DEBRECENI EGYETEM
MŰSZAKI KAR
GÉPÉSZMÉRNÖKI TANSZÉK

UNIVERSITY OF DEBRECEN
FACULTY OF ENGINEERING
DEPARTMENT OF MECHANICAL ENGINEERING

IPARI VR TARTALOMKÉSZÍTÉS TERVEZÉSE

SZAKDOLGOZAT

Kovács Otilia Beatrix

Járműipari folyamattervező specializáció

Debrecen

2024

Tartalomjegyzék

1 Bevezetés

A Virtuális valóság technológia rohamosan fejlődik, és az alkalmazási területek folyamatosan bővülnek. Ezáltal a virtuális környezetek és az interaktív élmények egyre nagyobb lehetőségeket kínálnak az iparban.

A vállalatok egyre inkább felismerik a VR technológia potenciálját, és egyre többet fektetnek bele, mivel segíti a hatékonyság és a versenyképesség növelését.

A mérnöki tudás soha nem áll meg, mindig fejlődésre és új fejlesztésekre van szükség, ezért a munkaerőnek folyamatosan szüksége van az alkalmazkodáshoz és az új készségek elsajátításához.

A szakdolgozatom oktatási céllal jött létre, tekintettel arra, hogy a munkavállalóknak szükségük van alkalmazkodó képességre és új készségek elsajátítására. Egy interaktív karbantartási tréninget hoztam létre, amelyben egy elektromotor szétszerelését és egy csapágy kicserélését mutatom be az Unity programozói környezetben, ezáltal az iparban dolgozók jobban megérthetik a munkafolyamatokat, megelőzhetik a hibákat és a baleseteket, illetve az elméletben megszerzett tudást gyakorolhatják a szimulált környezetben.

Az interaktív tréning létrehozásakor fontos szempont volt, hogy a felhasználók minimális hibalehetőséggel találják szembe magukat, ezért csak az utasításokat követve tudják befejezni az oktatóanyagot. A tréning elvégzése után a dolgozók magabiztosabbak lehetnek és könnyebben megoldhatják a valós szituációkban a problémákat.

Emellett bemutatom a tartalomkészítéshez szükséges eszközöket és szoftvereket, a hatékony használat érdekében.

Célom az, hogy az iparban dolgozók elsajátítsák az ipari tartalomkészítés alapjait, és átfogó képet kapjanak erről a technológiáról, ezáltal eredményesen tudják használni azt a munkájuk során.

A szakdolgozatom emellett áttekintést nyújt a jelenlegi és a jövőbeli irányokról, illetve segít, hogy innovatív módon gondolkodjunk az ipari folyamatokról.

Az előző cégemnél töltött idő alatt ismerkedtem meg a VR technológia fogalmával, ezután motivációt kaptam arra, hogy jobban megismerjem és elsajátítsam a használatát, ehhez a YouTube videótárát vettem segítségül.

Emellett az egyetemi tanulmányaim alatt lehetőségem nyílt szabadon választható tantárgyak keretein belül kipróbálni a VR technológia eszközeit. Ezek az élmények inspiráltak arra, hogy a szakdolgozatom során is ezzel a témával foglalkozzak.

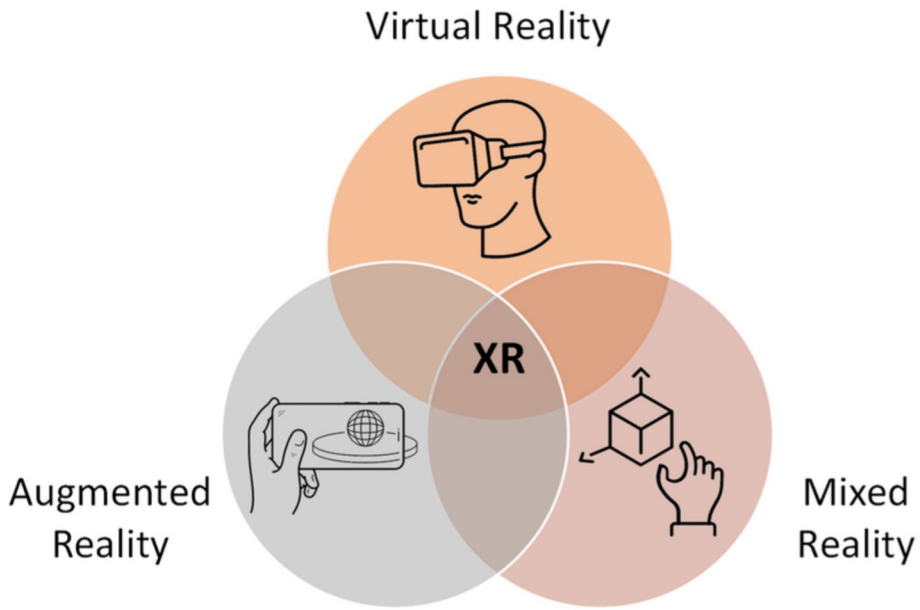
2 Kibővített, virtuális, kiterjesztett valóság a kevert valóság és társtechnológiák általános bemutatása

2.1 A kibővített valóság (XR)

Paul Milgram alkotta meg azt a szakkifejezést, amely az összes ember-gép interakcióra következtet. Tehát az Extended Reality vagyis rövíditve XR egy gyűjtőfogalom, amely magába foglalja a virtuális valóságot, a kiterjesztett valóságot és a kevert valóságot is.

Az XR gyorsan fejlődik és már sok területen alkalmazzák, mivel virtuális élményekkel lendíti fel a felhasználók valós élményeit. Minden jel arra utal, hogy ez a jövőben egyre dinamikusabban fog fejlődni. [1]

A fejlesztők törekednek arra, hogy a lehető legjobb tudásuk szerint járjanak el és új funkciókkal bővítsék a jelenlegi technológiákat, illetve izgalmasabb élményeket biztosítsanak a felhasználók számára.



1. ábra. A kibővített valóság [2]

2.2 A Virtuális Valóság (VR)

A virtuális valóság egy számítógép által generált környezet, amely lehetővé teszi, hogy a felhasználó mesterséges háromdimenziós környezettel lépjen kölcsönhatásba. [3]

Ezekben a virtuális környezetekben a felhasználó olyan élményeket élhet át, amelyek nem elérhetőek a valós életben. Interaktív eszközök segítségével, mozoghatnak és reagálhatnak az őket érintő eseményekre, ezáltal teljesen el tudnak merülni bennük.

Az élmény átéléséhez inkább VR fejszetteket használnak, de léteznek VR alkalmazások is, amelyek mobiltelefonokon, tableteken és számítógépen is használhatóak, de ezek természetesen kevesebb élményt tudnak biztosítani a felhasználóknak.

1838-ban Sir Charles Wheatstone volt az első, aki írt a sztereopszisról, majd létrehozta a sztereoszkópot. Azt jelenti, hogy a felhasználó két különböző képet lát, különböző szögből, ugyan arról a témáról, ez biztosítja a binokuláris parallaxis hatást, vagyis a térélmény kialakulását.

A későbbiek során többen is hozzájárultak a virtuális technológia fejlődéséhez, mint például Morton Leonard Heilig a „Sensora” -mát 1962-ben szabadalmaztatta, ehhez szükség volt egy 3D kamerára és egy kivetítőre és ehhez készített öt filmet.

1965-ben Ivan Sutherland előállt az „The Ultimate Display” ötletével, majd megalkotta a „Damoklész kardját”, ez volt az első hardveres megvalósítás. Ez sztereó látást támogatott, ami a felhasználó fejmozgásához igazodott. A szerkezetet egy rúddal mennyezethez rögzítették.

A Jaron Lainer nevéhez fűződik az 1980-as években a „DataGlove” és az „EyePhone”-t feltalálása. A DataGlove-ból optikai szálak futottak ki, amelyek fényjelzéseket továbbítottak azáltal, hogy a felhasználó hogyan mozgatta a kezét, majd ezeket a jelzéseket továbbította az „EyePhone” kijelzőre.

Napjainkaban a VR technológia fejlesztésébe egyre nagyobb összegeket ruháznak be, ennek köszönhetően a technológia rohamosan fejlődik. [4]

2. ábra. A virtuális valóság [5]

2.3 A Kiterjesztett Valóság (AR)

Az AR fogalma az 1990-es évekre tekinthető vissza. A kibővített valóság a valóságot veszi alapul keverve digitális objektumokkal. Azuma szerint az AR inkább kiegészíti a valóságot, mintsem, hogy teljesen helyettesítené azt. [6]

A kibővített valóság élménye okostelefonnal, tablettel, fejszettel és AR szemüveggel élhető át.

Pár évvel ezelőtt belépett a köztudatba a „Pokemon Go" nevezetű telefonos játék, ami az AR technológiát használva működik. A játéknak az a lényege, hogy a felhasználónak az őt körülvevő valós környezetben kell megkeresni a Pokemon labdákat.

Az okostelefonokba beépített szenzorok segítségével érzékelhető a játékos térbeli pozíciója és a helyzete, ezáltal az alkalmazás követni tudja a felhasználó pozícióját. Ilyen szenzorok például a kamera, digitális iránytű, gyorsulásmérő, giroszkóp és a lépésszámláló. [7]

3. ábra. A „Pokemon Go” nevű játék [8]

2.4 A Vegyes Valóság (MR)

A vegyes valóság (MR) egy hibrid technológia, ahol a valós világ egyesül a virtuális világgal és a felhasználók interakcióba tudnak lépni a valós környezetben lévő objektumokkal és emberekkel, és eközben virtuális elemek is megjelennek.

A vegyes valóság kifejezést Milgram és Kishino alkotta meg 1994-ben a „The Taxonomy of Mixed Reality Virtual Devices” című tanulmányában. Ez a technológia magában foglalja az AR és VR technológiákat is. [9]

2016-ban a Microsoft készítette el az első ilyen önálló szemüveget HoloLens néven, amelyen futtatható a Windows 10-es rendszer. Átlátszó holografikus lencséket használ és ezeket a lencséket úgy tervezték meg, hogy a generált több dimenziós hologramok megfelelően beilleszkedjenek a valós világba. [10]

4. ábra. A HoloLens szemüveg [11]

Az „Assessing augmented reality in production: remote-assisted maintenance with HoloLens” nevű tanulmányban azt vizsgálták, hogy hogyan valósítható meg egy távoli karbantartás a HoloLens szemüveg, illetve egy okostelefon segítségével. A tanulmány azt is vizsgálta, hogy ez a módszer milyen mértékben segíti elő a karbantartási feladatok elvégzését.

Az adatgyűjtéshez egy autóipari környezetet választottak, ahol az iparban dolgozók segítségével egy valódi 5 tengelyes feldolgozógépen szimulálták a megoldandó hibát.

A gép bizonyos elemeit manipulálták, hogy olyan hibákat idézzenek elő, amelyek az érzékelők elakadását jelentsék. Az egyik ilyen művelet egy sérült érzékelőt szimulált egy forgácsszállító egységnél, míg a másik eset egy eldugult szűrőt szimulált.

Távolról egy szakértőt vontak be élő közvetítésben, aki ezáltal szóban, illetve 3D virtuális megjegyzésekkel támogatta a társait. A kísérletben skype-ot használtak közvetítő alkalmazásként.

Elmondható, hogy a dolgozók mindegyike javulásról számolt be a HoloLens használata során. A dolgozók 67%-a teljes mértékben egyetértett azzal, hogy időt takarítottak meg a rendhagyó eljárással, 83%-a arról számolt be, hogy a hatékonyság javult, és 67%-a szerint a kockázat minimálisra csökkent.

A módszerrel egyszerűbb volt értelmezni a leírásokat és a bonyolult hibákat, valamint a dolgozók magabiztosabbnak érezték magukat a távkommunikáció során.

A HoloLens azok számára is hasznos lehet, akik nem kaptak képesítést vagy oktatást az adott gépre vagy pedig pályakezdők és tapasztalat nélkül kell

megoldaniuk a hibát. A verbális és a vizuális kommunikáció együttesen gyorsabb problémamegoldást biztosít.

Ugyanakkor tapasztaltak akadályokat is, mint például a sérülések kockázatát, mivel a forgó vagy forró elemek balesetveszélyesek, viszont a megfelelő kameraképhez megfelelő helyzetben kell az adott elemet vagy alkatrészt mutatni.

A telefonnal is akadt probléma, mivel a dolgozóknak le kellett tenni, ha minkét kezükre szükség volt, vagy addig a másik dolgozónak tartania kellett. Így összességében a telefon használata 20%-kal megnövelte a probléma megoldásának idejét. [12]

2.5 A 3D Hologram

A hologram egy háromdimenziós vetítési módszer, amely segítségével a létrejött kép bármilyen szögből és pozícióból megtekinthető, ezeket a fénysugarak interferenciája hozza létre és ezek valós fizikai tárgyakat tükröznek. [13]

Kétféle hologram létezik, az egyik a sztereotip hologram, ez számítógépen hozhatóak létre és a HoloLens holografikus szemüveget használják hozzá, ehhez a HoloStudio-ban készíthetünk hozzá tartalmakat.

A másik ilyen a valósághű hologramot, 1947.-ben Gábor Dénes találta fel, aki magyar származású fizikus volt. Amikor kifejlesztette ezt a technológiát akkor nem állt rendelkezésre koherens fényforrás, így 1960-ban a lézer volt az első elérhető fényforrás erre a célra, majd 1971-ben vehette át a Nobel-díjat a találmányáért. [14] [15]

Ez a találmány lehetőséget biztosít, hogy repülés közben egy golyóról vagy nyílról készüljenek hologramok.

Úgy történik, hogy a 3D objektumot lézerrel rögzítik, majd lézerrel megvilágítva, a hologramok képesek pontos 3D képet alkotni az objektumról, és megkettőzni annak jellemzőit.

A hologram pontos megjelenítéséhez a tér egy bizonyos pontján két fényhullámot kell koordinálni a mozgásban – egy referenciahullámot és egy tárgyhullámot. A referenciahullámot közvetlenül a fényforrás hozza létre, és a tárgyhullám visszaverődik a rögzített tárgyról. [17]

5. ábra. A 3D hologram [16]

A hologram technológia is segíthet az ipari környezetben dolgozók oktatásában, segíthet a munkavállalóknak megjeleníteni az útmutatókat a valós idejű munkafolyamat során.

Például egy műszaki rajz vagy egy hibaelhárítási útmutató jelenhet meg. Segíthet részletesen megvizsgálni a termékeket gyártás előtt. Akár a gyártási területen történő baleset esetén segít követni az evakuációs útvonalat.

Emellett a 2017-ben a Verizon és a Korea Telecom próbálták ki a távközlésben az első 5G technológiával a holografikus hívást, a hívásban a felhasználók gesztusait tudta közvetíteni.

A londoni St George's University 2013-ban kipróbálta egy tanórán az emberi test működő szerveit bemutatni hologramok segítségével.

A holográfia és a mesterséges intelligencia lehetővé teszi olyan digitális emberi modellek létrehozását, amit egyre nehezebb lesz megkülönböztetni a valódi modellektől. [17]

A kutatók szerint a holografikus megjelenítés közvetlenebb élményt tud majd biztosítani, mint a VR technológia, mivel a hordozható eszközök hátráltathatják a felhasználót, de a hologram a jövőben hasznos hozzávalója lehet a VR és az AR sisakoknak, emellett a hologramok adattárolásra is képesek lehetnek majd.

Jelenleg a hologramok minősége problémát jelent, mivel a hologramok kis méretben képesek tiszta élményt adni, nagyobb verzióknál még nem elég realisztikus a végeredmény. Kutatások és fejlesztések várnak még erre a területre. [15]

2.6 A 3D kivetítés

Arra használják, hogy a 3D objektumokat megjelenítsenek egy 2D síkfelületen, de 2D és 3D fizikai elem is válhat projektálható felületté.

A valóságban szobrokra, műalkotásokra, épületekre történik a vetítés. A fényjátékok, effektek, animációk kivetítéséhez projektort használnak, ezáltal lenyűgöző látványosságokat biztosítanak. [18]

Emellett az iparban is használják, mivel támogatja a gyártást, képes összetett információkat megjeleníteni az alkatrészekről.

Költséghatékonyan telepíthető és a gyártósoron dolgozók pontos munkautasításokat kapnak arról, hogy hogyan és hol lehet az alkatrészeket összeszerelni, ezáltal növeli a munkafolyamatok átláthatóságát. [19]

6. ábra. 3D kivetítés [20]

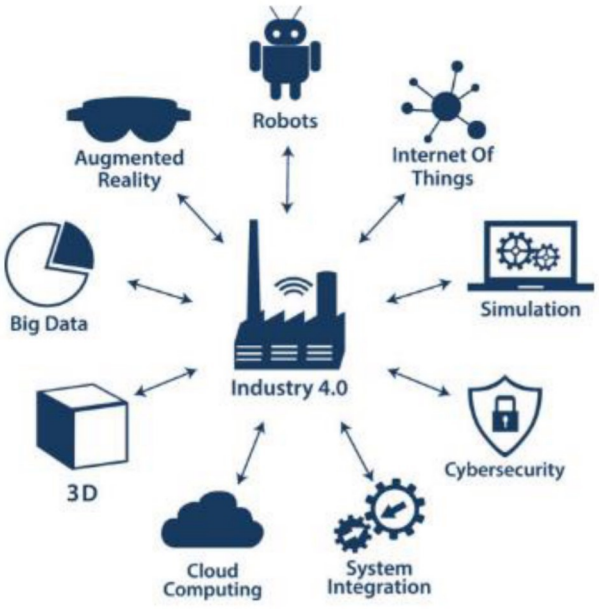
3 A Virtuális valóság ipari alkalmazási lehetőségei és ipari folyamatokban való alkalmazása

3.1 Az ipari folyamatok, képzés és szimuláció

Napjainkban, minden vállalatnak be kell építenie az innovációt a gyártási folyamatába, hogy fenntartható és megfelelő termelési rendszert biztosíthasson, amelyet rugalmasság és alkalmazkodó képesség jellemez.

Az Ipar 1.0 a víz- és gőzenergia, a 2.0 az elektromos energia és a 3.0 a számítási képességek megjelenését jelentette. Az ipar 4.0 kifejezést 2011 novemberében tette közé a német kormány, amely a negyedik ipari forradalomra utal és azóta is használják Európában.

Az Ipar 4.0 több technológiát foglal magába, mint például az adatok digitalizációját, a robotok széleskörű alkalmazását és -automatizációt, valamint a mesterséges intelligencia használatát. Ide tartoznak a ciber fizikai rendszerek, az adatok feldolgozása és -tárolása felhőben, intelligens rendszerek, szimulációk létrehozása, rendszerek összekapcsolása és a virtuális valóság eszközei. [22]



7. ábra. Az ipar 4.0 [21]

Célja az ipari folyamatok hatékonyságának és termelékenységének növelése, illetve jobb minőségű termékek előállítása.

Az új irány előnyök mellett hátrányokkal is rendelkezik, mint például magas kezdeti költségekkel, ami természetesen hosszú távon megtérül.

A VR eszközök viselése pedig sok esetben kényelmetlenséget okozhat, emellett technikai problémák is előfordulhatnak.

Az alkalmazások használata új készségek és ismeretek elsajátítását is igényli a munkavállalóktól, ami időt és erőforrást jelent a cégek számára. A befektetések megtérülését az is mutatja, hogy egyre több cégnél alkalmazzák az ipar 4.0 irányelveit.

