Debreceni Egyetem – Informatikai Kar

Mesterséges Intelligencia alkalmazása az ESport területén

SZAKDOLGOZAT

|  |  |
| --- | --- |
| Témavezető: Lakatos Róbert | Készítette: Zagyva Ákos |

DEBRECEN

2023

Tartalom

[Bevezetés 5](#_Toc150885853)

[Mesterséges Intelligencia az Esportban 8](#_Toc150885854)

[Adatfeldolgozás, adatelemzés 8](#_Toc150885855)

[Anti-cheat rendszerek 8](#_Toc150885856)

[Szervezés, megvalósítás 8](#_Toc150885857)

[Edzések segítése 8](#_Toc150885858)

[Játékbeli MI alkalmazása 9](#_Toc150885859)

[Megerősítéses tanulás 10](#_Toc150885860)

[Megerősítéses tanulás bemutatása 10](#_Toc150885861)

[A megerősítés típusai 10](#_Toc150885862)

[Pozitív megerősítés 10](#_Toc150885863)

[Negatív megerősítés 11](#_Toc150885864)

[Ágens 11](#_Toc150885865)

[Algoritmusok kategorizálása 11](#_Toc150885866)

[Modell alapú algoritmusok 11](#_Toc150885867)

[Modell nélküli algoritmusok 12](#_Toc150885868)

[Megfelelő algoritmus választása 12](#_Toc150885869)

[Autóverseny szimulátor 14](#_Toc150885870)

[Pályák és a környezet: 14](#_Toc150885871)

[Autók: 15](#_Toc150885872)

[Autóvezetési élmény: 15](#_Toc150885873)

[Bajnokságok: 16](#_Toc150885874)

[A legismertebb szimulátorok 16](#_Toc150885875)

[iRacing: 16](#_Toc150885876)

[Assetto Corsa: 16](#_Toc150885877)

[Dirt Rally: 16](#_Toc150885878)

[F1 széria: 16](#_Toc150885879)

[Fejlesztési környezet 18](#_Toc150885880)

[Hardver 18](#_Toc150885881)

[Asztali számítógép: 18](#_Toc150885882)

[Notebook: 18](#_Toc150885883)

[Operációs rendszer 18](#_Toc150885884)

[Alkalmazott Technológiák 19](#_Toc150885885)

[Python 19](#_Toc150885886)

[Anaconda 19](#_Toc150885887)

[Jupyter kNotebook 19](#_Toc150885888)

[Virtuális környezet 20](#_Toc150885889)

[VJOY 20](#_Toc150885890)

[Pandas 20](#_Toc150885891)

[Pygame 21](#_Toc150885892)

[Numpy 21](#_Toc150885893)

[DirectInput 21](#_Toc150885894)

[Stable Baselines 21](#_Toc150885895)

[MatPlotLib 22](#_Toc150885896)

[TensorBoard 22](#_Toc150885897)

[Környezet felépítése 23](#_Toc150885898)

[Környezet inicializálása 23](#_Toc150885899)

[Megfigyelés és cselekvési tartomány meghatározása 23](#_Toc150885900)

[Képernyőképek készítése 24](#_Toc150885901)

[Interfész 24](#_Toc150885902)

[Telemetria adatok kinyerése 25](#_Toc150885903)

[A pálya meghatározása 25](#_Toc150885904)

[Interpolálás 27](#_Toc150885905)

[Az autó poziciója 27](#_Toc150885906)

[A környezet egyszerűsitése 27](#_Toc150885907)

[Cselekvés 27](#_Toc150885908)

[Jutalom számítása 28](#_Toc150885909)

[Visszaállítás 28](#_Toc150885910)

[Tanítás 29](#_Toc150885911)

[Modell kiértékelése 31](#_Toc150885912)

[Összefoglalás 33](#_Toc150885913)

[Köszönetnyilvánítás 35](#_Toc150885914)

# Bevezetés

A szakdolgozatom a mesterséges intelligencia alkalmazásához, felhasználásához kapcsolódik az esport területén. A választásom többek között azért esett erre a témára, mert maga az esport hobbi szinten közel áll hozzám, ugyanis online verseny szimulátorokkal én magam is játszok és indultam már kisebb drift bajnokságokon is. Ehhez társult az, hogy az egyetemi tanulmányaim során betekintést nyertem a mesterséges intelligencia világába. Az egyetemi órákon megismerkedtem a mesterséges intelligencia alapjaival, hogy hogyan is kell elképzelni magát a mesterséges intelligenciát, hogy hogyan működik, betekintést nyertem a gépi tanulásba. A hobbimhoz köthetően az órák egyikén egy vizuális szemléltetését láttam Lakatos Róbert tanár úr által, ahogyan egy, a megerősítéses tanulás módszereivel feltanított ágens egy profi szimulátor pilótát óriási időkülönbséggel győz le. [1] Az óriási időkülönbség közel egy másodpercet jelent, amely az autósport ezen világában jelentős differencia. Ekkor nyílt fel igazán a szemem a téma irányába és kezdett el foglalkoztatni maga a mesterséges intelligencia.

Az elmúlt évtizedben az esportra világszerte egyre nagyobb hangsúlyt fektetnek, a COVID-19 eseményei pedig a homeoffice és a távoktatás okán jelentősen megnövelték az esport játékosok számát a világon. Ez magával vonzotta azt is, hogy egyre több online verseny és rendezvény kerül lebonyolításra, valamint napjainkban az online játékok élveznek elsőbbséget a játékgyárak szemében is. Az esport lehetővé teszi egyes területeken, például a szimulátor versenyzésben, hogy olyanok is átültethessék valóságba a tudásukat akiknek az anyagi feltételek hiányoztak ahhoz, hogy több tízmillió forintos versenyautót vezessenek egy versenypályán. Erre egy remek példa Jann Mardenborough aki 19 éves korában a Gran Turismo Academy eddigi legfiatalabb győzteseként került ki az esport világából, majd a Nissan felajánlásával igazi versenyautóba ülhetett, és teljesítményeivel egészen a 24 órás, híres Le Mans-ig jutott, ahol elképesztő, harmadik helyezést tudott elhozni. [2] Valamint ide tudnám sorolni Rhys Tatterson-t aki a 2021-es(DMVC) Drift Masters Virtuális Bajnokság nyertese. Őt egy 900 lóerős Nissan Silvia S15 drift autó volánja mögé ültették be úgy, hogy előtte csak szimulátorban vezetett drift autót. [3] Meglepő módon pár perc alatt megszokta az új járművét, amelyet Piotr Więcek lengyel származású Drift Masters Európa Bajnoktól kapott meg, valamint őt kapta meg koordinátornak az anyósülésre.

Viszont ne csak az autóversenyzést említsem, a mai világban egyre kevesebb figyelmet kapnak a televíziós műsorok, leváltják őket az online streamek, közvetítések, melyek hatása, hogy milliók szegezik a tekintetüket a kijelzőnek, és bizony elég sok esetben azt nézik, hogy ki, hogyan játszik. A legjobb játékosok pedig többek között ennek köszönhetően a megélhetés szintjén tudják az esportot űzni, csak úgy, mint egy profi football játékos, olimpikon, vagy akár egy profi extrém sportoló. Ahhoz, hogy az egyik legjobb legyen valaki egy adott területen pedig rengeteg gyakorlás szükséges. Ehhez a gyakorláshoz tartom a egy megfelelő eszköznek a mesterséges intelligencia használatát. Természetesen lehet online játékokban mások ellen gyakorolni illetve a legtöbb játék felkínál AI ellenfeleket is vagy akár versenyezhetünk a más vagy a saját szellemünk ellen is, lehet visszamenőleg elemezni több lejátszott meccsünket, vagy megtett körünket a pályán, de lehetne egy feltanított mesteréges intelligencia ágens is, ami segít a saját gyengeségeink kiküszöbölésén.

Személy szerint mivel az autózás áll közel a szívemhez illetve ennek látom a hétköznapi életben is a felhasználhatóságát, így ezt ragadtam meg a szakdolgozatom konkrét témájának. Egy játékbeli AI (Artifical Intelligence, azaz mesterséges intelligencia) természetesen beprogramozható, hogy minden alkalommal tökéletesen végrehajtsa ugyan azt a mozdulatsorozatot. Annak érdekében, hogy ne egy robotot észleljünk mikor ezzel szembe találkozunk a játékok/szimulátorok használata közben, a fejlesztők néhány tucatnyi előre meghatározott szabályt vagy heurisztikát hoztak létre, melyek meghatározzák a játékbeli gépi ellenfelünk következő lépéssorozatait.[4]Ezt a gépet is lehet annyira nehézre faragni persze, hogy legyőzhetetlen legyen, és labdába se tudjon egy ember ellene rúgni, vagy megadni minden lépésre a választ kód formájában, hogy miként reagáljon a gép a mi mozdulatsorunkra, viszont minden lehetséges mozdulatot megadni valamint a környezeti változókat is szem előtt tartani rendkívül bonyolult lenne, mert végtelen számú megoldáson kellene gondolkozni a fejlesztőcsapatnak egy játék esetében. Ebből az okból kifolyólag pont az előre meghatározott döntéssorozatok azok, amik miatt ezekben az esetekben nem beszélhetünk mesterséges intelligenciáról, csak egy előre felprogramozott algoritmusról, amely nem fog önálló döntéseket hozni a folyamatban lévő események és környezeti változók alapján.

