zoznamu dostupných serverov. Tento zoznam pošleme do Available Match List, ktorý pomocou Match List Panel vytvorí panel na zobrazenie všetkých Join Button objektov, predstavujúcich tlačidlo na pripojenie k serveru.

4.9.2 Menu výberu tímu



Obr. 4.12: Používateľske rozhranie pre menu výberu tímu

Pokiaľ hráč vytvoril alebo sa pripojil na server dostane sa do druhého menu, v ktorom si musí vybrať tím, do ktorého sa chce pridať. Toto menu môžeme vidieť na obrázku 4.12. Hra je vytvorená pre osem hráčov, a teda maximálny počet hráčov pre jeden tím sú štyria hráči. Preto sa neustále pozeráme na to, koľko hráčov sa nachádza v ktorom tíme, pokiaľ je hráč v tomto menu. Ak tím už má štyroch hráčov skryjeme tlačidlo na pripojenie hráča do tohto tímu.

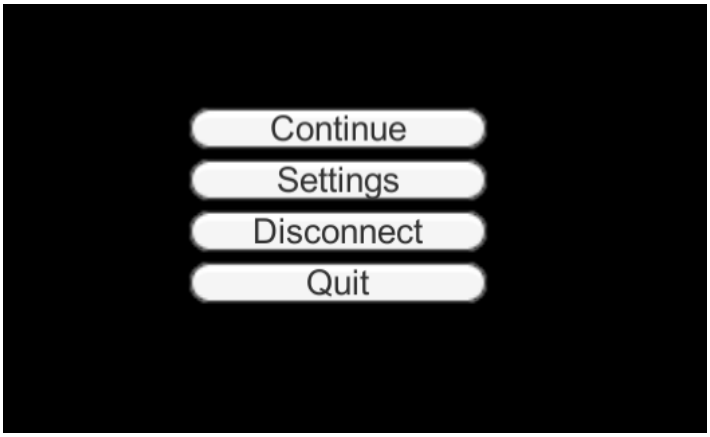


Obr. 4.13: Diagram tried potrebných pre menu výberu tímu

V tomto prípade sa o interakciu s hráčom starajú skripty, ktoré sú zobrazené na obrázku 4.13. Handle UI skript slúži ako prostredník, ktorý bude volať funkcie nachádzajúce sa v Player Team Handler. Skript Handle UI je potrebný, pretože po vytvorení tlačidla v Canvas objekte, musíme priradiť tomuto objektu určité správanie, a čo sa má stať po kliknutí. Keďže hráč nie je ako objekt vo svete pred zapnutím hry, nevieme priradiť toto správanie priamo z Player Team Handler. Preto budeme priradzovať funkcie z Handle UI skriptu, ktorému po vytvorení

lokálneho hráča priradíme premennú udržiavajúcu inštanciu objektu lokálneho hráča. Následne pri výbere tímu voláme funkciu z Player Team Handler, ktorá hráčovi priradí tím.

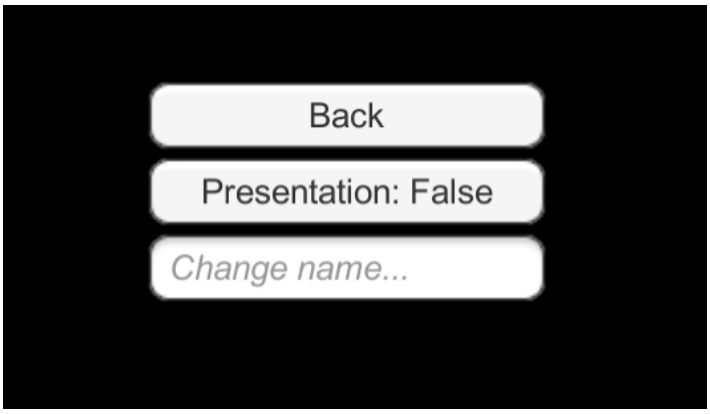
4.9.3 Hlavné menu



Obr. 4.14: Používateľské rozhranie hlavného menu

Do hlavného menu sa hráč dostane po stlačení klávesy ESC. Celé hlavné menu je ukázané na obrázku 4.14. Ako môžeme vidieť nachádzajú sa v ňom štyri tlačidlá. Po stlačení prvého, respektíve najvrchnejšieho tlačidla, sa hlavné menu vypne a hráč pokračuje v hre. Po stlačení druhého tlačidla sa hráč dostane do menu s nastaveniami. Tretie tlačidlo odpojí hráča zo servera, čím sa dostane do menu pre online pripojenie. Posledné tlačidlo slúži na odpojenie hráča zo servera a následnom vypnutí hry.

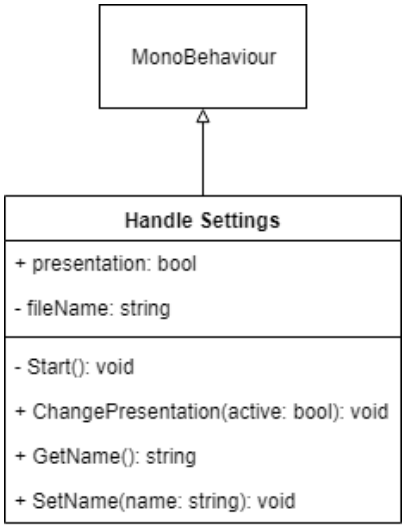
4.9.4 Menu pre nastavenia



Obr. 4.15: Používateľské rozhranie pre menu s nastaveniami

Ako to je zobrazené na obrázku 4.15 v tomto menu si hráč dokáže zapnúť prezentačný mód ako aj zmeniť svoje meno.