Megfigyelhető, hogy a digitalizáció és a vizualizációs folyamat számos lehetőséget biztosít a gyártóknak, hogy innoválják termelési rendszereiket és versenyképesebbek legyenek a piacon. [22]

Az ipari képzésekbe is egyre inkább beépülnek az innovatív módszerek, mivel sokkal egyszerűbb valós interakció által megtanulni valamit, mint könyvekből vagy a hagyományos módszerekkel. A munkavállalók így VR környezetben gyakorolhatják, az új ismereteiket.

A munkavállalók virtuális környezetben próbálhatják ki először az adott feladatot elkerülve ezzel a veszélyes környezetben való munkát, a kezdeti selejt termékek gyártását, a ciklusidő túllépéseket és a termelő sorok beragasztását a tanulási fázis miatt. Ezen felül a dolgozók előre gyakorolhatják a biztonsági protokollokat és a helyes eljárásokat is. [23]

A VR lehetővé teszi a karbantartási és javítási folyamatok virtuális valóságban történő bemutatását és navigációját. Ezáltal a munkavállalók pontosan megismerhetik a berendezések szerkezetét, működését és azokhoz kapcsolódó eljárásokat, amelyek segítik a karbantartókat a gépek vagy javításban és karbantartásában. A virtuális útmutatók segítik a hatékonyabb és pontosabb munkavégzést.

8. ábra. A Virtuális technológia használata az iparban [24]

A VR lehetőséget biztosít a termékek és komponensek virtuális ellenőrzésére és tesztelésére, még a fizikai prototípusok elkészítése előtt, ezzel segíthet a magasabb minőségű termékek előállításában és a költségek csökkentésében.

A gyártósorok és a munkaterületek virtuális modellezése nagyban segíti az strukturális kialakítást, hogy például a megfelelő ergonómiai normák érvényesüljenek, vagy éppen azt optimalizálják. Valamint mozgáselemzések készítésére is használható a VR, amivel a folyamatok hatékonysága növelhető, ezzel lean munkaállomásokat kialakítva.

Megvalósítható ellenőrző állomások létrehozása, ahol a termékek és alkatrészek minőségét és megfelelőségét ellenőrizhetik virtuálisan. Ez hatékonyabbá, gyorsabbá és fenntarthatóvá teszi a minőségellenőrzési folyamatokat. [23]

4 A modellkészítés lehetőségei

4.1 VR fejszett típusok és jellemzőik

A VR tartalom lefuttatásához immerzív eszközökre van szükségünk, a szakdolgozatomban a HTC VIVE PRO nevű fejszettet használtam, amely az egyetemen található és szabadon használható a diákok számára egyetemi projektek keretein belül.

Ez az eszköz rendelkezik két bázisállomással és két kontrollerrel. A két kontroller segítségével a menük között tudunk navigálni, ezekkel a VR környezetekben tudjuk vezérelni az objektumokat, interakciók esetén megfogni és mozgatni az objektumokat. Emellett a teleportálásban is nagy szerepük van, hiszen lehetővé teszik a mozgás irányítását egyik helyről a másikra.

A bázisállomásoknak nagy szerepük van a térbeli követésben és pozícionálásban. Ezek infravörös fényeket bocsátanak ki, amelyeket a fejszetten és a kontrollereken található szenzorok érzékelnek.

A bázisállomásokat a felhasználó magasságánál magasabbra kell elhelyezni, és ajánlott a terem sarkaiba helyezni úgy, hogy ne legyen akadály, ami miatt megszakadhat a kapcsolat.

A HTC VIVE PRO fejszettnek szüksége van egy külső erős számítógépre a működéshez, a csatlakozás pedig kábeleken keresztül történik, ami nagyban megnehezíti a mozgást, mivel a használata közben oda kell figyelnünk a kábelek elkerülésére.

A berendezés nagy felbontású (2880x1600 pixel) AMOLED kijelzőkkel és kiváló minőségű hangzással rendelkezik. A piacon lévő VR eszközök között még mindig kiemelkedő VR élményt biztosít. [25]

9. ábra. HTC VIVE PRO eszköz [26]

Egy másik VR eszközt is említésre méltónak tartok, ez pedig a Meta Quest 2, ami szoros kapcsolatban áll a közismert Facebook Metaverzum-mal, ezért a használata Facebook regisztrációt igényel.

A mesterséges intelligencia és a technológiai áttörések a napjainkban felgyorsultak és lassan több időt töltőnk az online térben, mint a fizikai valóságban.

A Metaverzum egy 3D virtuális tér, ahol még többet lehetünk együtt online a barátainkkal, dolgozhatunk, cikkeket olvashatunk, élő koncertekre mehetünk, sportolhatunk a kanapéból és gazdálkodhatunk a pénzünkkel.

Igaz, hogy fejlesztés alatt ál, de nem tudhatjuk, hogy mit hoz a jövő. Manapság nagyon gyakoriak az élő közvetítések, az online chat és -munkavégzés, ami pár évvel ezelőtt még elképzelhetetlen volt. Ugyanakkor mérlegelnünk kell, hogy mi az a szint ameddig ez az irány egészséges számunkra. [27]

10. ábra. A Meta Quest 2 eszköz [28]

A Meta Quest 2-nek nincs szüksége kábelekre és külső érzékelőkre, mivel ez egy All in One eszköz, így nem korlátozza a felhasználó mozgásterét. Ennek köszönhetően például otthoni edzésekhez is tökéletes választás lehet, hiszen a VR eszköz használata előtt a felhasználónak meg kell rajzolnia a teret, amin belül mozogni fog.

Ha ezt átlépi akkor az eszköz figyelmezteti a felhasználót. Ez lehetővé teszi a biztonságos és akadálymentes mozgást, elmerülve a VR nyújtotta élményekben.

A fejszett viselése komfortosabb lett, és a lencsék egymás közötti távolsága is állítható. A felbontása szemenként 1832x1920 pixel és LCD kijelzővel rendelkezik.

Fontos megemlíteni, hogy a Meta Quest 2-ből 128GB és 256GB-os verzió választható, és akár közvetlenül a készülékre is telepíthetőek alkalmazások és fájlok.

Mindkét eszköz a „6DOF" Six Degrees of Freedom, vagyis hat szabadságfok elvén működik - ami a térbeli mozgás hat fokát jelenti -, lehetővé téve ezzel a szabadon mozgást. [29]

Ez az eszköz minimálisan olcsóbb, mint a HTC Vive Pro, hiszen nem igényel kiegészítő hardvert. A Meta Questhez egy okostelefon szükséges és a Meta Quest applikáció, amelyhez párosítani kell az eszközt.

A választás attól függ, hogy jobb grafikát és magasabb teljesítményt szeretnénk vagy inkább mobilitást és vezeték nélküli független élményt.

4.2 Modellkészítés lehetőségei

Napjainkban a gyártók a termékek tényleges gyártása előtt komoly számításokat és terveket készítenek - csökkentve ezzel a gyártási kockázatot -, és gyakran 3D modelleket is.

Ezek a modellek konfigurálhatóak, szimulációkat lehet rajtuk lefuttatni - akár végeselem szimulációt is - és így a gyártási költségeik jelentősen csökkenthetőek. [30]

A modellkészítésre több lehetőségünk is van, ilyen például a CAD szoftverek, vagyis Solidworks, Fusion 360, Inventor, amelyeket alkatrészek és gépek ipari tervezéshez használnak.

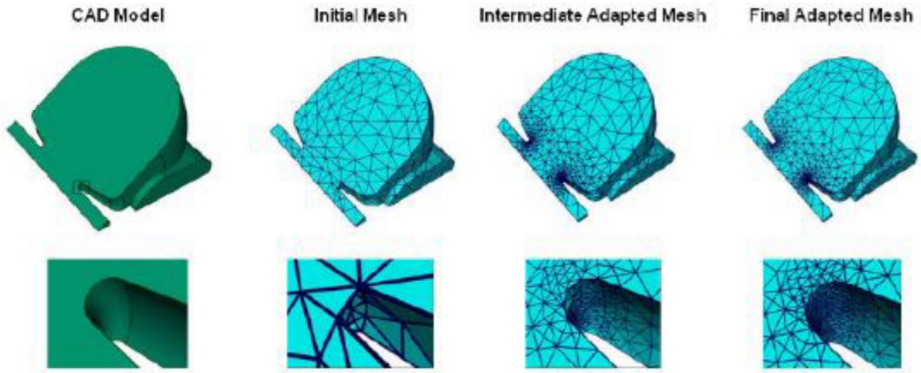
Ahhoz, hogy az elkészített CAD modelleket megfelelően használni tudjuk a VR tartalmakhoz gyakran szükség van Mesh modellé való átalakításra, mivel azok

könnyebben kezelhetőbbek, alacsonyabb felbontásúak, de megőrzik az alapvető geometriai tulajdonságaikat, így nem terhelik annyira a hardvert. Ezek a modellek 3D nyomtatáshoz, animációkhoz és játékfejlesztéshez is nagyon hasznosak.

A CAD modelleket lehet exportálni STL vagy OBJ formátumba, ezek a formátumok jellemzik leginkább a Mesh modelleket. Akár külső konvertáló szoftvereket is lehet használni, de a legtöbb CAD alkalmazás már képes rá ezzel az integrált funkcióval.

Az ábrán látható első zöld modell a CAD modell és mellette a különböző felbontású Mesh modellek kék színnel. Ki lehet választani, hogy milyen részletes legyen a konvertálás, vagyis, hogy milyen pontosan kövesse le az egyszerűsített modell az eredeti geometriát.

Érdemes mérlegelni, hogy milyen sűrűségű hálómodellt generálunk, mivel ennek függvényében változik a hozzá szükséges idő és tárhely is.



11. ábra. Cad és Mesh modell közötti különbség [31]

Továbbá, ha nem akarunk saját modellt létrehozni vagy túl sok idő lenne, használhatunk előre elkészített 3D modelleket is online 3D modell könyvtárakból, ahonnan letölthetjük a kívánt formátumokba a mások által elkészített modelleket. Ilyen 3D modell könyvtárak például a OneShape, Sketchfab, CGTrader és sok más, amelyek számos témában kínálnak modelleket.

Egy másik modellkészítési lehetőség a 3D szkennelés, ami már létező fizikai tárgyak 3D beolvasására szolgál és az adatokat digitális formátumba konvertálja. Ezek a modellek pedig felhasználhatóak számos célra és gyakran alkalmazott különféle iparágakban.

A 3D szkennereknek két fő típusa van, érintéses és érintésmentes. Az érintéses szkennerek fizikai érintéssel vizsgálják az objektumokat, ezalatt az idő alatt az objektumot egy rögzítőelem támasztja alá vagy egy precíziós síkon fekszik.

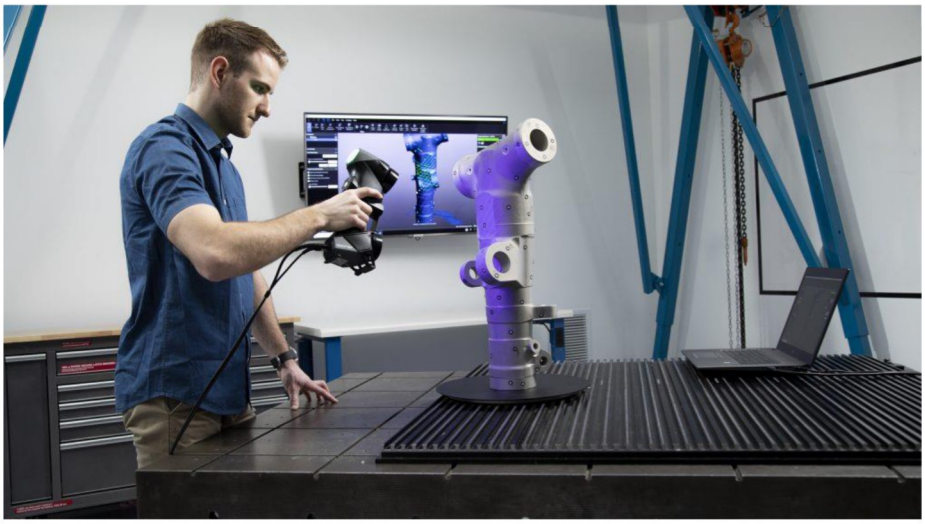
Az érintésmentes szkennerek fehér fénnyel vagy lézervonalakkal világítják meg az objektumot és a kamerák érzékelik a tükröződést és ebből kapnak információkat a tárgy textúrájáról és a geometriájáról. Ezt 3D szögelésnek nevezik, mivel a lézervisszaverődés, a lézersugárzó és a kamera egy háromszöget alkotnak. [32]

Lézer trianguláció alapú, lézer impulzus alapú strukturált fény, fotogrammetria, kontakt alapú, optikai alapú szkennerek. [33]

Gyors és precíz digitális eszköz, ami fény vagy lézerforrás segítségével rögzíti az objektumok alakját, szélességét, mélységét és magasságát és CAD modellekké alakítva azokat. Az x, y, z koordináta rendszert használja alapul és ebben hoz létre adatpontokat.

Hasznos lehet nehézgépeknél, lifteknél, gyártásban lévő alkatrészeknél az újra tervezésben vagy ha az alkatrész eredeti rajza már nem látható.

Továbbá a fogászatban, művészetben, játékokban karakter fejlesztésére és a filmek vizuális effektusaiban is használják, mivel tiszta képet adnak és ezáltal időt és költségeket spórolhatnak vele a felhasználók. [34]



12. ábra. 3D szkennelés [35]

4.3 Animációk készítése

Az animáció készítéshez több lehetőségünk is van, mivel több szoftver is rendelkezésünkre áll. Ha a meglévő modellünkhöz szeretnénk animációt társítani akkor a Blender, Unity, Maya, 3ds Max, Cinemas stb. szoftverek segítségével tehetjük azt meg.

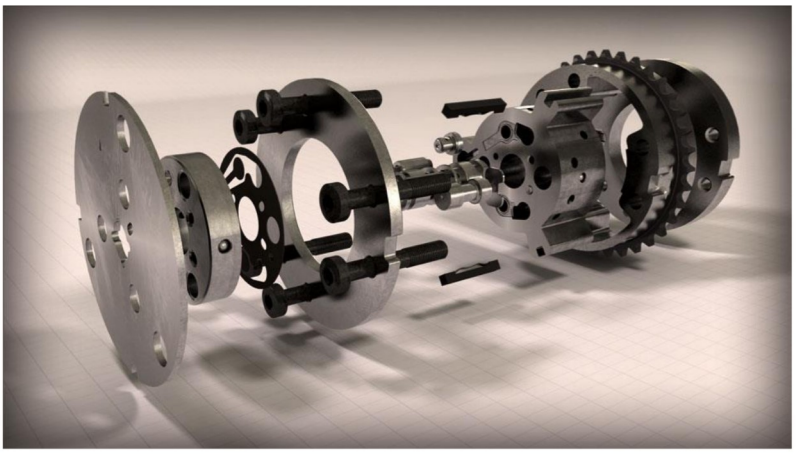
Ezek közül a Blender-t és a Unity-t ismerem és használtam már. A Blender sokoldalú alkalmazás, segítségével létre lehet hozni 3D modelleket is, már meglévő modellhez animációt társítani, textúrázni vagy megvilágítást szimulálni.

A Blender hasonlít egy videóvágó szoftverhez ezért könnyű vele dolgozni, akár kis filmek vágására is alkalmas. Továbbá vizuális effektek létrehozására, modell robbantás animálására is használható, vagy akár tűzijátékot is lehet vele készíteni az animációkhoz vagy filmekhez.

Az animációkat meg leheti valósítani a Unity-ben is és elég egy szoftvert használni hozzá, de Blender-ben is meglehet valósítani, mivel ez a szoftver inkább az animációk készítésére szolgál. A Unity-ben az animációkat lehet kódokból is irányítani, de az Unity inkább egy játékfejlesztő motor.

Arra kell figyelni, hogy az elkészült animációt render-elni kell ahhoz, hogy a kívánt grafikai és minőségi állapotban legyen a modell. Ezt követően exportálni kell FBX formátumba, mivel a Unity ezt a formátumot támogatja leginkább.

Az animáció készítő szoftverekben két irány van, az első a mozgás animáció. A mozgás animáció a tárgyak és a karakterek mozgását, forgását és viselkedését kontrollálja.



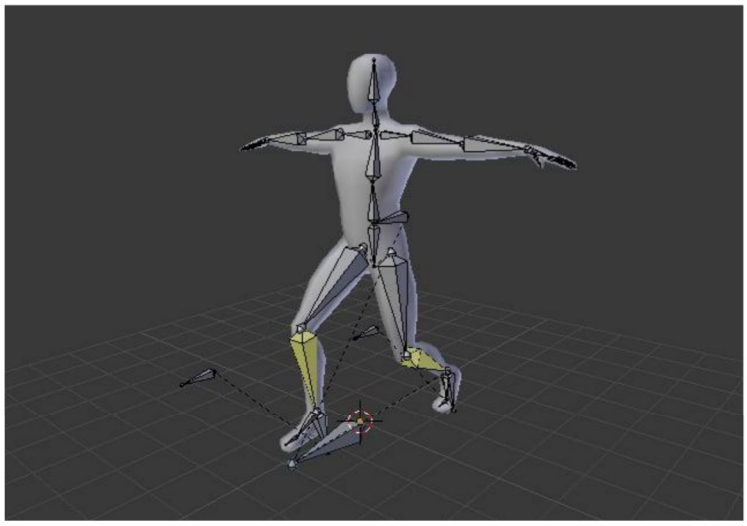
13. ábra. Mozgás animáció [36]

A másik a mechanizmus animáció, ez inkább a karakterek belső szerkezetét használja fel, ezáltal sokkal élethűbbé téve azokat.

Ha készítettünk egy karaktert vagy letöltöttünk egyet, akkor a karakter animációhoz hozzá kell rendelni csontvázolást, ami kulcsfontosságú szerepet játszik a karakter mozgása és viselkedése során. Ezáltal tudjuk vezérelni a karakter animációkat.

A csontvázolás egy hierarchikus csontrendszer, amelyek csontok és ízületekből állnak.

Létre kell hozni a csontvázat és összekapcsolni a karakter részeivel. Ezután a következő lépés, hogy kulcsszámokat adjunk hozzá, amelyek rögzítik a karakter különböző állapotait és pozícióit. Megadhatjuk, hogy egy adott időpillanatban a karakter milyen pozíciót vegyen fel, például, ha szeretnénk bemutatni a karakter segítségével egy tánc koreográfiát.



14. ábra. Csontvázolás [37]

4.4 A Keretrendszer bemutatása

A szakdolgozatomban az Unity videójátékfejlesztő motort használtam, ami az egyik legelterjedtebb és a vezető a játékkészítés és -fejlesztés világában. Az Unity mellett létezik az Unreal Engine, Phaser, Godot Engine stb. platformok.

Azért választottam, mivel logikusan felépített, könnyen kezelhető a rengeteg oktatóanyagnak köszönhetően, ami a témában készült, és mivel felhasználóbarát, gyors és fejlett a grafikája. Mivel több mint 2 millióan használják, ezért amikor egy problémába ütköztem, akkor könnyű volt tájékozódni és megoldást találni a problémára a Unity fórumokon.

