

TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

VIRTUÁLIS VALÓSÁG JÁTÉKFEJLESZTÉS ÉS OPTIMALIZÁLÁS KÜLSŐ KÉZMOZGÁS ÉRZÉKELŐ SZENZORRAL

Szerző:

Dobra Erik

mérnökinformatikus BSc. Szak

Konzulens:

Nagyné Dr. Hajnal Éva

egyetemi docens

Tartalom

Absztrakt	1
1 Bevezető	2
1.1 Célkitűzés	2
1.2 Törekvések a virtuális valóságban kézi interakciók megvalósítására	3
2 Játék koncepciója	7
3 Alternatív mozgás meghatározása	11
3.1 Mozgási modell megvalósítása	15
3.2 Alternatív megoldások a kézmozdulatokkal irányított virtuális mozgás kivitelezésére	17
3.2.1 Csuklómozgással történő irányítás	17
3.2.2 Magicsee R1 controller	19
3.2.3 Virtuális VR billentyűlenyomás kézfelismeréssel	19
3.3 Mozgás kézi vezérlés módszerek összehasonlító elemzése	20
4 Felhasznált technológia	23
4.1 Fejlesztőkörnyezet ismertetése	26
5 Játékoptimalizálás, eredmények	28
5.1 fejlesztési lehetőségek	31
6 Összefoglalás	32
7 Summary	33
8 Köszönetnyilvánítás	34
Irodalomjegyzék	35

ABSZTRAKT

A virtuális valóság egy számítógép által szimulált környezet, amellyel az ember kölcsönhatásba léphet.

A VR a valóságot egy elképzelt világgal helyettesíti, amelyet speciális fejre helyezhető eszközön (HMD) keresztül láthatunk. Ebben a környezetben a kézmozgások követése (hand tracking) fontos feladat. Több eszköz nem rendelkezik ilyen funkcióval, vagy ez a funkció nem elegendő pontosságú. Gyakori, hogy a kézmozgás követése egy speciális kézi kontroller követésével van kialakítva, holott sok alkalmazás megkövetelné a saját kezünk használatát.

A dolgozatban a VR szemüveg kiegészítését szeretném bemutatni egy kézfelismerő eszközzel. A feladatban Windows Mixed Reality típusú VR szemüveget bővítettem egy szemüvegre illesztett leap motion szenzorral, amely segítségével a kéz követése kézi kontroller nélkül is megoldható. A megvalósított megoldást egy mintaalkalmazásban teszteltem. A játék témája horror, amely a játékost ösztönözi a gyors reakciókra. Az alkalmazásban a virtuális térben való mozgásnak egy alternatív megvalósítását használtam. A virtuális térbe újratervezett vezérlőegység fejlesztése egy Unity alapú projektben készült. A rendszer lehetővé teszi a játékosoknak, hogy a hüvelykujj és a mutatóujj segítségével speciális gesztusokat hajtsanak végre, kifejezetten a gombok megnyomására. A szoftver értékelését a beépített profilozó rendszerrel végeztem el, ami megmutatja, hogy milyen jövőbeli optimalizációk lehetségesek a projekten belül. Összefoglalva, sikerült egy olyan kézfelismerő rendszert alkotni, amely valós időben leköveti a pontos kézmozgást, ami a rendszerhez gyártott driverhez képest jelentős előrelépés a kézfelismerés implementálásban.

1 BEVEZETŐ

1.1 CÉLKITŰZÉS

Az volt a célom, hogy egy olyan VR játékot készítsek, amit egyszerre élvezetes használni, és egy új lehetőséget nyithat arra, hogy a felhasználók hogyan használhatják a játékon belül az kézi interakciókat és a játékon belüli mozgást. Azért választottam a virtuális valóságot témának, mert ez az ágazat jelenleg egy fordulópont előtt áll. A fejlesztőknek most kell adaptálódniuk a hardverhez, és ugyanúgy a hardverek is még folyamatos fejlődésben vannak. Fejlesztői szemmel tekintettem a problémára, hogy hogyan tudok a speciális hardverek felhasználásával az igényeknek megfelelő szoftvert kreálni, mennyire nehéz, hogyan kivitelezhető. Ahogyan a dolgozatomban több helyen kifejtettem, az alapvető gondolat a VR kapcsán, hogy minél több érzékszervi tapasztalatot átültessünk virtuális térbe, és azt immerzív formában élje át a felhasználó. A végtagjaink bevonása a virtuális térbe egy lépcsőfoka ennek, de nem az utolsó.

Természetesen hozzá kell tennem, hogy a virtuális valóság immerzív megvalósítása nagyon sok helyen megállja a helyét, nem csak a szórakozatóiparban, hanem több, iparág hasznára tud válni a szimuláció terén: veszélyes vagy nehézipari alkalmazása is ismert a mai tanulmányaink alapján. Adott a lehetőség, hogy az én megalkotott rendszerem később használható lesz más iparág számára is.

Hozzátehetek sokat az is, hogy megnőtt a tendencia arra, hogy az emberek egyre többet maradnak zárt térben (a jelenlegi egészségügyi helyzet miatt kifejezetten), és ebből kifolyólag az ember kereste a lehetőséget, hogyan szakadjon ki a mindennapi valóságból, így a virtuális valósághoz való csatlakozás iránt megnőtt a hajlandóság[1]. Így szereztem be én is az első VR eszközeimet, ami aztán előrevitt a témában.

A játékon belül finom kézmozgással kapcsolatos feladatok kerültek megvalósításra, mint például egy konzol irányítása, ami feladatok elé állítja a játékost. Három különböző módszert is megalkottam a játék irányítására. Ezeknek összemérését egy általam lemerített tanulási görbén mutatom be. A játék stabilitásának tesztelésére a fejlesztőkörnyezet által kínált profilozót használtam, amivel képes voltam olyan problémákra fényt deríteni, amikre másképp nem lettem volna képes.

1.2 TÖREKVÉSEK A VIRTUÁLIS VALÓSÁGBAN KÉZI INTERAKCIÓK MEGVALÓSÍTÁSÁRA

Különböző tudományos cikkek alátámasztják azt a törekvést, hogy a VR és kézérzékelő szenzorok együttes alkalmazása a mindennapokban pozitív hatással van a felhasználókra. Olyan cikk is publikálásra került, miszerint sztrókon átesett betegek terápiás kezelésében használták a virtuális valóság és kézérzékelő szenzorokat. A kísérlet során VR Headsetet és Leap Motion szenzort használva szabályos testeket kellett a kísérleti alanyoknak felismerni és megérinteni. A publikáció arról számolt be, hogy a felhasználók élvezték ennek a komplex eszköznek a használatát, viszont a felmérésben résztvevők alacsony létszáma miatt a pontos statisztikai kiértékelés nem volt lehetséges. [2]–[4]

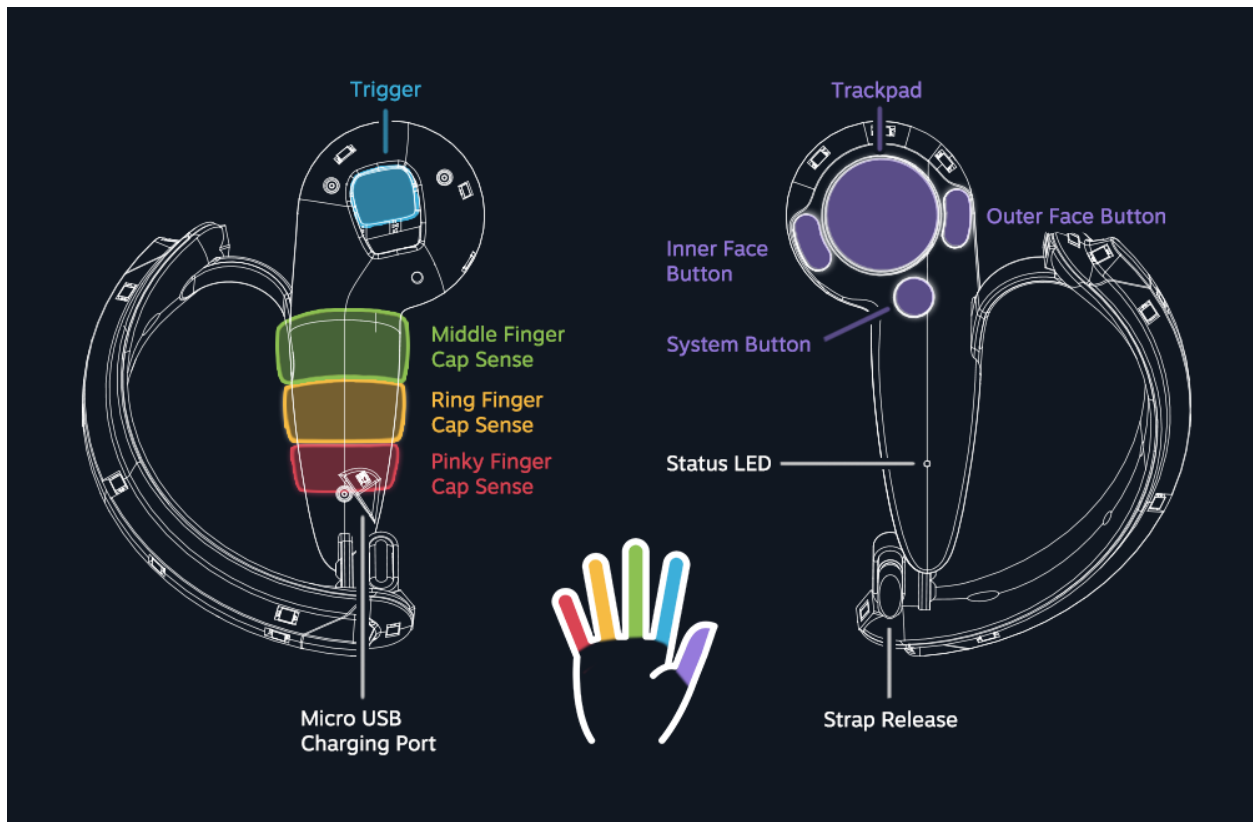
A mozgáskultúra és rehabilitáció mellett az is fontos szempont, hogy a mozgás szabadsága megvalósuljon, viszont ehhez az is kell, hogy kiiktassuk a kábeleket, amik a gép mellett tartanak. Folyamatos törekvés ennek megvalósítása, és még emellett megmaradjon a Leap Motion által kínált kézfelismerés is. Sajnos az Android operációs rendszer nem tudta feldolgozni a Leap motion adatait, ezért kénytelenek voltak közvetíteni az adatokat egy PC vagy egy MAC eszközre – derült ki egy tudományos cikkből, amit a kolozsvári tudományegyetem publikált[5]. Mivel egyre több nagyteljesítményű IoT eszköz készül, ezért hamar találunk erre megoldást. Az első nagyobb fejlesztést ebben az irányban az Oculus Quest 2-nél tapasztaltuk. Ez az eszköz rendelkezik beépített kézfelismeréssel, viszont ez csak a menüben teszi lehetővé az irányítást, más esetben nem használható[6]. Később ezt kibővítették, és elérhetővé vált egy SDK is, amivel alkalmazást lehet fejleszteni rá. Ha ezt az eszközt összemérjük a Leap Motionnel, akkor azt mondhatjuk, hogy a Leap Motion sokkal pontosabb és szofisztikáltabb, könnyebben kezel olyan nehéz helyzeteket, mint például a kezek átfedése. Ez fontos szempont, mert komplexebb mozdulatok esetén hibát véthet az eszköz, ami kritikus felhasználás esetén problémát okozhat. Ilyen kritikus esetek az ipari felhasználásban lehetnek jelen.