A kutatómunkám során felfedeztem olyan megoldásokat, melyek 2D felülnézetes egyedileg megalkotott környezetben valósítanak meg valamilyen szintű önvezetést, manőverezést egy pályán. Ezek fizika nélküli játékok voltak, ahol csak az objektum (autó) eltolása volt megoldott egy ágens segítségével egy rajzolt pályán, amely körül egy falat definiáltak, amibe ha bele ütközik a mozgatott jármű, akkor véget ér a játék. Az én gondolatom az volt, hogy legyen ez megoldva egy 3D-s környezetben, ahol a játékon belül valamilyen fizika is implementálva van, ami lehetőleg a valósághoz egészen közel áll. Az által, hogy hobbi szinten szimulátorozok eléggé kézenfekvő volt, hogy nem egy egyedileg elkészített környezetben, hanem már egy meglévő versenyszimulátorban szeretném ezt megvalósítani ahol a hétköznapi vezetéshez hasonló helyzetet tudunk megteremteni, valamint az alapvető inputok az autóvezetésnek megfelelnek. Tehát tudunk gázt adni, fékezni, kanyarodni finom mozdulatokkal korrigálni.

A fejlesztés megkezdésével hamar realizáltam, hogy ez a terület eléggé felfedezetlen. Egyetlen fejlesztőt találtam, aki gépi tanulási módszerrel próbál feltanítani egy modellt, szimplán arra, hogy egyáltalán haladjon a játékban valamerre az autó. Kiindulási alapnak ezt vettem alapul viszont a megerősítéses tanulás irányába indultam el, itt viszont egyáltalán nem találtam semmilyen modellt, tehát a saját modellemet kellett felépítsem, megfigyelnem a fejlődését. Ebből az okból adódóan nem volt más viszonyítási alapom, nem tudtam egyéb modellekkel összehasonlítani. A saját modell fejlesztése pedig rengeteg időt és energiát emésztett fel.

A mesterséges intelligencia napjainkban történő térhódításával úgy vélem egy az enyémhez hasonló projekt nagy sikereket tud elérni, és amint a szimulációban is megállja a helyét, könnyedén átültethető a valóságba is és felhasználható akár balesetmegelőzéshez, az ehhez megfelelő szenzorok segítségével. Az autózás része az emberek mindennapjainak, sokan autóval járnak munkába, boltba, rengetegen vannak az utakon ebben a pillanatban is de elenyésző azon emberek száma, aki kontrollálni is tudja az autóját szélsőséges körülmények között. Egy túlkormányzottság esetén egy megfelelően feltanított modell akár korrigálhatja a gépjárművet, hogy annak a megpördülését esetlegesen megakadályozza.

# Mesterséges Intelligencia az Esportban

A mesterséges intelligencia (röviden MI) az esport területén sokoldalúan alkalmazható és számos előny származhat belőle a játékélmény felpezsdítése, a teljesítmény növelése vagy a játékok felügyelete érdekében.

## Adatfeldolgozás, adatelemzés

Az MI segítségével hatalmas mennyiségű adatot tudnak a szakemberek gyorsan és eredményesen feldolgozni. Meccselemzésnél vagy telemetria adatok feldolgozásánál az MI segítségével a játékosok teljesítményét könnyedén lehet elemezni, kiértékelni, megtalálni az erősségeiket és gyengeségeiket, amik alapján a későbbiekben finomhangolni tudják a képességeiket vagy új stratégiát felépíteni.

## Anti-cheat rendszerek

Mesterséges intelligencia használatával monitorozhatók a játékosok és kiszűrhető közülök az, aki valamilyen segédprogramot azaz csalást próbál használni a játék során, Ez segíthet abban, hogy a játékmenet tiszta maradjon és csak a játékosok teljesítményén, képességén múljon a végkimenetel.

## Szervezés, megvalósítás

Az esport rendezvények többségéhez már nem kell a helyszínen is jelen legyünk, a legtöbb rendezvény manapság már online kerül megszervezésre leszámítva a nevesebb rangadókat. A csaló programok megszűrése ezért is jelentős, valamint MI alkalmazásával könnyedén összeállítható a versenynaptár, generálható a játékosok teljesítményére alapozva ágrajz, továbbá az online közvetítéshez is nagy segítséget tud nyújtani.

## Edzések segítése

Mesterséges intelligencia programok felhasználásával monitorozni tudjuk a játékosok testi és mentális állapotát, mely segíthet időben felismerni a túlterhelés vagy a stressz tüneteit, ezzel megelőzve egészségügyi problémákat. Továbbá stratégiákat, játékterveket tudunk összeállítani, optimalizálni, az ellenfél elleni taktikák kidolgozását nagyban elő tudja nekünk segíteni. Ajánlásokat kaphatunk, hogy miben, hogyan fejlődhetünk. Egy ilyen program rámutat a hibáinkra, amit esetlegesen nem vennénk észre, ha utólag visszanézünk egy játékot.

## Játékbeli MI alkalmazása

NPC [5] vagyis non playable character azaz nem játszható karakter. Minden játékban jelen van. Előre meghatározott döntéssorozatok halmazaiból cselekszik. Főleg régebbi játékokban megfigyelhető az ismétlődés, hogy ezek a karakterek például mindig ugyan azon az útvonalon járnak fel-alá, ezáltal könnyen kiismerhetővé válik az adott játék, pedig az NPC -k azért kerültek a játékokba, hogy azt változatosabbá tegyék és kihívást vagy további élményeket adjanak. Ezek továbbfejlesztéséhez remekül alkalmazható az MI. Könnyedén adaptálódnak a játékosok képességéhez, ezzel elkerülve az irreálisan nehéz vagy éppen túl könnyű ellenfeleket, így élvezetesebb és feszesebb lesz a játékmenet.

Ezen alkalmazások és területek jól bemutatják, hogy mennyire sokoldalú felhasználása lehet az esport területén a mesterséges intelligenciának, valamint, hogy mennyire jelentősen hozzájárulhatnak a játékosok teljesítményének optimalizálásához.

# Megerősítéses tanulás

A tanulás három alapvető típusa a felügyelt tanulás, nem felügyelt tanulás, illetve a megerősítéses tanulás. [6] Én az utóbbival, a megerősítéses tanulással foglalkoztam a szakdolgozatom elkészülte alatt.

## Megerősítéses tanulás bemutatása

A mesterséges intelligencia területén a gépi tanulás másik két módjához viszonyítva egy kevésbé közismert ágazat a megerősítéses tanulás azaz reinforcement learning. A felügyelt tanulással szemben, ami előre rögzített adatkészletből dolgozik és határozza meg a kimenetet, a megerősítéses tanulás nem rendelkezik ilyen adatkészlettel, hanem saját magának fedezi fel ismétlődő próbálgatások sorozatán keresztül, hogy milyen cselekmény minősül helyesnek és mi helytelennek egy adott környezetben. A cselekmények sorozata alatt megfigyeli a korábbi kimeneteleket, ezek alapján döntve el, hogy milyen irányba próbálkozzon tovább. Egyszerűbben fogalmazva, a kimenet függ a jelenlegi állapottól, az jelenlegi cselekvéstől, a következő cselekvés pedig függ az előző cselekvés eredményétől. A próbálkozások száma akár milliós számra is rúghat, függően attól, hogy mennyire komplex a környezetünk.

## A megerősítés típusai

Két [7] típust különbözetünk meg a megerősítéses szakaszban. A megerősítés lehet végállapotban, lehet menet közben, vagy mind a két esetben is.

### Pozitív megerősítés

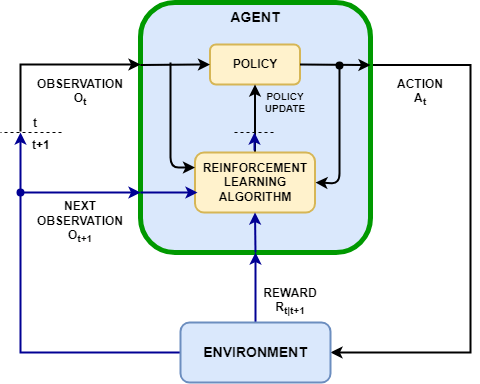
Pozitív megerősítésről akkor beszélünk, amikor egy esemény egy adott viselkedés után növeli a viselkedés gyakoriságát és intenzitását. Tehát pozitív hatással van a viselkedésre. Eredményül segítve a megerősítés során teljesítmény maximalizálását, tartós változást eredményezve hosszú távon. A túlzott erősítés azonban túlterheltséget okozhat, amely ronthatja az eredményeket.

### Negatív megerősítés

Negatív megerősítésről akkor beszélünk, amikor egy negatív hatással járó állapotot megállítunk vagy elkerülünk. Segítségével tudjuk maximalizálni a teljesítményt, ad egy minimum teljesítményszintnek való megfelelést.

## Ágens

A mesterséges intelligencia rendszert tanítjuk, hogy a döntések hatásait megtanulja. Az ilyen AI rendszert ágensnek [8](agent, vagy magyarra fordítva ügynök) nevezzük. Az ágens lép interakcióba a környezettel, célja pedig, hogy maximalizálják a jutalmat. A megerősítéses tanulás során tehát az AI ágens kísérletezik és tanul azáltal, hogy különböző akciókat hajt végre és ezeknek az akcióknak kiértékeli az eredményét.