Használhatjuk az elkészített kétdimenziós és háromdimenziós játékokat Xbox 360, PlayStation, Windows, MACoS, Iphone, Android szoftvereken és eszközökön, de lehetőséget nyújt interaktív projektek létrehozására is, ilyen például a VR és AR alapú tartalmak készítése. [38]

A Unity weboldaláról az is kiderül, hogy a 2022-es évben a legjobb 1000 mobiljáték 70%-a Unity-vel készült. Ezenkívül, az Unity-t nagyobb és ismertebb gyártók is használják az iparban, mint például az Audi, Mercedes-Benz, Airbus, vagy a Bosch. [39]

Az Unity-t használják repülőszimuláció és járműszimuláció során, az orvoslásban, hogy könnyebben átlátható legyen egy műtét, valamint VR környezetek létrehozásához és az oktatáshoz is.

Ahhoz, hogy létrehozzunk egy projektet szükségünk van egy játékfejlesztő motorra, jelen esetben az Unity lesz az, ahol a játék elemeket össze tudom pakolni és amire emellett szükségünk lesz az pedig a kódok. Programoznunk kell benne és programozás által adunk utasításokat az objektumainknak.

A Unity-ban a legelterjedtebb és a fejlesztők szerint is a legajánlottabb a C sharp programozási nyelv, mivel a dokumentációk és az útmutatók is ezen a nyelven íródtak. Ezt a nyelvet a Microsoft fejlesztette ki, és egy objektum orientált nyelv, ami lehetővé teszi a fejlesztési idő csökkenését, az utasítások egymás utáni végrehajtását.

A kódok megírásához a Microsoft Visual Studio 2022 lesz a segítségemre, ami egy szerkesztői környezet és szorosan együttműködik az Unity-vel.

A Unity-t megnyitva az Editorban vagyis a szerkesztő felületen találjuk magunkat, itt helyezkednek el az alapvető tevékenységekhez szükséges eszközök.

Az ábrán látható fontos területekből áll a kezelőfelület. A Unity Learn weboldalán található egy hasonló ábra a 2021.-es verzióhoz, de azóta az Asset Store vagyis az Áruház, ahonnan a bővítményeket és eszközöket lehet letölteni, elköltözött a Windows menübe.

Először is a Hierarchy ablak arra szolgál, hogy hierarchikusan tudjuk elrendezni az objektumainkat. Ha behívunk egy objektumot akkor ide kerül és itt átlátható lesz. Ha az objektumunk több elemből tevődik össze akkor a fő elemhez alelemeket rendelhetünk, vagyis a gyerekeket hozzáadjuk. [40]

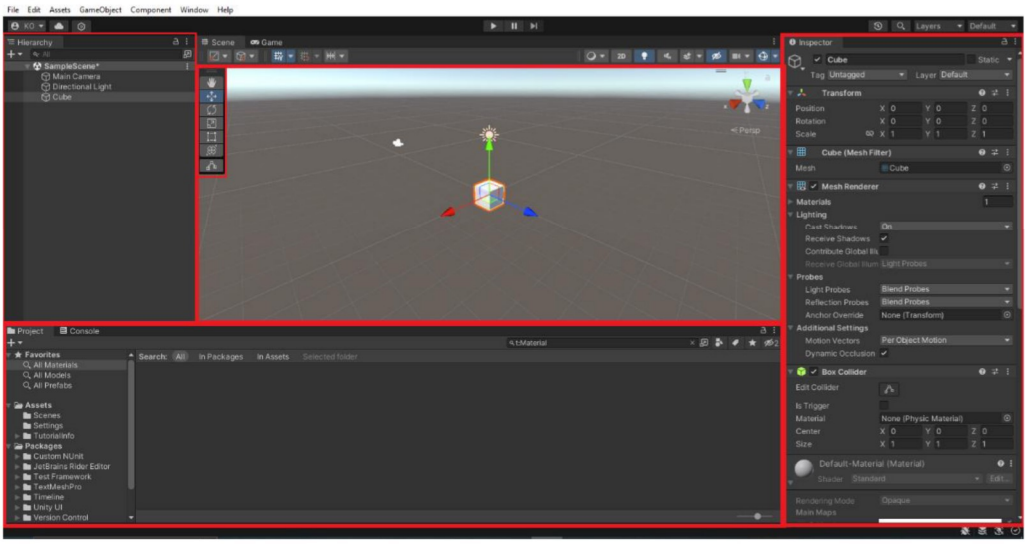
A Toolbar vagyis az eszköztár segítségével a modelljeinket mozgatni, forgatni tudjuk, ezenkívül hozzátartozik a játéknézet elindításához és a leállításához szükséges eszközök a felső sávban. Ha rákattintunk az indítás gombra akkor azt látjuk, amit a játékos fog látni, ilyenkor nem lehet szerkeszteni.

A Scene térben, vagyis a pályán készítjük elő a játéknak az elemeit, ahol összetudjuk rakni őket és szerkeszteni, ebben a nézetben adunk hozzájuk animációkat és interakciókat és mindent, ami szükséges a játék megfelelő

működéséhez. Ahogy előzőleg is említettem a Game vagyis játéktérben pedig játékosként tudjuk megtekinteni azt, amit készítettünk s Színtérben.

A Project ablak úgy működik, mint egy fájlkezelő, rendezett mappákban itt található minden olyan elem, amit használhatunk a projektünkben, mint például zenék, animációk vagy anyagok. Ide kerülnek a letöltött bővítmények mappái is. Ez az ablak abban különbözik a Hierarchy-tól, hogy itt azt is tároljuk, amit esetleg nem fogunk használni a projektben, míg a Hierarchy-ban a projektben jelenlévő aktuális elemek láthatóak. A projekt ablakból könnyedén be tudjuk húzni az elemeket a pályára.

Az Inspector ablakban, vagyis az Ellenőr ablakban az objektumaink méretét, helyzetét, fizikai tulajdonságait módosíthatjuk. [41]



15. ábra. A Unity Editor (Saját készítésű ábra)

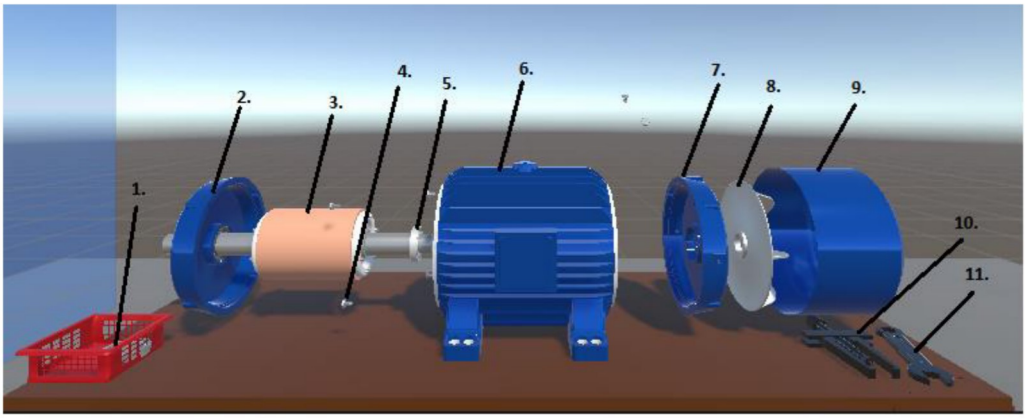
5 Példa folyamat meghatározása és elkészítése

5.1 A feladat meghatározása

Olyan gépet vagy eszközt szerettem volna választani, ami egyszerű, könnyen szerelhető és amelyen jól szemléltethetőek a szerelési folyamatok, illetve egy alkatrész cseréje és a Unity használata megfelelően bemutatható egy gyakorlati feladaton keresztül. Ezért végül egy elektromotorra esett a választásom, amelyet gyakran használnak az iparban.

Mivel edukációs céllal szeretném bemutatni az elektromotor szerelését VR környezetben ezért a szétszerelés és az egyik csapágy cseréjének folyamatáról fogok részletes bemutatást adni.

Egy VR karbantartási tréninget hoztam létre, ahol az iparban dolgozók vagy tanulók a már meglévő elméleti tudásukat a gyakorlatba ültetnék, magabiztosabban felkészülve a valódi szerelésre.



16. ábra. Az elektromotor részei (Saját készítésű ábra)

Egy egyszerűsített elektromotor modellt használtam a VR ipari tartalom létrehozásához.

A projektben a következő elektromotor részeit és eszközöket használtam: 1. csapágy, 2. pajzs, 3. a rotor és a tengely, 4. csavarok, 5. az elhasználódott csapágy, 6. az elektromotor ház, 7. pajzs, 8. ventilátor, 9. burkolat, 10. csapágykihúzó, 11. villáskulcs.

Az elektromotor egy olyan eszköz, ami elektromos energiából mechanikai energiát hoz létre. Az elektromágneses indukció elvére alapul, ez azt jelenti, hogy

elektromos áram hatására mágneses mező jön létre, ami mozgatja a tekercseket a motor belsejében, ezáltal forgómozgást eredményez.

Az elektromotort ipari környezetben használják például elektromos autókban, de gyakran használják a háztartásokban is, mosógépekben, hűtőkben.

5.2 Az Unity projekt inicializálása

A Unity-nak a 2021.3.15f1 verzióját használtam, mivel videókból próbáltam elsajátítani a használatát, ezért egy régebbi verziót választottam, hiszen rengeteg videó és anyag van fent az interneten ehhez a verzióhoz. Fórumokban is könnyebb volt tájékozódni, hiszen a feltett kérdésekre azóta már érkeztek válaszok és a felhasználók is tapasztaltabbak a gyakori hibákat illetően.

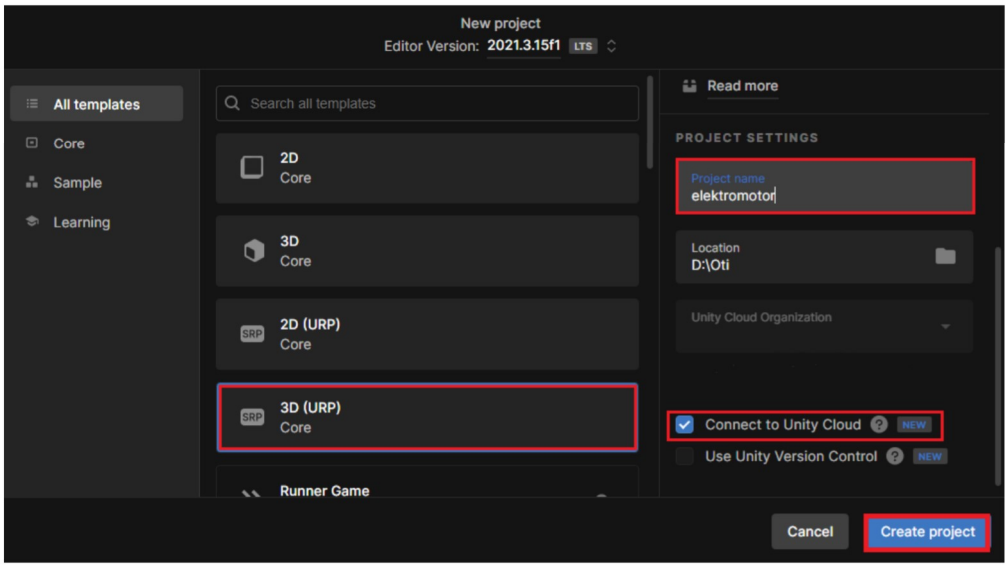
Ahhoz, hogy létrehozzak egy új projektet meg kell nyitnom a Unity Hub-ot, ezt a Unity Technologies fejlesztette ki, ebben lehet kezelni a projekteket a verziókat és a licenszeket tudjuk módosítani. Természetesen előzetesen letöltöttem a Unity.com weboldaláról a Unity-t és ehhez tartozik a Unity Hub is, majd regisztráltam. Mivel hallgató vagyok ezért az egy éves ingyenes diákverziót választottam.

Az új projekt létrehozása során a Unity 3D (URP)-t grafikus motort választottam vagyis Unity Rendering Pipeline-t, ami azt jelenti, hogy Unity renderelési csomópont. Ez jobb teljesítményt és kezelhetőbb grafikus megjelenítést biztosít.

Több Unity motor közül lehet válogatni, attól függ, hogy mi a célunk, kétdimenziós, háromdimenziós grafikus motort, például: Android Core-t is választhatunk játékok és az alkalmazások létrehozásához és fejlesztésére.

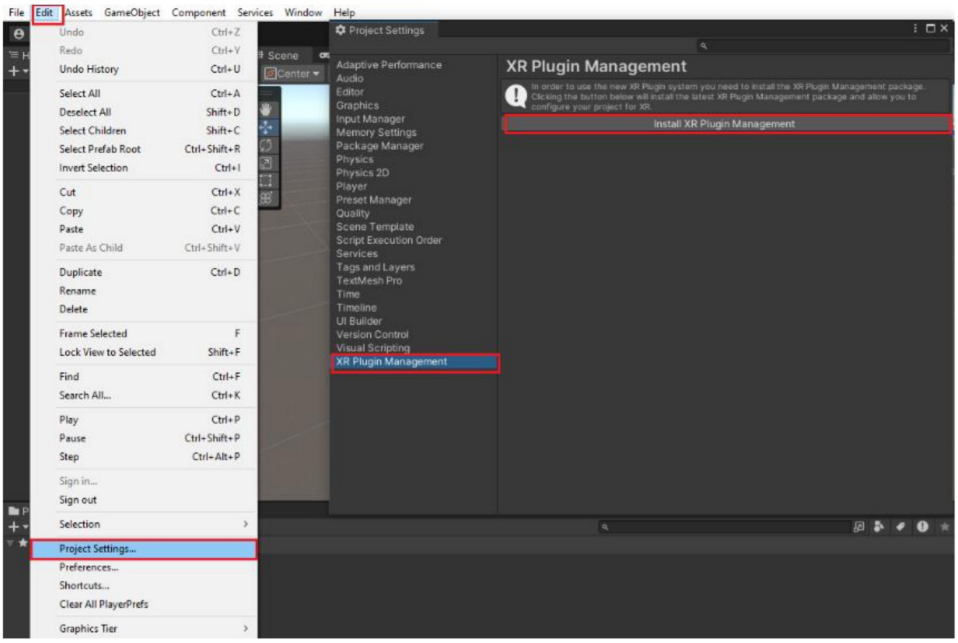
A projektem elneveztem „elektromotornak” és bepipáltam azt, hogy megosztható legyen a Unity Cloudban, ez a felhő alapú megoldás a projektek menedzselését segíti elő.

Ezután csak a Create Project, vagyis a projekt létrehozás parancs következett.



17. ábra. Új Unity projekt létrehozása (Saját készítésű ábra)

Miután betöltötte a Unity a projektet, a következő lépés az volt, hogy VR kompatibilissé kellett tenni, ehhez pedig az Edit menüben, vagyis a szerkesztés menüben a Project Settings-et, vagyis a projekt beállításokat kellett kiválasztanom és azon belül pedig az XR Plugin Managament-et le kellett töltenem, amely lehetővé teszi a VR és az AR platformjainak az integrálását.

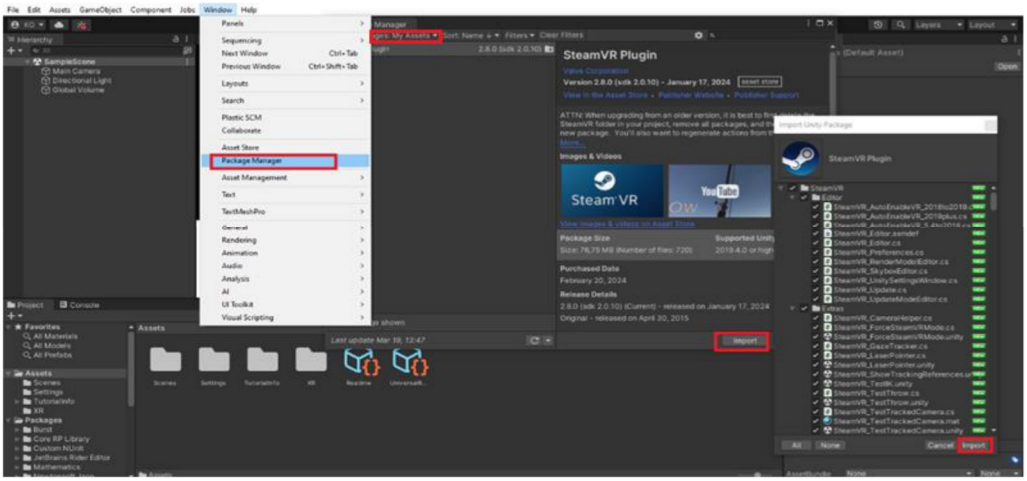


18. ábra. A projekt XR kompatibilissé tétele (Saját készítésű ábra)

Miután ez megtörtént, a Windows, vagyis Ablak menüpontban a Package Manager-t vagyis a csomagkezelőt kell kiválasztani. A My Assets-nél vagyis a saját eszközöknél pedig meg kell keresni a SteamVR Plugin-t és importálni kell. Ez a SteamVR platform lehetőséget biztosít arra, hogy zökkenőmentesen tudjuk kezelni a VR projekteket és futtatni tudjuk a HTC Vive fejszetten.

A működéshez elengedhetetlen a SteamVR letöltése a számítógépre, ezt pedig a hivatalos oldaláról tehető meg, ahol regisztrálni kell egy új fiókot.

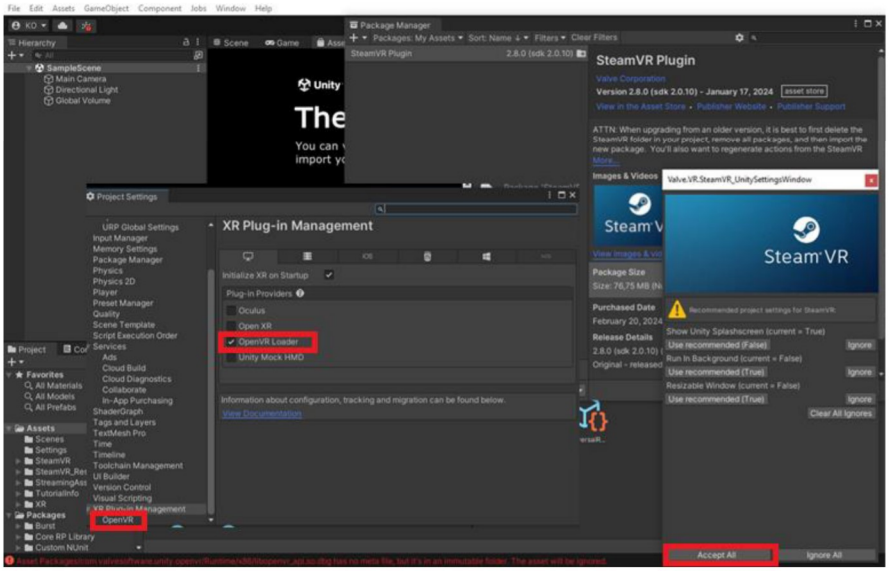
Ha nem található meg a saját eszközöknél, akkor le kell tölteni az Asset Store-ból, ez egy online Unity áruház, ahonnan eszközöket és bővítményeket lehet letölteni. Mivel én már használtam egy másik projektemben így megtalálható volt a saját eszközöknél.



19. ábra. A Steam VR importálása (Saját készítésű ábra)

Sikeres elfogadás után megjelenik az OpenVR, amely egy keretrendszert nyújt és egységesíti a fejlesztést a különféle VR eszközökön, ez SDK-nak vagyis szoftver fejlesztői készletnek felel meg, ami szoftver csomagot tartalmaz.