A virtuális valóság ipari integrációja még gyerekcipőben jár, de egy helyen biztosan megállja a helyét, a szimulációkban. Pilóták kiképzése során nélkülözhetetlen, hogy minél alaposabb jártasságot elsajátítsanak biztonságos körülmények között, és csak utána szálljanak fel a magasba.

Ám, egy markoló vagy egy daru működtetése is szakértelmet igényel, ezért fontos, hogy a szakképzettség megszerzése biztonságos körülmények között történjen meg. Készült olyan tudományos publikáció, ami a Leap Motion szenzort használva megmutatja és szimulálja a markolók működését a virtuális valóságban. A kísérletben részt vevők fele gondolta úgy, hogy hasznára vált az előzetes szimuláció.[7]

A virtuális valóság legerősebb húzóágazata természetesen a szórakoztatóipar az ipari integráció mellett, ezért fontos, hogy minél inkább immerzívvé tudjuk tenni a virtuális valóságot mindenki számára. Több megvalósítás történt már, amely próbálta újraformálni az emberi interakciót a virtuális valóságban, többek között haptikus szenzorok segítségével. Említést érdemel a Senso Glove eszköz, ami kesztyűként viselve közli a kéz állapotát a gazda számítógéppel, felismerve a gesztusokat. Sajnos több visszajelzés azt a tendenciát mutatta, hogy a tesztalanyok teljes egyetértésben nem preferálják a használatát, hanem inkább a Leap Motion szenzort vagy az eredeti Joystick használatát részesítették előnyben. [8]

A Valve, az egyik legnagyobb játék-kibocsájtó és fejlesztő cég 2019 nyarán piacra dobta saját fejlesztésű VR headset-ét, a Valve Index-et, amelynek a joystickje (1. ábra) ujjfelismerést tartalmaz. Ezt azért fontos megjegyezni, mert még nem volt ebben az időben trend a finom kézmozgások felismerése, viszont ez nem optikai módon működött, hanem nyomásérzékeny szenzorokat alkalmaztak. [9]



1. ábra: Valve Index ujjfelismerésének sematikus rajza

forrás: <https://uploadvr.com/latest-valve-index-controller-firmware-brings-noticeably-improved-finger-tracking/>

Tulajdonképpen amint a fenti grafikonon is látszik, csupán három ujjnak a nyomását volt képes felismerni nyomásérzékeny gumírozott felületen, a hüvelyk-és a mutatóujj csak gombokat kezel.

A valóságban a megvalósítás többnyire megfelelt a közönségnek, viszont többen jelezték vissza a fejlesztőknek, hogy kényelmetlen a joystick használata, mivel kemény pánt fogja át az ember tenyerét és ez egy idő után kellemetlenül nyomja a kéztőcsontokat, ezért a Valve fejlesztői adtak ki egy hivatalos extendert az eszközhöz.[10] Fontos azt is megjegyezni, hogy a Valve Index az egyik, ha nem a legdrágább VR Headset ami kapható a piacon, így nem sok lehetősége van az átlagembernek használni ezt a funkciót.

Az egyetlen játékkonzolhoz megvalósított HMD szemüveg a Sony Playstation VR volt, amivel nem volt lehetőség finomkézmozgás felismerésére, csak a joystick helyzetét volt képes megállapítani az eszköz végén kibocsájtott fény segítségével, amit egy pozícionált kamera észlelt – tehát a kamera nem a fejegységen szerepel, hanem a szobában volt elhelyezve ahol használják. Az egyetlen ok, amiért elterjedt lett, az az ára volt: Az első olyan fogyasztóknak szánt VR eszköz,

ami megközelíthető áron volt a többség számára, így a fejlesztés is elterjedtebbé vált virtuális valóság témában.

Oculus (ma már Meta) cég kínálja a legszélesebb palettát ha HMD eszközöket keresünk. Egyik legnagyobb sikere az első Oculus Rift-el volt, ami nem tartalmazott külön joysticket, hanem a csomagban egy Microsoft Xbox kontrollert társítottak a termékhez.[11]

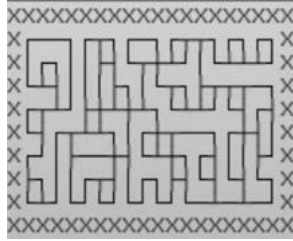
Későbbi nagy befutója az Oculus Quest 2 lett, ami már tartalmaz kézfelismerést, és céljuk hogy finomítsák annyira hogy a finomkézmozgásoknak is megfeleljen.[12]

Az általam használt eszköz, a HP Mixed Reality Headset Developer headset nem rendelkezik finomkézmozgás érzékelővel, csak joystick-el, amit két kamera követ le, ami a fejegységen található.

például: a „—” karakternek a bal szomszédjai lehetnek(3. ábra):

3. ábra: példa a labirintus generálás lehetséges opcióira.

8

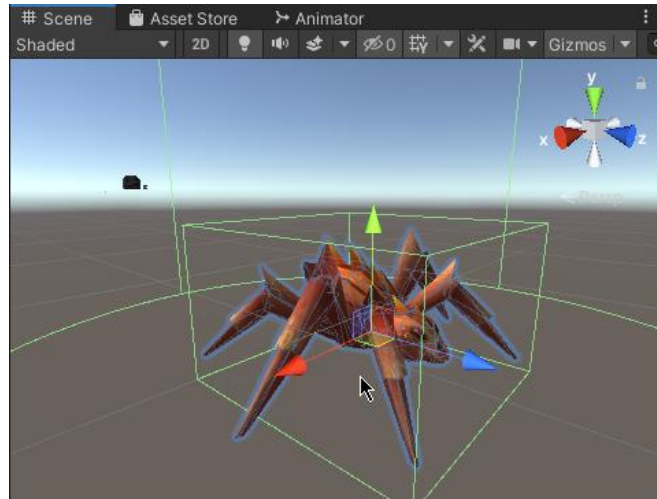


4. ábra: A labirintus generálás előzetes működése

Ez tulajdonképpen az alapja annak, amit felhasználtam, utána még ezt kibővítettem azzal, hogy ha van hely középen akkor tegyen le egy szobát, aminek csak balra és jobbra van kijárata, illetve hogy ha zsákutca van akkor legyen egy vége is, ahol egy teleport kapu várja a felhasználót. Ahhoz, hogy tudja használni a kaput és véghez vigye a játékot, ahhoz meg kell találnia a szobát, benne feladatokkal. Feladatok megoldása után aktiválja a kaput, amivel befejezheti a játékot.

Témájának valamifajta horror tartalmat szántam, hogy ösztönözze a játékost a játék teljesítésére. Következő gondolat, ami ezzel kapcsolatban felmerült, hogy ezzel van egy baj. Mégpedig, ha az ember megismeri a pályát, akkor onnantól kezdve tudja, hogy hogy lehet a lehető legkevesebb idő alatt lejátszani a játékot, aminek következtében elveszik a játék varázsa. Ennek kiküszöbölésére találtam azt ki, hogy a játék egy véletlenszerű pálya generálást valósítana meg, így akárhányszor próbálkozik valaki, mindig új kihívás előtt találja magát. Akkor megpróbáltam lefektetni a játékszabályokat, amelyek a következők.

- Játékos menekül egy ijesztő karakter elől, ha az elkapja, akkor vége a játéknak,
- ellenben a játéknak valamifajta kihívás elé kell állítani a játékost, és a játék irányításának és az interakcióknak valamifajta újra gondolt technikát kell megvalósítania. Az ijesztő karakternek a saját, illetve sok más ember főbiáját tartottam jó ötletnek, ezért az arachnofóbia karakterére, a pókra esett a választásom (5. ábra).