. ábra: Megerősítéses tanulás - ágens működése

## Algoritmusok kategorizálása

A megerősítéses tanulás területén két fő kategóriára tudjuk bontani az algoritmusokat. [9] Vannak modell alapú algoritmusok és modell nélküli algoritmusok.

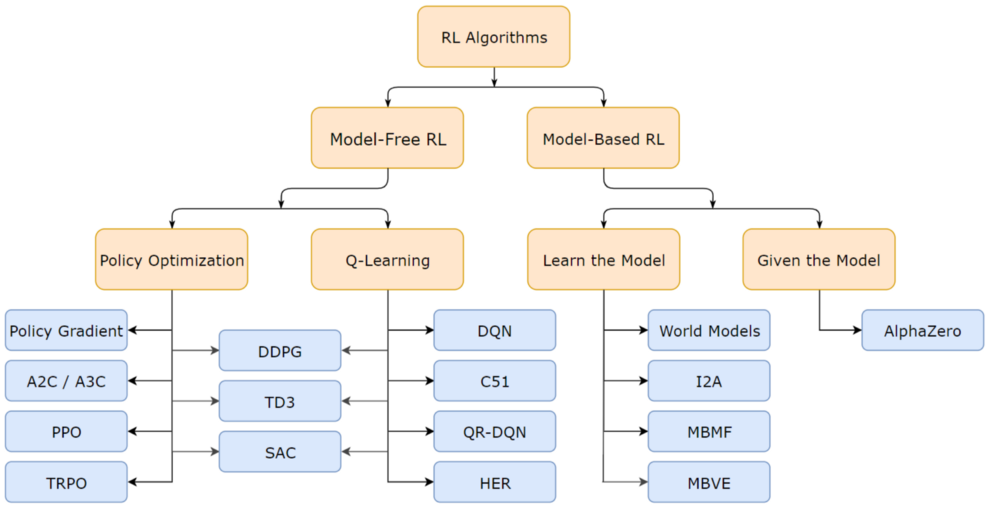
### Modell alapú algoritmusok

A környezet viselkedését egy modell segítségével próbálják megjósolni. Fő céljuk, hogy minden lépésnél maximalizálja a jutalmat.

### Modell nélküli algoritmusok

Ezek az algoritmusok nem használnak előre elkészített modellt a környezet viselkedésének előrejelzésére. Fő céljuk a teljes jutalom maximalizálása.

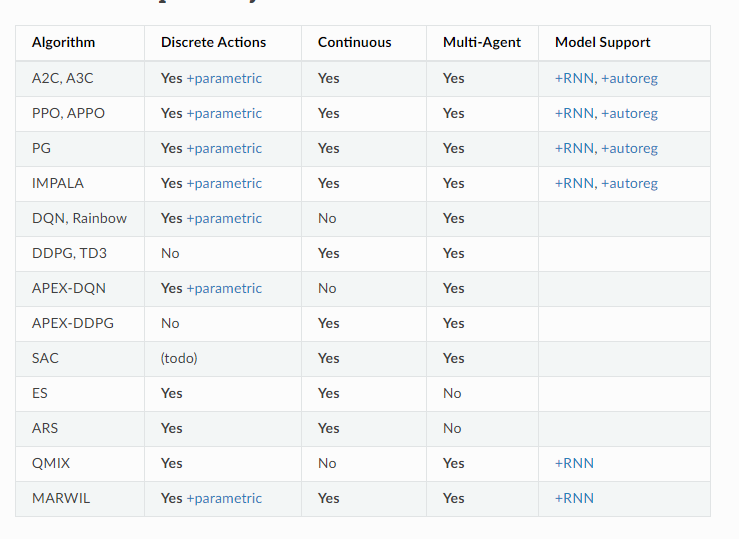
Ez a két összefoglaló csoport többfele ágazik, kisebb csoportokat alkotva melyek esetén vannak átfedések. Néhány főbb algoritmus az OpenAI [10]említésében kategorizálva látható a 2. ábrán.



. ábra: Algoritmusok kategorizálása a megerősítéses tanulásban

## Megfelelő algoritmus választása

A számunkra megfelelő algoritmus kiválasztása elengedhetetlen. Sok fajta algoritmus közül választhatunk, azonban fontos tisztában lenni azzal, hogy melyik, miként tud kommunikálni a környezetével. Éppen ezért megkülönböztetünk diszkrét és folytonos cselekvési tartományt, mely a környezetünk egy lépésének a felvehető értékeit jelenti.  
Autóvezetést szeretnék szimulálni. Elméletben a diszkrét és a folytonos tartomány is megvalósítható, mind a két esetben tudunk adni az autónak utasításokat. A gyakorlatban azonban tudunk gázt adni fékezni és jobbra vagy balra kanyarodni, melyekhez finomhangolt cselekedetekre, mozgásokra van szükség. Éppen ezek miatt a finomhangolt mozdulatok miatt van szükség az én környezetemben a folytonos cselekvési tartományra (continous action space).



. ábra: algoritmusok cselekvési tartományai

Ennek vonzatában, például a DDPG, TD3, PPO, SAC, TRPO, A2C algoritmusok [11]jöhettek szóba, mivel ezek folyamatos cselekvési tartománnyal tudnak dolgozni. Sajnálatos módon multi ágensre nem volt lehetőségem a környezetem adottságai miatt, ezért ezt nem kellett figyelembe vegyem.

# Autóverseny szimulátor

Az autóverseny szimulátorozás a Formula 1 és a rally sportágaknak köszönhetően nagy népszerűségnek örvend. Hétköznapi emberek és profi versenyzők is egyaránt használják ezeket a szoftvereket edzés, vagy kikapcsolódás gyanánt. Különböző autóversenyzés stílusokat utánoznak, például forma autózás, rally, drift. Nem összekeverendőek [11] az árkád versenyzős játékokkal, mint a Need For Speed vagy Forza Horizon videójáték sorozat. Ezek a szoftverek valósághű fizikai modellezéssel készültek és sok aspektusban hasonlítanak a valóságos versenyzéshez.

## Pályák és a környezet:

A legtöbb ilyen szimulátorban megtalálhatóak pontos reprodukciója a valós pályáknak pl Nürburgring, Spa vagy Mondello Park. Személyre szabható vagy dinamikusan változó beállításokkal rendelkeznek, melyek a valós időjárási körülményeket, valós környezetet próbálják meg reprodukálni. Például ha esik az eső akkor könnyebben megcsúszik az autó, vagy ha fűre megyünk szintén nagy eséllyel elveszítjük az uralmat a járművünk fölött. A pályák a mai technológiáknak -mint a 3D modellezés vagy photogrammetry, [12] azaz 3D leképezés- köszönhetően egyes játékokba utólag is integrálhatóak. Tehát ha kedvet kapunk a blenderhez és van egy okostelefonunk mondjuk lidar szenzorral, vagy megbarátkozunk a Google Street View alkalmazásával, akkor akár a saját városunk utcáit is szelhetjük a szimulátoron belül. [Assetto Corsa Photogrammetry](https://www.youtube.com/watch?v=f4zxp2IdXQo&t=2s&ab_channel=ISavic)



. ábra: Photogrammetry - autó 3D leképezése játékokhoz

## Autók:

Az autóverseny szimulátorokban jellemzően licenszelt autókat találunk, ezek széles választékban elérhetőek, de utólagos modifikációknak köszönhetően már saját magunk is tudunk autót készíteni, csak úgy mint pályákat. Az autók kinézete mellett természetesen azok fizikai tulajdonságai is nagyon fontosnak bizonyulnak a szimulátorozás világában. Éppen ezért sok helyen a futóművet tudjuk finomhangolni, a guminyomást állítani, a futófelületet változtatni a tapadás növelése vagy éppen csökkentése érdekében, leszorító erőt módosítani, vagy éppen a váltó áttételt vagy a differenciálmű zárását megváltoztatni.

## Autóvezetési élmény:

Az autóvezetéshez hasonló élmény biztosítása érdekében nem kontrollerrel történik a vezetés, hanem kormánnyal, pedálsorral. Természetesen fontos az eszközök megválasztása is. Egy pár tízezer forintból könnyedén hozzájuthatunk egy kezdő szetthez de nem fog olyan élményt nyújtani, mint ha beruháznánk egy felsőbb kategóriás termékre. A kormányokba direct drive rendszer vagy ennek egy alternatív (olcsóbb) megoldása a force feedback [13] van jelen, mely szimulálja a valósághű kormányzást. Továbbá a pedálban is változtatható a rugóút, mellyel keményíthetjük vagy lágyíthatjuk a pedálokat. Komolyabb szerelésekhez természetesen versenyülést is használnak, ezeknek is vannak már olyan megoldásai, amelyek térben is hidraulikus rendszerrel mozgatják a széket ezzel a G erőket szimulálva.



. ábra: : Profi szimulátor felszerelés

## Bajnokságok:

A legtöbb autóverseny szimulátor támogat multiplayer azaz többjátékos általában online játékmódokat ezáltal lehetővé téve az online bajnokságok, versenyek megszervezését.