4.10 Nastavenia

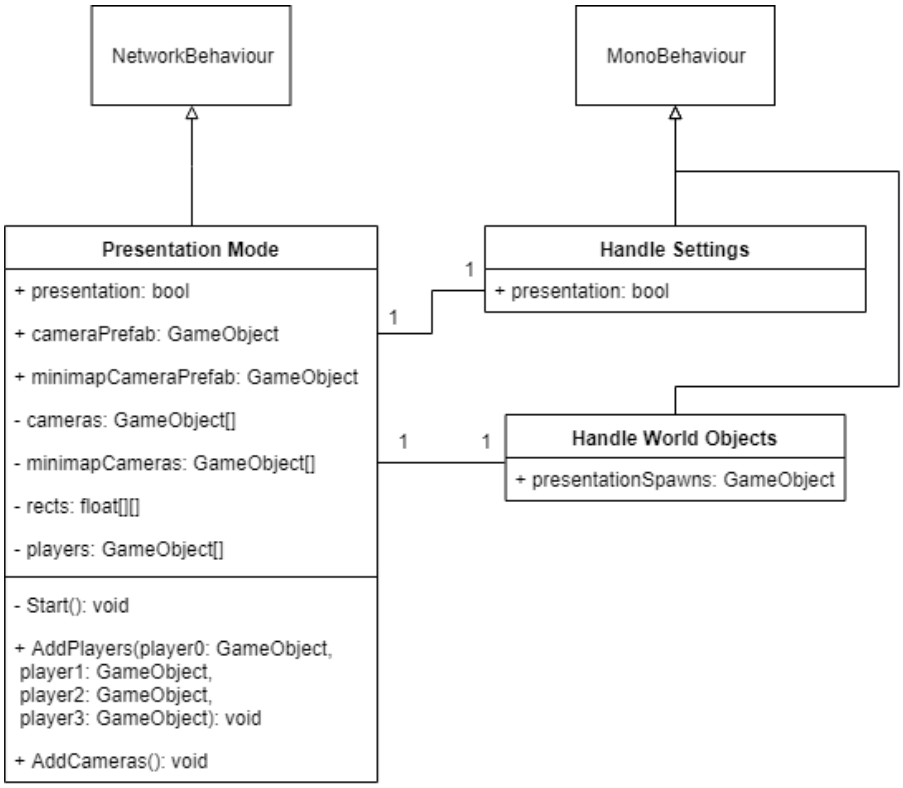


Obr. 4.16: Diagram tried potrebných pre nastavenia

Na obrázku 4.16 môžeme vidieť triedu, respektíve skript Handle Settings starajúci sa o ukladanie nastavení hráča, čo zahŕňa jeho meno a informáciu o tom, či je pre-

zentačný mód vypnutý alebo zapnutý. Na ukladanie budeme používať jednoduchý textový dokument, ktorý po spustení hry načítame, respektíve ak neexistuje ho vytvoríme s preddefinovanými hodnotami. Ak budeme meniť hodnoty mena a zapínať či vypínať prezentačný mód, tento textový dokument sa nám bude celý prepisovať s novými hodnotami.

4.11 Prezentačný mód



Obr. 4.17: Diagram tried potrebných pre prezentačný mód

O prezentačný mód sa nám stará skript Presentation Mode, ktorý je aj so všetkými ostatnými potrebnými skriptami zobrazený na obrázku 4.17. Po spustení hry sa pozrieme pomocou Handle Settings na to, či je prezentačný mód zapnutý. Ak je zapnutý, server vytvoríme okamžite. To je z dôvodu, že chceme aby hra s prezentačným módom nemusela mať žiadnu interakciu s klávesnicou a myškou. Následne vytvoríme tri nové kamery, ktoré budú kamerami pre hráčov v prezentačnom móde. Nemusíme vytvárať štyri kamery, pretože jedna kamera sa

už nachádza vo svete. Tieto kamery nastavíme vzhľadom na obrazovku na pozície pravá spodná, pravá stredná, stredná spodná a stredná časť obrazovky. Okrem kamier musíme mať aj samostatné používateľské rozhranie pre tento mód, ktoré bude mať taktiež 4 časti reprezentujúce používateľské rozhranie pre všetkých hráčov, keďže každý hráč bude potrebovať zobrazovať svoje vlastné informácie. Tabuľka so skóre, stav hry či QR kódy môžeme dať na inú časť obrazovky, keďže tieto informácie sú pre každého hráča rovnaké.

4.12 Mobilný ovládač

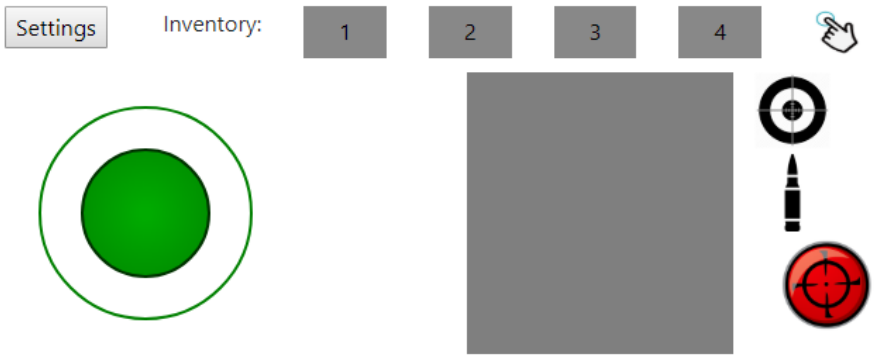
V tejto kapitole sa pozrieme na mobilný ovládač potrebný pre riadenie postáv hráčov pomocou mobilného zariadenia. Na mobilný ovládač použijeme internetový protokol MQTT, ktorý nám zaručí posielanie takmer okamžitých správ do našej hry. Vďaka týmto správam budeme vedieť aké akcie hráči chcú vykonávať so svojimi postavami. Najprv sa pozrieme na to, ako je potrebné pripraviť hru na získavanie správ z MQTT servera, následne sa pozrieme na všetko potrebné pre pripojenie mobilného zariadenia, na ktorom bude bežať náš ovládač. Ako posledné sa pozrieme na to, ako budeme spracovávať prijaté správy v hre.

4.12.1 Pripojenie hry na server

Mobile Controller po zapnutí hry vytvorí ID hry a pripojí sa na MQTT server školiaceho pracoviska. Následne sa pomocou vytvoreného ID prihlási k odberu témy, a teda táto téma bude mať tvar nášho predvoleného prefixu, herného ID a mriežky. Mriežka je znak vďaka čomu hra vie, že sa na tomto mieste môže nachádzať hocijaký reťazec. Následne vytvoríme QR kódy, do ktorých zakódujeme URL stránku, na ktorej sa nachádza náš mobilný ovládač spolu s herným ID ako parameter v URL. Keďže máme štyroch hráčov v hre a chceme aby si hráč mohol ľubovoľne zvoliť, ktorú postavu chce ovládať musíme vytvoriť štyri QR kódy pre každý objekt hráča. A aby sme tieto postavy rozlíšili musíme pridať do URL v QR kóde ešte jeden parameter, ktorý bude značiť ID postavy. Toto ID vytvoríme pri ich vkladaní do servera.

4.12.2 Mobilné zariadenie

Hráč pomocou mobilného zariadenia oskenuje QR kód z hry, ktorý ho presunie na stránku zakódovanú v QR kóde. Ako prvé sa na stránke zoberú URL parametre poslané z hry. Následne sa ovládač pripojí na MQTT server, ako to bolo aj pri hre. Keďže neposielame žiadne údaje do mobilného zariadenia, nie je potrebné aby sme sa s týmto zariadením pripojili na tému, ako to bolo v hre. Namiesto toho budeme iba posielať hráčov vstup do hry.