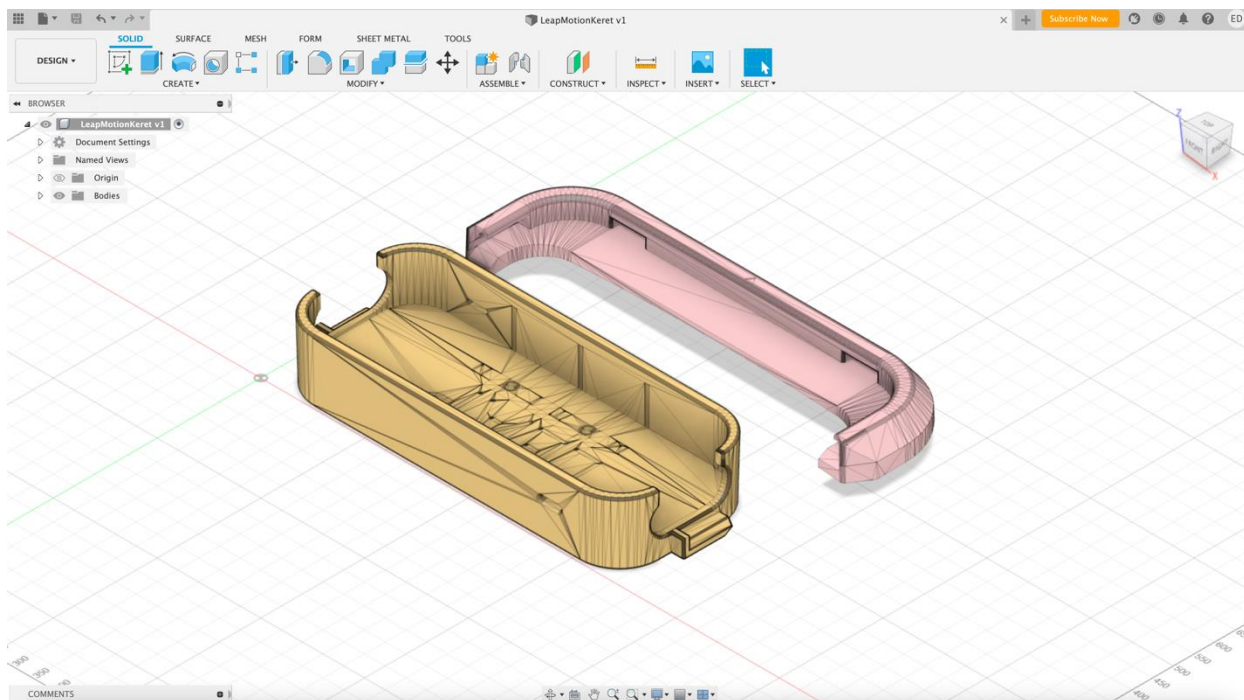
5. ábra: Pók modell.

- Azt is eldöntöttem idő közben, hogy a játékosnak legyen három élete, mert szeretném, ha a játékos tudna a feladatokra is koncentrálni, így hozva egyensúlyba a rettegés faktort és a feladatmegoldást a játékon belül.
- Ahhoz, hogy ne csak egy embernek legyen lehetősége játszani, gondoltam, hogy implementálok egy stoppert is a játékon belül, aminek a mérése akkor ér véget mikor a játékos végigviszi a játékot, így van lehetősége a játékosoknak egymás ellen versenyezni, hogy ki tudja a legjobb időt elérni a játék végbeviteléhez.

Természetesen mivel itt véletlenszerű pályagenerálásról van szó, így az lesz a leggyorsabb, aki ismeri a játékot és céltudatosan tudja játszani, habár a szerencsének is van kis szerepe, mivel olyan pályát gyorsabb végigjáráni, amelyik egyenesebb és nem kell sokat keresni benne a feladat elemeit.

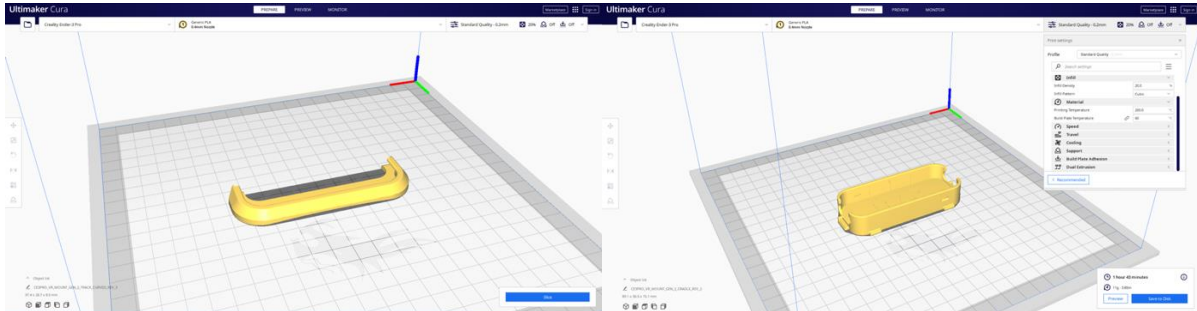
3 ALTERNATÍV MOZGÁS MEGHATÁROZÁSA

Már fentebb említettem hogy az Oculus Quest volt az egyetlen olyan VR eszköz, ami tartalmaz egy kezdetleges kézfelismerést, de azzal nem volt kivitelezhető a fejlesztés, és még a pontossága sem éri el azt a szintet amire más eszközök képesek, pontosabban kifejtve a Leap Motion milliméteres pontosságot garantál[14], amíg a Quest és Quest 2 csak megközelíti ezt az értéket. Egyetlen hátránya a Leap Motionnek a látótér a Questekhez képest, de az is bőven elegendő ahhoz, hogy megfelelő VR élményt nyújtson[15]. Mérlegelés után elsősorban a Leap Motion vezérlőegységen gondoltam végig a megvalósítást, de ehhez az kellett, hogy fizikailag összekössem a Leap Motiont a fej-egységgel. A Leap Motion gyártói ajánlása alapján a HMD elülső frontjának közepére erősítettem fel, és a rögzítéséhez terveztem egy adaptert, a gyártó weboldalán[13] megtalálható adatok alapján, és utána Fusion 360 programban modelleztem ezt. A modell köré keretet vontam, majd a port helyét kimetszettem(6. ábra). A keret mentén csináltam kettő domborulatot mindkét oldalon, majd a VR Headset ívére illeszkedően egy tartóelemet alkottam, ami illeszkedik a domborulatok pontjain, így biztosítva a tartását.



6. ábra: modellezés Fusion 360 szoftverben.

Mindezt Ultimaker Cura programba tettem, ami a modellnek a nyomtatási hálóját valósítja meg. Ezek után kerültek kinyomtatásra az elemek(7.-8. ábra).



7. ábra: 3D modell nyomtatás előtt

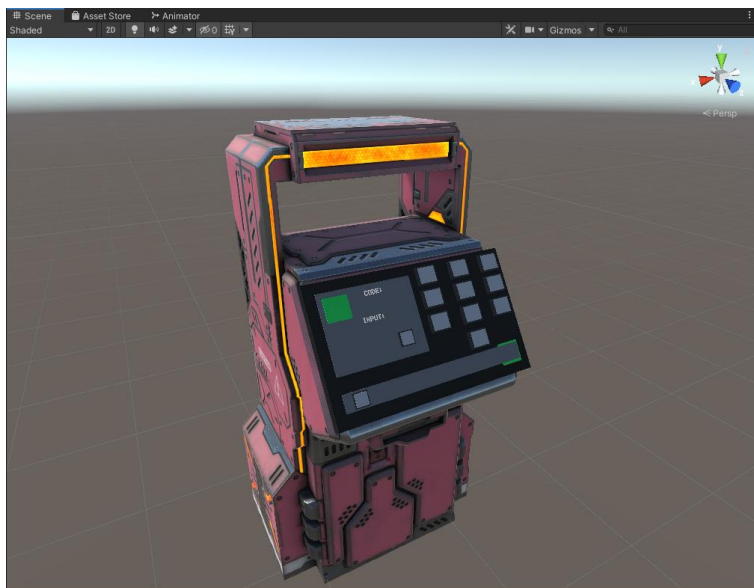


8. ábra: Nyomtatott Leap Motion tartó különböző szögekből.

Ezek után meg kellett terveztem, hogy milyen eseteket, VR funkciókat lehet megvalósítani, ha az ember a saját kezét használja a virtuális térben.

Konzol(9. ábra):

Létrehoztam bankautomatához hasonló modelleket, amelyeken gombokat helyeztem el. Kérdésem, hogy felismeri-e az ujjakat és lehet-e velük interakciókat kialakítani. A gombok mellett elhelyeztem csúsztatható kockákat is, amiket el kell húzni ahhoz, hogy folytatni lehessen a játékot.



9. ábra: Interaktív gombok a játékban belül.

A tesztelés során azt tapasztaltam, hogy amennyiben a VR HMD használatba vételekor a magasság helyesen volt beállítva, abban az esetben egészen kellemes, felhasználóbarát volt használni ezeket az elemeket. Ha nem volt helyesen beállítva a testmagasság akkor nem lehetett elérni a konzolt. Ha nem volt helyesen beállítva (például túl nagy volt az ATM a felhasználóhoz képest), akkor fel kellett nyúlni a felhasználónak a feje fölé, akkor a Leap Motion kézfelismerő szenzor kezdett bizonytalan lenni és nem regisztrálta rendesen a gombnyomásokat.

Speciális kijelző(10. ábra):

Emellett elhelyezésre került egy életszintjelző rendszer is, ami a kéztőcsont-fordulást figyeli a bal karon, és akkor jelenik meg, ha magunk felé fordítjuk a tenyerünket. Amint már feljebb említettem, a játékosnak három élete van, és ez itt került jelzésre a játékos felé.



10. ábra: Csuklómozgásra illesztett élet-mutató.

Kézi mozgásvezérlési rendszer

Létrehoztam saját, kézzel vezérelhető mozgásirányító rendszert, ennek elvét a következő fejezetben ismertetem.