Az autóverseny szimulátorok tehát lehetővé teszik az autózás megszállottjainak, hogy saját otthonukban fejlesszék a reakcióidejüket, megtapasztalják a versenyzés élményeit, és gyakorolhassanak anélkül, hogy valós veszélynek lennének kitéve. Fontos megjegyezni, hogy ezek a játékok közel állnak a valósághoz, azonban nem helyettesítik a valós autóvezetés és autóversenyzés tapasztalatát.

## A legismertebb szimulátorok

### iRacing:

A professzionális autóverseny szimulátorozás területén ez az egyik ha nem a legnépszerűbb platform. [14] Valósághű a fizikai modellezés valamint a hangsúly az online versenyzésen van és a valós versenyekhez hasonlóan itt is ranglisták, bajnokságok valamint hivatalos pályák vannak.

### Assetto Corsa:

Egy másik nagyon népszerű verseny szimulátor. [15] Szintén híres a fizikájáról, valamint a pályák, autók sokaságáról és arról, hogy mennyire testre szabható különböző modifikációkkal, melyeket a játékos bázis alakított ki.

### Dirt Rally:

Ha a rally versenyzés érdekli az embert, akkor ez egy remek választás. Rally versenyekre specializálódott a játék, mint a nevéből is kivehető. [16] A valós versenyeket próbálja utánozni, bár ennél a játéknál kicsit messzebb áll a fizika a valóságtól.

### F1 széria:

A Formula 1 megszállotjainak remek választás, ugyanis a lienszelt járművek csak itt találhatóak meg. Évente jön új széria, egész jó fizikával rendelkeznek, sok profi versenyző használja. Meg szeretném említeni Bereznay Dániel nevét aki F1 Esports Series versenysorozatban első magyarként jutott el az elődöntőkig, és a Mercedes AMG Petronas Motorsport online esport pilótájaként versenyez azóta.

Én személy szerint az Assetto Corsa-t ismerem a legjobban valamint a játékhoz tartozik egy Content Manager nevű indító alkalmazás, mely lehetővé teszi egyedi pályák utólagos hozzáadását a játékhoz, melyre szükségem volt, a környezetem egyszerűsítése érdekében, így a szakdolgozatomhoz ezt a játékot választottam.



. ábra: Assetto Corsa autóverseny szimulátor logója

# Fejlesztési környezet

## Hardver

Az asztali számítógépemet valamint a notebookomat is felhasználtam mind a kódok írása során, mind a tesztelés, tanítás során. Ennek az oka, hogy míg az egyik gépen próbáltam az ágenst futtatni, akár órákon vagy napokon át, hogy milyen eredményre jut, addig próbáltam a másik gépen előrébb lendíteni valamit a kódon. Futási teljesítményben viszont volt már a két gép között is különbség, viszont érezhető volt, hogy kevés a gépek hardveres teljesítménye.

### Asztali számítógép:

Processzor: Intel Core I7-8700 CPU @3.20 GHz

CPU magok száma: 6

CPU szálak száma: 12

RAM: 32GB GSkill Trident Z @3000mHz DDR4

GPU: Aorus Master RTX 3070 8GB

### Notebook:

Notebook: MSI GF63 Thin 9SC

Processzor: Intel Core I7-9750H CPU @2.60 GHz

CPU magok száma: 6

CPU szálak száma: 12

RAM: Samsung 16GB @2400mHz DDR4

GPU: NVIDIA GeForce 1650 4GB

## Operációs rendszer

Mind a két eszközön Windows 10 operációs rendszer van telepítve, ezen fejlesztettem és futtattam a programot is. Az operációs rendszerről további információk a hivatalos Microsoft weboldalon olvashatók: <https://www.microsoft.com/>

## Alkalmazott Technológiák

### Python

Évek óta az egyik legnépszerűbb programozási nyelv a Python. Munkaerő piacon ez az egyik legkeresettebb nyelv, emellett népszerű a kezdő programozók körében a könnyen érthető szintaxisának köszönhetően. [17] Guido van Rossum, holland származású programozó a nyelv megalkotója, akinek nem titkolt szándéka volt a nyelvet úgy formázni, mintha egy angol szöveget olvasna az ember. 1991-ben jelent meg az első hivatalos Python 0.9.0 verzió. A Python jelenleg két fő verzióval rendelkezik, a Python 2 és a Python 3. A programozási nyelv mögött egy hatalmas aktív és segítőkész közösség áll. Mivel a Python 2 támogatása 2020-ban lejárt, a közösség ösztönzi a fejlesztőket a Python 3 használatára. A Pythonnal először az egyetemen találkoztam a Mesterséges Intelligencia tárgynál, viszont csak kódelemzés szinten. Kód írására nem került sor, így otthon, a szakdolgozatom készítése alatt fejlesztettem a tudásom a programozási nyelvvel kapcsolatban is. Szerencsére objektum orientált programozási nyelvvel már találkoztam előzőleg, így nem okozott gondot megérteni, hogy hogyan működik.

### Anaconda

Az Anaconda [18] egy nyílt forráskódú platform, mely sok másik nyelv között tartalmazza a Pythont is. Továbbá többek között tartalmazza a fejlesztéshez használt interaktív shell környezetet, a Jupyter Notebookot. Valamint tartalmaz rengeteg fejlesztéshez szükséges Python könyvtárat, például NumPy, pandas, matplotlib. A virtuális környezetek egyszerű kezelését a Conda nevezetű csomagkezelőjével teszi lehetővé. Ennek segítségével hozhatunk létre, aktiválhatunk, inaktiválhatunk, telepíthetünk vagy frissíthetünk könyvtárakat és csomagokat.

### Jupyter Notebook

A Jupyter Notebook eredetileg az IPython keretein belül indult mint egy interaktív shell környezet a Python számításokhoz. [19]2014-ben vált különálló projekté, amikor elérhetővé vált több programozási nyelv is például a Julia és R. A Jupyter ezen programozási nyelvek nevének kezdőbetűiből alakult ki.

Azért választottam ezt a fejlesztői környezetet, mert a Mesterséges Intelligencia kurzuson is ezt használtuk, ebben néztünk meg kódokat és megtetszett a cellás kialakítás, valamint az egyszerűsége. Utólag amióta már dolgozom is, mint frontend fejlesztő, úgy vélem nem ez a legjobb választás, valamint több nehézségem is adódott a környezettel a kód futtatása során, valamint kompatibilitási gondjaim is felmerültek, akadtak érdekes anomáliák is, például az egyik gépemen működött a kód a másikon viszont hibára futott. Szerencsére ezeket a problémákat tudtam végül orvosolni.

### Virtuális környezet

Virtuális környezetek olyan eszközök a Python programozási nyelvben, amelyek lehetővé teszik a projektjeinkhez elkülönített és független fejlesztési környezetek létrehozását, amikor több Python projektet vagy alkalmazást fejlesztünk. Egy-egy ilyen elkülönített környezetben a projektekhez specifikus függőségeket telepíthetünk. Előfordulhat, hogy különböző projektek más verziószámmal rendelkező könyvtárat igényelnek és ezt a virtuális környezeteknek köszönhetően megengedhetjük magunknak, nem kell minden egyes projekt fejlesztése előtt újra telepíteni a környezetet.

### VJOY

A VJOY egy olyan szoftver, melynek segítségével virtuális kontrollereket hozhatunk létre. Segítségével interaktálhatunk olyan alkalmazásokkal, melyek támogatják a joystick használatát. A szoftver képes gombokat valamint tengelyeket is szimulálni.  
A viruális joystickot a VJOY segítségével valósítottam meg, mely segítségével interaktáltam a játék szoftverrel.

### Pandas

A Pandas segítségével exportáltam és dolgoztam fel a versenypálya koordinátáit. A Pandas egy Python könyvtár adatfeldolgozáshoz, adatmanipulációhoz. Főként táblázatos adatok analizálásához tervezték. Lehetővé teszi adatok beolvasását, szerkesztését, törlését, elemzését. Két alapvető adatszerkezetete van, a Series és a Dataframe. [20]

#### Series

Egy egydimenziós adatszerkezet, gyakorlatilag egy listát vagy tömböt jelent.

#### DataFrame

Egy kétdimenziós táblázat, amely sorokat és oszlopokat tartalmaz.

### Pygame

A Pygame egy nyílt forráskódú Python könyvtár, melynek használatával lehetőség van grafikus alkalmások, multimédiás projektek fejelsztésére. Fontos kiemelni, hogy platform független, tehát futtatható Windows, Linux és macOS rendszereken egyaránt. Az én esetemben az autó irányításában nyújtott szerepet, amikor manuálisan akartam irányítani. Ennek segítségével kérdeztem le a Logitech G29 kormánybázisom input adatait.

### Numpy

A Numpy [21] [22] egy nyílt forráskódú Python kiegészítő csomag számítások elvégzéséhez, numerikus műveletekhez. Lehetővé teszi a véletlenszám-generálást, többdimenziós tömbök és mátrixok használatát a magas szintű matematikai függvénykönyvtárával.