Obr. 4.18: Používateľské rozhranie mobilného ovládača

Na obrázku 4.18 je zobrazený mobilný ovládač. Nachádza sa v ňom jedna páčka, ktorá pri zmene pozície relatívnej na stred pošle informáciu o jej polohe do hry. Keďže hráči budú chcieť so svojimi postavami chodiť a zároveň aj pozerať okolo seba, je potrebné náš ovládač pripraviť na viac dotykovú interakciu. To dosiahneme tak, že sa budeme stále pozerať na ID dotyku, ktoré začalo interakciu s obrazovkou a následne pri zmene pozície dotyku na obrazovke budeme stále porovnávať toto ID s ID, ktoré zmenilo polohu. Na pohľad nebudeme používať páčku, ale si zoberieme inšpiráciu z rôznych mobilných hier na podobný žáner a vytvoríme plochu, ktorá je na obrazovke zobrazená sivým štvorcom. Ak sa hráč začne dotýkať danej plochy začne sa interakcia s ovládačom. Po zmene pozície sa pošle údaj ohľadom zmeny pohybu relatívne k poslednej zaznamenanej pozícií a tak zistíme, do ktorého smeru a ako rýchlo sa hráč chce otočiť so svojou postavou a následne tieto údaje môžeme posielať do hry. Na pravej strane ovládača môžeme

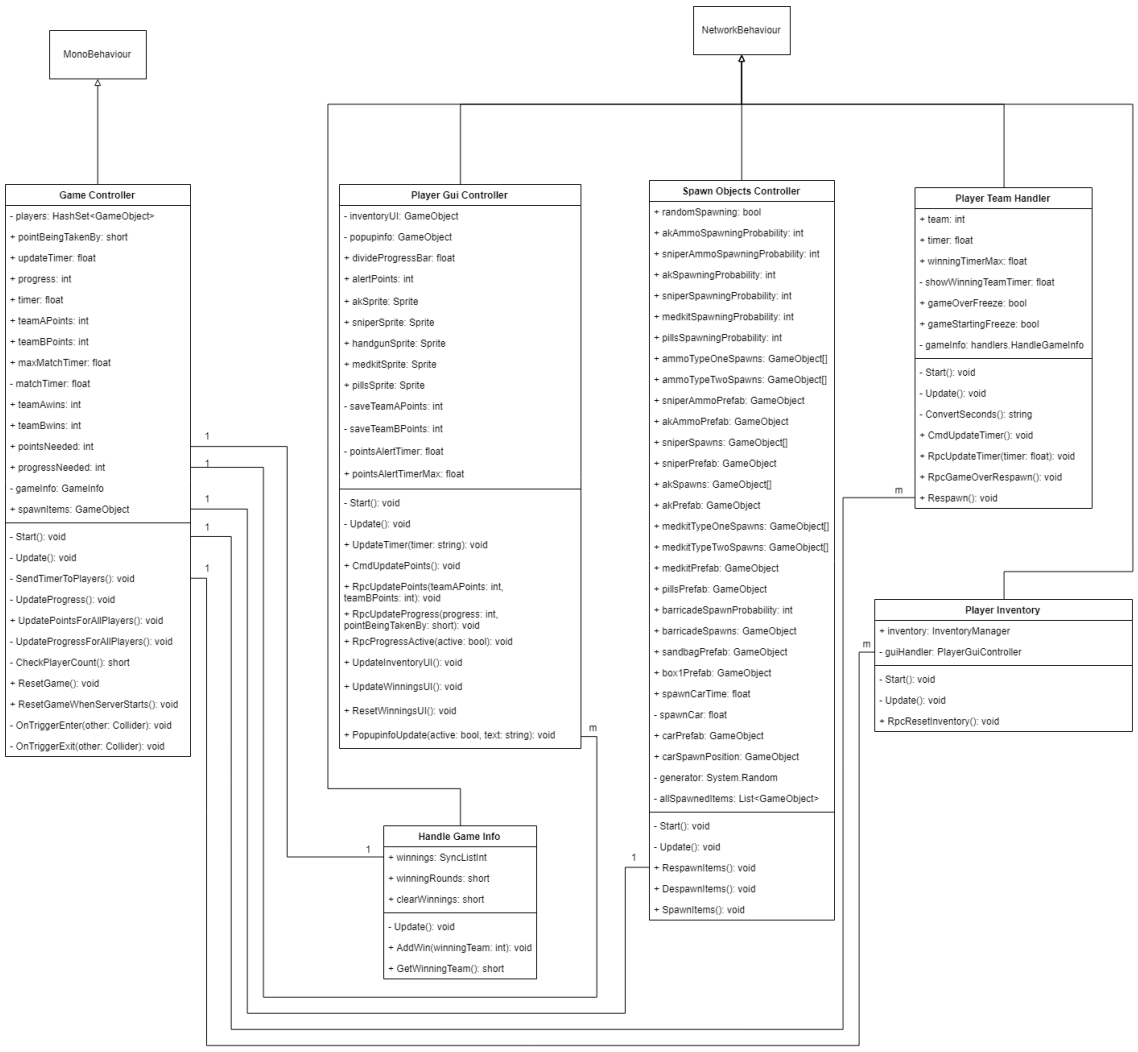
vidieť štyri ikonky. Zhora dole tieto ikonky posielajú správy o zobratí predmetu zo zeme, zamierení so zbraňou, prebití nábojov a strieľaní. Informácie o priblížení so zbraňou, prebití zbrane a zobratí predmetu zo zeme sa pošlú hneď ako hráč začne interakciu s ikonkou. Pri streľbe sa pošlú dve správy, prvá pri začiatku interakcie s ikonkou a druhá po jej skončení. Na vrchnej časti môžeme vidieť päť tlačidiel. Prvé tlačidlo zobrazí menu, v ktorom si hráči môžu zmeniť meno a invertovať rotáciu pohľadu. Posledné štyri tlačidlá na mobilnom ovládači slúžia na ovládanie inventára. Pokiaľ hráč stlačí tlačidlo spracuje sa inventár na danej pozícií ako to bolo aj pri inventári v hre. Na výmenu predmetov v inventári hráč musí kliknúť na jedno tlačidlo a potiahnuť ho nad druhé tlačidlo. Na vytvorenie tohto správania musíme zistiť ID dotyku, ktoré začalo interakciu s ovládačom a následne po skončení interakcie pozrieme, nad ktorým tlačidlom sa dotyk nachádzal. Ak to je iné tlačidlo, pošleme príkaz na výmenu predmetov v inventári. Na vyhodenie predmetu z inventára budeme pozerať či sa pri ukončení interakcie nachádzal dotyk nad tlačidlom, čo nám zabezpečí že na vyhodenie z inventára musí hráč potiahnuť tlačidlo smerom hore.