3.1 MOZGÁSI MODELL MEGVALÓSÍTÁSA

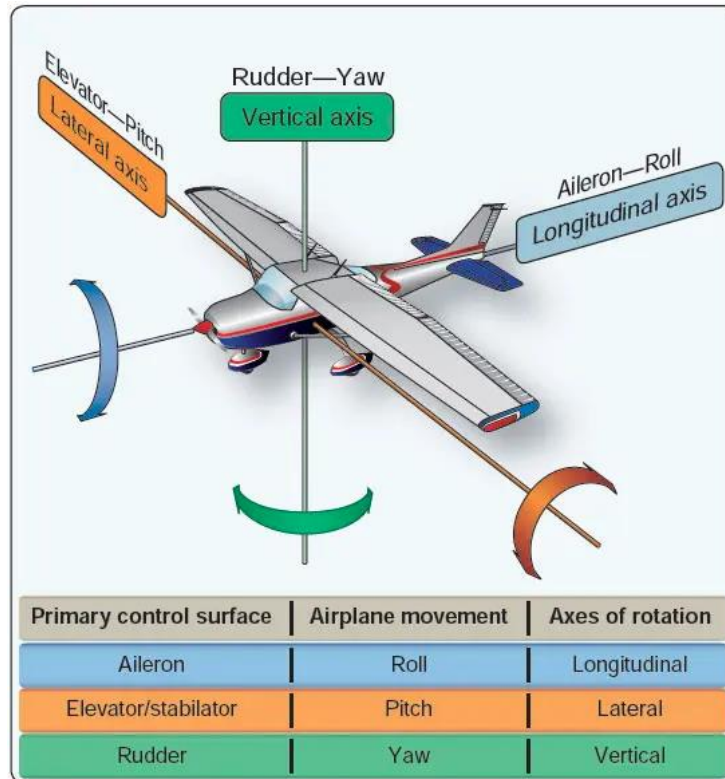
Elsősorban el kellett képzelnem, hogy milyen síkon mozog az ember, és emellett hogyan van elhelyezve a térben. A fejlesztőkörnyezetben a következő fogalmakkal kell tisztában lennünk:

három dimenziós térben való mozgás és forgásvektorok meghatározása, geometriával, és diszkrét matematikával.

Ezen belül a legfontosabb a forgásvektorok – Euler szögek és kvaterniók számítása.

Az Euler-szögszámítás azért hasznos és fontos, mert a tétel szerint leírható minden test forgása három darab szöggel. Mivel a térben három egymásra merőleges koordináta tengelyt tudunk megkülönböztetni (X, Y, Z), ezért három különálló elforgatást hajthatnak végre az egyes tengelyek körül (A fejlesztőkörnyezetben a sorrend Z, X, Y). Elfogadható alternatíva a kvaterniók használata, amik segítségével orientációtól függetlenül van lehetőségünk bármely elforgatást megvalósítani, egy adott szöggel és a vektorral kapott forgatási értékkel. Az első megoldást jobban preferáltam, mivel ennek a számítása sokkal komplexebb, de a Unity ennek a számításoknak a nehezét kiszámolja nekünk előre. Ennek ellenére is, jó ha tisztában vagyunk, hogy mégis mit szeretnénk elérni, mi a célunk a forgatással.

Ezt az elvet felhasználva lehetne irányítani a karakterünket, és tudván, hogy a kézfelismerés implementáció sikeres volt, így mertem merészen gondolkozni, és az alábbi ábrán látható repülő mozgását lehetne implementálni a mozgás kézi irányításához.



11. ábra: Repülőgép mozgása és forgatása térben[14].
Forrás: flightliteracy.com

Amint a fenti képen is látszik, három tengelyen tud forogni, amely segít neki a térben való mozgásban. Yaw, Pitch, Roll, tehát magyarul oldalirányú, függőleges és hosszirányú forgása van(11. ábra). Ha kitesszük magunk elé a kezünket, mi is tudjuk ebben a három síkban elforgatni a csuklónkat, a csuklómozgás ugyanis 3 szabadsági fokú mozgás. Ha megfeleltetjük ezt a logikát a fent említett kézfelismerő szenzorba, akkor nem csak elvben, hanem gyakorlatban is működésre bírható a mozgás „gesztusokkal”.

3.2 ALTERNATÍV MEGOLDÁSOK A KÉZMOZDULATOKKAL IRÁNYÍTOTT VIRTUÁLIS MOZGÁS KIVITELEZÉSÉRE

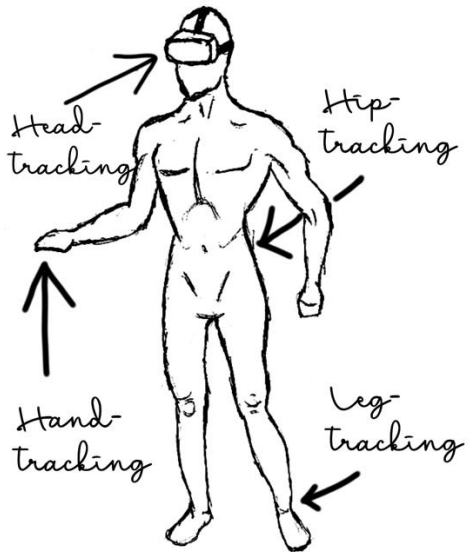
3.2.1 CSUKLÓMOZGÁSSAL TÖRTÉNŐ IRÁNYÍTÁS

Ahhoz, hogy egy mozgást le tudjuk követni (motion capture), ahhoz szükségünk van arra, hogy legyenek szenzorjaink, amikkel meg tudjuk állapítani a test irányát és állapotát.

Elsődleges probléma a globális/lokális északpont kijelölés volt. Melyik szenzor fogja megmondani hogy merre nézek, „merre van észak”? A Leap Motion sajnos nem rendelkezik olyan szenzorokkal, amelyek segíthettek volna nekem, esetleg egy optikai mozgásfelismeréssel tudtam volna, de az nagyon erőforrásigényes feladat lett volna.

Az alap elgondolás, hogy hova érdemes elhelyezni trackereket (szenzorokat, amelyek mozgást és irányt közvetítenek) négy pozícióban érdemes keresni(12. ábra)[15]:

- Head-tracking – Fejmozgás érzékelés
- Hand-tracking – Kézmozgás érzékelés
- Hip-tracking – csípőmozgás érzékelés
- Leg-tracking – lábfej érzékelés.



12. ábra: Tracker elhelyezés lehetőségei egy ember testén.

A HMD Headset elvileg jó megoldás, viszont van egy probléma ezzel is. Sajnos írásom ideje alatt Unity nem támogatja a dinamikus északpont kijelölés integrációját, ha ezt a két eszközt használom egyszerre, és akárhogy próbáltam végére járni a problémának, mindig ugyanoda jutottam.

Ellenben arra is gondoltam hogy kipróbálom a HTC által fejlesztett Vive Tracker nevű jeladót, amit az emberre erősítve képes az adatok leközlésére. Elgondolásom sajnos hamar elvetettem, mikor megtudtam hogy ennek kiépítése mennyibe kerül, mivel ezek az eszközök csak Base Station-nel működnek. A Base station tulajdonképpen úgy működik, hogy folyamatosan nem látható fénnel árasztják el a helyiséget. A nyomon követendő eszközök receptorai befogadják a kibocsájtott fényt, és ebből meg lehet állapítani, hogy hol vannak a Bázisállomásokhoz képest[19].

Elsőre azt gondoltam, hogy a globális északpont kijelölés hiányában a relatív mozgás lesz a nehezítés a játékban, ámbár használat közben hamar kiderült, hogy nagyon nehéz hozzászokni ehhez a mozgási sémához.

Nem adtam fel, alternatív megoldásokhoz folyamodtam. A kettő irányítási elvnek (joystick és kézfelismerés alapúnak) az ötvözetét vettem alapul.

Két megvalósítás is született, amit sikeresnek tapasztaltam:

3.2.2 MAGICSEE R1 CONTROLLER

Az első, amelyben egy apró, Magicsee R1 joystick gyűrűt alkalmaztam. Ezen a gyűrűn van egy analóg kar, ami el tud bújni a kéz belső vagy külső felére, ami nem zavar be a kézfelismerésben, így mindkét világból megmarad előnyként, ami a saját specialitása. A finom kézmozgást továbbra is lehet használni a játékon belül feladatok megoldásához, a mozgásban viszont a joystick és a Headset vesz részt.

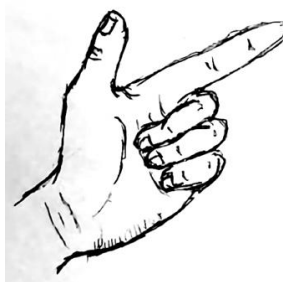
Ez egy biztonságos megközelítése a problémának, ami rendeltetésszerű, garantált működést fog biztosítani.

3.2.3 VIRTUÁLIS VR BILLENTYŰLENYOMÁS KÉZFELISMERÉSSSEL

Másik, merészebb megközelítés, ami nem biztosított a gyártók által, sőt, saját implementációval átlépjük a keretrendszer határait. Mivel egyes elemek nincsenek implementálva a Unity fejlesztőkörnyezetben, így egy-két helyen improvizációhoz hasonló helyzetek alakultak ki.

A következőképp működik ez a megoldás:

- Ha a játékos kinyújtja a hüvelyk és a mutatóujját (pisztolyt formál) mindkét kezével, akkor aktiválódik egy script, ami egy virtuális billentyűlenyomást eredményez (az én esetemben „W”)(13. ábra).



13. ábra: Aktiváló kézmozdulat illusztráció.