### DirectInput

A DirectInput a [23] Microsoft által fejlesztett alkalmazás programozási interfész, röviden API. Lehetővé teszi hogy közvetlenül kommunikáljunk input eszközökkel, mint például a billentyűzet. A DirectInput már elavult eszköz, azonban a PyInput könyvtára nem volt használható számomra. A játékon belüli visszaállítás a kezdőpontra történik meg a DirectInput segítségével, mely gomblenyomás eseményt szimulál a billentyűzeten, melynek segítségével az autót visszaállítjuk az epizód végén a kezdeti pozícióba.

### Stable Baselines

A Stable Baselines [24]egy Python könyvtár, mely az OpenAI Baselines-ra azaz az OpenAI alapértelmezett modelljeire épül és különböző megerősítéses tanulási algoritmusokat kínál a tanításhoz. Ezek olyan algoritmusokat foglalnak magukba, mint például a PPO, DDPG, A2C, amelyek népszerűek a megerősítéses tanulási feladatok megoldásánál. A Stable Baselines használja a PyTorchot.

### MatPlotLib

A MatPlotLib egy Python könyvtár, mely lehetővé teszi 2D grafikák létrehozását. Segítségével diagramokat, ábrákat, grafikákat készíthetünk adatok reprezentálására.

### TensorBoard

A TensorBoard a [25] TensorFlow keretrendszer egy eszköze, modell adatok eredményeinek vizualizálására hozták létre. Segítségével megvizsgálhatóak a tanulási görbék.

# Környezet felépítése

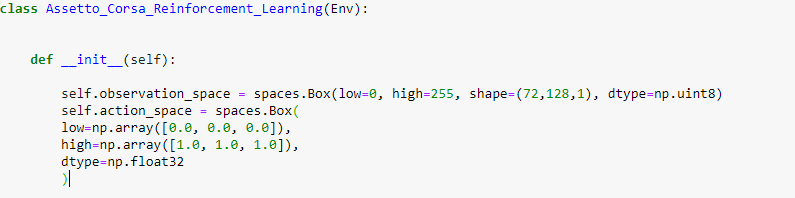
A környezet célja, hogy lehetővé tegye egy megerősítéses tanulásos ágens tanítását és annak értékelését. Az ágens feladata, hogy egy virtuális autóval kell megtanulni egy pályán körbe menni az Assetto Corsa verseny szimulátor játékban. További célkitűzéseim között szerepel, hogy ha megfelelőnek látom a feltanított ágens teljesítményét, akkor megtanítsam az autó megcsúszásának korrigálására valamint szabályzására, tehát a driftelésre.

## Környezet inicializálása

Először is szükségem volt a környezetem kezdő állapotának a beállítására.

## Megfigyelés és cselekvési tartomány meghatározása

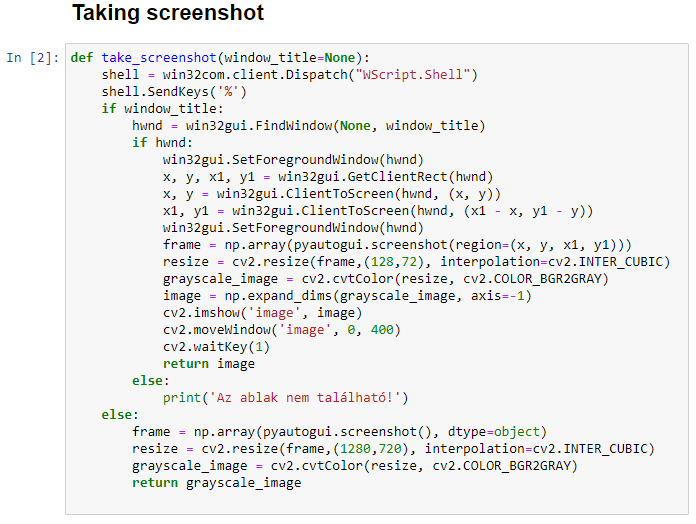
A megerősítéses tanuláshoz szükség van egy megfigyelésre (observation) és az cselekvési tartományra (action space). A megfigyelések jelen esetben a képernyő képek lettek, mivel a játékban nincs lehetőségem egyéb kamerákat felvenni, vagy szenzort használni. Eleinte egészen nagy felbontással kezdtem el dolgozni (1280 x 720), de a teljesítményt ez nagyban lassította. Jelenleg 128x72 pixelt rögzítek minden lépés folyamán. Az inicializáláshoz szükségem volt egy 0-255 közötti egész számokból álló 3 dimenziós tartományra. Mivel szürkeskálásra transzormált képet használok, így a 3. dimenzió csak 1 hosszúságú. A cselekvési tartományom 3 változóból állt, melyek a kormányzást, fékezést, gázadást reprezentálják. Ezek az értékek 0-1 között lebegőpontos számokként lehetnek jelen.



. ábra: Inicializálás - observation, action spaces

## Képernyőképek készítése

A képernyőképek rögzítésére a pyautogui könyvtárát használom, pontosabban a screenshot metódusát. Sajnos ez a megközelítés eléggé leterheli a hardvert, mivel minden lépésben lefut, emiatt célszerű lenne, ha rendelkezésre állna szenzoros adat, de sajnos a játék ezt nem teszi lehetővé.



. ábra:Képernyőkép készítése

## Interfész

Az interfész segítségével tud a környezet kommunikálni a játék szoftverrel. Mivel a valós vezetéshez hasonlóan folytonos tartománnyal szerettem volna dolgozni, ehhez nem volt elég a hagyományos W, A, S, D irányítás.

A VJOY szoftverét használtam fel, mellyel egy virtuális kontrollert tudtam létrehozni és annak kiadni a megfelelő utasításokat. A VJOY-t kormányként állítottam be a játékon belül az irányítópultban. Azért esett a választásom erre a szoftverre, mert open source, azaz nyílt forráskódú és szabadon felhasználható. A VJOY felhasználásával a későbbiekben a saját szimulátor felszerelésemet is sikerült rákötnöm a programra bár ez kompromisszumokkal működik jelenleg, éppen ezért található egy változó is az inicializálásban, amelynél meg lehet adni, hogy gépi vagy emberi vezérlésre kerül majd sor.

## Telemetria adatok kinyerése

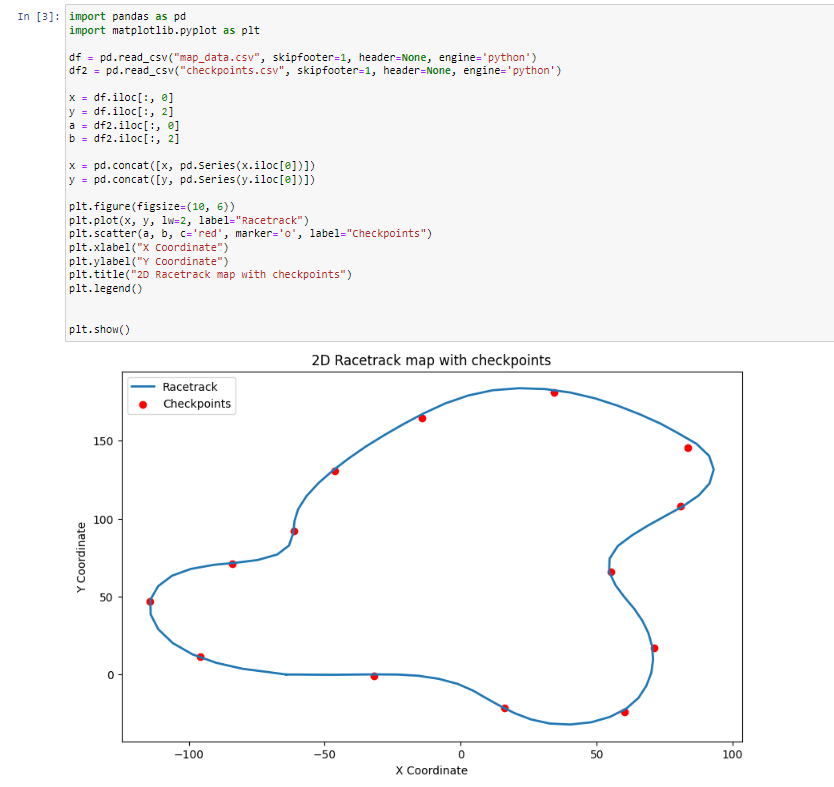
A telemetria adatok kinyerésére az Assetto Corsa Telemetry Interface-t használom, melyet az Assetto Corsa közösség tagja KLGreene nevű felhasználó (Latch Dimitrov) készített a közösség számára. A telemetria adatok segítségével tudom begyűjteni például a következő adatokat:

* Az autó pillanatnyi sebessége
* A pályát elhagyó kerekek száma
* Az autó pillanatnyi sebességfokozata
* Az autó pillanatnyi fordulatszáma
* Az autó pillanatnyi kormányzási szöge
* Az autó által megtett távolság (a játék indításától kezdve)
* Az autó által teljesített körök száma
* Az autó pillanatnyi gázállása
* Az autó pillanatnyi fékereje
* Az autó X,Y,Z koordinátája
* Az autó maximális fordulatszáma

Sok további telemetria adatra is szert lehet tenni, melyekkel az ágensemet a későbbiekben esetlegesen tovább tudnám fejleszteni, például megtanítani a megcsúszás korrigálására, azonban egyelőre az elsődleges cél az volt, hogy találjon körbe a pályán.