4.12.3 Spracovanie správ

Správy sa posielajú do hry, ktorá ich prijíma asynchrónne. Unity však nepodporuje asynchrónnu zmenu stavu objektov. Preto sú správy vložené do zoznamu, ktorý sa vykonáva v metóde *Update*(). Prvú správu, ktorú dostaneme z mobilného zariadenia je informácia o pripojení nového hráča do hry. Pri tejto správe priradíme ID hráča z ovládača k hráčovmu objektu. Následne mobilné zariadenie bude posielať správu o tom, že je hráč stále pripojený každých päť sekúnd. Ak hra nedostane túto správu po dvanástich sekundách odpojí daného hráča z hry a je pripravená na nové pripojenie. Dôvod pre dvanásť sekúnd je, že predpokladáme stratu tejto informácie. Ak by hra čakala iba päť sekúnd, mohlo by to viesť k omylným odpojeniam aktívnych hráčov. Preto ich odpojíme až po dvoch neúspešných prijatiach správy, pričom máme vyhradené dve sekundy pre možné internetové oneskorenie. Pri každej novej správe pozrieme na ID hráča, ktorý poslal správu, a porovnáme ho s ID v hráčových objektoch. Ak sú ID rovnaké, vykonáme na tomto objekte požadovanú akciu podľa správy, ktorá prišla z mobilného ovládača. Ak správa obsahuje viac informácií, ako napríklad zmena pozície alebo zmena mena, vytvoríme túto správu s určitým znakom a následne pomocou tohto znaku daný

reťazec rozdelíme a vykonáme jednotlivé príkazy.

4.13 Priebeh hry



Obr. 4.19: Diagram tried pre priebeh hry

O priebeh hry sa stará skript Game Controller, ktorý môžeme vidieť na obrázku 4.19. Game Controller ako aj Spawn Objects Controller sú skripty nachádzajúce sa v objekte, ktorý je zničený na každom klientovi. Tieto skripty bežia teda iba na serveri. Game Controller skript má v metóde *Update*() odpočet. Pokiaľ tento odpočet je rovný alebo menší nule, pozrieme na počet hráčov nachádzajúcich sa na vlajke pre jednotlivé tímy. To dosiahneme udržiavaním zoznamu všetkých hráčov

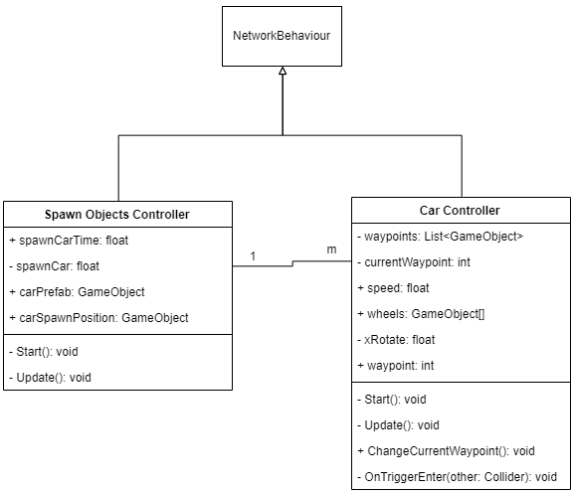
na vlajke. Do tohto zoznamu vložíme každého hráča, ktorý vošiel do vlajky. Ak hráč vyšiel z vlajky, odoberieme tohto hráča zo zoznamu. Tím, ktorý má viac hráčov na vlajke začne preberať vlajku, pokiaľ túto vlajku už nevlastní. Tím získava bod pokiaľ sa skončil časovač a vlajka je prebratá pre daný tím.

Server každú sekundu posiela klientom informácie o aktuálnom počte bodov a každému hráčovi na vlajke informáciu o stave vlajky.

Hra má nastavený čas pre hru na päť minút. Po uplynutí tohto času vyhráva hru tím s najväčším počtom bodov. Tento čas sa neposiela klientom periodicky ale pošle sa iba raz, a to po pripojení klienta. Následne sa každému klientovi odpočítava tento čas lokálne.

Výhra každého kola sa zapisuje do zoznamu, ktorý sa synchronizuje online. Hru vyhráva tím, ktorý dosiahol tri výhry. Po výhrach sa resetuje každý predmet po svete a každému klientovi sa pošle príkaz na resetovanie inventára.

4.14 Auto



Obr. 4.20: Diagram tried pre auto

Auto je jednoduchý objekt pohybujúci sa predurčenou rýchlosťou. Tento objekt sa vytvára v skripte Spawn Objects Controller, ktorý môžeme vidieť na obrázku 4.20. Nato, aby tento objekt vedel chodiť po ceste a aj do zákrut, použijeme už spomínane objekty Waypoint. Auto po vytvorení nájde objekt Waypoints, ktorý je rodičom objektov Road ako to bolo ukázané v návrhu riešenia. Následne náhodne

vyberie jednu z ciest a vloží každý Waypoint do zoznamu. V metóde *Update*() auto začne prechádzať týmito Waypoint objektami, respektíve sa k ním začne pomaly približovať kým do nich nevojde, čím vyhodí aktuálny Waypoint a začne sa pohybovať k ďalšiemu. Na poslednom Waypoint objekte sa auto zničí.

4.15 Používateľské rozhranie

Aj keď už vieme z jednotlivých kapitol, ako sú implementované jednotlivé elementy používateľského rozhrania, musíme vyriešiť jeden nedostatok. Keďže každý element má fixnú veľkosť, ak by používateľ hral na príliš malej alebo naopak na príliš veľkej obrazovke, naše požívateľské rozhranie by nebolo veľmi praktické. Preto je potrebné Canvas Scaler komponent v Canvas objekte nastaviť na hodnotu Scale With Screen Size. Táto funkcia nám zaručí aby sa každý element, respektíve objekt, menil v závislosti na veľkosti obrazovky. Táto funkcionalita funguje tak, že Canvas Scaler zoberie základné rozlíšenie, ktoré mu nastavíme a pomocou toho zväčšuje elementy používateľského rozhrania podľa veľkosti obrazovky. Napríklad ak nastavíme základné rozlíšenie na hodnotu 1280x720 a hráč by hral na rozlíšení 1920x1080 tento komponent by mu zväčšil každý element používateľského rozhrania jeden a pol krát.

4.16 Agent

Agent používa skript Agent Controller. Tento skript pridá agentovi život, podobným spôsobom ako to bolo pri hráčovi. Po smrti agenta sa tento skript stará o vytvorenie predmetov okolo mŕtveho agenta. Agent bude využívať umelú inteligenciu spolu s formálnymi metódami, riešené v súvisiacej bakalárskej práci Formálne metódy a strojové učenie vo virtuálnom priestore: autonómna entita.