- A billentyűlenyomás egy előrehaladást fog kikényszeríteni a virtuális billentyűzet által, majd a headset északpontját felhasználva lehet vele előre haladni. Ha valamelyik kezünkkel már nem mutatjuk a gesztust, akkor az előrehaladás abbamarad.

Fontos megjegyezni, még egyszer, hogy ez a megoldás nem natív, alapesetben nem megoldható a mostani fejlesztőkörnyezetben, viszont reprezentálja azt, hogy ez az összeállítás mire lenne képes. Számomra ezt azért volt fontos kiemelni, mert koncepciónak alkalmassá vált a program, viszont olyan megoldás, ami hatással lehet a számítógép stabilitására azt nem tudnám végérvényesnek tekinteni (az én esetemben a virtuális gomblenyomást azt jelentette, hogy ha minimalizáltam a programot, akkor is figyelte a kézmozgást, és. a programon kívül is garantálta a billentyűlenyomást).

3.3 MOZGÁS KÉZI VEZÉRLÉS MÓDSZEREK ÖSSZEHAONLÍTÓ ELEMZÉSE

Egy nem-reprezentatív, általam tapasztalt összeállításban összemértem a tanulási görbáját a fent említett megoldásoknak. A három különböző locomotion technológiát amit implementáltam, a játékban használt időmérő segítségével hasonlítottam össze. Minden egyes sikeres futamot rögzítettem, lemértem és egy táblázatban rögzítettem, az egyszerűség kedvéért percekre kerekítve. A 25 futamot minden típusból normalizáltam, hogy jól látható legyen, hogy milyen teljesítményt lehet elérni e három mozgásfajta mellett.

Leap Motion - Fingergun	F3 /min	Leap Motion - kéztőcsont	F2 / min	Leap Motion + Joycon	F1 / min
-27%	19	-36%	20	1%	16
-27%	19	-27%	19	9%	15
-27%	19	-27%	19	9%	15
-27%	19	-27%	19	27%	13
-9%	17	-18%	18	27%	13
-9%	17	-9%	17	27%	13
-9%	17	-9%	17	36%	12
0%	16	-9%	17	36%	12
0%	16	0%	16	45%	11
9%	15	0%	16	55%	10
9%	15	0%	16	55%	10

9%	15	0%	16	55%	10
27%	13	9%	15	64%	9
36%	12	9%	15	64%	9
45%	11	9%	15	64%	9
55%	10	9%	15	64%	9
64%	9	27%	13	64%	9
64%	9	27%	13	73%	8
82%	7	27%	13	73%	8
82%	7	27%	13	73%	8
91%	6	27%	13	82%	7
91%	6	36%	12	82%	7
91%	6	45%	11	91%	6
91%	6	64%	9	100%	5
100%	5	73%	8	100%	5

A következő a képlet a számításhoz:

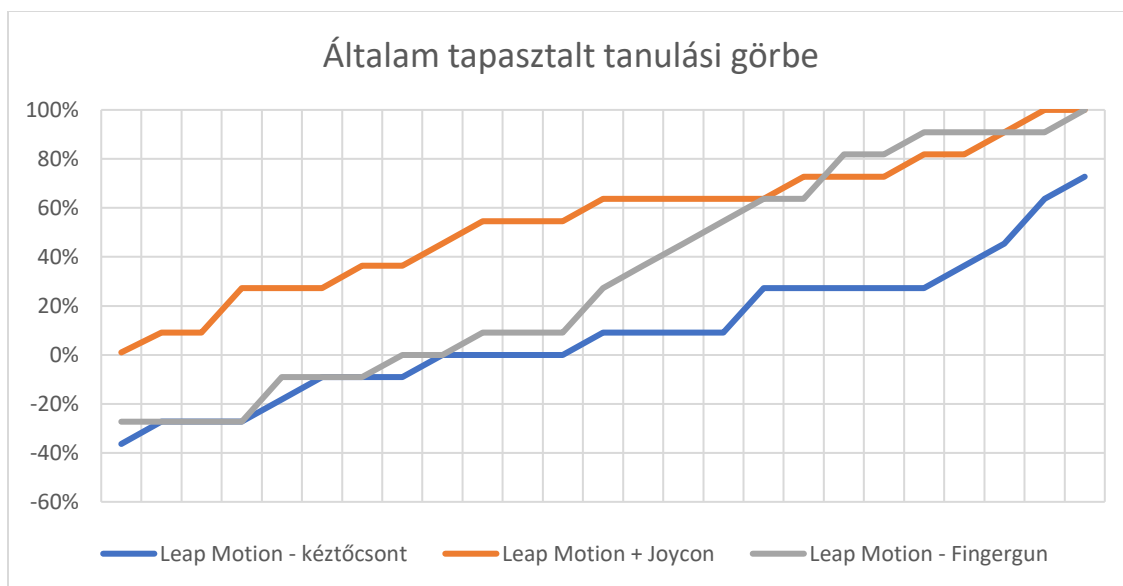
$$z_i = - \frac{x_i - x(\max_n)}{x(\max_n) - x(\min_n)}$$

ahol az

x_i = az aktuális mért érték

és

n = legkisebb intervallummal és legkisebb szórással rendelkező halmaz.



14. ábra: Általam tapasztalt tanulási görbe

A tesztet mindegyik esetben 25 alkalommal végeztem el, ezzel szemlélítve a fejlődés lehetőségét. Ahogy a fenti ábra is mutatja, a legnehezebben tanulható mozgástechnika a csuklómozgás alapú, ami a fent említett dinamikus északpont kijelölés hiánya miatt még jobban nehezítette a tanulhatóságát. A joycon megoldás a legeredményesebb, de ezzel még továbbra sem mondtunk le arról, hogy a kezünk üres legyen, ezért az arany középút a „fingergun” módszer, ami tanulhatóságát tekintve nehezebb mint a joycon, viszont hasonló végeredményt garantált.

4 FELHASZNÁLT TECHNOLÓGIA

Egyetemi éveim alatt volt lehetőségem több alkalommal játékot fejleszteni különböző projektek keretében, ami lelkessé tett abban, hogy mélyebben foglalkozzak vele. Ezért merült fel az ötlet, hogy lehetne nem csak egy átlagos játékot fejleszteni, legyen benne valami kihívás is. Konzulensem javaslatára virtuális valóság játékot alkottam, de legyen benne egy olyan csavar,

miszerint gondoljuk át a rendszer irányítását; vajon lehetséges lenne, ha nem lenne a kezünkben semmi és mégis lehetne irányítani valamiképp a játékot. Illetve fontos megemlíteni, a játéknak stabilan kell futnia egy átlagos hardverspecifikációval rendelkező számítógépen, illetve, ha lehetséges, el kell érnie legalább 90 FPS (képkocka / másodperc) válaszütemet, ugyanis ez alatt nagyobb eséllyel alakulna ki utazási betegség, vagy ismertebb nevén 'tengeribetegség' a HMD használata során. [16]

Egyre többen használnak virtuális valósághoz tartozó eszközöket (későbbiekben: VR) otthonukban, de még továbbra is azt mondhatjuk, hogy gyerekcipőben jár ez a technológia, és a felhasználók tapasztalják még ennek a fejlődését[17]. Éppen ezért merült fel bennem az a feltételezés, miszerint létezik-e jobb beviteli eszköz, mint amivel a mostani VR eszközök készülnek. Ezért kezdtem el azon gondolkodni, hogy milyen opciók vannak most a piacon, és milyen lehetőségek állnak rendelkezésre, ha most fejlesztőként és mérnökként valaki alkotna egy VR eszközkészletet. Több lehetőség is felmerült a kutatásom során, de végül egy – illetve két – megoldással tudtam abszolválni a feladat teljesítését.

Későbbiekben ki fogok térni bővebben az eszközök tartalmára, de ismertetésképpen szeretném bemutatni a fizikai eszközöket, amelyekkel volt szerencsém dolgozni:

- HP Windows Mixed Reality Headset – fejlesztői kiadás[18]
- Ultraleap Leap Motion Controller kézérzékelő szenzor[13]
- Magicsee R1 Mini Ring Bluetooth 4. Rechargeable Wireless VR Remote Game Controller Joystick Gamepad.[19]

A HP VR Headset-el már a projekt előtt rendelkeztem, a többi eszközt később szereztem be a projekt megvalósítása során.

A játék fejlesztéséhez Unity 2019 keretrendszert alkalmaztam, illetve a programkódokat Visual Studio 2018-ban írtam.

A számítógép, amin a fejlesztést végeztem a következő hardverspecifikációval rendelkezett:

- Alaplap: MSI B450M BAZOOKA / Gigabyte B550M DS3H
- Processzor: AMD Ryzen 5 3600
- Memória: Crucial 16 GB DDR4 3200MHz
- Videókártya: Gigabyte Nvidia GTX 1060 6GB / Asus Republic of Gamers RX 470 4GB
- Operációs Rendszer: Microsoft Windows 10 Pro for Workstations 64 bit ver. 21H1

Ez azért egy jó és reprezentatív felépítés, mert összhangban van az átlagos játékokra szánt konfigurációkkal, amik részt vesznek a negyedéves összesítésben (15. ábra)[20]:

AUGUST 2021 (CLICK LINE ITEM TO SEE MORE DETAIL)			
ITEM	MOST POPULAR	PERCENTAGE	CHANGE
OS Version	Windows 10 64 bit	86.64%	-3.02%
System RAM	16 GB	49.65%	+4.12%
Intel CPU Speeds	2.7 Ghz to 2.99 Ghz	17.71%	+2.90%
Physical CPUs	4 cpus	36.79%	-1.75%
Video Card Description	 NVIDIA GeForce GTX 1060	10.14%	+1.24%
VRAM	6143 MB	26.28%	+2.44%
Primary Display Resolution	1920 x 1080	68.42%	+1.22%
Multi-Monitor Desktop Resolution	3840 x 1080	70.56%	-0.22%
Language	English	33.56%	-4.40%
Free Hard Drive Space	100 GB to 249 GB	24.01%	+0.03%
Total Hard Drive Space	Above 1 TB	55.02%	+3.00%
VR Headsets	Oculus Quest 2	32.15%	-0.42%
Other Settings	SSE3	100.00%	0.00%

15. ábra: Valve általános számítógép konfigurációs listája.
Forrás: Steam.com

Szeretném megjegyzésként beleírni, hogy az írás közben viharkárt szenvedett a gépem, ezért új alaplap és videokártya vált szükségessé, ám ez mérsékelten befolyásolta csak a teljesítményt.