## A pálya meghatározása

A pálya meghatározására jelenleg előre rögzített adatokat importálok be egy CSV fájlból a pandas könyvtár segítségével. A pálya adatok rögzítésére az említett telemetria adatok kinyerésére szolgáló programot használom. Ennek segítségével a pályán manuálisan lassan körbe vezetek és másodpercenként rögzítem az autó koordinátáit, melyeket egyből egy .csv állományba töltök bele. Ezt az állományt dolgozom fel a calculate\_center\_line függvény segítségével, hogy meghatározzam a pálya középvonalát interpolációval, amihez a lépések során majd viszonyítom az autó pozícióját. A későbbiekben a versenyzéshez ezt a vonalat később fel lehet venni mint ideális ív, vagy akár a drifteléshez meg lehet adni hasonló módon érintési pontokat, zónákat.



. ábra: Pálya kirajzolása

## Interpolálás

Az interpoláció egy matematikai eljárás, amely jelen esetben a felvett pontok közötti ismeretlen értékeket becsüli meg. Az interpolate\_points\_between függvény segítségével az interpoláció az adott két pont közötti további pontok kiszámolását jelenti, hogy egy görbét hozzunk létre, amely a pálya vonalát jelenti számunkra, melyhez viszonyítani tudjuk az autó pozícióját.

## Az autó poziciója

A kezdőállapot jelen esetben a játékból adódóan a kiinduló pozíciója az autónak, mely egy statikus helyet jelent a versenypálya egy pontján. Az autó kezdeti sebessége tehát nulla, az időjárási változók jó esetben optimálisak, és jó esetben a pálya haladási irányába néz. A jó esetet úgy értem, hogy vannak pályák, ahol parkolóból vagy boxutcából kell indulni az autónak, vagy éppen viharos, esős az időjárás, amely némi nehézséget eredményez a tanulási fázis megkezdésében.

## A környezet egyszerűsitése

A környezet egyszerűsítése érdekében saját pályát készítettem, amely mentesül a különböző pályaelemektől, felfestésektől, akadályoktól, tapadási felület változásoktól. A pályát Blenderben készítettem el, viszonylag egyszerű vonalvezetéssel rendelkezik, nincsenek bene éles visszafordítók, vagy nehéz kanyarok. Kicsit szűkebb lett, mint egy átlagos versenypálya, de könnyedén végig lehet rajta vezetni az autót. Továbbá kikapcsoltam a játékban a gumik kopását, illetve az autó mechanikus rongálódását, valamint bekapcsoltam az Traction Control-t azaz menetstabilizátort gyári beállításra, mely szabályozza a kerekek túlzott kipörgését nagy gázadás esetén, illetve automatikus váltást és kuplungolást állítottam be, annak érdekében, hogy ne kelljen túl sok inputot kezelni a modellnek, ugyanis nagyban nehezítenék a tanulást.

## Cselekvés

A cselekvés minden lépését átadom egy szűrőnek, amely segít gépi irányítás esetén abban, hogy minimalizáljam a kiugrások mértékét. Ezzel előidézve a finomabb gázadást, fékezést illetve a kormánymozdulatokat. Ez a szűrő azért volt szükséges, mert az autó nagyon hamar megpördült és a tanulási folyamatot nehezítette.

Ez a szűrő tovább adja a VJOY eszköznek a kormányzás, gázadás, fékezés értékeit, amely a játékkal interaktál, kiadva az utasításokat az autónak. A lépések során frissítem a telemetria adatokat minden lépésnél. Minden lépésre csinálok egy képernyőképet a játékról, amely a megfigyelést frissíti, melyet az ágens figyelembe vesz a következő lépés meghatározásához. Minden lépésre számítok egy jutalmat, amelyet az ágens szintén figyelembe vesz a következő lépésnél.

## Jutalom számítása

* Meghatározok egy maximális sebességet annak érdekében, hogy optimalizáljam a tanulást.
* Meghatározok egy maximális távolságot a középvonaltól, szintén a tanulási folyamat optimalizálására.
* Amennyiben az autó a maximális távolságon kívül vezet, negatív jutalmat kap.
* Amennyiben az autó elhagyja 2 kerékkel a pályát negatív jutalmat kap.
* Amennyiben 2-nél több kerékkel hagyja el a pályát abban az esetben az epizód véget ér, új epizód indul.

Azon lépések esetén amikor a fenti felsorolás eseményei nem következnek be, abban az esetben pedig pozitív pontszámmal jutalmazom az ágenst. A jutalom mértéke a középponttól való távolságtól illetve az autó pillanatnyi sebességétól függ. Minél gyorsabban megy a pálya középvonala mentén annál nagyobb pontszámot kap a lépésért.

## Visszaállítás

Mivel az ágens ciklikusan fog tanulni, így szükség van az autó visszaállítására a kezdő pozícióba. Amennyiben az epizód véget ér, tehát a „done” változó értéke igaz lesz és a maximális epizód vagy lépésszám még nem következett be, abban az esetben a környezetet visszaállítom kezdeti pozíciójába, és új epizód veszi kezdetét.

# Tanítás

A megerősítés tanítás során az ágens kezdetben véletlenszerűen választja meg az akciókat, így próbál meg mozogni a pályán. A visszajelzésekből tanulja meg, hogy melyik irányba érdemes tovább haladni. A tanulási folyamat a megfigyeléseken és a jutalom mértékén alapul. A tanulás során egy frissíti a stratégiáját (policy) a visszajelzések alapján.

Három algoritmussal próbálkoztam főleg a tanítások kapcsán, melyek a következők voltak:

* A2C
* DDPG
* PPO

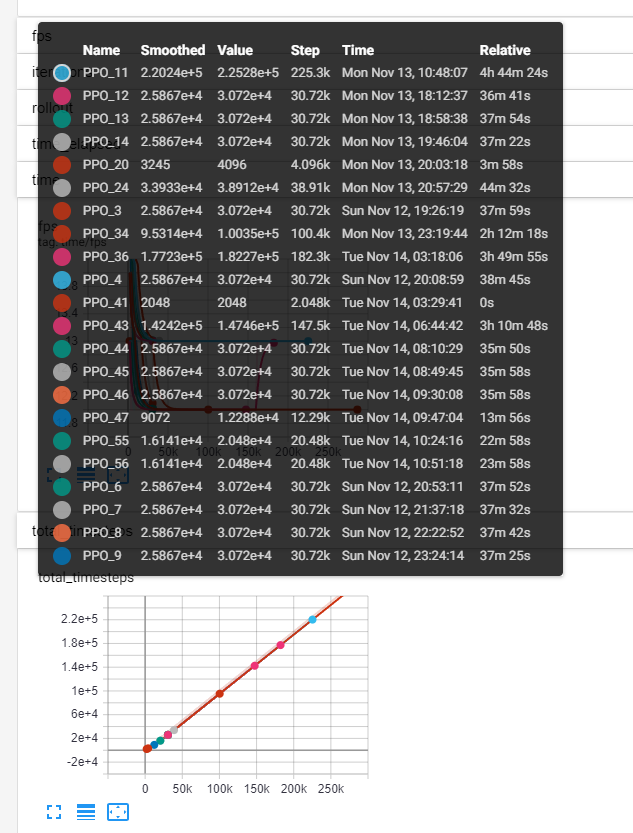
A PPO algoritmus bizonyult a leghatékonyabbnak számomra, mivel ezt használva volt a leggyorsabb az ágens. 13-15 képkocka/másodperc volt az ágens sebessége, amely ugyan ennyi lépésszámot jelentett másodpercenként. A többi algoritmussal ez a szám 3-7 közötti tartományba esett, illetve volt olyan eshetőség, hogy 0 volt. Hardveresen személy szerint úgy gondolom, hogy legalább az asztali számítógépemnek jobban kellene teljesíteni ennél. Gyanítom, hogy a kódban is van még potenciál, tervezem is a sebesség optimalizálását, de jelenleg nem ez volt az elsődleges, a 13 képkocka/másodperc elegendő volt ahhoz is, hogy én is körbe tudjak kényelmesen vezetni a pályán.

A tanítást különböző lépésszámokban próbáltam megvalósítani. Próbálkoztam egészen kevés 2000 környéki lépésszámmal is, amely nagyjából egy epizódnak felel meg, mely időben nagyjából 5 percet vesz igénybe, valamint próbálkoztam milliós nagyságrendű számokkal is, amik olykor napokon át futottak az eszközökön. Végül 30000 lépést határoztam meg, amely optimálisnak bizonyult. Egy ilyen ciklus kb 40 percet vesz igénybe. Ennek valamint a kód csiszolásának következtében a model stratégiája látványos fejlődésnek indult, az autó minden tanítási ciklus után egyre messzebb jutott el a pályán.

Bevezettem a logolást a Tensorboard segítségével. A log adatok külön fájlba mentődnek minden iteráció során. A tanítást eredetileg automatizált ciklusokban végeztem a logolás bevezetése előtt. A naplózás viszont okozott egy kisebb problémát melynek köszönhetően az első ciklus végeztével a kernel „megfagy”. Az idézőjelekre azért van szükség mert nem fagy meg teljesen, csak a virtuális kontrollerrel nem interaktál tovább, így a játéknak nem ad vezérlés. Kis probléma lehet, viszont annál kellemetlenebb. Mivel fejlesztőként dolgozom, a munkámból kifolyólag arra tudok következtetni, hogy egy feliratkozás nem zárul le, amely esetlegesen az irányításért felel. Ahhoz, hogy eredményeket tudjak felmutatni első körön manuálisan folytattam a tanításokat, minden ciklust manuálisan indítva.