4.16.1 Umelá inteligencia

Umelá inteligencia je dosiahnutá pomocou prostriedku od Unity - ML-Agents. V hre máme objekt akadémie, ktorý riadi učenie umelej inteligencie. V tomto objekte vieme aj zrýchliť učenie až na sto násobnú rýchlosť. Učenie je založené na spracovaní odmien a pozorovaniach, ktoré agent získava. Po zapnutí učenia sa agent

začne pohybovať k nami definovanému cieľu. Tento cieľ je jednoduchá kocka vo svete, ktorú hráči na svojich kamerách nevidia. Naučený model sa ukladá do .nn alebo .byte súborov, ktorý vieme použiť ako naučený mozog pre agenta.

4.16.2 Formálne metódy

Počas hry riadi pohyb agenta natrénovaný model strojového učenia. Môžu nastať situácie, kedy sa agent dostane príliš blízko objektom vo svete. Preto je jeho správanie kontrolované pomocou komponentu vytvoreného formálnymi metódami. Agent počas učenia dostáva akcie, ktoré má vykonávať. Predtým ako sú vykonane, sú informácie o stave agenta a prostredia poslane do verifikovaného komponentu. Ak sú splnene podmienky definovane v tomto komponente, vráti sa nová rotácia, do ktorej sa ma agent natočiť. Ak nie sú splnene, agent pokračuje s akciou obdržanou z natrénovaného modelu. Podmienky, na ktoré agent pozerá sú vzdialenosti medzi agentom a najbližšou prekážkou ako aj fakt či sa nachádza cieľ agenta v jeho blízkosti bez akýchkoľvek prekážok.

5 Vyhodnotenie

V tejto kapitole sa pozrieme na to, ako prešla naša hra a agent v nej testovaniami, ktorým boli podrobené. Najprv sa však pozrieme na náročnosť vývoja tejto hry.

5.1 Vývoj hry

V tejto časti sa pozrieme na náročnosť pre vývoj hry pre viacerých hráčov. Náročnosť budeme určovať časom potrebným na vývoj danej časti.

Vyhodnotenie vývoja					
Názov	Vývoj	Synchr-onizácia	Testova-nie	Oprava	Celkovo
Online komponent	2	0	2	1	5
Terén	5	0	0.5	0	5.5
Pohľad a pohyb	2	0	0.5	0	2.5
Fyzika	1	0	0.5	0	1.5
Zdravie	1	2	2	0.5	5.5
Streľba	1	4	5	2	12
Inventár	2	0	2	2	6
iné hráčové mechaniky	3	4	6	2	15
Používateľské rozhranie	16	0	4	7	27
Minimapa	1	0	0.5	0	1.5
Menu	4	0.5	2	2	8.5
Nastavenia	2	0	1	0.5	3.5
Prezentačný mód	8	3	10	5	26
Mobilný ovládač	6	0	3	8	17
Priebeh hry	4	10	6	4	24
Auto	1	3	2	2	8
Predmety	2	10	4	3	19
Agent	1	4	5	1	11
Celkovo	62	40.5	56	40	198.5

Tabuľka 5.1: Vyhodnotenie náročnosti vývoja

Tabuľka 5.1 zobrazuje čas potrebný na vývoj jednotlivých častí hry. Tieto časy sú zobrazované v hodinách.

Z tabuľky môžeme vyhodnotiť, že najdlhšia časť tvorby hry bol samotný vývoj. Do tohto vývoja zaraďujeme prvotný vývoj bez predbežného testovania. Za vývojom je samotné testovanie, kde je potrebné podotknúť, že toto testovanie nebolo používateľské a bolo teda testované iba vývojármi. Na používateľské testovanie sa pozrieme v nasledujúcej kapitole 5.2 Testovanie hry. Najväčší problém pri testo-

vaní hry bola samotná synchronizácia, keďže bolo potrebné neustále zapínať dve inštancie hry na otestovanie prostredia na dvoch klientoch. Tretím najdlhším vývojovým procesom bola synchronizácia hry, aby každý klient dostával správne informácie do hry. Tento proces bol najdôležitejší pri priebehu hry a používateľskom rozhraní, keďže hráči sa do hry môžu napojiť v hocijakej fáze hry a stále si musia správne zosynchronizovať informácie, aby sa im zobrazili v používateľskom rozhraní. Oprava zahŕňala najkratšiu časť vývoja. Do tejto časti zaradujeme každú opravu kódu, ktorá bola výsledkom nájdenia určitého nedostatku v hre.

Najdlhší čas vývoja zabral vývoj používateľského rozhrania. To je hlavne preto, že do tejto časti započítavame aj tvorbu používateľského rozhrania pre menu a prezentačný mód. Synchronizácia pri tejto časti nezabrala žiaden čas, pretože táto časť sa zaoberá iba samotným zobrazovaním údajov na obrazovku. Predpokladáme teda, že každý hráč už má potrebné údaje zosynchronizované.

Najzložitejšie časti hry pre synchronizáciu boli scenár hry a predmety. Pri scenári hry bolo potrebné synchronizovať každú informáciu o stave hry. Tie tvoria informácie o výhrach v jednotlivých kolách a všetko potrebné pre aktuálne kolo. U predmetov bolo potrebné synchronizovať aktuálnu hráčovu zbraň, každý predmet, ktorý sa nachádza vo svete ako aj barikády. Po hráčovom vyhodení predmetu z inventára bolo potrebné tento predmet vytvoriť pre každého klienta, ako aj novo napojeného klienta a zničenie každého predmetu po jeho zobraní hocijakým hráčom.

Najdlhší čas testovania zabralo testovanie prezentačného módu. Dôvodom je nutnosť zapnutia tohto módu a neustáleho pripájania každého hráča do hry a následné testovanie jednotlivých mechaník.

Mobilný ovládač bolo potrebné najviac opravovať, pretože bolo potrebné tento ovládač neustále prispôsobovať novým mechanikám.

Nakoniec je potrebné podotknúť, že časy pre agenta zahŕňajú iba jeho vývoj a synchronizáciu pre život a tvorbu predmetov po smrti. Taktiež zahŕňajú časy pre synchronizáciu pohybu a testovanie a opravu týchto elementov. Tieto časy teda nezahŕňajú jeho tvorbu odmien, proces učenia ani vývoj formálnych metód, keďže to nie je súčasťou tejto bakalárskej práce.