A szoftver csomagban lehetnek eszközök, kiegészítők, dokumentációk és példa programkódok. Ha be van pipálva az azt jelenti, hogy onnantól kezdve a projektünk VR kompatibilis.



20. ábra. Open VR letöltése (Saját készítésű ábra)

A projektemben használtam egy elengedhetetlen kiegészítőt is, az Auto Hand-et, amely lehetővé teszi, hogy az objektumokkal könnyedén interakcióba lépjünk, mivel jól kezeli az összetett kézmozgásokat, könnyedén megtudjuk fogni az objektumokat és mozgatni tudjuk, emellett a teleportálásban is segítségünkre van és valósághűen szimulálja a kézmozgásokat.

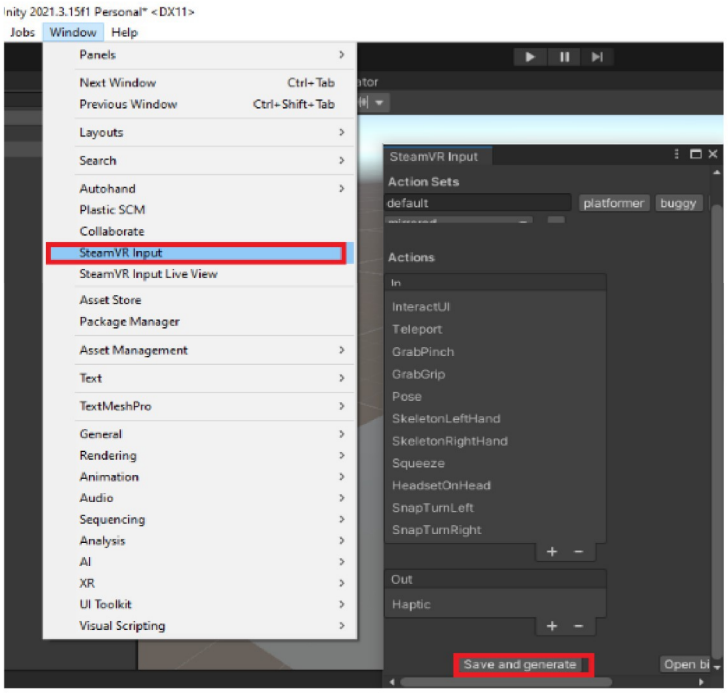
21. ábra. Az Auto Hand [42]

Az Auto Hand kiegészítő segít az interakciók létrehozásának egyszerűsítésében, mivel kevesebb script-et, vagyis kódot kell írni, valamint ez a bővítmény rendelkezik a megfoghatósághoz és a teleportáláshoz szükséges programkódokkal.

Ezáltal felgyorsítja a folyamatokat és azok számára is élvezhetővé formálja a VR tartalomkészítést, akik kevésbé jártasok a programozásban.

Az Auto Hand tartalmaz olyan script-eket, amelyeket elég csak ráhúzni az adott objektumra, hozzáadni a Rigidbody komponenest, ami egy olyan komponens, amely az objektum fizikai tulajdonságait szimulálja, mint például a sebességét, súlyát. Továbbá jól kezeli az ütközéseket és a mozgásokat. A Mesh Collider komponens pedig az ütközés detektálására szolgál, ezt is hozzáadva elérjük azt, hogy az objektum jól reagáljon a környezettel való interakcióra.

Az Auto Hand-et meg kell keresni az Asset Store-ban, onnan kell letölteni és hasonlóan kell eljárni, mint a SteamVR-nál, mivel a kettő együtt kompatibilis, ezért csak új lépésként a Windows menüben a SteamVR inputot kell menteni és generálni, ez pedig a 22.-es ábrán látható.

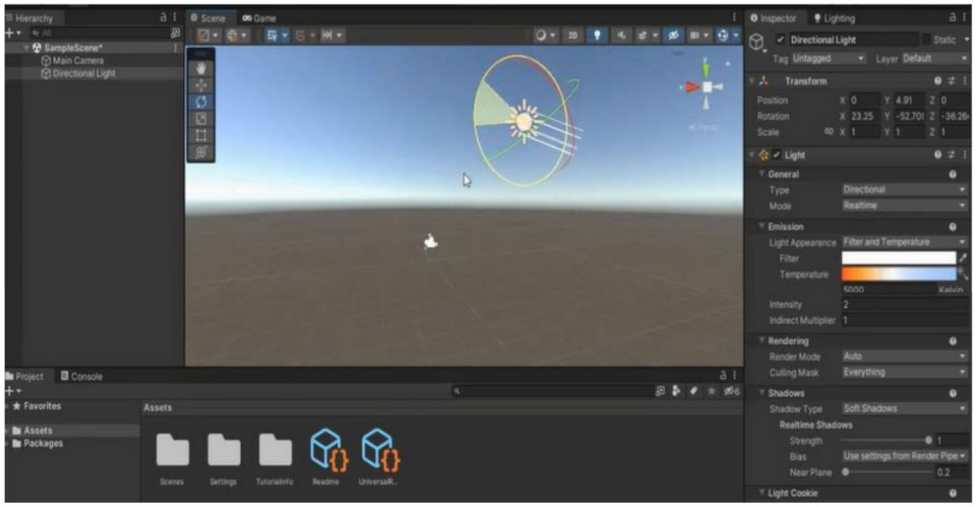


22. ábra. SteamVR Input (Saját készítésű ábra)

5.3 Környezet felépítése, modellek importálása a tulajdonságaik és fizikai viselkedésük meghatározása

Ahhoz, hogy a VR projektünk megfelelően működjön létre kell hozni egy környezetet. Az ábrán látható, hogy van egy Main Camera, vagyis fő kamera, amely a látószöget jelenti, amit a játékos fog látni.

A Directional Light pedig a napot jelenti, ahol be tudjuk állítani, hogy merre süssön, vagy azt, hogy éjszaka legyen.

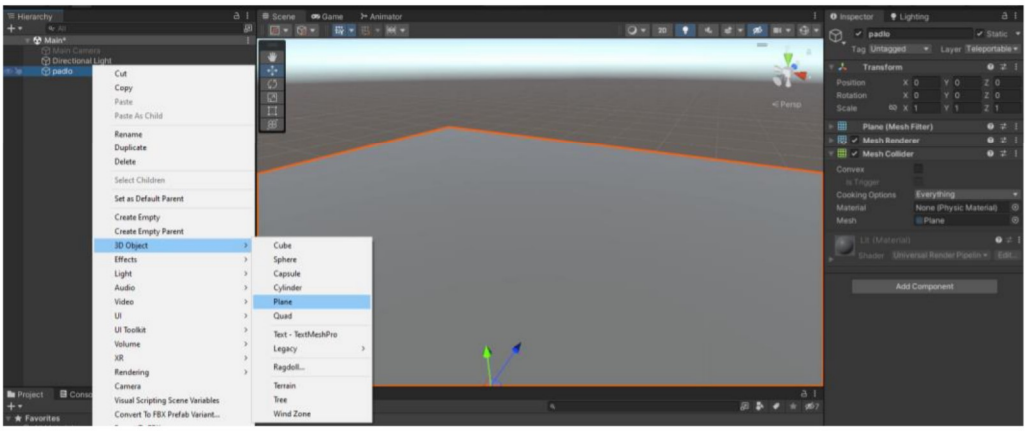


23. ábra. Unity-ban a fények beállítása (Saját készítésű ábra)

A talajt úgy kell létrehozni, hogy jobb egérgombot nyomunk a Hierarchy ablakon belül, majd ki kell választani a 3D Object-et, aztán pedig a Plane-t, vagyis a talajt.

A talaj jobban optimalizáltabb, mint a Cube vagyis a kocka és egyszerűbb vele dolgozni. Bár a kockából is lehet készíteni talajt, csak módosítani kell a méreteit és a pozícióját a Transform menüpontnál, vagyis az átalakításnál, ami a Unity-ben a jobb felső sarokban az Inspector ablakon belül található.

Hozzá lett automatikusan adva a Mesh Renderer, tehát már rendelkezik fizikai tulajdonsággokkal, illetve adtam hozzá Mesh Collider komponenst, az ütközések detektálására.



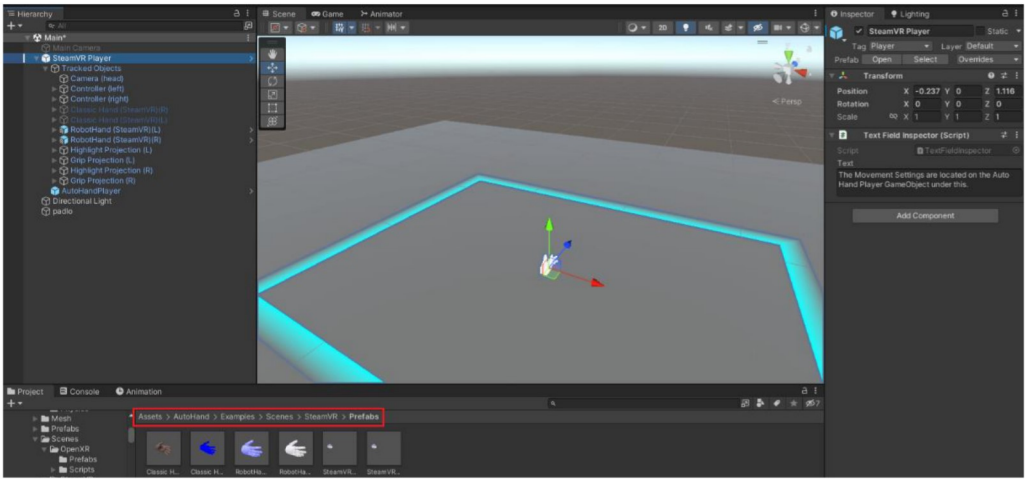
24. ábra. A talaj létrehozása (Saját készítésű ábra)

Hozzá adtam az Auto Hand mappából a SteamVR Player komponenst, vagyis a kezet, ezt csak át kell húzni a Hierarchy ablakba, ez a VR kontrollerem, amit a VR környezetben használok. Én ezt az egyszerű robot kezet választottam. Az elérési útvonal a mappához a 25.-ös ábrán látható.

Ki lehet választani, hogy csak egy kézzel, melyik legyen az kéz, ha eldobjuk akkor milyen erővel dobjuk el az objektumot, az Auto hand segítségével.

A képen látható világos kék színű keretet Tracked Object-nek, vagyis nyomon követett objektumnak nevezik. Ez adja meg azt a teret, amelyen belül mozoghatunk, és ezt a Transform menüben a helyére igazítottam.

Az Auto Hand-nek köszönhetően már ez így is működik, például tudom a kezeket mozgatni a környezetben.

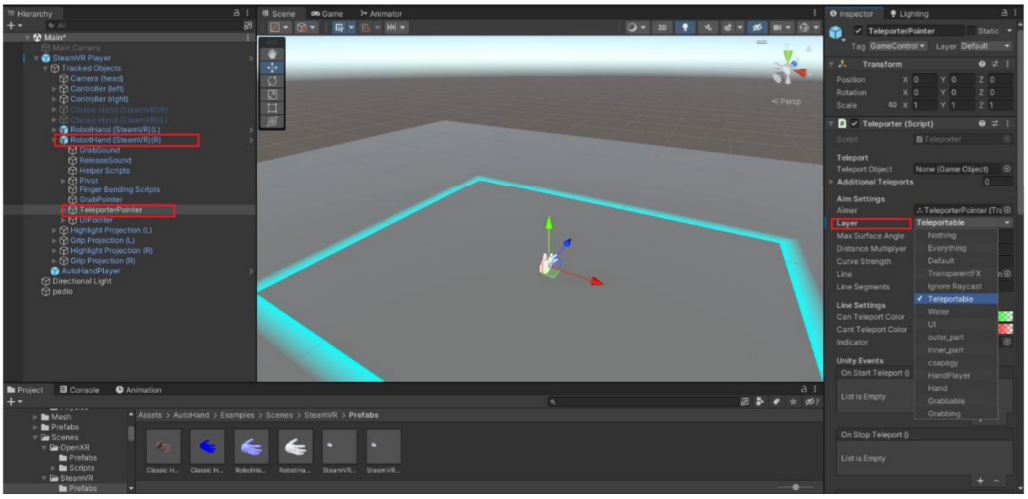


25. ábra. Teleportálás irányítása jobb kézzel (Saját készítésű ábra)

Ha lenyitom a SteamVR Player fület, akkor látszik, hogy hozzá van adva, hogy a jobb és a bal kéznek milyen cselekvések engedélyezettek.

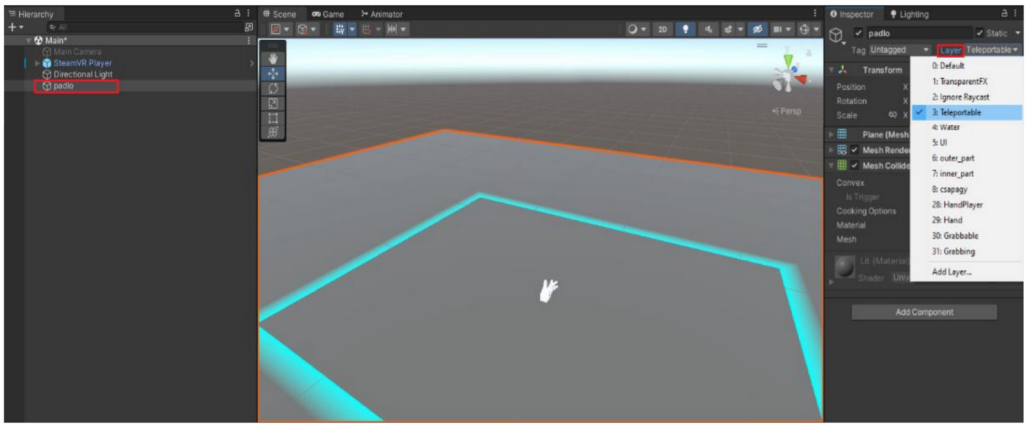
A jobb kézre raktam a Teleporter Pointer-t. A Teleport Pointer egy olyan funkció, amely segítségével a felhasználó rámutat egy célra a környezetben és ezáltal a kijelölt helyre tud teleportálni. A felhasználó megnyomja a jobb kontrolleren a gombot és ezáltal irányítani tudja a teleportációt,

A teleportálásra szükségünk lesz, mivel az elektromotort egy asztalra fogom helyezni, hogy még valósághűbb legyen a környezet és egy valódi szerelőműhely érzését keltse. Ezáltal jobban átélhető a VR élmény, mintha csak a levegőben lennének elhelyezve az elektromotor részei és az eszközök. Az asztal nagy lesz, hogy minden elférhető legyen rajta és az asztalon lévő tárgyak elérése teleportálással lesz elérhető.



26. ábra. Teleportáció helyének meghatározása (Saját készítésű ábra)

Ki kell választani a Teleport Pointer-nél egy Layer-t vagyis réteget és a padlón is ki kell választani. Én a Teleportable layer-t hoztam létre, amelyre azért van szükség, hogy megadja azt a helyet, amelyen tudok járni. Az Auto Hand nélkül rengeteg idő lett volna, így pedig csak pár kattintás az egész.



27. ábra. Layer létrehozása a padlón (Saját készítésű ábra)

A modelljeimet az internetről töltöttem le. Az elektromotort és a piros kosarat a CGTrader-ről, fbx formátumban. Az asztalt, a villáskulcsot, csapágykihúzót pedig a OneShape oldalról szereztem be.

Fontos volt az elektromotor modell kiválasztásánál, hogy minden komponense külön részekre legyen bontva és hogy műszakilag helyes legyen.

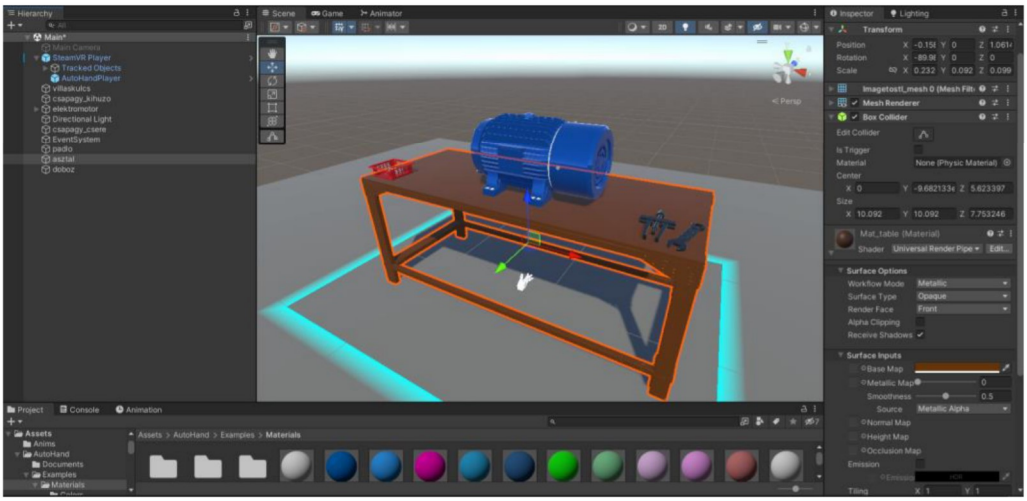
Amit nem tudtam egyből fbx formátumba letölteni azt webes konvertáló segítségével konvertáltam át, mivel ahogy korábban említettem a Unity jól kezeli ezt a formátumot és így csak ezzel dolgoztam.

Lent látható a 28.-as ábrán, hogy minden modellemet behívtam, amit használni fogok a projektben és ezt úgy tettem meg, hogy lenyitottam az Assets menü-t és itt import New Assets-t választottam ki.

A kéz aktiválásával kezdtem, mivel az adott egy viszonyítási pontot és ehhez tudtam igazítani a nagy méretű modelleket.

A Project ablakon belül az Assets-nél a Material vagyis az anyagokon belül tudtam megváltoztatni az objektumok színét és a textúráját.

Az asztalnak adtam egy barna színt, hogy jobban látható legyen. Ugyanígy jártam el a villáskulccsal, csapágykihúzóval és a csere csapággyal is. Módosítottam a méreteiken a Transform menüben, mivel hatalmasak voltak az objektumok.



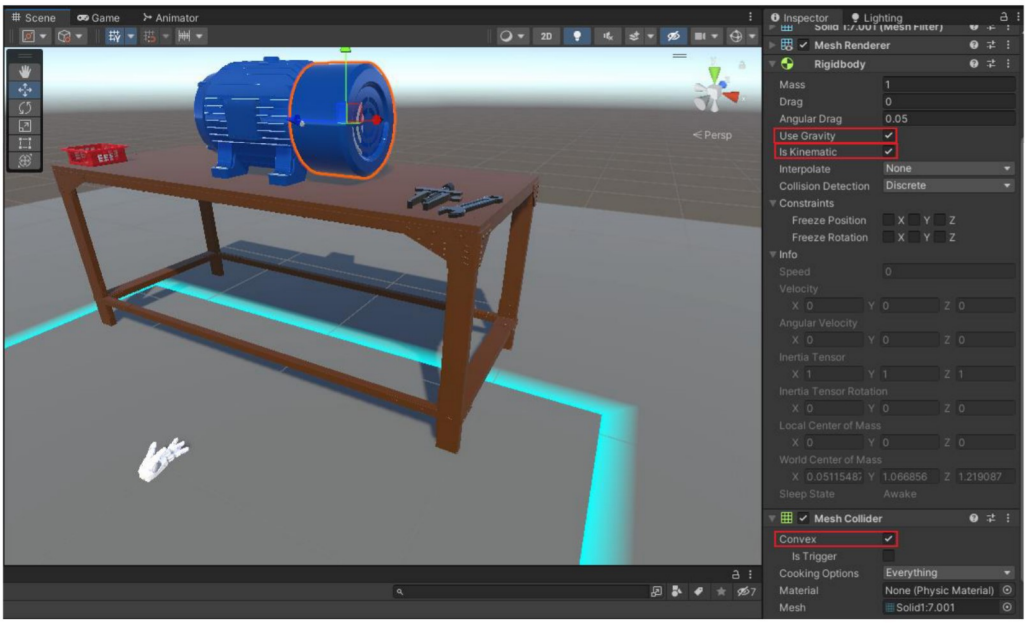
28. ábra. Modellek konfigurálása (Saját készítésű ábra)

Ahhoz, hogy az objektumok megfoghatóak legyenek, fizikai tulajdonságokkal kellett ellátnom őket. Az Inspector ablakban az Add Component-nél lehet megkeresni és hozzáadni a komponenseket.