A tanítások és tesztelések során több zsákutcába is belefutottam, melyek esetében minden alkalommal változtatni kellett a környezeten, vagy a jutalmazáson. Ezek a változtatások a modell teljes újra tanítását vonzották magukkal, mely ilyenkor órákba teltek. Próbálkozások során sikerült a környezetembe integrálni egy sávdetektáló rendszert, mely a fehér folytonos vonalakat felismeri, és egyenest illesztve ezekre a kontúr vonalakra kirajzoltam, valamint egy darabig próbáltam a megfigyelésbe integrálni ezt a funkciót. A környezetben itt még nem volt ismert a pálya középvonala, amelyet helyettesíteni próbáltam a két illesztett egyenes átlagolásával. Mivel kamera képről van szó a megfigyelésekben és ez a sávdetektálás is ebből a kameraképből dolgozta fel az adatokat, amint az autó és vele együtt a kamera elfordult, melynek következtében az egyik sáv nagy része nem látszódott a képen, a vonalillesztés megbukott és használhatatlan volt.

# Modell kiértékelése

A jelenlegi modell összegezve ~20 óra tanításon van túl. A tanítás eredményesnek tekinthető. A legjobb teljesítménnyel rendelkező modell a következő linken megtekinthető:

. ábra: Tanítási idő

A kezdeti negatív pontszámokból kiindult modell eltudott jutni 7-800 pontos átlag teljesítményig, ami közel 4-500 méternyi megtett távolságot jelent a pályán, ezzel teljesítve több kanyart is mielőtt elhagyta volna a pálya területét. Gyorsnak és kifinomultnak egyáltalán nem tekinteném még ezt az állapotot, de ígéretesnek tűnik a modell és látok potenciált a későbbi fejlesztésekben.

# Összefoglalás

A mesterséges intelligencia vagy röviden AI/MI kifejezést nap mint nap hallom, olvasok róla, szembe jön velem az internetes felületeken. Az elmúlt években hatalmas szerepet kapott és elkezdett a hétköznapi élet egyes területeire betörni. A munkám során betekintést nyerhettem egy olyan világba, mely szintén érinteni fogja a jövőben a mindennapjainkat véleményem szerint.

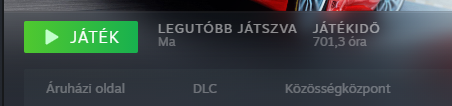
Azt gondolom, hogy egy igen nehéz és összetett dolgot szerettem volna megvalósítani, melyben rengeteg potenciál rejlik, viszont nem gondoltam volna az elején, hogy ennyire összetett és nehéz lesz egy valamennyire is működő modellt megalkotni. A végcélomtól sajnos nagyon messze vagyok., bár a jelenlegi modell az eddigi próbálkozásaimhoz képest ígéretes és végre azt látom, hogy fejlődik és halad körbe a pályán. Úgy gondolom, hogy megfelelő mennyiségű tanítással el lehetne érni egy olyan szintre, ahol tényleg gyorsabban is körbe tud érni a pályán, mint én vagy akár szimulátor versenyző. A megcsúszás korrigálása azonban egy összetettebb és komplexebb feladat, amelyhez véleményem szerint egyéb szenzoros adatra is szükség lenne, mint például oldalgyorsulás, illetve a tapadásvesztés mértéke.   
 A jövőben szeretnék még foglalkozni ezzel a programmal, szeretnék megoldást találni arra, hogy hogyan lehet ténylegesen megvalósítani a driftelést vagy a megcsúszások korrigálását egy kamerakép illetve a pálya ismeretével, valamint néhány szenzoros adattal, amelyet az autó vagy egy mérőműszer esetlegesen biztosíthat a valós körülmények között is, ugyanis ez azt jelentené, hogy átültethető és megoldható egy autó vezetése is ilyen módon.

Az én véleményem, hogy a mesterséges intelligenciának ez a része még gyerekcipőben jár, és a következő években hatalmas előre lépés fog bekövetkezni, csak úgy, mint az ágazat többi területén. Az utóbbi napokban találkoztam a Thyssenkrupp csoporttal, akik Magyarországon önvezető autókkal, kifejezetten a megcsúszások, határesetek korrigálásával leküzdésével foglalkoznak. Továbbá a Toyota is kifejlesztett az elmúlt évben egy technológiát ahol megvalósult a driftelés sofőr nélkül.

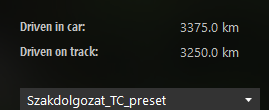
A program elkészítése folyamán hasznosítottam az egyetemen valamint a munkahelyemen megszerzett tudásokat, tapasztalatokat, valamint új technológiákat ismertem meg. A program fejlesztése, a modell tanítása rettentő izgalmas ugyanakkor rejtelmes és ennek okán stresszes volt. Sokszor előfordult, hogy a tanítás zsákutcába futott, melynek a miértjét csak sok napnyi fejtörés árán tudtam megfejteni. Természetesen ez a programozói léttel együtt jár. A hibakeresés része, a fejlesztésnek.

Készitettem egy rövid montázs videót is arról, hogy hogyan is néz ki egy ilyen szimulátoros kör, ahol saját magam driftelek, valamint hozzáfűztem, hogy a modell hogyan indult ki illetve, hogy milyen állapotra sikerült feltanítanom. A modell tanítása és tesztelése sajnos alacsony felbontáson kellett, hogy történjen, ezért a felvételek minősége miatt elnézést kérek.

Lezárásként érdekességként megemlíteném, hogy a Steam felhasználómon kb 200 órám volt összesen a játékban, azóta az összegyűlt órák 90%-ban a szakdolgozatom készítése során gyarapodtak. Valamint az Nissan 350Z amit kizárólagosan a tanítások során használtam látható, hogy mekkora távolságot tett meg összességében, míg idáig eljutott a program.



. ábra: Játékidő (óra)



. ábra: Az autó által megtett távolság (km)

# Köszönetnyilvánítás

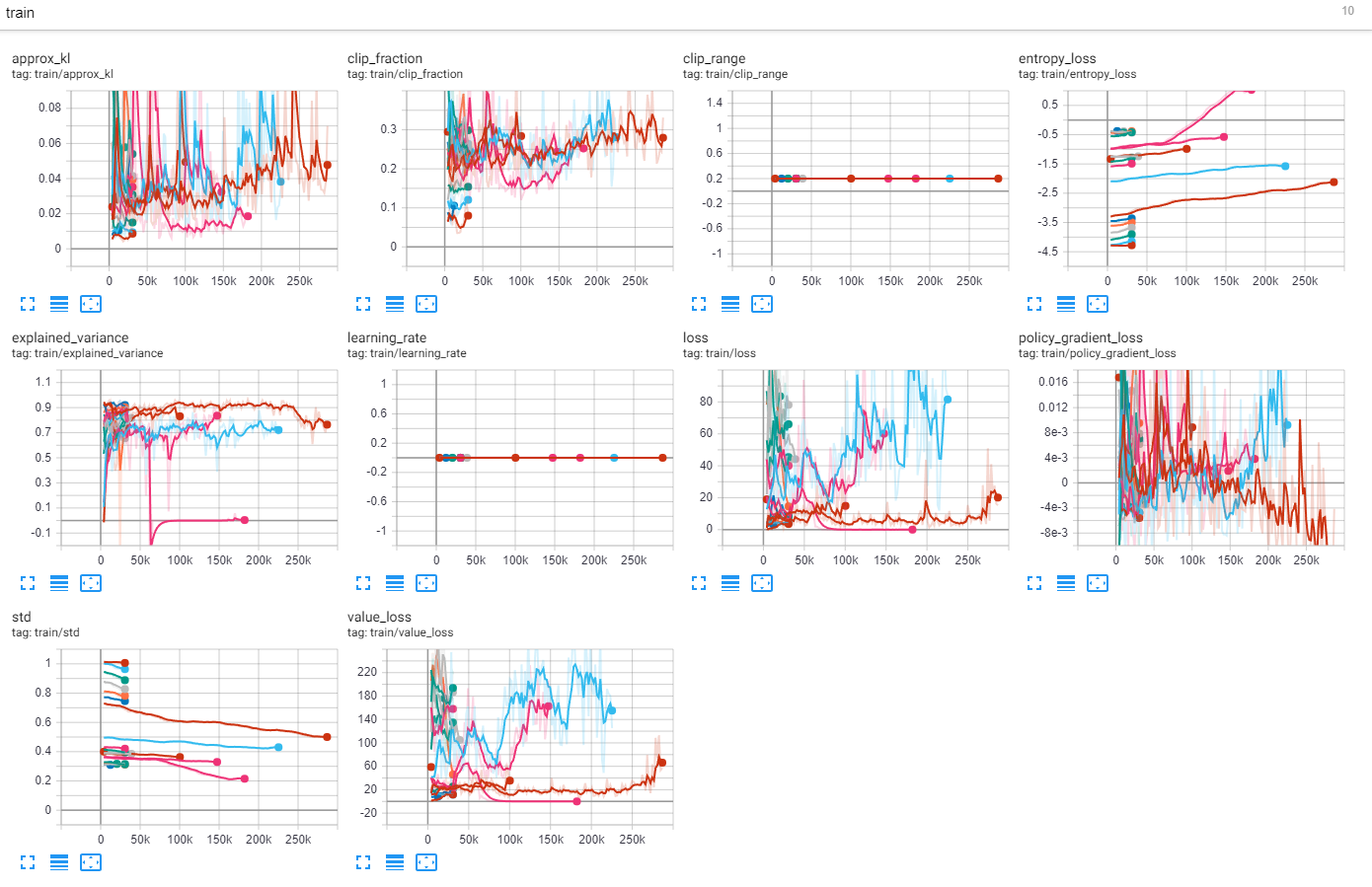
Szeretném megköszönni Lakatos Róbert témavezetőmnek, hogy felkeltette az érdeklődésemet a mesterséges intelligenciával kapcsolatban! Köszönöm a segítségét, a tanácsait, a kérdéseimre adott válaszait.