5.2 Testovanie hry

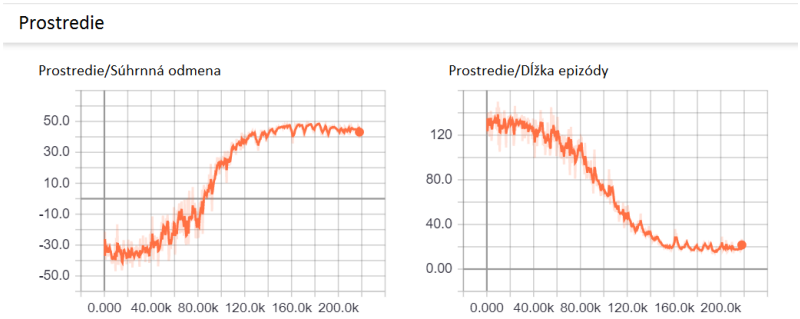
Hra sa testovala v troch iteráciách. Tieto iterácie boli testovanie hry bez prezentačného módu, s prezentačným módom a zároveň bez aj s prezentačným módom.

V prvej iterácií bez prezentačného módu bolo prítomných v hre päť hráčov, ktorí sa pripojili na jeden server z rôznych zariadení na rôznych sieťach. Každý hráč mal zosynchronizovaný každý predmet vo svete ako aj každý novo vytvorený objekt. Pri hraní mali hráči najväčší problém s manipuláciou inventára. Hráči nevedeli ako vymeniť alebo vyhodiť zbrane z inventára. To bolo zapríčinené tým, že v hre nie je prítomné žiadne menu, ktoré by ovládanie hry vysvetľovalo.

V druhej iterácií sa hralo na hre so zapnutým prezentačným módom kde hrali traja hráči. Každý hráč vedel, že na pripojenie do hry im stačilo načítať QR kód. Následne hráči začali ovládať svoje postavy s čím nemali žiadne problémy. Ovládanie inventára v tomto prípade nebolo problematické, pretože poťahovanie elementov nad iné elementy na obrazovke je pri mobilných zariadeniach už štandardom.

V tretej iterácií bol server vytvorený s prezentačným módom, na ktorý sa pripojil hráč s vlastným počítačom bez prezentačného módu. Pri tomto testovaní sme zistili, že hranie s mobilným ovládačom proti hráčom na vlastných počítačoch nie je úplne ideálne a môže byť až frustrujúce. Vďaka tomuto testovaniu máme nové možnosti na vylepšenie mobilného ovládača. Možné vylepšenie je napríklad automatické zameriavanie. Ich implementácia nebola súčasťou tejto bakalárskej práce, ale môže byť v budúcnosti pri jej nadviazaní.

5.3 Testovanie agenta



Obr. 5.1: Grafy trénovania agenta

Na obrázku 5.1 je zobrazený proces učenia nášho agenta. Pravý graf ukazuje priemernú odmenu, ktorú agent dostal za akcie pri určitom počte krokov. Pri hodnote približne 82 tisíc sa agent dostal do kladných hodnôt pre odmeny. Následne agent potreboval ešte približne 40 tisíc krokov aby sa dostal k najlepším odmenám, ktoré je možné dosiahnuť v našom virtuálnom priestore. Druhý graf zobrazuje počet krokov potrebných na nájdenie cieľa pri určitých počtoch krokov. Ako môžeme vidieť najprv agent potreboval až 120 krokov na nájdenie cieľa. Po jeho natrénovaní a teda pri 140 tisíc krokoch potreboval na nájdenie cieľa iba 20 krokov. Toto učenie trvalo 4 hodiny pri 19 násobnom zrýchlení hry.

Trénované modely			
Model	Úspech	Neúspech	Úspešnosť
Naučený bez formálnych metód spustený bez formálnych metód	3292	65	98,0255
Naučený bez formálnych metód spustený s formálnymi metódami	3175	61	98,0787
Naučený s formálnymi metódami spustený bez formálnych metód	4419	101	97,7144
Naučený s formálnymi metódami spustený s formálnymi metódami	4970	79	98,41

Tabuľka 5.2: vysledky testovania rôznych naučených modelov

Na otestovanie boli vytvorené štyri modely. Tieto modely môžeme vidieť v tabuľke 5.2. Ako z tabuľky vyplýva model natrénovaný a zároveň spustený s formálnymi metódami mal najlepšiu úspešnosť. Tento model taktiež našiel najviac cieľov, čo značí že sa agent najmenej zasekával, a teda išiel stále priamo za cieľom. Model

natrénovaný s formálnymi metódami ale spustený bez nich rýchlejšie našiel ciele. Stále išiel priamo za cieľom, no ako môžeme vidieť podľa neúspešných nájdení cieľa sa tento model zasekával najviac zo všetkých. Pri trénovaní bez formálnych metód, modelu trvalo stále najdlhšie nájdenie cieľa. Úspešnosť však majú väčšiu ako pri natrénovaní s formálnymi metódami a spustený bez formálnych metód. Aj tieto dva modely sa teda menej zasekávali.

6 Zhrnutie práce

Na projekte sme pracovali v dvojčlennom tíme. Našou úlohou bolo vloženie autonómnej entity do virtuálneho prostredia. Mojou úlohou a teda úlohou tejto bakalárskej práce bolo vyhodnotenie rôznych hier a herných rámcov, do ktorých by sme takúto autonómnu entitu vedeli vložiť a otestovať. Prešli sme si dve hry, do ktorých by sme mohli vložiť našu autonómnu entitu. Jedna z hier bola Minecraft, kde by sme pomocou pluginov vedeli pridať rôzny obsah na náš vytvorený server. Druhá hra bola Dota 2. V tejto hre by sme vedeli vytvárať samostatné módy, ktoré by hráči vedeli hrať v prostredí hry Dota 2. Následne sme sa pozreli na dve hry, ktoré majú verejne dostupné zdrojové kódy. Tieto hry boli PlaneShift a Worldforge. Obsah pre PlaneShift by sme dokázali vytvárať pomocou rámca Unreal Engine. Pri hre Worldforge sa obsah do hry vytvára v ich klientovi, v ktorom sa kóduje pomocou jazyka Python. Poslednú možnosť, na ktorú sme sa pozreli bola rámec Unity. Každá z týchto možností mala svoje výhody aj nevýhody, no nakoniec sme sa rozhodli vytvárať naše virtuálne prostredie v rámci Unity. Dôvody pre výber tohto rámca zahŕňali obchod prostriedkov Asset Store, ML-Agents, voľnosť a Unet. Obchod prostriedkov nám pomohol pri vkladaní bezplatných prostriedkov do našej hry bez akýchkoľvek problémov. ML-Agents je prostriedok, pomocou ktorého kolega v druhej bakalárskej práci vytváral autonómnu entitu. Tento prostriedok poskytuje priamo Unity a teda je jednoducho implementovateľný do Unity hier. Voľnosť nám poskytla možnosť vytvorenia hry pre OpenLab na Technickej univerzite v Košiciach. Unet je jednoduchý rámec, vďaka ktorému sme vedeli jednoducho prepojiť hráčov online a synchronizovať ich hry.