Érdekességgéppen utánanéztem hogy lett volna rá lehetőségem hogy esetleg más platformon is működjön a játékom. A válasz egyben igen is és nem is. Unityben van arra lehetőség hogy Mac és Linux platformokra publikáljunk játékot, csak mivel Windows Mixed Reality Headsetet használok, ez erősen ragaszkodik Windows platformhoz. Tehát ha a játékot nem VR-ban játsza a

felhasználó, akkor van arra lehetősége hogy más platformon is képes legyen elindítani a programot, de a WMR csak Windows alatt fog működni, a MacOS nem is ismeri fel, Linuxon belül pedig egy OpenHMD API-val lehetséges képet megjeleníteni, de nagyon limitált lehetőségek mellett működne[25].

4.1 FEJLESZTŐKÖRNYEZET ISMERTETÉSE

Unity, a Unity Technologies által fejlesztett 3D-s motor, ami legfőképp videóanyag készítésére és játékfejlesztés céljával alkottak meg, amelyet először 2005. júniusában jelentettek be és jelentettek meg az Apple Inc. világméretű fejlesztői konferenciáján Mac OS X exkluzív játékmotorként. 2018-tól a motort kibővítették, hogy több mint 25 platformra támaszkodjon [26]. Fontos elismerni, hogy képes nagy adatbázisok kezelésére, geometriai eszközcsomagok közvetítésére, a fényhatások előzetes kiszámítására vagy valós idejű megvalósítására, interakciós és animációs lehetőségek biztosítására, különböző scriptek és objektumok viselkedésének beállítására, miközben a végtermék továbbra is stabil és gördülékenyen használható marad. A motor felhasználható háromdimenziós, kétdimenziós, VR és kibővített valóság (AR) játékok, valamint szimulációk létrehozására. Az egyszerűség kedvéért nyugodtan mondhatjuk, hogy a Unity a világ egyik legnépszerűbb játékmotorja. Nagyon sok funkció elvégzésére képes, és elég rugalmas ahhoz, hogy szinte bármilyen játékot elkészítsen bárki, csak a fantáziánk szabhat határt.

A Unity több fázison ment keresztül mire a mai formáját kapta – Régen JavaScript-ben (sajátos formavilágának köszönhetően saját nevet kapott: UnityScript) lehetett a tartalmat programozni, ma már C# programozási nyelvet használ, illetve a végtermék C++ binárisokban kerül összeállításra. A motort a videojátékokon kívüli iparágak, például a film, az autóipar, az építészet, a mérnöki és az építőiparban is alkalmazták[27]. A Unity ingyenes verziót is kínál, így a fejlesztők kiadhatják a 'Unity Personal' licensz alatt készített tartalmakat anélkül, hogy a szoftverért fizetnének, feltéve, hogy kevesebb, mint 100 000 dollárt keresnek a Unity-vel készült játékkal. Volt lehetőségem élni a 'Student Discount' programjukkal, ami azt jelenti, hogy ha oktatásban szereplő veszi igénybe a Unity motort, akkor akkreditációs programon belül felhasználható a Unity Pro verziója a motornak, feltéve, hogy nem eladásra készül a program.[21]

Ahhoz, hogy ebben tudjak dolgozni, fontos importálni olyan script csomagokat a projektbe, amik biztosítják az eszközeimmel a kommunikációt – hasonlóképp működik, mint a Visual Studio-ban belül a NuGet Package technológia[22] – így lehetségessé válik különböző eszközök programozhatósága a projekten, avagy játékon belül. A következő scriptek kerültek elhelyezésre, amelyeket felhasználtam a projektmen belül:

- Leap Motion Software Development Kit – Leap Motion szenzorhoz[23],

- Unity XR / AR Foundation kit – VR fejlesztéshez[24],
- Unity NavMesh Component kit – útvonal-kereső algoritmushoz[25],
- Unity TextMesh Pro kit – UI kialakításához.
- Unity Asset Store – Helyettesítő / helyőrző elemekhez.

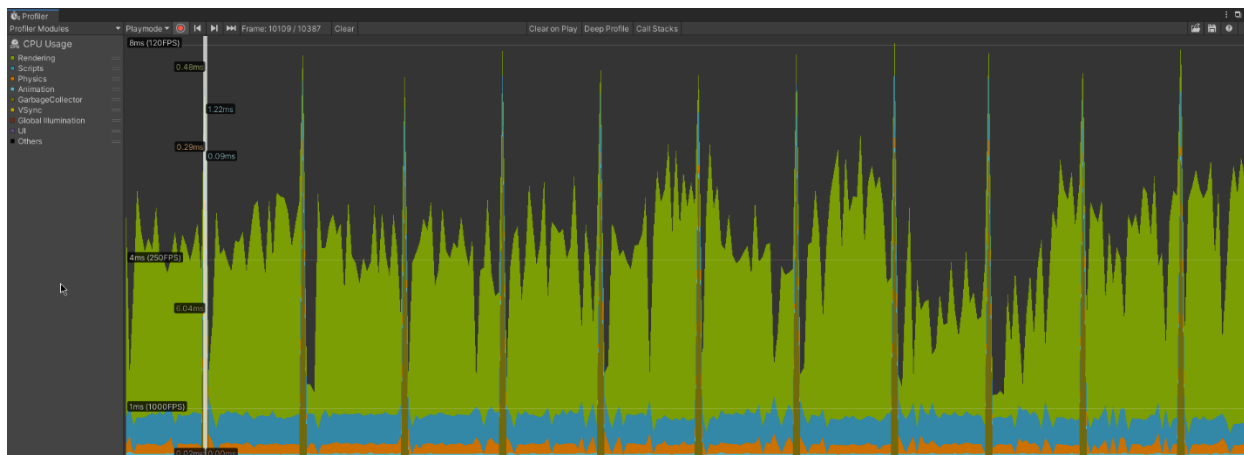
Fontos azért megjegyeznem, hogy alapvetően több scriptet implementáltam a projektembe, de nem használtam fel végül, például a Microsoft Kinect Software development kit -et – erre kitérek a következő fejezetben, hogy miért.

5 JÁTÉKOPTIMALIZÁLÁS, EREDMÉNYEK

Különböző technikákat kellett alkalmaznom ahhoz, hogy a VR játékelmény mindenki számára a lehető leghatékonyabb élményt nyújtsa. A szoftver problémák kiértékelését a Unity fejlesztőkörnyezetbe épített profilozóval oldottam meg[29]. Ez elsősorban azt mutatja meg képkockákra lebontva, hogy milyen tevékenységet/folyamatokat dolgoz fel éppen a gép. Itt lehet kiugró adatokat elemezni, hogy minél gördülékenyebben fusson a játék, ne legyenek benne lassulást okozó folyamatok.

Occlusion culling

Elsősorban a következő problémát kellett megoldanom, amivel szembesültem miközben használtam a játékot (18. ábra):



16. ábra: Unity profilozó grafikonja 0-8 ms közötti intervallumon.
Zöld: Renderelés, Kék: Scriptek, Narancs: Fizika szimuláció, Türkiz: Animáció.

Amint a grafikon is mutatja, a játék első verzióiban kevésbé optimalizált volt az élmény, gyakran akadt meg a játék, amit a grafikonon látható felcsapó élek is mutatnak. A zöld színnel jelzett érték a renderelés, ami azt jelenti, hogy a képkockához szükséges a legtöbb erőforrás. Az élek arra utalhatnak, hogy ha egy olyan irányba nézek, ahol több játékelem van, annak feldolgozása hirtelen terhelésnek teheti ki a gépet, tehát olyan feldolgozandó elemek voltak a pályán, amiket akkor is számításba vett a program, miközben nem is volt az ember a közelében (vagy nem látott rá).

Ennek a megoldása trükkös a Unity-n belül, ugyanis én tudtam, hogy occlusion cullingot, vagyis magyarra fordítva takarás alapján történő megjelenítést jelent – kellett megvalósítani. Ezt

kibontva, annyit tesz a programunkkal, hogy azok a dolgok/textúrák amikre nem lát rá a játékos, azt nem veszi számításba és inaktívvá teszi. Fontos leszögezni, ha van dinamikus tartalom a pályán (például egy ellenség, ami mozog), akkor azt meg kell jelölni a programon belül hogy arra ne vonatkozzon ez az eljárás, különben megfagyaszthatja azt az entitást addig ameddig nincs a játékosnak rálátása.

Ezzel is ugyancsak az a probléma, hogy ezeket az adatokat a Unity szereti „belesütni” a programba, tehát még a program lefutása előtt tudja, hogy mire vonatkozik és mire nem. Az én programom objektumai dinamikusan a futás ideje alatt generálódnak, így a Unity-nak nincs fogalma arról, hogy a pályáról miképp kellene a takarás-számítást végezni.