Köszönöm a családomnak, kedvesemnek és a barátaimnak a biztatást, hogy mellettem álltak, támogattak, segítettek az egyetemi éveim alatt.

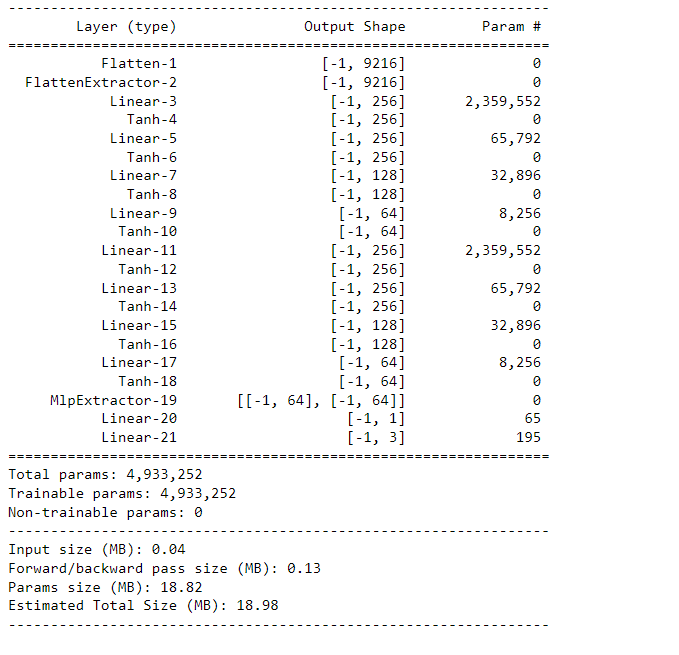
# Hivatkozások

|  |  |
| --- | --- |
| [1] | Y. S. E. K. D. S. P. D. Florian Fuchs, "Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning," 2021. [Online]. Available: https://ai.sony/publications/Super-Human-Performance-in-Gran-Turismo-Sport-Using-Deep-Reinforcement-Learning/. |
| [2] | „Gran Turismo,” Gran Turismo, [Online]. Available: https://www.gran-turismo.com/hu/academy/graduates/jann/. [Hozzáférés dátuma: 11 2023]. |
| [3] | T. S. Staff, „theshopmag,” December 2021. [Online]. Available: https://theshopmag.com/features/can-a-sim-racer-drive-a-900-horsepower-drift-car/. [Hozzáférés dátuma: 11 2023]. |
| [4] | unknown, „Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Artificial\_intelligence\_in\_video\_games#cite\_note-Yannakakis2012-2. [Hozzáférés dátuma: 11 2023]. |
| [5] | „Lexiq,” 16 10 2022. [Online]. Available: https://lexiq.hu/npc. [Hozzáférés dátuma: 11 2023]. |
| [6] | F. Richárd, „Szegedi Tudományegyetem,” [Online]. Available: https://www.inf.u-szeged.hu/~rfarkas/ML20/alapfogalmak.html. [Hozzáférés dátuma: 11 2023]. |
| [7] | P. Bajaj, „Geeks For Geeks,” 04 2023. [Online]. Available: https://www.geeksforgeeks.org/what-is-reinforcement-learning/. [Hozzáférés dátuma: 11 2023]. |
| [8] | „Mathworks,” The MathWorks, Inc., [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html?fbclid=IwAR2nDt8OKCu3WWNPjT1V1XV95bH2rCeBPhij7AnJ7vNjU\_d\_P7WvcrizOCo. [Hozzáférés dátuma: 11 2023]. |
| [9] | baeldung, „Baeldung,” 03 2023. [Online]. Available: https://www.baeldung.com/cs/ai-model-free-vs-model-based. [Hozzáférés dátuma: 11 2023]. |
| [10] | m. OpenAI, „OpenAI Spinning Up,” OpenAI, 2018. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl\_intro2.html. [Hozzáférés dátuma: 11 2023]. |
| [11] | T. R. Team, „RAY,” The Ray Team, 2023. [Online]. Available: https://docs.ray.io/en/latest/rllib/rllib-algorithms.html. [Hozzáférés dátuma: 11 2023]. |
| [12] | M. Jahn, „Overtake,” Overtake.gg, 03 2020. [Online]. Available: https://www.overtake.gg/tutorials/simracing-vs-arcade-vs-simcade/. [Hozzáférés dátuma: 11 2023]. |
| [13] | P. Hanaphy, „Artec 3D,” Artec 3D, 03 2023. [Online]. Available: https://www.artec3d.com/learning-center/photogrammetry-for-games. [Hozzáférés dátuma: 11 2023]. |
| [14] | „SimXPerience,” SimXPerience, [Online]. Available: https://www.simxperience.com/blog/simxperience-news-1/direct-drive-vs-belt-drive-vs-gear-drive-70. [Hozzáférés dátuma: 11 2023]. |
| [15] | „IRacing,” IRacing, [Online]. Available: https://www.iracing.com/overview/. [Hozzáférés dátuma: 11 2023]. |
| [16] | „Assetto Corsa,” KUNOS Simulazioni Srl , [Online]. Available: https://assettocorsa.gg/. [Hozzáférés dátuma: 11 2023]. |
| [17] | „Dirt Game,” Codemasters, [Online]. Available: https://dirtgame.com/dirtrally/uk/about. [Hozzáférés dátuma: 11 2023]. |
| [18] | „Wikipedia,” [Online]. Available: https://hu.wikipedia.org/wiki/Python\_(programoz%C3%A1si\_nyelv). [Hozzáférés dátuma: 11 2023]. |
| [19] | „Domino,” Domino Data Lab, [Online]. Available: https://domino.ai/data-science-dictionary/anaconda. [Hozzáférés dátuma: 11 2023]. |
| [20] | A. Eberle, „University of Bonn,” [Online]. Available: https://wt.iam.uni-bonn.de/fileadmin/WT/Inhalt/people/Andreas\_Eberle/AlMa17/JupyterIntroNew.html. [Hozzáférés dátuma: 1 2023]. |
| [21] | „Pandas,” [Online]. Available: https://pandas.pydata.org/docs/user\_guide/10min.html. [Hozzáférés dátuma: 11 2023]. |
| [22] | „Wikipedia,” 1 12 2022. [Online]. Available: https://hu.wikipedia.org/wiki/NumPy. [Hozzáférés dátuma: 11 2023]. |
| [23] | „Python for Scientists and Engineers,” 2011. [Online]. Available: https://www.computer.org/csdl/magazine/cs/2011/02/mcs2011020009/13rRUx0xPMx. [Hozzáférés dátuma: 11 2023]. |
| [24] | „Microsoft,” 2011. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ee418273(v=vs.85). [Hozzáférés dátuma: 11 2023]. |
| [25] | „Stable Baselines,” Stable Baselines, [Online]. Available: https://stable-baselines3.readthedocs.io/en/master/. [Hozzáférés dátuma: 11 2023]. |
| [26] | „Tensorflow,” [Online]. Available: https://www.tensorflow.org/tensorboard. [Hozzáférés dátuma: 11 2023]. |

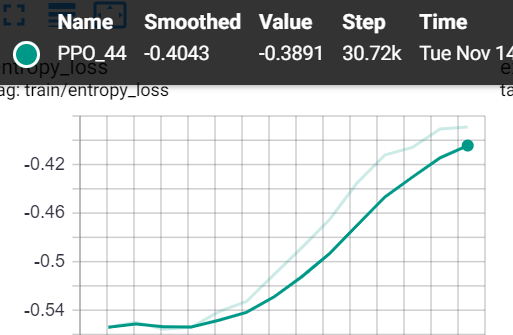
# Függelék



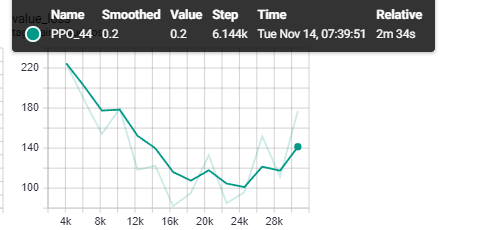
. ábra: Tanulási folyamatok grafikonjai



. ábra: Model paraméterei



. ábra: Model entropy loss



. ábra: Model value loss