Po výbere Unity na tvorbu virtuálneho prostredia sme začali navrhovať hru. Navrhnutá hra bola strieľačka z pohľadu prvej osoby, kde hráči by mali cieľ prebrať vlajku v strede mapy. V tejto časti by sa nachádzala aj autonómna entita. Prešli sme si mechaniky, ktoré chceme implementovať do našej hry a objekty potrebné

na tieto mechaniky.

Keď sme už mali predstavu o tom ako chceme aby naša hra vyzerala začali sme túto hru implementovať. Mapa je jediná potrebná časť hry, ktorú agent potrebuje na naučenie pohybu. Preto sme ako prvé vytvorili dôkladnú mapu sveta. Po vytvorení mapy kolega mohol začať trénovať autonómnu entitu a mojou úlohou bolo vytvorenie mechaník a cieľa pre hráčov.

Po vytvorení mapy sme pridali do hry online komponent poskytujúci pripojenie hráčov online. Keď už hráči vedeli vytvárať a pripájať sa na servery, začali sme vytvárať objekt, ktorý budú títo hráči ovládať. Tomuto objektu sme priradili kameru, tak aby objekt hráča bol z pohľadu prvej osoby. Následne sme hráčovmu objektu pridali fyziku v mape. Konečne sme hráčovi mohli začať pridávať potrebné mechaniky. Tieto mechaniky zahŕňali pohyb, zdravie, streľbu, inventár a pripájanie do tímov. K týmto mechanikám sme vytvorili používateľské rozhranie, ktoré zobrazuje potrebné informácie. Tieto mechaniky bolo potrebné synchronizovať online.

Keď už sme mali hráčov, ktorý sa vedeli pohybovať po mape a zabíjať nepriateľsky tím, mohli sme začať vytvárať cieľ pre hráčov. Vytvorili sme vlajku, ktorá sa nachádza na serveri a počíta hráčov nachádzajúcich sa na tejto vlajke, čím vedeli hráči pre svoj tím prebrať vlajku. Tímom sme pridali body. Hráči tieto body získavajú pre svoj tím prebratím vlajky alebo zabitím nepriateľského tímu. Následne sme vytvorili vyhratie hry. Výhra hry nastane vtedy, keď tím vyhral tri kolá, pričom kolo sa vyhrá po získaní potrebného počtu bodov.

Po vytvorení cieľa sme pridali do hry predmety, ktoré hráč dokáže zobrať zo zeme a pridať si ich do inventára. Tieto predmety sú rôzne zbrane a liečivé predmety. Okrem týchto predmetov sme do hry pridali náboje, ktoré hráč zoberie po prejdení cez objekt nábojov. Spolu s týmito predmetmi sa vytvárajú aj barikády, ktoré slúžia ako krytie pre hráča. Všetky tieto predmety sa vytvárajú náhodne, čo vytvára mapu viac dynamickú a nepredvídateľnú. Okrem statických prekážok sme vytvorili aj auto, ktoré prechádza z jednej strany mapy na druhú. Vďaka týmto barikádam a autu je mapa medzi všetkými hrami rozdielna, čo vytvára mapu ťažšie naučiteľnú pre autonómnu entitu.

Keď sme už sme mali hru, v ktorej hráči mali aj cieľ mohli sme začať prispôsobovať našu hru pre OpenLab. To sme dosiahli vytvorením špeciálneho módu, ktorý nazývame prezentačný mód. Pri tomto móde sme vytvorili štyroch hráčov

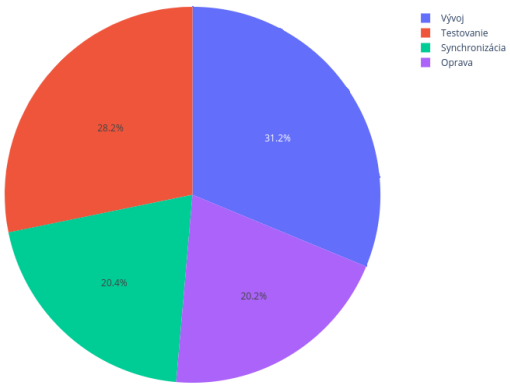
do hry a rozdelili obrazovku pre jednotlivých hráčov. Nakoniec sme rozdelili aj používateľské rozhranie na zobrazovanie informácií pre jednotlivých hráčov. K tomuto prezentačnému módu sme vytvorili webovú stránku, ktorá bude prispôsobená pre mobilné zariadenia na ovládanie týchto hráčov. Ovládanie hráčov bolo dosiahnuté posielaním krátkych správ do hry a následným vykonávaním týchto príkazov na hráčovom objekte. Do hry sme nakoniec pridali štyri QR kódy pre jednotlivých hráčov, v ktorých je zakódované URL s týmto ovládačom spolu s informáciou, ktorý objekt hráča bude ovládať po načítaní tohto QR kódu.

Na zvýšenie používateľského zážitku sme vytvorili štyri menu pre hráčov, ktoré sú:

- Menu pre online pripojenie - slúži na vytvorenie a pripojenie na server
- Menu pre výber tímu - slúži na vyber tímu po pripojení k serveru
- Hlavné menu - slúži na odpojenie zo servera, vypnutie hry a prejdenie do nastavení
- Menu pre nastavenia - slúži na zapnutie prezentačného módu a pridanie mena

Všetky nastavenia ukladáme do textového dokumentu, z ktorého sa tieto nastavenia načítavajú pri zapnutí hry.

Nakoniec sme do hry finálne pridali agenta, ktorému sme pridali životy a vypadávanie predmetov po jeho smrti.



Obr. 6.1: Graf vyhodnotenia náročnosti vývoja

Tvorba hry pre viacerých hráčov nebol náročný proces, no aj napriek tomu zabral veľa času. Na obrázku 6.1 môžeme vidieť percentuálne, ktorá časť tvorby hry bola najzdĺhavejšia.

Pri testovaní hry a agenta sme zistili, že hra je plne funkčná na rôznych počítačoch na rôznych sieťach. U agenta sme zistili, že využitie naučeného modelu s formálnymi metódami a následne spustený s formálnymi metódami je najideálnejšie riešenie.

7 Záver

Virtuálne prostredie sa v dnešnej dobe využíva na rôzne účely, ako architektúra či filmový priemysel. Účelom nášho projektu bolo zamyslenie sa nad myšlienkou využitia virtuálneho prostredia na overenie funkčnosti softvéru. Táto myšlienka vznikla hlavne z dôvodu veľkého rastu herného priemyslu a nevyužitého času stráveného nad hrami. Takéto hry by teda mohli slúžiť ako skúšobná pôda, kde by nič netušiaci hráči otestovali funkčnosť nami testovaného softvéru.