A Mesh Collider, amely következtében nem tudunk átnyúlni az objektumon és amint említettem detektálja az ütközéseket. Egyedül az asztalhoz és a csere csapágyhoz adtam a Box Collider-t, a Rotorhoz pedig Capsule Collider-t, mivel jobban leköveti az alakjukat.

Amikor Collidert raktam egy objektumra, akkor azon belül be kellett pipálnom, hogy Convex legyen, azért, hogy megfelelően működjön a fizikai szimulációval. Ha ez nem történik meg, akkor hibás viselkedést eredményezhetnek az objektumok.

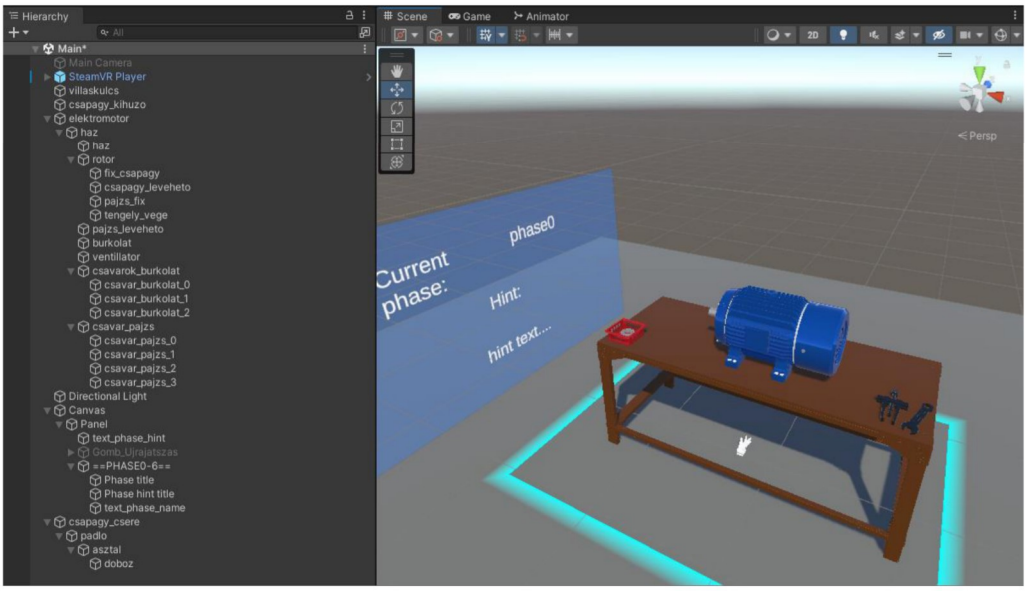
A Rigidbody-n pedig bekapcsoltam az Is Kinematic-ot, ami azt eredményezi, hogy a fizikai motor nem fogja kezelni, hanem saját magam tudom irányítani kódokból. Emellett a Use Gravity-t is bekapcsoltam, ezáltal hatni fog rá a gravitáció. A Mesh Renderer-t pedig már az Unity automatikusan hozzáadta a jobb megjelenítés érdekében.



29. ábra. Az objektumokhoz fizikai tulajdonságok hozzáadása (Saját készítésű ábra)

A logikának megfelelő elrendezésekben hierarchikusan csoportosítottam az objektumokat és a projektben a mappákat rendezetten hoztam létre, hogy minden könnyedén megtalálható legyen. Ezek a mappák a Project ablakban az Assets-en belül láthatóak.

A Hierarchy ablakban pedig látható, hogy az elektromotor a fő objektum és így ehhez parent-eltem, vagyis hozzákapcsoltam a hozzá tartozó részeit, a child, vagyis a gyerek objektumokat. Ezáltal, ha az elektromotort mozgatom akkor vele együtt a child objektumok is mozognak. A csavarokat a helyük alapján neveztem el, annak függvényében, hogy a pajzson vagy a burkolaton helyezkednek el.



30. ábra. Az objektumok csoportosítása (Saját készítésű ábra)

5.4 Folyamatgráf megtervezése és megvalósítása

Az elektromotor szétszereléséhez és a csapágy kicseréléséhez különböző fázisokat hoztam létre. Úgy hoztam létre a szimulációt, hogy kevés hibalehetőséggel keljen szembe találnia magát a felhasználónak, hogyha például egy pályakezdő próbálja ki, akkor neki is siker élménye legyen a használata során. Ezért az alkatrészeket csak a megfelelő sorrendben és a megfelelő eszközzel lehet szétszerelni.

Az egyik fázisból a másikba úgy lehet átlépni, ha sorban teljesülnek a lépések. A phase 0 az alapállapot, ekkor még az elektromotor egyben van, ebben az összes csavart ki kell csavarozni a villáskulcs hozzáérintésével, phase 1 fázisban a burkolatot le kell venni, phase 2-ben a ventilátort kell levenni, phase 3-ban a pajzsot kell levenni, phase 4-es fázisban ki kell venni a rotort a pajzzsal és a csapággyal együtt, a phase 5-ben le kell venni a tengelyen lévő elhasználódott csapágyat a csapágykihúzóval, phase 6 fázisban fel kell tenni az új csapágyat és végül a phase 7-ben megjelenik a panelen a „Újraindítás” gomb.

Csak azután lehet átlépni a következő fázisba, miután megtörtént az adott, jelenlegi fázis.

Erre a célra hoztam létre a Phase Controller nevű script-et, ami a folyamatgráf megtervezéséhez szükséges és az alapja az interakciós lehetőségek létrehozásának a projektemben.

A számozást azért 0-val kezdtem mivel a programozásban is 0-val kezdünk. A programok angol nyelven íródtak, mivel a Unity is angol nyelven működik, ezért próbáltam ehhez tartani magam. Az objektumokat viszont igyekeztem magyarul elnevezni a könnyebb megértés érdekében.

Script-et a Transform menüben az Add Component-nél, vagyis komponens hozzáadásánál New script-et vagyis új programkódot hoztam létre. Ezután betöltött a Visual Studio-t, amiben a későbbiekben dolgoztam.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public enum Phase { phase0, phase1, phase2, phase3, phase4, phase5, phase6, phase7 }
public class PhaseController : MonoBehaviour
{
    public Phase CurrentPhase;
    int boltsUnscrewed = 0;

    private void Update()
    {
        if(CurrentPhase == Phase.phase0 && boltsUnscrewed == 7)
        {
            SetCurrentPhase(Phase.phase1);
        }
    }

    private void Start()
    {
        CurrentPhase = Phase.phase0;
    }

    public Phase GetCurrentPhase()
    {
        return CurrentPhase;
    }

    public void SetCurrentPhase(Phase nextphase)
    {
        CurrentPhase = nextphase;
    }

    public void UnScrewedBolt()
    {
        boltsUnscrewed++;
    }

}
```

31. ábra. A Phase Controller script (Saját készítésű ábra)

Ebben a script-ben tárolom a fázisokat. A Phase Controller script-re épül az egész szimuláció, és ez felügyeli az egész projektet. Az elektromotorra tettem rá, ami egy üres Game Object, ez az elején aktiválódik és ez állandóan fut, mivel az elektromotor elemeit szerelem le ezért a szimulációt az elektromotor tudja felügyelni.

A „public enum Phase” tárolja azokat a fázisokat, amelyek előfordulhatnak a folyamat közben, a „CurrentPhase” pedig segít nekünk abban, hogy jelenleg melyik fázisban vagyunk. A többi script képes lesz ezzel kommunikálni ezeken a metódusokon keresztül.

Le tudja kérni a jelenlegi fázist a „GetCurrentPhase”, be tudja állítani a jelenlegi fázist egy másik fázisra „SetCurrentPhrase”, illetve ellenőrzi, hogy átléphet-e a nulladik fázisból az első fázisba és így tovább.

Ebbe a script-be raktam bele azt is, hogy ellenőrizze, hogy minden csavar ki van-e csavarva.

A Unity script-ek 95%-a Mono Behavior osztályból származik. A script-ek annyiszor fognak önállóan futni ahányszor egy objektumra föltettük.

A Unity-nek van két fontos metódusa. A „Start” metódus egyszer lefut a legeslegelején a 0 fázisban, amint a script aktiválódni tud, ez a metódus csak az Unity-ben van jelen.

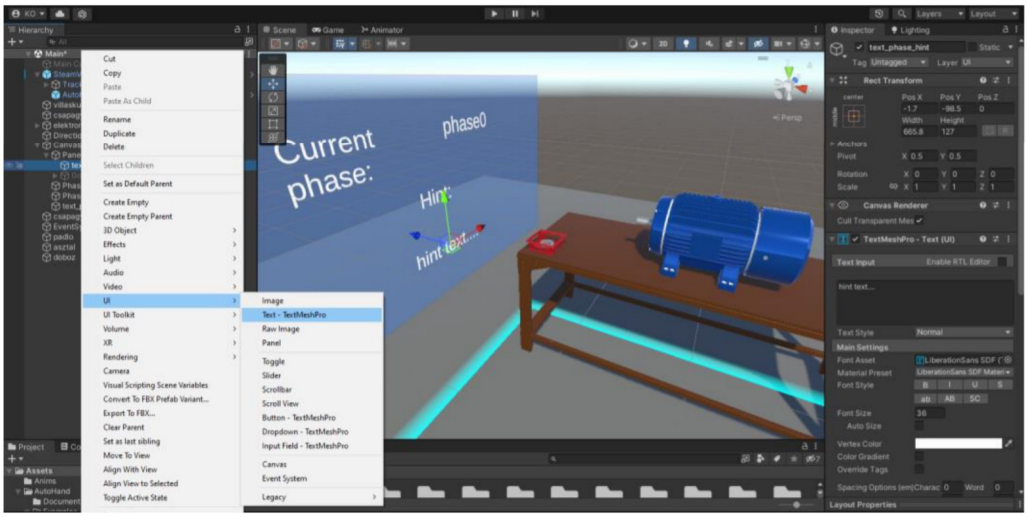
A másik metódus az „Update”, ami automatikusan lefut és nem csak a legelején fut le, hanem akkor is, amikor aktív a GameObject, amelyhez kapcsolva van a script. Lefut minden egyes Frame-ben.

Egy 3D játék számos Frame-ből épül fel. Frame-ben mérik az időt. Például, ha egy játék 30 FPS-el fut az azt jelenti, hogy 30 Frame, vagyis képkocka, fut le. Ez arra jó, hogy állapotokat tudjunk ellenőrizni, mondhatni folyamatosan. Például azt ellenőrzi, hogy levettük-e már a burkolatot, mert akkor átléphetünk a következő fázisba.

Ezt követően egy UI panelt hoztam létre, amely kiírja, hogy hányadik fázisban vagyunk és hogy mi az a lépés, amit teljesíteni kell a felhasználónak. Úgy hoztam létre, hogy a Hierachy ablakban jobb egérgomb megnyomással az UI-re nyomtam, ezután pedig a Panel-t választottam ki.

A Panelt elhelyeztem, de mivel eltakarta a kamerát, ezért beállítottam a World Space opciót a Transform menüben, ezután méretre kellett igazítanom és be kellett állítanom a pozícióját figyelve arra, hogy ne tükröződve legyen a panelen megjelenítve a szöveg, hanem balról jobbra legyen olvasható.

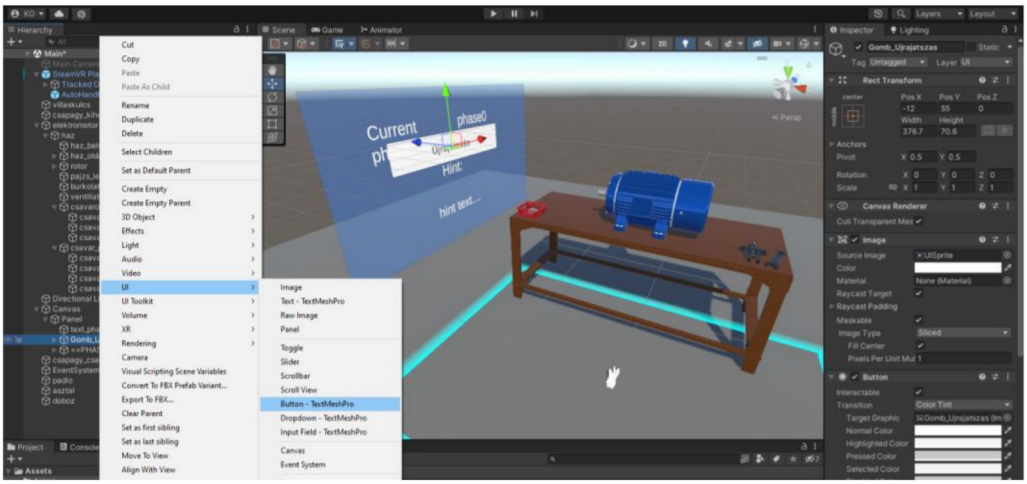
Majd szövegeket adtam hozzá, ami kiírja, hogy hányadik fázist kell követni és annak a lépéseit teljesíteni. A phase0_6_obj üres Game Object-be helyeztem el a panelre kiírt szövegeket, így csoportosítva azokat, ezáltal könnyedén egyszerre kikapcsolható a script-ben a panelre kiírt szöveg.



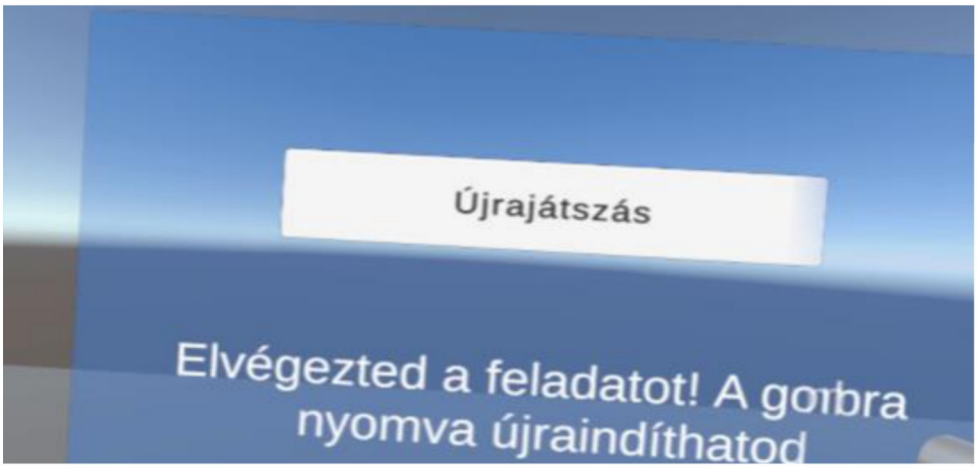
32. ábra. Az UI panel-ra szöveg létrehozása (Saját készítésű ábra)

Miután teljesült az utolsó fázis is, megjelenik a kijelzőn egy újrajátszás gomb, ami az interaktív oktatóanyag újraindítását biztosítja, ezáltal a felhasználó többször is gyakorolhatja a feladatot.

A játék újraindításához az UI menüpontban a Button, vagyis gomb lehetőséget választva hozhatjuk létre. Az „Újrajátszás” gombot a jobb kezünkben lévő kontrollerrel lehet megnyomni és újraindul az oktatóanyag.



33. ábra. Az ,,Újrajátszás gomb” létrehozása (Saját készítésű ábra)



34. ábra. Az ,,Újrajátszás" gomb VR környezetben (Saját készítésű ábra)

A 35. ábrán látható script-et használtam erre a célra, UI néven hoztam létre, majd ezt is az Inspector ablakban az Add Component-nél hozzáadtam a panel objektumhoz.

A „current Phase" a jelenlegi fázist jelenti. A „current hint" pedig az utasításokat jeleníti meg a panelen:„ Csavard ki az összes csavart", „Vedd le a burkolatot", „Vedd le a ventilátort", ,,Vedd ki a rotort a pajzzsal együtt", ,,Vedd le a csapágyat", „Tedd fel a csere csapágyat", Elvégezted a feladatot a gombra nyomva újraindíthatod". illetve a panelen látható az is, hogy hanyadik fázisban vagyunk.

A „switch" megvizsgálja a fázisokat és attól függően, hogy melyik phase-ben jár a felhasználó, az adott utasítás fog lefutni a panelen.

A végén pedig biztosítja a Scene, vagyis a pálya lefutását újra. A „phase0_6_obj" pedig kikapcsolódik és az „Újrajátszás" gomb pedig megjelenik, vagyis aktiválódik.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class UIDisplay : MonoBehaviour
{
    public TextMeshProUGUI currentPhase;
    public TextMeshProUGUI currentHint;
    public GameObject phase0_6_obj;
    public GameObject gomb_ujrajatszas;

    private List<string> hints = new List<string> { "Csavard ki az összes csavart", "Vedd le a burkolatot", "Vedd le a ventillátort", "Vedd le pajzsot", "Vec
    private void Update()
    {
        Phase currentphase = FindObjectOfType<PhaseController>().GetCurrentPhase();
        currentPhase.text = currentphase.ToString();

        switch (currentphase)
        {
            case Phase.phase0: currentHint.text = hints[0]; break;
            case Phase.phase1: currentHint.text = hints[1]; break;
            case Phase.phase2: currentHint.text = hints[2]; break;
            case Phase.phase3: currentHint.text = hints[3]; break;
            case Phase.phase4: currentHint.text = hints[4]; break;
            case Phase.phase5: currentHint.text = hints[5]; break;
            case Phase.phase6: currentHint.text = hints[6]; break;
            case Phase.phase7: currentHint.text = hints[7]; phase0_6_obj.SetActive(false); gomb_ujrajatszas.SetActive(true); break;
        }


    }

    public void Ujrajatszas()
    {
        SceneManager.LoadScene(0);
    }
}
```

35. ábra. Az UI script (Saját készítésű ábra)

5.5 Interakciós lehetőségek

Interakcióra azért van szükség, hogy a felhasználó az objektumokkal interakcióba tudjon lépni.

Ahhoz, hogy ténylegesen megfogható legyen egy objektum az AutoHand -nek a Grabbable script-jét is hozzá kell adni és onnantól kezdve gond nélkül megfogható. Ezt a Transform menüben lehet megtenni az Add Component-nél pedig ki kell keresni.