Erre alkottam egy egyszerű algoritmust:

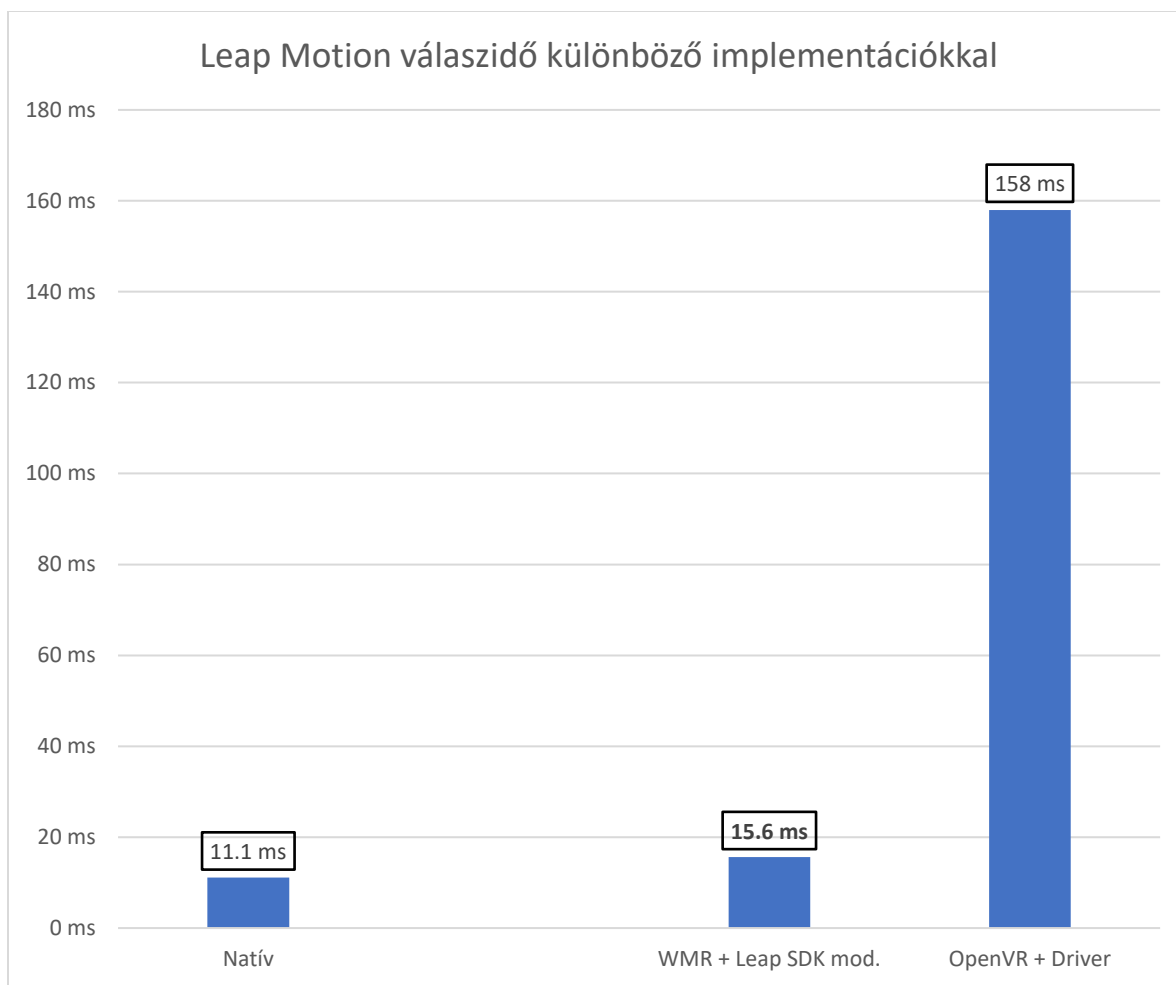
Vesszük a játékost, ráhúzunk egy láthatatlan, áttetsző téglatestet (collider), és ezt felskálázzuk egy nagyobb méretre. Ez lesz a mi „látótávolságunk”. Ami ezzel a téglatesttel fizikailag ütközik, az megjelenítésre kerül, minden más, ami nincs jelölve dinamikusnak, az nem kerül megjelenítésre.

Ez a megoldás működik, viszont annyit hozzá kell tennem, hogy ha nyílt térben vagyunk, akkor előfordulhat, hogy textúrahiányokat lehet látni a horizonton. Ennek orvoslására el lehet ködösíteni a távolabbra eső teret, hogy ez ne legyen annyira zavaró.

Leap Motion integráció

Következő optimalizációt amit végeztem, azt magával a Leap Motion integrációval végeztem. Amikor valaki elkezd a Unityban VR eszközre fejleszteni, alapvetően az OpenVR alapot kell megjelölni mint opció, ennek az előnye a kompatibilitás, viszont erősen visszafogja a hardverek működését mint funkcióban, mint reszponzivításban.

Én mindennek ellentmondva, kikapcsoltam ezt a kompatibilitási réteget és manuálisan integráltam bele az SDK-k tartalmát és ezzel azt tapasztaltam, hogy sokkal kisebb lett a válasz ideje a hardvernek, mint ha a fejlesztőkörnyezetre bíztam volna.



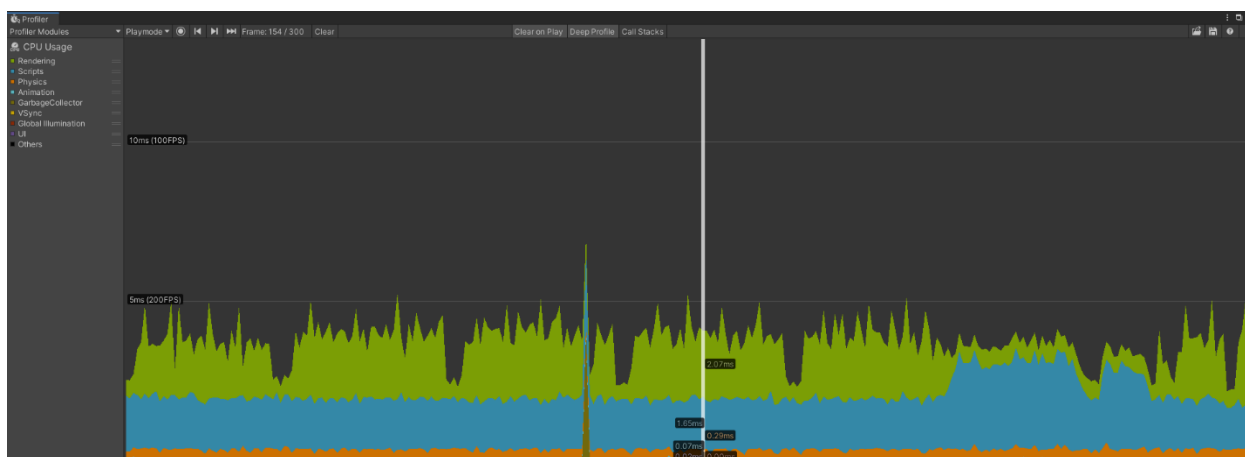
17. ábra: Leap Motion válaszidő különböző implementációkkal.

Végeztem egy mérést, amiben külső és belső felvételt készítettem arról ahogyan a két metódust használva melyik mennyire reagál azonnal az eredeti, kamerához igazított képhez képest. 240 FPS-ra állítottam a kamerát és a belső felvételt, majd összehasonlítottam az értékeket a natív vizualizátorral, illetve egymással.

Amint a fenti ábra is mutatja, sikerült közel natív implementációt létrehozni, ahhoz képest, hogy az OpenVR milyen lehetőséget garantált.

5.1 FEJLESZTÉSI LEHETŐSÉGEK

Ez a két megoldás ötvözetével sikerült egy jól működő, stabil, és optimalizált rendszerű játékot létrehozni. A profilozó is már egy sokkal simább és nagyobb képfrekvenciájú játékot prezentál (20. ábra). Sajnos a videokártya csere miatt az átlagosnál kicsit kevesebb lett a képkockamennyiség (4-5% veszteség), de még így is benne van abban hibahatárban, hogy akár egy háttérben futó alkalmazás is okozhat ilyenfajta romlást a mérés kapcsán.



18. ábra: Profilozó ábra optimalizálás után.

Fejlesztési lehetőségként én elsősorban arra tudnék gondolni, hogy több, színesebb akadályt elhelyezhetnék a pályán. Gondoltam arra is, hogy több ellenfél közül lehetne választani, és mindegyik másképp viselkedne, ami miatt más taktikához kell folyamodni a játék során. Esetlegesen, az új karakterekhez új kihívások is társulnának, így akár a felismerés finomítása, esetleg újabb kézmozdulatok felismerése is elképzelhető fejlesztési pont lehet.

6 ÖSSZEFOGLALÁS

A célom az volt, hogy megoldjam a VR környezetben a kéz használatát, kiváltva a kézi kontrollereket. Témaválasztásomat az indokolta, hogy igen sok alkalmazásban szükség lehet erre. Ezt az irodalomkutatás is alátámasztotta, viszont ez nagyon sok HMD eszköznél nincsen megoldva. Ahol meg van oldva, ott jellemzően a kézkövetés és gesztus felismerés pontossága hagy kívánni valót maga után. A feladat részeként több eszközt tesztelve kiválasztottam egy kézfelismerő eszközt, a Leap Motion szenzort, amit a már meglévő VR szemüvegemhez illesztettem. Az összeillesztéshez készült 3D nyomtatással egy tartóelem, ami biztosította az eszköz rögzítését. Az irodalom alapján áttekintettem, hogy milyen lehetőségek állnak rendelkezésre, ha be tudnánk vonni a kézfelismerő eszközt a játékon belüli mozgáshoz és interakciókhoz. Ezt követően megvalósítottam egy virtuális konzolt, amelyet gombokkal lehet kezelni, és egy speciális kijelzőt, amely gesztus kiértékelés alapján működött. A térbeli mozgás irányításához 2+1 külön fajta megoldást valósítottam meg a, ezek használhatóságát összehasonlítottam. Készítettem egy játékot, ami véletlenszerűen generált egy labirintust és tartalmazza az összes általam fejlesztett funkciót. Az alkalmazást leteszteltem, hogy mennyire stabil profilozó segítségével, majd optimalizáltam. Megvizsgáltam, hogy melyik SDK-val lehet a legjobb eredményt elérni és fejlesztettem egy egyszerű occlusion culling algoritmust, ami VR rendszerben használható. Vizsgálataim alapján egy alacsony késleltetéssel rendelkező kézfelismerőrt tudtam implementálni, ami előnyös előrelépés a beépített kompatibilitási réteghez képest. Bővítési lehetőségként ajánlottam további új programfunkciókat, amik sajátos viselkedéssel is rendelkezhetnek.