Cieľom tejto bakalárskej práce bolo vytvorenie virtuálneho prostredia, v ktorom by sme dokázali naučiť a otestovať tak správanie autonómnej entity. K naplneniu tohto cieľa sme využili rôzne prostriedky poskytované herným priemyslom.

Naše virtuálne prostredie sme sa snažili vytvoriť čo najviac dynamické. To sme dosiahli pridaním online komponentu do hry, čo nám zabezpečilo pripojenie viacerých hráčov. Hráčom sme vytvorili rôzne mechaniky a ciele aby sme ich v hre a okolo našej autonómnej entity udržali čo najdlhšie. Okrem pridaní viacerých hráčov do hry sme vytvorili rôzne prekážky, ktoré sa vytvárali s určitou náhodnosťou.

Vytvorenie hry prebehlo úspešne. Pri testovaní s rôznymi hráčmi na rôznych zariadeniach a sieťach bolo pripojenie bezproblémové. Každý hráč mal celú hru zosynchronizovanú a plne funkčnú.

Testovanie autonómnej entity prebiehalo v štyroch iteráciách, kde sme porovnávali správanie agenta s a bez použitia formálnych metód. Pri tomto testovaní boli formálne metódy úspešne a agent sa správal najlogickejšie v iterácií, kedy sa naučil aj bežal s formálnymi metódami.

Literatúra

1. CLEARSY. *Atelier B Translators* [Stiahnuteľné pdf]. Dostupné tiež z: https://www.atelierb.eu/wp-content/uploads/sites/3/ressources/DOC/english/translators-user-manual.pdf.
2. ING. ŠTEFAN KOREČKO, PhD. BKPI. Dostupné tiež z: http://it4kt.kpi.fei.tuke.sk/obsah/bkpi.
3. KLEVTSOV, Vadim. Lua java byte code translator. Dostupné tiež z: https://github.com/vad23klev/lua-java-translator.
4. MELHASE, Troy. Simple but effective library to translate Java source code to Python. Dostupné tiež z: https://github.com/natural/java2python/.
5. TEAM, PlaneShift. About. Dostupné tiež z: https://www.planeshift.it/About.
6. TEAM, Unity. Machine Learning. Dostupné tiež z: https://unity3d.com/machine-learning.
7. TEAM, Unity. Whether you're a professional team or freelancer, a hobbyist, or a total beginner, there's a Unity plan for you. Dostupné tiež z: https://store.unity.com/.
8. TEAM, Worldforge. CREATE YOUR OWN WORLD. Dostupné tiež z: https://www.worldforge.org/index.php/create/.
9. WATER, Jens van de. HOW TO CREATE AN ONLINE MULTIPLAYER GAME WITH PHOTON UNITY NETWORKING. Dostupné tiež z: https://blogs.unity3d.com/2018/08/02/evolving-multiplayer-games-beyond-unet/.
10. BARRON, Christine. Pass the Butter // Pancake bot. 2018. Dostupné tiež z: https://connect.unity.com/p/pancake-bot?_ga=2.19617916.533531763.1544913214-507508564.1537962041.

11. GILBERT, Ben. 'Minecraft' is still one of the biggest games in the world, with over 91 million people playing monthly. 2018. Dostupné tiež z: https://www.businessinsider.com/minecraft-has-74-million-monthly-players-2018-1.

12. HOUSE, Brandi. Evolving multiplayer games beyond UNet. 2018. Dostupné tiež z: https://blogs.unity3d.com/2018/08/02/evolving-multiplayer-games-beyond-unet/.

13. OPENAI. OpenAI Five Benchmark: Results. 2018. Dostupné tiež z: https://openai.com/blog/openai-five-benchmark-results/.

14. ROBLOX. Data Stores. 2018. Dostupné tiež z: https://www.robloxdev.com/articles/Data-store.

15. TEAM, Bukkit Dev. Bukkit. 2018. Dostupné tiež z: https://dev.bukkit.org/.

16. BUSCH, David. Hide / Escape - Avoidance of Pursuing Enemies. 2017. Dostupné tiež z: https://connect.unity.com/p/hide-escape-avoidance-of-pursuing-enemies?_ga=2.19617916.533531763.1544913214-507508564.1537962041.

17. OPENAI. Dota 2. 2017. Dostupné tiež z: https://openai.com/blog/dota-2/.

18. SOFTWARE, Valve. Dota 2 Workshop Tools/Scripting/API/Global.CreateHTTPRequest. 2017. Dostupné tiež z: https://developer.valvesoftware.com/wiki/Dota_2_Workshop_Tools/Scripting/API/Global.CreateHTTPRequest.

19. TEAM, PlaneShift. An Unreal Arcade. 2017. Dostupné tiež z: https://www.planeshift.it/article/2017/An%20Unreal%20Arcade.

20. THIEN, Chau Chi. Metal Warfare - Real Time Strategy game. 2017. Dostupné tiež z: https://connect.unity.com/p/metal-warfare-real-time-strategy-game-special-edition-for-ai-ml-challenging?_ga=2.19617916.533531763.1544913214-507508564.1537962041.

21. LINN, Allison. Project Malmo, which lets researchers use Minecraft for AI research, makes public debut. 2016. Dostupné tiež z: https://blogs.microsoft.com/ai/project-malmo-lets-researchers-use-minecraft-ai-research-makes-public-debut/.

22. MALÝ, Martin. Protokol MQTT: komunikační standard pro IoT. 2016. Dostupné tiež z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/.

23. GREG BROCKMAN, Ilya Sutskever; OPENAI. Introducing OpenAI. 2015. Dostupné tiež z: https://openai.com/blog/introducing-openai/.

24. NOYA. Beginners Guide to Dota Scripting. 2015. Dostupné tiež z: http://moddota.com/forums/discussion/135/beginners-guide-to-dota-scripting.

25. THURSTEN, Chris. Dota 2 Reborn and why the MOBA will never be the same again. 2015. Dostupné tiež z: https://www.pcgamer.com/dota-2-reborn-and-why-the-moba-will-never-be-the-same-again/.

26. VANBROCKLIN, Tyler. How to Learn Roblox and Roblox Studio. 2012. Dostupné tiež z: https://gamedevelopment.tutsplus.com/articles/how-to-learn-roblox-and-roblox-studio--gamedev-2304.

Zoznam príloh

Príloha A CD médium

- `doc` - Systémová príručka, používateľská príručka a bakalárska práca v elektronickej podobe
- `tex` - Zdrojové súbory bakalárskej práce v LaTeX formáte
- `lib` - Dokumentácia mobilného ovládača v anglickom jazyku
- `bin` - Spustiteľná verzia hry
- `src` - Zdrojové súbory pre Unity projekt a mobilný ovládač

Príloha B Používateľská príručka

Príloha C Systémová príručka - kvôli veľkému rozsahu iba v elektronickej podobe na priloženom CD médiu