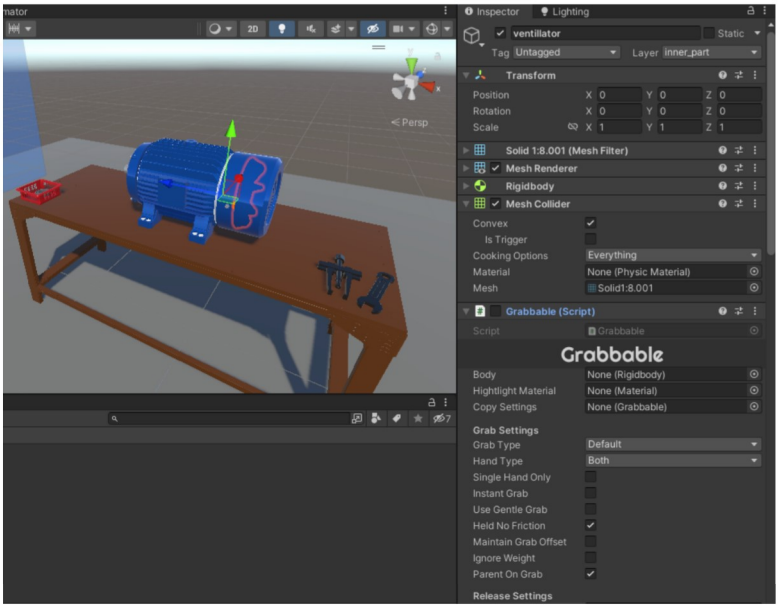
A Grabbable script-et hozzáadtam minden olyan elemhez, amit meg fogunk fogni. Az asztalt és a kosarat nem fogjuk meg, illetve magát az elektromotort sem, ami egy üres Game Object, viszont az elektromotor elemeit igen, mivel azokkal dolgoztam együtt.

Az Auto Hand Grabbable script-je abban különbözik az egyszerű megfogástól, hogy a tárgyaknak a fizikáját érzékeli például a súlyát, a gravitáció hogyan hat rá, hol van az adott objektumnak a határai, ahol ütközni tud, VR környezetben a kezet úgy tudja megformálni, mint ahogyan azt a való életben megfognánk és nem csak azt érzékeli, hogy ott van egy pont vagy ott van valami.

De egyelőre kikapcsoltam, mivel scriptből irányítom, említettem, hogy ehhez be kell pipálni a Rigidbody-n az Is Kinematic-ot és ezáltal kinematikus lesz és scriptből várja az utasításokat.

Az animációnál azért lesz hasznos az IS Kinematic mert ameddig nem indul el az animáció addig meg tudjuk akadályozni, hogy a csavar a helyéről kiessen.

A 36. ábrán látható, hogy az Inspector ablakban jelenik meg a hozzáadott script.



36. ábra. Grabbable script hozzáadása (Saját készítésű ábra)

A folyamatgráf megtervezésénél beszéltem arról, hogy hány folyamatra bontottam a VR projektem.

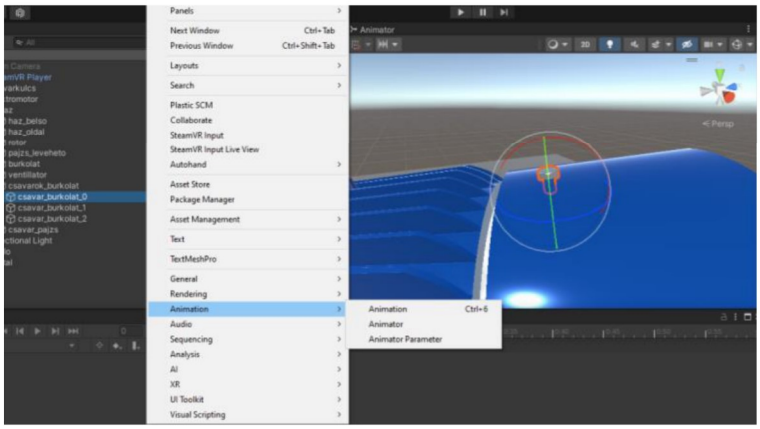
A csavarok kiszedésével kezdtem, ami a villáskulccsal fog történni. A villáskulcs és a csavarok interakcióba lépnek egymással, majd az interakció hatására elindul egy-egy animáció.

Becsavarozott állapotban nem lehet megfogni a csavarokat, nem hat rá a fizika, ha a villáskulcsot hozzáérinti a felhasználó, akkor ki fognak jönni forogva, aztán kiesnek az asztalra és meglehet fogni őket, azután lehet pakolgatni, ha megfogtuk akkor bekapcsolódik a fizikája is.

Az elektromotorról nyolc darab csavart kellett volna leszednem ahhoz, hogy szét tudjam szerelni a modellt, de az egyik nagyon nehezen hozzáférhető helyen van, ezért úgy döntöttem, hogy hét darab csavarral dolgozom.

A Window fült lenyitottam és azon belül az Animációt választottam vagy megnyomom a Ctrl6 billentyű kombinációt. Aztán rá kattintottam a meganimálni kívánt komponensre, majd elneveztem.

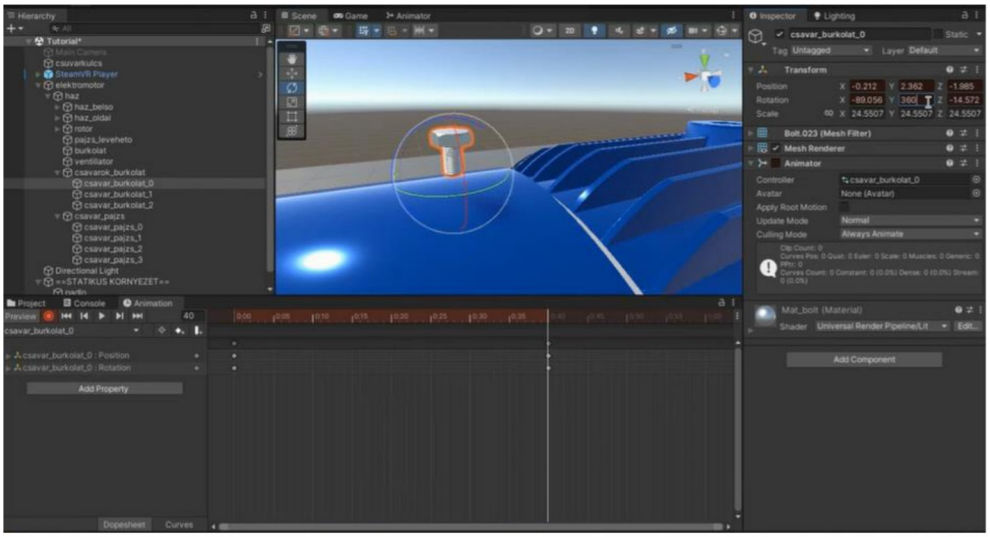
Ahogy már említettem ez olyan, mint egy videóvágó program, a Unity-ben is könnyű az animáció készítése, egyszerű vele dolgozni.



37. ábra. Animáció létrehozása (Saját készítésű ábra)

El kell indítani a piros gombot az Enable/Disable gombot és be kell állitani, hogy az adott milliszekundumig milyen pozícióban legyen a csavar.

Azt akartam, hogy forogjon, ezért én felhúztam a csavart és beállítottam, hogy 40-en milliszekundumig a megadott mozgásokat vegye fel. A Transform menüben beállítottam 360-ra a pozícióját, ezt követően elmentettem a mozgásokat. Ha visszatekerem és lejátszom akkor kijön és forog a csavar. Ezt meg kellett ismételnem hétszer mivel a csavarok máshol helyezkednek el az elektromotoron.



38. ábra. Az Animáció elkészítése (Saját készítésű ábra)

Miután mind a hét csavarhoz hozzárendeltem az animációkat, el kellett készítenem az Animátorban a folyamat ábrát.

Az Animátort a Window fülön az Animation opciónál az Animator lehetőségnél lehet megnyitni és ebben tudjuk kezelni az animációkat.

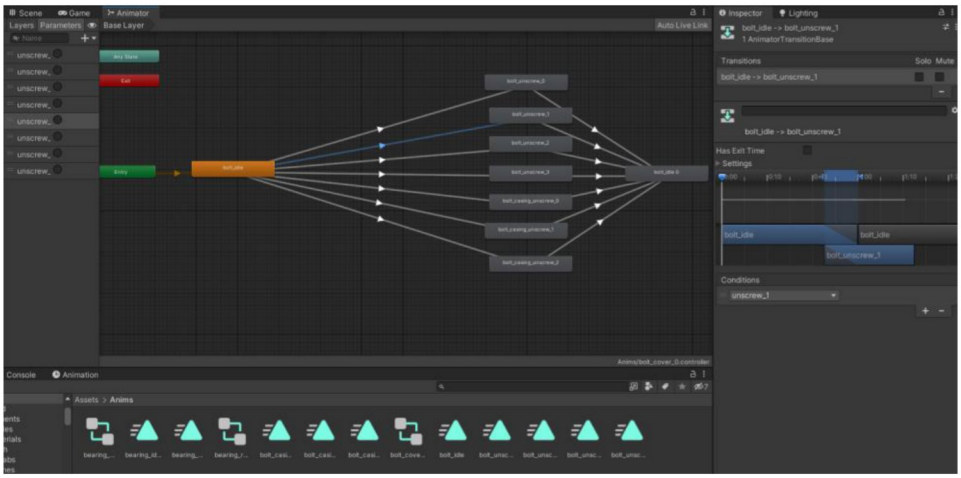
A folyamat ábrán látható a megtervezett logika, különböző feltételekhez és paraméterekhez szabva az animáció lejátszását.

Be tudjuk állítani a nyilak segítségével, hogy mi mihez kapcsolódjon, az átmenetekbe tudunk feltételeket kötni, jobb klikk Make a Transition segít ebben.

Ha rákattintok egy nyílra akkor lent a jobb sarokban látható, hogy mi a feltétele ahhoz, hogy elinduljon az animáció, innentől kezdve scriptek-ből, vagyis kódokból is lehet irányítani.

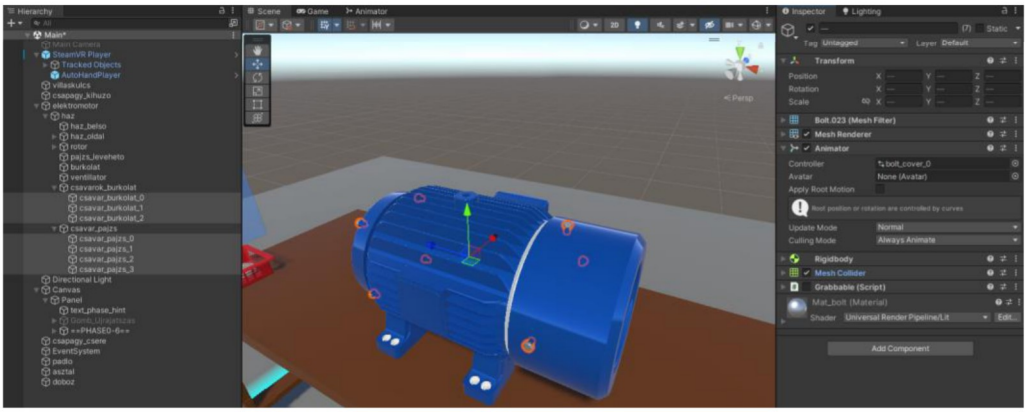
Tehát ha meghívom az Animation Triggername-t, vagyis azt a paramétert, ami beazonosítja az adott animációt és behívja a scriptbe, akkor kódból lehet irányítani az animációt. Például, ha a script-be meghívom az unscruw_1 paramétert, akkor a bolt_unscruw_1 animáció le tud játszódni és így tovább. Ezek a paraméterek a 39.-es ábrán láthatóak a bal oldalon a Parameters-nél.

Létrehoztam egy Entry állapot, ami bemegy egy alapértelmezett animációba, ami egy üres animáció bolt_ idle néven, ez az animáció nem csinál vele semmit, de nem is engedi, hogy történjen vele bármi, ott tartja, de szükséges ahhoz, hogy az Entry-ből ki tudjunk lépni. Miután az animációk lefutottak átlépnek bolt_idle 0 általános állapotba, ahova ki kell léptetni őket.



39. ábra. Bolt Animator Controller (Saját készítésű ábra)

A csavarok függetlenek egymástól, bármelyiket kicsavarozhatjuk a sorrend nem számít, ezért elég egy kódot írni, de nem lehet az animációkat általánosítani.



40. ábra. Animátor komponens

Kijelöltem a csavarokat és rájuk tettem az Add Component-ből az Animator komponenst és ezután bepipáltam és rátettem a Bolt Animátor Controller-t.

A következő script-et használtam hozzá, amit Bolt néven neveztem el, angolul csavart jelent és ezt mindegyik csavarhoz hozzá kell adni.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Autohand;

public class Bolt : MonoBehaviour
{

    public string animationTriggerName;
    public Animator animator;
    private bool isBoltOut = false;

    private void OnCollisionEnter(Collision collision)
    {
        if(collision.collider.name == "villaskulcs" && !isBoltOut && FindObjectOfType<PhaseController>().GetCurrentPhase() == Phase.phase
        {
            isBoltOut = true;
            animator.SetTrigger(animationTriggerName);
            StartCoroutine(ScrewCooldown());
            FindObjectOfType<PhaseController>().UnScrewedBolt();
        }
    }

    IEnumerator ScrewCooldown()
    {
        yield return new WaitForSeconds(1f);
        animator.enabled = false;
        gameObject.GetComponent<Grabbable>().enabled = true;

    }
}
```

41. ábra. Bolt script (Saját készítésű ábra)

„OnCollisionEnter” metódus, azt jelenti, ha összeütközik a csavar a villáskulccsal akkor meghívódik egy metódus és ellenőrzi, hogy mivel ütközött össze, ha ténylegesen a villáskulcs volt az, amivel összeütközött, akkor még ellenőrizni kell a többi feltételt is, amik tovább engedik ebben az elágazásban. Az

egyik ilyen az „isBoltOut” kivan e már csavarva az adott csavar, másik ilyen feltétel pedig, hogy a phase 0 fázisban vagyunk, ebben a Phase Controller script segít.

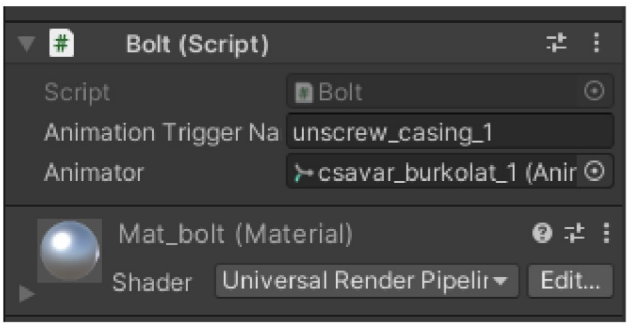
A „Set.Trigger” beállítja a csavaroknak az „animationTriggerName”-t, annak függvényében, hogy melyik csavart érintettük meg, ezáltal a megfelelő animáció fog lejátszódni.

A Unity-ben az „IEnumerator” annyit csinál, hogy egy külön szálon engedi futtatni tovább a kódot, így ezáltal nem akad el a program és így tud tovább működni megfelelően.

A „WaitForSeconds”-ra azért van szükség mert 40 milliszekundumig tart az animáció, de ennek segítségével egy másodpercet tudunk várni, hogy a csavaroknak legyen idejük leesni az asztalra. Tehát ameddig az animáció le nem játszódik legyenek megfoghatatlanok a csavarok.

Ezután az „animator. enabled” kikapcsolódik és az animációt nem engedi tovább játszani.

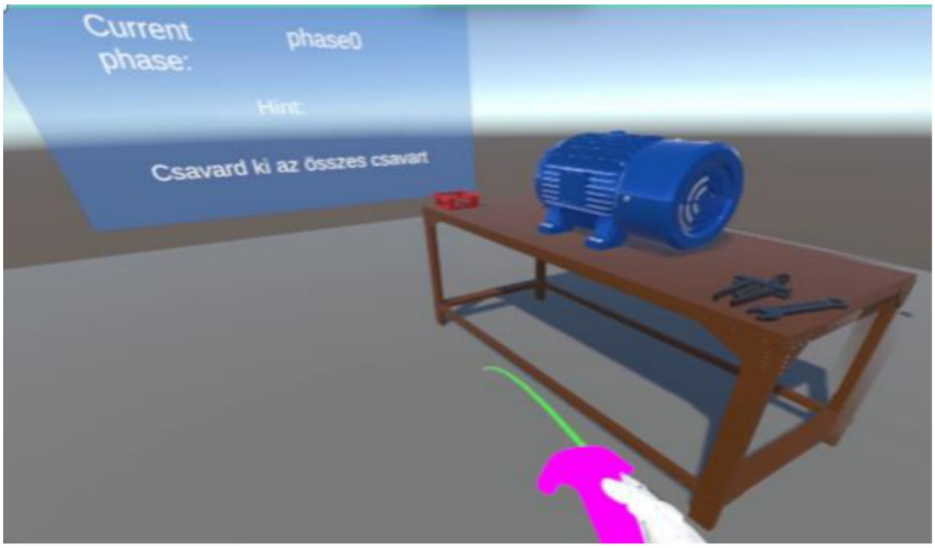
A „Grabbable” bekapcsolódik rajta és megfoghatóak lesznek a csavarok és lehet pakolgatni őket.



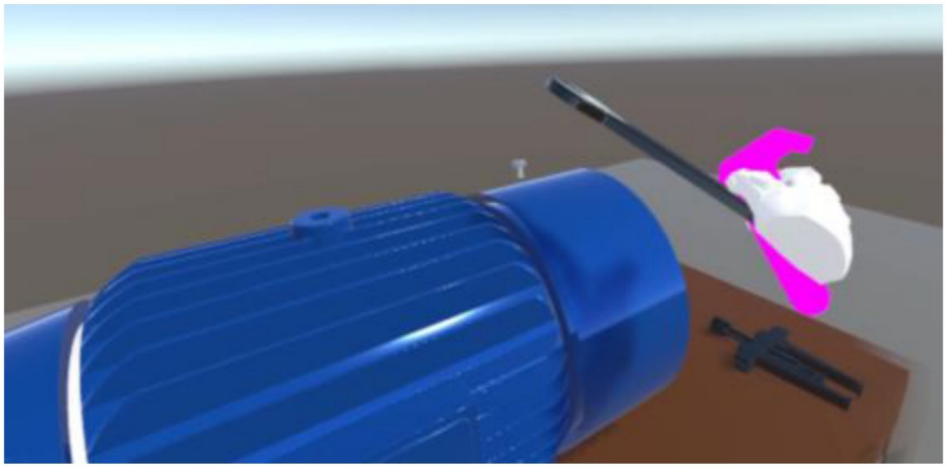
42. ábra. A csavarok lejátszásához szükséges feltételek (Saját készítésű ábra)

Miután a Bolt script-et hozzáadtam mindegyik csavarhoz, ezt követően a Bolt script-en belül hozzá kellett rendelni egyenként az Animation Trigger Name-t vagyis az animációk paramétereit.

Az Animator-hoz pedig a megfelelő csavart adtam hozzá. Figyelmesen kell eljárni, látható, hogy a script ugyanaz, de a csavar folyamatok teljesen másak.



43. ábra. A panelra írt utasítás követése VR környezetben (Saját készítésű ábra)



44. ábra. A csavarok kiszedése villáskulcs segítségével VR környezetben (Saját készítésű ábra)

Ezt követően a phase 1, phase 2, phase 3, phase 4 fázisokban egyszerűbb dolgom volt. Innentől kezdve általánosíthatóak a fázisok, ehhez pedig létrehoztam a „Part” scriptet, vagyis részekre bontást. Levettem a burkolatot, a ventilátort, a pajzsot és kivettem a rotort a másik pajzzsal és a csapággyal együtt.

Ehhez pedig a következő script-et használtam:

```
using Autohand;

public class Part : MonoBehaviour
{
    public Phase requiredPhase;
    public Phase nextPhase;

    private void Update()
    {
        if(FindObjectOfType<PhaseController>().GetCurrentPhase() == requiredPhase)
        {
            gameObject.GetComponent<Grabbable>().enabled = true;
        }
    }

    public void Grabbed()
    {
        if (FindObjectOfType<PhaseController>().GetCurrentPhase() == requiredPhase)
        {

            gameObject.GetComponent<Rigidbody>().isKinematic = false;

            FindObjectOfType<PhaseController>().SetCurrentPhase(nextPhase);
        }
    }
}
```

45. ábra., Part" script (Saját készítésű ábra)

Ameddig nem történt meg a 0-dik fázis, addig nem tudunk belépni az 1-es fázisba, addig nem lesz levehető a burkolat, de amint kicsavartuk a csavarokat, automatikusan átlép a következő fázisba és így kell tovább haladni egészen a 4. fázisig.

Itt is jelen van az „Update" metódus, folyamatosan lefut ezért különböző állapotokat vizsgálhatunk meg vele.

Ebben ellenőrzi, hogy ez az a fázis, amit megadtunk a „requiredPhase" -nél. A „nextPhase" pedig a következő fázist jelenti és itt is szükségünk van a „Phase Controller" script-re, mivel lekéri a jelenlegi fázist.

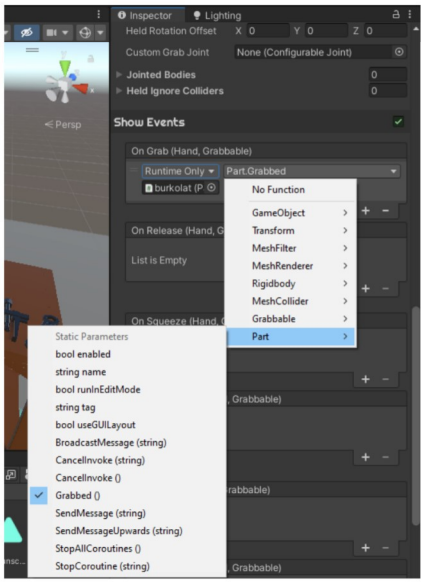
Aztán megnézi, hogy ez a jelenlegi fázis és hogy egyenlő-e azzal, amit megadtam fázisként és ha igen akkor a „Grabbable" komponensét bekapcsolja és onnantól kezdve meglehet fogni az adott objektumot.

A megfogás után, meghívódik a „Grabbed" metódus, ahol megfogható, de nem szabad, hogy másik fázisba ugorjon és újra ellenőrzi, hogy tényleg abban a fázisban vagyunk, amiben kell.

A „Rigidbody" komponensen kikapcsolja az „Is Kinematic"-ot, ezután lehet pakolgatni az elektromotor részeit, ezt követően átlép a következő fázisba.

A „Grabbable"-nek van egy eseménye, amit létre kell hozni és az 1-es fázistól a 4-es fázisig alkalmazni kell. A burkolatnál, a ventilátornál, a pajzsnál és a rotor kiszedésénél, a Part script-et rátesszük ezekre a részekre és a Grabbed metódusát

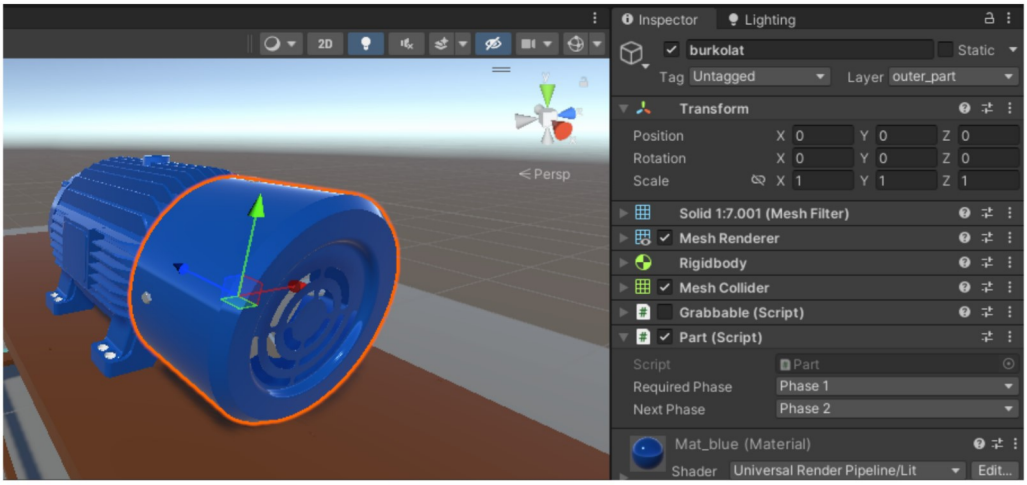
kiválasztjuk, ez csak játékidőben jelenik meg, erre azért van szükség mert ez detektálja, hogy az elemet megfogtuk e már.



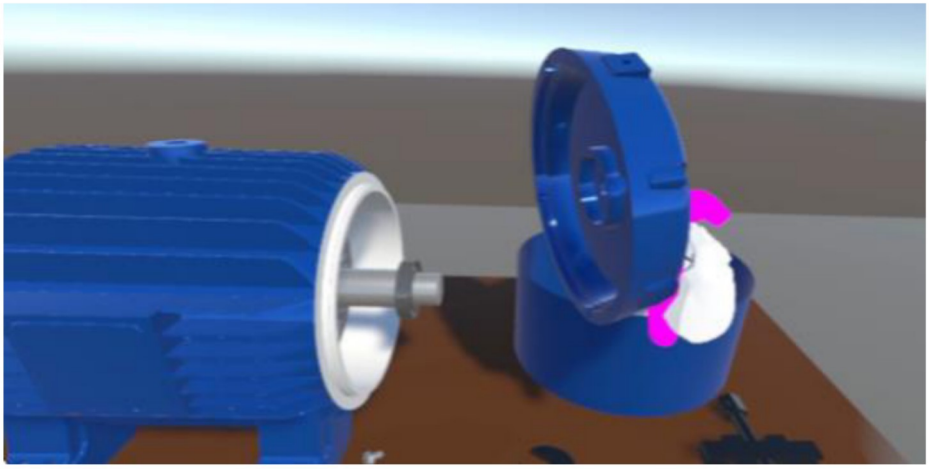
46. ábra. Esemény létrehozása (Saját készítésű ábra)

Az ábrán látható, hogy a Part script-nél a „Required Phase” -nél meg kell adni, hogy melyik fázisban szeretnénk ahhoz lenni, hogy interakcióba tudjunk vele lépni és a „Next Phase” -nél pedig meg kell adni, hogy hányas fázisba lépjen át ezután.

A 47. ábrán látható esetben, le kell venni a burkolatot ehhez a phase 1.-ben kell lenni és a következő fázis a phase 2, amelyben a ventilátort kell levenni.



47. ábra. A fázisok átlépéséhez szükséges feltételek megadása (Saját készítésű ábra)

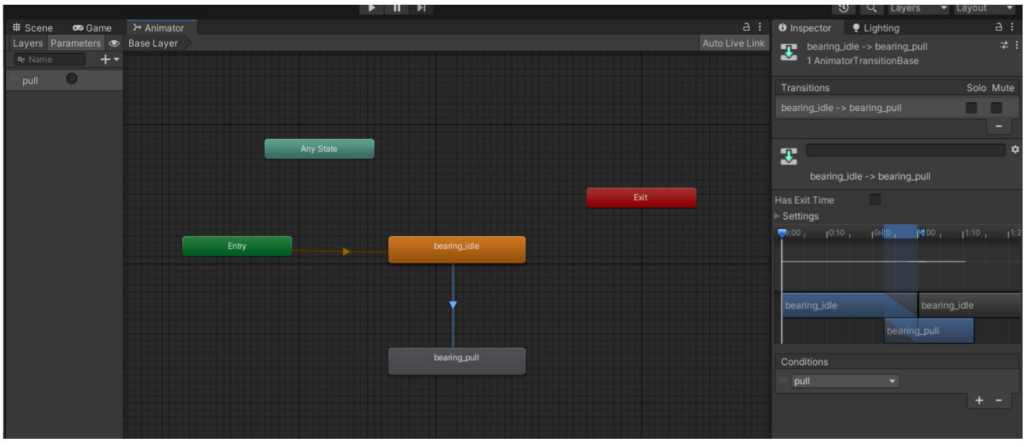


48. ábra. Az elektromotor részeinek leszedése VR környezetben (Saját készítésű ábra)

Az 5-dik fázisban az elhasználódott csapágyat kell levenni. A csapágyhoz készítettem egy animációt, amiben le kell venni a csapágyat a csapágylehúzóval és ahogy vége az animációnak a csapágy leesik az asztalra.

A csapágynál ugyanúgy jártam el, mint a csavaroknál az animáció létrehozásánál, megadtam, hogy az adott milisecumdum-nál milyen pozícióban legyen a csavar. Az animációnak a bearing_pull, vagyis csapágy kihúzás nevet adtam, az animáció paraméter neve pedig „pull” mint húzás.

Itt is van egy Entry bemeneti állapot és az bearing_idle itt is azt jelenti, hogy tétlen állapot.



49. ábra. bearing_pull Animator Controller (Saját készítésű ábra)

Ahhoz, hogy a csapágykihúzó interakcióba lépjen a csapággyal egy script-et kellett létrehoznom. Ezt a script-et elneveztem Bearing-nek vagyis csapágynak, ehhez a script-hez hozzá kell adni az animátort, ez lesz a referencia.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Autohand;

public class Bearing : MonoBehaviour
{
    public Animator animator;
    private bool isBearingOut = false;


    private void OnCollisionEnter(Collision collision)
    {
        if(collision.collider.name == "csapagy_kihuzo" && FindObjectOfType<PhaseController>().GetCurrentPhase()==Phase.phase5 && !isBearingOut)
        {
            isBearingOut = true;
            StartCoroutine(StartPull());
        }
    }
    IEnumerator StartPull()
    {
        animator.SetTrigger("pull");
        yield return new WaitForSeconds(1f);
        animator.enabled = false;
        transform.SetParent(null, true);
        gameObject.GetComponent<Rigidbody>().isKinematic = false;
        gameObject.GetComponent<Grabbable>().enabled = true;
        FindObjectOfType<PhaseController>().SetCurrentPhase(Phase.phase6);
    }
}
```

50.ábra. Bearing script (Saját készítésű ábra)

„isBearingOut”-ban tároljuk azt, hogy lejött-e már a csapágy, ha még alapesetben van akkor hamis, ha kicsavarodott akkor igaz lesz.

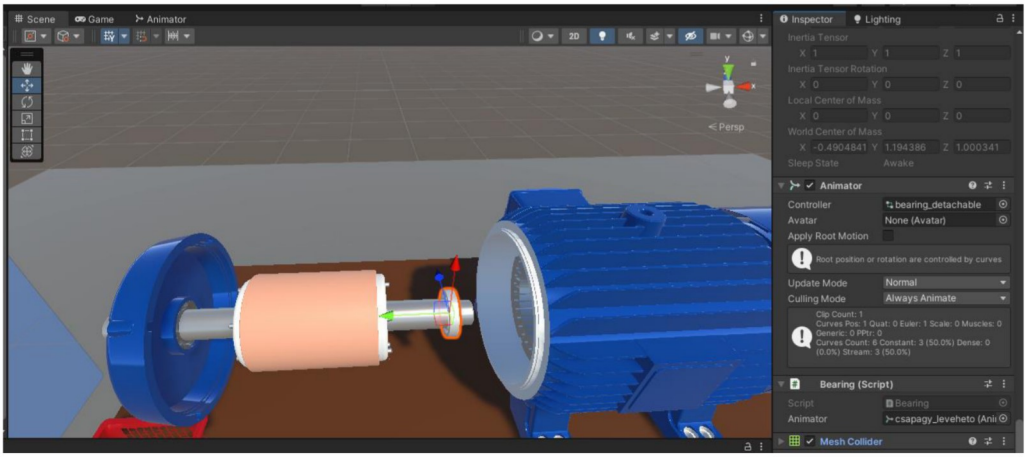
Az „OnCollisionEnter”, ami itt is azt vizsgálja, hogy mivel ütközik a csapágy, ha a csapágykihúzóval ütközött, akkor lekérdezi hanyadik fázisban vagyunk a „PhaseController” script segítségével, ehhez az 5-dik fázisban kell lenni és az „isBearingOut”-nak hamisnak kell lenni. Ha nem teljesülnek ezek a feltételek akkor nem fut tovább a script.

Ha lefut, akkor az „isBearingOut”-nak igaznak kell lenni, mivel csak egyszer futhat le és többé már nem lehet hamis.

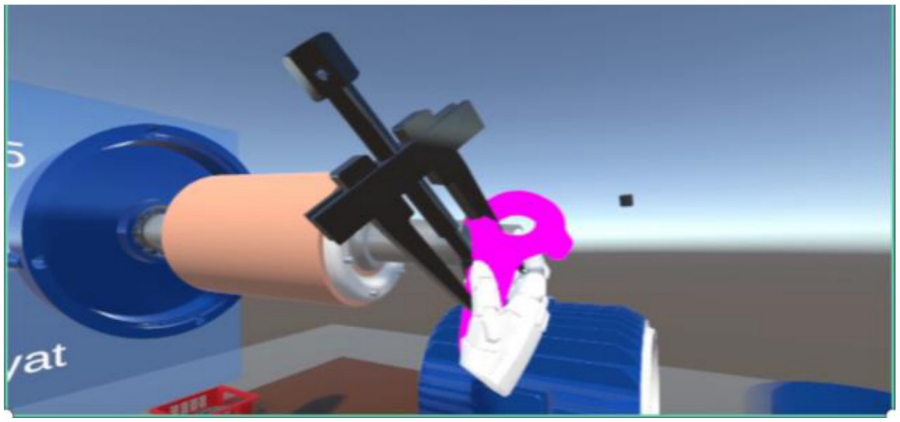
A „WaitForSeconds” segítségével van időnk késleltetni a futást és ezáltal tudunk várni egy másodpercet, hogy befejeződjön az animáció. Ezt követően az animátor kikapcsol.

Azután a „Rigidbody” -ban az „Is kinematic” ki lesz kapcsolva és így le tud esni az asztalra, ezt követően bekapcsolódik a „Grabbable” és ezáltal megtudjuk fogni a csapágyat, ezt követően átugrunk a 6-os fázisba.

Létrehoztam egy animátor komponenst, amihez hozzáadtam az animátor controller-jét, illetve hozzáadtam még a csapágyhoz a „Bearing” script-et és ehhez hozzáadtam a csapágy animációját.



51. ábra. Az elhasználódott csapágy levétele (Saját készítésű ábra)



52. ábra. Az elhasználódott csapágy levétele VR környezetben (Saját készítésű ábra)

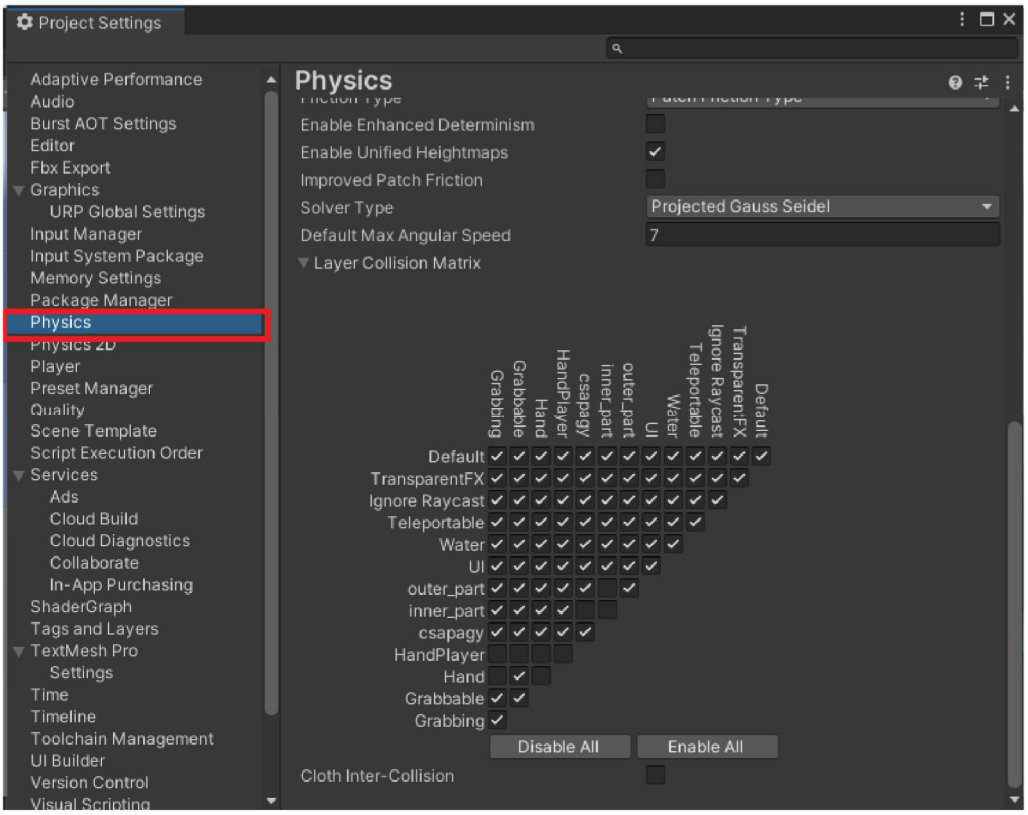
Létrehoztam egy Csapágy nevű layer-t, ezt ugyanúgy hoztam létre, mint a padlónál, erre tettem rá a két csapágyat, mivel nem akartam, hogy az asztallal ütközzenek.

Az Inner part layer-en a rotor a pajzzsal, másik pajzs, ventilátor van rajta, az Outer part layer-en a burkolat, a többi objektum Default layer-en van.

A Layer-ek arra jók, hogy csoportosítani tudjuk az objektumokat és ezáltal kezelni tudjuk, hogy az objektumok milyen műveleteket hajthatnak végre egymással.

Project Settings-en belül a Physics-nél látható egy mátrix, amiben láthatóak a Layer-ek vagyis a rétegek. Itt az interakciókat lehet korlátozni vagy hagyni, hogy szabadon kölcsönhatásba lépjenek egymással egyes Layer-ek, például ütközhetnek vagy érintkezhetnek. Erre a csoportosításra azért is van szükség, mivel előfordulhat, hogy két Layer ütközése nem megfelelő reakciót vált ki az objektumoknál.

Általában, amik itt vannak Layer-ek azokat bővítmények hozták ide, például Auto Hand stb.