7 SUMMARY

My goal was to solve the problem of using hands in the VR environment, replacing handheld controllers. My choice of topic was motivated by the fact that there are many applications where this might be necessary. This was supported by the literature research, but it is not solved in many HMD devices. Where it is solved, the accuracy of hand tracking and gesture recognition typically leaves something to be desired. As part of this exercise, I tested several devices and selected a hand recognition device, the Leap Motion sensor, to pair with my existing VR glasses. To fit it together, I 3D printed a mounting element to secure the device. I reviewed the literature to see what options were available if we could incorporate the hand recognition device for in-game movement and interaction. I then implemented a virtual console that could be controlled by buttons and a special display that worked based on gesture evaluation. I implemented 2+1 separate types of solutions for controlling spatial movement and compared their usability. I created a game that randomly generated a maze and included all the features I developed. I tested the application to see how stable it is using a profiler and then optimized it. I investigated which SDK would give the best results and developed a simple occlusion culling algorithm that could be used in VR. Based on my tests, I was able to implement a low-latency hand recognition algorithm, which is a beneficial improvement over the built-in compatibility layer. As an extension, I proposed additional new program features that could have specific behaviours.

8 KÖSZÖNETNYILVÁNÍTÁS

Köszönöm elsősorban Nagyné Dr. Hajnal Évának a folyamatos rendelkezésre állását, és hogy végig vezetett és segített ezen a nehéz témán, köszönöm Dobra Rebekának az illusztrációkat, és Illetve Kiss Dániel és Ács Márton kollégáimnak a visszajelzését a programról, és Jakab István László kollégámnak is, aki felajánlotta segítségét a modellezésben és rendelkezésre bocsájtotta a 3D nyomtatóját!

IRODALOMJEGYZÉK

- [1] "Coronavirus sent us home. Will VR bring us back together? - Protocol — The people, power and politics of tech." <https://www.protocol.com/vr-headset-future-of-work> (accessed Nov. 10, 2021).
- [2] D. Holmes *et al.*, *Usability and performance of Leap Motion and Oculus Rift for upper arm virtual reality stroke rehabilitation Communication Style and Exercise Compliance in Physiotherapy (CONNECT) View project PhD Motion Tracking in Digital Images View project Usability and performance of Leap Motion and Oculus Rift for upper arm virtual reality stroke rehabilitation*. 2016. [Online]. Available: <https://www.researchgate.net/publication/312868469>
- [3] A. G. D. Corrêa, N. R. Kintschner, V. Z. Campos, and S. M. Blascovi-Assis, "Gear VR and leap motion sensor applied in virtual rehabilitation for manual function training* an opportunity for home rehabilitation," in *PervasiveHealth: Pervasive Computing Technologies for Healthcare*, Sep. 2019, pp. 148–151. doi: 10.1145/3364138.3364169.
- [4] D. E. Holmes, D. K. Charles, P. J. Morrow, S. McClean, and S. M. McDonough, "Using fitt's law to model arm motion tracked in 3D by a leap motion controller for virtual reality upper arm stroke rehabilitation," in *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, Aug. 2016, vol. 2016-August, pp. 335–336. doi: 10.1109/CBMS.2016.41.
- [5] M. Chifor and T. Stefanut, "Immersive Virtual Reality application using Google Cardboard and Leap Motion technologies."
- [6] "Oculus Quest 2 Hand Tracking Has So Much Untapped Potential." <https://www.thegamer.com/oculus-quest-2-hand-tracking-has-so-much-untapped-potential/> (accessed Nov. 10, 2021).
- [7] F. Nainggolan, B. Siregar, and F. Fahmi, "User Experience in Excavator Simulator using Leap Motion Controller in Virtual Reality Environment," in *Journal of Physics: Conference Series*, Jul. 2020, vol. 1566, no. 1. doi: 10.1088/1742-6596/1566/1/012093.
- [8] F. Fahmi, K. Tanjung, F. Nainggolan, B. Siregar, N. Mubarakah, and M. Zarlis, "Comparison study of user experience between virtual reality controllers, leap motion controllers, and senso glove for anatomy learning systems in a virtual reality environment," in *IOP Conference Series: Materials Science and Engineering*, May 2020, vol. 851, no. 1. doi: 10.1088/1757-899X/851/1/012024.
- [9] Valve Corporation, "Valve Index Controllers: Your hands, in VR," <https://www.valvesoftware.com/en/index/controllers>, Jun. 28, 2019.
- [10] G. 'Pierre-Loup A., "Valve Index controller Extender," https://github.com/ValveSoftware/IndexHardware/blob/master/Controller/index_controller_example.stl, Jun. 25, 2019.
- [11] W. 'Mason, "Palmer Luckey Speaks Candidly About Xbox One Controllers Being Bundled with the Rift," <https://uploadvr.com/palmer-luckey-speaks-candidly-about-xbox-one-controllers-being-bundled-with-the-rift/>, Jun. 12, 2015.
- [12] "Quest Gets Hand Interaction SDK & Tracked Keyboard SDK." <https://uploadvr.com/quest-interaction-sdk-tracked-keyboard-sdk/> (accessed Apr. 17, 2022).
- [13] T. 'Jean-Luc and M. 'Scott, "GeiselVR." Apr. 12, 2018.
- [14] C. Khundam, V. Vorachart, P. Preeyawongsakul, W. Hosap, and F. Noël, "A comparative study of interaction time and usability of using controllers and hand tracking in virtual reality training," *Informatics*, vol. 8, no. 3, Sep. 2021, doi: 10.3390/informatics8030060.
- [15] "Leap Motion Hand Tracking vs Oculus Quest One vs Vive One | PaleBlue." <https://pale.blue/2020/07/09/leap-motion-hand-tracking-vs-oculus-quest-one-vs-vive-one/> (accessed Nov. 10, 2021).

- [16] "Tracking | Leap Motion Controller | Ultraleap." <https://www.ultraleap.com/product/leap-motion-controller/> (accessed Nov. 10, 2021).
- [17] "Flight Control Systems – Primary Flight Controls (Part One)." <https://www.flightliteracy.com/flight-control-systems-primary-flight-controls-part-one/> (accessed Nov. 10, 2021).
- [18] "The Pros and Cons of VR Full Body Tracking." <https://exyte.com/blog/vr-full-body-tracking> (accessed Nov. 10, 2021).
- [19] "Base Stations & VR Arcades. The Basics, The Do's, and The Don'ts | by SpringboardVR | Medium." <https://medium.com/@springboardVR/base-stations-vr-arcades-1dffac2abf05> (accessed Dec. 02, 2021).
- [20] Lili Feyerabend, "VR's Motion Sickness Problem — And How To Hack It," *medium.com - DangerousTech* (<https://medium.com/dangeroustech/vrs-motion-sickness-problem-and-how-to-hack-it-with-the-help-of-your-cat-a040cf72a7f3>), Sep. 2017.
- [21] "Can advertising scale in VR? | TechCrunch." <https://techcrunch.com/2021/07/08/can-advertising-scale-in-vr/?guccounter=1> (accessed Nov. 10, 2021).
- [22] "Mixed Reality Headset / Augmented Reality (AR) Headset | HP® Store." <https://www.hp.com/us-en/shop/cv/mixed-reality-headset> (accessed Nov. 10, 2021).
- [23] "Magicsee R1 Bluetooth Controller".
- [24] "Steam Hardware & Software Survey." <https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam> (accessed Nov. 10, 2021).
- [25] "GitHub - OpenHMD/OpenHMD: Free and Open Source API and drivers for immersive technology." <https://github.com/OpenHMD/OpenHMD> (accessed Dec. 02, 2021).
- [26] "Unity game engine goes cross-platform." <https://www.macworld.com/article/176746/unity-3.html> (accessed Dec. 01, 2021).
- [27] "3D Software for Architecture, Engineering & Construction | Unity." <https://unity.com/solutions/architecture-engineering-construction> (accessed Dec. 11, 2021).
- [28] "Unity Student Plan – Access the real-time 3D development platform and workflows that professionals use to create immersive experiences across industries. | Unity." <https://unity.com/products/unity-student> (accessed Nov. 10, 2021).
- [29] "What is NuGet and what does it do? | Microsoft Docs." <https://docs.microsoft.com/en-us/nuget/what-is-nuget> (accessed Nov. 10, 2021).
- [30] Ultraleap, "Ultraleap Developer Leap Motion Unity Manual," https://developer.archive.leapmotion.com/documentation/v2/unity/api/Leap.HandList.html#csharpclass_leap_1_1_hand_list_1aa88b64262c71a8f1252fd0b7d5fde07d, 2013.
- [31] "Unity's AR Foundation Framework | Cross platform augmented reality development software | Unity." <https://unity.com/unity/features/arfoundation> (accessed Nov. 10, 2021).
- [32] "GitHub - Unity-Technologies/NavMeshComponents: High Level API Components for Runtime NavMesh Building." <https://github.com/Unity-Technologies/NavMeshComponents> (accessed Nov. 10, 2021).
- [33] "Unity - Manual: Profiler overview." <https://docs.unity3d.com/Manual/Profiler.html> (accessed Nov. 10, 2021).