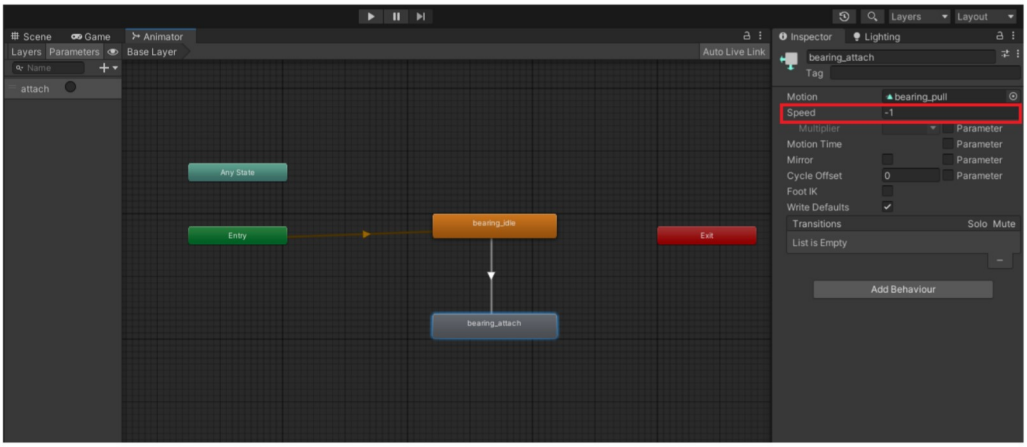


53. ábra. A layer-ek bemutatása (Saját készítésű ábra)

Következett a 6-dik fázis, ebben lecseréltem a csapágyat a piros dobozban lévő csere csapágyra. Ez a csapágy egy Prefab, vagyis többször felhasználható objektum. A Prefab készítéshez a Project ablakba kell behúzni az objektumot vagy a Hierarchy ablakban a Create menüből a Prefab lehetőséget kell kiválasztani.

Ehhez pedig egy új animátor controller-t hoztam létre, ugyan úgy jártam el itt is. Ezt bearing_attach néven hoztam létre, ennek az animátor paraméterét attach -nak neveztem el.

Felhasználtam hozzá a bearing_pull vagyis az előző levehető csapágy animációját. A sebességet levittem -1-re, ezáltal visszafelé fogja lejátszani az animációt.



54. ábra. bearing_attach Animator Controller (Saját készítésű ábra)

A csere csapágyhoz is hozzáadtam egy animátor komponenst, de nem kapcsoltam be és ráraktam a bearing_attach kontrollert.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Autohand;

public class CsereCsapagy : MonoBehaviour
{
    public Animator animator;
    public GameObject rotor;
    public GameObject csapagy_csere;
    private bool csapagyBenn = false;


    private void Update()
    {
        if(FindObjectOfType<PhaseController>().GetCurrentPhase() == Phase.phase6 && !csapagyBenn)
        {
            csapagy_csere.GetComponent<Grabbable>().enabled = true;
            csapagy_csere.GetComponent<Rigidbody>().isKinematic = false;
        }
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.name=="csapagy_csere" && FindObjectOfType<PhaseController>().GetCurrentPhase()==Phase.phase6 && !csapagyBenn)
        {
            csapagyBenn = true;
            csapagy_csere.GetComponent<Grabbable>().enabled = false;
            csapagy_csere.GetComponent<Rigidbody>().isKinematic = true;
            csapagy_csere.transform.SetParent(rotor.transform, true);

            animator.enabled = true;

            animator.SetTrigger("attach");
            StartCoroutine(Cooldown());
        }
    }
    IEnumerator Cooldown()
    {
        yield return new WaitForSeconds(1f);
        FindObjectOfType<PhaseController>().SetCurrentPhase(Phase.phase7);
    }
}
```

55. ábra. Csere csapágyhoz script (Saját készítésű ábra)

Ez a Csere csapágy script, amit a tengelyre raktam rá, ebben a script-ben a rotor és a csapágy_csere objektum is referencia lesz és hozzá kell adni őket.

Ebben a script-ben már az animáció lejátszása előtt hozzálehet érni az objektumhoz, interakcióba lehet vele lépni, majd, amikor a csere csapágyat hozzáérintjük a tengely végéhez akkor játszódjon le az animáció visszafelé, vagyis a csere csapágy a helyére kerül.

A script figyeli hanyadik fázisban vagyunk és ha elértük a 6-dik fázist akkor engedi megfogni a piros dobozban lévő csere csapágyat, a „Grabbable”-t bekapcsolja, az „Is Kinematic”-ot kikapcsolja, mivel ne legyen saját fizikája.

Ha hozzáérintjük a csapágyat a tengelyhez, akkor ellenőrzi, hogy a „csapágy_csere” tárggyal ütközött, 6-dik fázisban vagyunk és hogy nincs még bent a csapágy.

„Csapágybent” igaz, ez azt jelenti, hogy csak egyszer futhat le és tárolja, hogy rajta van e már a csere csapágy.

Ezt követően kikapcsolja a „Grabbable”-t és nem lesz megfogható, bekapcsolja az „Is kinematic”-ot.

A „SetParent” segítségével hozzá parent-eljük a rotorhoz, és bekapcsoljuk az animátorját, az „attach” paraméter pedig segít meghatározni, hogy a megfelelő animáció lejátszódjon.

Majd „IEnumerator Cooldown” elindul, ezáltal egy másodpercet lehet késleltetni az animációt, majd átlépünk a 7-es fázisba, ahol megjelenik az „Újrajátszás” gomb.

5.6 Tesztelés és Ellenőrzés

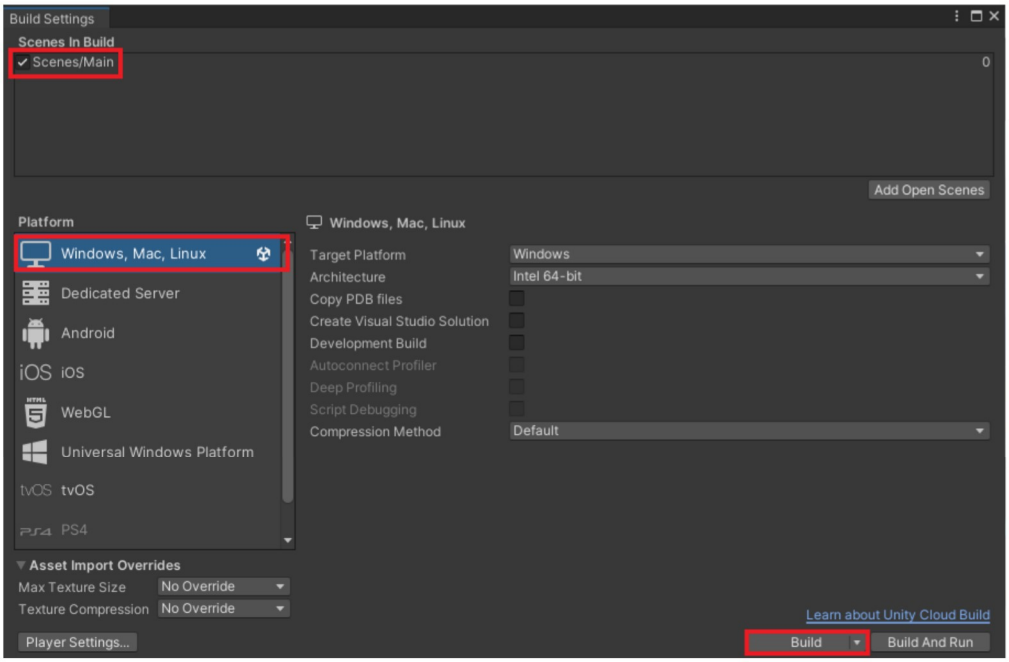
A tesztelés során fontos volt a fejszett megfelelő pozícionálása a szoba méreteihez képest. Ehhez a SteamVR alkalmazásban be kellett állítanom a szoba elrendezését, hogy a virtuális tér és a valós tér megfelelően illeszkedjen egymáshoz.

A felvétel készítéséhez a halkítás és hangosítás gombokat egyszerre kellett nyomva tartani a fejszett készüléken, és az Obs Studio segítségével lehetett rögzíteni azt.

A felhasználói interakciók ellenőrzése során a megfelelő lépéseknél érintettem meg az objektumokat, és ellenőriztem, hogy a műveletek megfelelően működnek-e a virtuális környezetben. Végül többször is ellenőriztem az alkalmazás működését, és meggyőződtem róla, hogy a projekt megfelelően működik.

Az exportálás a File menüben a Built Settings-nél található, itt ki kell választani, hogy mire szeretnénk menteni a fájlt. Több lehetőség közül is lehet választani, és én a Windows-t választottam.

Fontos, hogy a Scenes/Main legyen kiválasztva, mivel ez a jelenlegi pálya. Ezután a Build-re kell kattintani és kiválaszthatjuk a helyet, ahova menteni kívánjuk.



57. ábra. A projekt exportálása (Saját készítésű ábra)

6 Összefoglalás, jövőbeli célok

A szakdolgozatomban átfogó képet adtam arról, hogyan lehet VR tartalmat készíteni ipari felhasználásra és reményeim szerint inspirációt nyújt majd azoknak, akik ezt elolvassák, hogy elkészítsék a saját projektjüket és alkalmazni tudják a munkahelyükön.

A szakdolgozat készítése közben számos kihívással szembesültem, volt köztük hardveres és szoftveres probléma is. Főleg a programkódok írása okozott nehézséget, de az interneten található nagy mennyiségű és megfelelő minőségű oktatóvideó sokat segített a feladat megoldásában. A nehézségek ellenére megértettem, hogy milyen széleskörű felhasználási lehetőséget biztosít a bonyolult programozási felületért cserébe.

Az elkészítése közben rengeteg új dolgot tanultam és felfrissíthettem az angol tudásomat is. Célként tüzöm ki azt, hogy hasznosítani tudjam a technológiát a következő munkahelyemen, a jövőben várhatóan még szélesebb körben alkalmazzák majd a VR alapú tréningeket a munkahelyi képzések során, illetve a későbbiek során a mesterképzésem alatt is szeretnék a VR területtel foglalkozni, remélve azt, hogy addigra a technológia még nagyobb fejlődésen megy keresztül.

A jövőbeli célom az oktatóanyag továbbfejlesztése, és az elektromotor teljes összeszerelésével szeretném folytatni, illetve szeretném, ha a szakdolgozatomban készült interaktív oktatóanyagot akár nyílt napok alkalmával bemutassák az érdeklődőknek és a leendő mérnök hallgatóknak.

Irodalomjegyzék

[1] Liu, Li-Lun, Chiang, Chung Hang: *Patent Trend Analysis: Extended Reality (XR) and Future Virtual Adventure*, Scientific Press International Limited 2023.

[2] A kiterjesztett valóság, Forrás: https://www.researchgate.net/figure/Extended-Reality-XR-technologies_fig1_354858662 (Letöltés időpontja: 2024.03.21. 18 óra 54 perc)

[3] Hamad, A.; Jia, B.: *How Virtual Reality Technology HasChanged Our Lives: An Overviewof the Current and Potential Applications and Limitations*, Int. J.Environ. Res. Public Health, University of Michigan-Dearborn,2022

[4] Dr. Németh A. – Virágh K.: *Virtuális valóság és haderő I. rész*, in *HADITECHNIKA Folyóirat*, 2021, 55(2). pp. 5-11.

[5] A virtuális valóság, Forrás: https://www.verdict.co.uk/why-is-virtual-reality-important-to-the-metaverse/ (Letöltés időpontja: 2024.03.22. 18 óra 24 perc)

[6] Vasarainen, M., Paavola, S., & Vetoshkina, L.: *A Systematic Literature Review on Extended Reality: Virtual, Augmented and Mixed Reality in Collaborative Working Life Setting,* International Journal of Virtual Reality, 21(2), 1-28., Helsinki,2021

[7] Pokemon Go, Forrás: https://hu.wikipedia.org/wiki/Pok%C3%A9mon_Go (Letöltés időpontja: 2024.02.25. 22 óra 24 perc)

[8] A „Pokemon Go" nevű játék, Forrás: https://www.ox.ac.uk/news/2016-11-16-what-can-pok%C3%A9mon-go-teach-world-conservation (Letöltés időpontja: 2024.03.02. 09 óra 10perc)

[9] Dipesh Gyawali: *Mixed Reality: The Interface of the Future*, Department of Computer Science Louisiana State University, Baton Rouge, Louisiana, United States

[10] A Hololens szemüveg, Forrás: https://blogs.windows.com/devices/2016/02/29/announcing-microsoft-hololens-development-edition-open-for-pre-order-shipping-march-30/ (Letöltés időpontja: 2024.02.25. 22 óra 24 perc)

[11] A Hololens szemüveg, Forrás: https://news.microsoft.com/europe/features/microsoft-hololens-comes-to-europe/ (Letöltés időpontja: 2024.03.01. 09 óra 55 perc)

[12] Wolfgang Vorraber, Johannes Gasser, Helena Webb, Dietmar Neubacher, Philipp Url: *Assessing augmented reality in production: remote-assisted maintenance with HoloLens*, In: 13th CIRP Conference on Intelligent Computation in Manufacturing Engineering, p.139–144, 2019

[13] Asan Baker: *3D Holographic Projection,* Cihan University Sulaimaniya, 2019

[14] Abid Haleem a,Mohd Javaid a,Ravi Pratap Singh szül,Rajiv Suman c,Shanay Rab a: *Holography and its applications for industry 4.0: An overview*, Internet of Things and Cyber-Physical Systems, 2. kötet,p. 42-48., 2022

[15] A hologramok, Forrás: https://raketa.hu/gabor-denes-hologram-optika (Letöltés időpontja: 2024.02.25. 22 óra 24 perc)

[16] A 3D hologram, Forrás: https://www.vision3d.in/blog/3d-holographic-technology/ (Letöltés időpontja: 2024.04.01. 09 óra 30 perc)

[17] A hologram technológia, Forrás: https://www.respeecher.com/blog/holograms-real-life-technology-works-industry-use-cases(Letöltés időpontja: 2024.02.25. 22 óra 24 perc)

[18] A 3D kivetítés, Forrás: https://www.researchgate.net/publication/328809838_Historic_Urban_Settings_LED_Illumination_and_its_Impact_on_Nighttime_Perception_Visual_Appearance_and_Cultural_Heritage_Identity (Letöltés időpontja: 2024.01.15. 00 óra 24 perc)

[19] A 3D kivetítés az iparban, Forrás: https://www.extend3d.com/en/video-projection-in-the-industry/ (Letöltés időpontja: 2023.12.10. 16 óra 40 perc)

[20] 3D kivetítés, Forrás: https://hu.pinterest.com/pin/416653403003515531/ (Letöltés időpontja: 2024.04.01. 12 óra 30 perc)

[21] Az ipar 4.0, Forrás: https://www.sciencedirect.com/science/article/pii/S1877705092201246 7 (Letöltés időpontja: 2024.04.01. 14 óra 50 perc)

[22] Mamad Mohamed*: Challenges and Benefits of Industry 4.0: An overview*, In: International Journal of Supply and Operations Management, Volume 5, Issue 3, p. 256-265, 2018,

[23] Abid Haleem a, Mohd Javaid a, Ravi Pratap Singh b, Shanay Rab a, Rajiv Suman c, Lalit Kumar d, Ibrahim Haleem Khan e: *Exploring the potential of 3D scanning in Industry 4.0: An overview*, In: International Journal of Cognitive Computing in Engineering, p.161–171., KeAi chinese roots global impact 2022

[24] A Virtuális technológia használata az iparban, Forrás: https://www.banelec.com/five-ways-virtual-and-augmented-reality-can-enhance-industrial-processes/ (Letöltés időpontja: 2024.04.28. 17 óra 50 perc)

[25] HTC VIVE PRO szemüveg, Forrás: https://dronozok.hu/htc-vive-pro/ (Letöltés időpontja: 2024.04.12. 12 óra 45 perc)

[26] A HTC VIVE PRO eszköz, Forrás: HTC Vive Pro Bázisállomás 2.0 Base Station – Dronozok (Letöltés időpontja: 2024.04.30. 16 óra 28 perc)

[27] Mi a metaverzum, Forrás: https://www.penzcentrum.hu/tech/20221030/mi-a-metaverzum-mutatjuk-melyik-facebook-metaverzum-befektetes-otletek-a-legnepszerubbek-1130281 (Letöltés időpontja: 2024.01.25. 22 óra 24 perc)

[28] A Meta Quest 2 eszköz, Forrás: https://www.businessinsider.com/guides/tech/meta-quest-2-price-increase-2022-7 (Letöltés időpontja: 2024.04.02. 17 óra 00 perc)

[29] Meta Quest 2, Forrás: https://www.meta.com/quest/products/quest-2/tech-specs/#tech-specs (Letöltés időpontja: 2024.04.12. 12 óra 45 perc)

[30] Jakob J. Korbel: *Creating the Virtual: The Role of 3D Models in the Product Development Process for Physical and Virtual Consumer Goods* Technische Universität Berlin, Chair of Information and Communication Management, Berlin, Germany, 2021

[31] Cad és Mesh modell közötti különbség, Forrás: https://www.researchgate.net/publication/318250772_Fracture_Modelling_directly_from_Computer-Aided_Design_CAD_by_the_Extended_Isogeometric_Finite_Element_Method_X-IGA_FEM_with_trimmed_NURBS (Letöltés időpontja: 2024.04.02. 17 óra 45 perc)

[32] 3D szkennelés, Forrás: https://www.creaform3d.com/blog/3d-scanning-for-inspection-a-smart-choice/ (Letöltés időpontja: 2024.03.21. 17 óra 54 perc)

[33] Mohd Javaida, Abid Haleema, Ravi Pratap Singhb, Rajiv Suman: *Industrial perspectives of 3D scanning: Features, roles and it'sanalytical applications,* Sensors International, KeAi chinese roots global impact 2021

[34] Mohd.Fairuz Shiratuddin Abdul Nasir Zulkifli: *Virtual Reality in Manufacturing*, School of Information Technology University Utara Malaysia, 2001

[35] 3D szkennelés, Forrás: https://www.creaform3d.com/blog/pun5th75ef_wp/wp-content/uploads/HandySCAN3D_Inspection_Metrology_Lab_1-scaled.jpg (Letöltés időpontja: 2024.04.02. 11óra 25 perc)

[36] Mozgás animáció, Forrás: https://www.indiacadworks.com/blog/how-do-businesses-benefit-from-3d-animation/ (Letöltés időpontja: 2024.04.03. 08 óra 25 perc)

[37] Csontvázolás, Forrás: https://design.tutsplus.com/tutorials/building-a-basic-low-poly-character-rig-in-blender--cg-16955 (Letöltés időpontja: 2024.04.05. 08 óra 00 perc)

[38] A Unity, Forrás: Unity (játékmotor) - Wikiwand (Letöltés időpontja: 2024.02.25. 22 óra 24 perc)

[39] A Unity ipari alkalmazása, Forrás: https://unity.com/industry (Letöltés időpontja: 2024.02.25. 22 óra 24 perc)

[40] A Unity Editor bemutatása, Forrás: https://learn.unity.com/tutorial/explore-the-unity-editor-1# (Letöltés időpontja: 2024.03.28. 19 óra 24 perc)

[41] Editor ablak részei | Játékfejlesztés Unity-ben, Forrás: https://www.inf.u-szeged.hu/~vargalg/inProgress/Unity/editor_ablak_rszei.html (Letöltés időpontja: 2024.03.21. 15 óra 54 perc)

[42] Auto Hand, Forrás: https://earnestrobot.itch.io/auto-hand (Letöltés időpontja: 2024.04.28. 17 óra 50 perc)