EDDIE - 3D SZKENNELÉS ÉS KITERJESZTETT VALÓSÁG OKTATÁSI CÉLRA

Tudományos diákköri dolgozat

2018. november 5.

Muhi Kristóf
mérnökinformatika szak
Juhász Csaba
gépészmérnök szak

Neumann János Egyetem
GAMF Műszaki és Informatikai Kar
Konzulensek:
Dr. Johanyák Zs. Csaba tudományos dékánhelyettes
Dr. Líska János, főiskolai docens
Dr. Pásztor Attila, főiskolai tanár

TARTALOMJEGYZÉK

1. Absztrakt 3

2. Előzmények 4

2.1. A kiterjesztett valóság 4

2.1.1. Történeti áttekintés 4

2.1.2. Felhasználási lehetőségek 5

2.2. Okostelefonos alkalmazásfejlesztés mint korunk új platformja 6

3. Eddie alkalmazás tervezési szempontjai 8

3.1. Fejlesztőkörnyezet kiválasztása 8

3.1.1. AR fejlesztő könyvtárak 8

3.1.1.1. ARCore 8

3.1.1.2. ARKit 9

3.1.1.3. Vuforia 9

3.1.2. Okostelefonos alkalmazás fejlesztői környezete 9

4. A webplatform szerkezeti felépítése 11

4.1.1. A platform használata 11

4.1.2. Lista létrehozás 12

5. Az okostelefonos alkalmazás szerkezeti felépítése 14

5.1. AR tartalom szinkronizálása 15

5.2. 3D tartalom megjelenítése és váltása AR-ben 16

5.3. Azure CosmosDB adatbázis-struktúra kialakítása 17

6. Reverse engineering 18

6.1. Reverse engineering bemutatása egy példán keresztül 19

6.1.1. A geometria rekonstrukciója 20

7. Testmodellezés alapjai példán keresztül 22

8. 3D szkennelés ismertetése 27

Leggyakoribb felhasználási területek: 27

8.1. Szkennerek típusai 27
8.1.1. Sense 3D szkenner 28
8.1.2. Sense 3D szoftver 29
9. Biológiai tárgyak beszkennelése 30
9.1. Megfelelő környezet és modellek kiválasztása 30
9.2. Szkennelési folyamat 31
9.3. Beszkennelt modellek szerkesztése és színezése 32
10. Összegzés és további tervek 34
11. Irodalom jegyzék 35

1. ABSZTRAKT

Dolgozatunkban egy olyan, általunk fejlesztett, oktatási platformot mutatunk be, mely ötvözi a 3D szkennelés és a kiterjesztett valóság adta lehetőségeket. Az alkalmazás az *Eddie* nevet kapta, mely az angol *education* szóból ered. A platform két fő részre bontható: egy okostelefonos alkalmazásra és egy weboldalra.

***1.a ábra** - Eddie használat közben*

A weboldalon lehetőségünk van kiválasztani azt, hogy mely kétdimenziós képekre mely háromdimenziós objektumokat szeretnénk megjeleníteni a kiterjesztett valóságban *(AR – Augmented Reality)*.

A kép egy megjelenítő felületként szolgál, speciális jelölőként viselkedve az okostelefonos alkalmazás képes felismerni azt. A weboldal segítségével az adott felhasználó tetszőlegesen hozhat létre különböző listákat, melyben a megjelenítendő felületet (képet) összekötheti a megjeleníteni kívánt 3D objektummal. A felhasználó az okostelefonos alkalmazásba belépve a weboldalon összeállított listát megnyitva képes elérni az általa előzőleg összekapcsolt 3D tartalmat, amit az alkalmazás *AR-ben* jelenít meg. A megjelenített tartalom lehet egy forgatható 3D-s modell, animáció, videó, illetve bármilyen digitális tartalom.

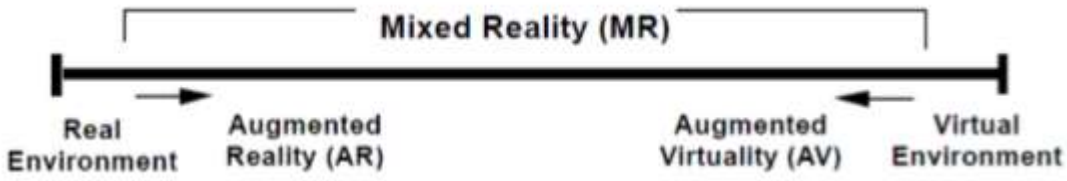
Dolgozatunkban részletezzük azt, hogy hogyan kapcsolódik a weboldal és az alkalmazás egymáshoz, hogyan készítettük el a megjelenített 3D-s objektumokat, illetve, hogy milyen technológiákat használtunk a fejlesztés során.

Mi, a dolgozat szerzői, oktatási tevékenységet végzünk általános iskolás gyerekek körében, így személyes tapasztalatból tudjuk, hogy a digitális oktatási módszerek bevezetésével a mai fiatal tanulók számára élménydúsabb, interaktívabb és érdekesebb tanulási folyamatot valósíthatunk meg.

2. ELŐZMÉNYEK

2.1. A kiterjesztett valóság

A kiterjesztett valóság (Augmented Reality, AR) a valóság egyfajta virtuális módján való kibővítése. A technológia gyakorlati használatához szükségünk van egy kamerával és megfelelő hardverrel ellátott okostelefonra vagy egy erre a célra kifejlesztett szemüvegre. Ezen eszközök használatával képesek vagyunk a céleszköz valós idejű kamera képén keresztül, a valós környezetbe különböző virtuális elemeket vetíteni.



[1] ***2.1.a ábra*** *– A kevert valóság (Mixed Reality, MR) kategorizálása*

A kiterjesztett valóság alapvetően két nagy csoportba osztható, a virtuális elem helyzetét meghatározó algoritmus és a megjelenítéstől függően. Az egyik a *pozíció és irány alapú AR*. Célja, hogy a kijelzőn, a már meglévő valós idő kameraképet valamilyen plusz információval egészítse ki. Ezeket az információkat úgynevezett POI-k (Point of Interest) tartalmazzák. Ezek olyan pontok helyét tárolják, melyek a felhasználó számára érdekes, hasznos lehet. A képernyőn megjelenő információk pontos helyének meghatározása a GPS pozíció, a beépített iránytű és a gyorsulásmérő adatinak feldolgozásával történik meg.

Az AR másik fajtája a *marker alapú AR*. Ilyenkor egy speciális jelölőt (általában képet, képrészletet) keresünk. A marker pozíciója és helye meghatározható az okostelefon valós idejű kameraképe által, ezáltal tetszőleges virtuális objektumokat helyezhetünk rá. Alkalmazásunk működése is ebbe a kategóriába sorolható, a saját marker kezelésünk a dolgozatunk további részeiben bővebb kifejtésre kerül.

2.1.1. Történeti áttekintés

Ivan Sutherland 1968-ban [2] fejlesztette ki az első fejre erősíthető képernyőrendszert. A rendszer képes volt különböző *wireframe* (hálóterves) grafikák megjelenítésére. 1974-ben Myron Krueger megépítette a *Videoplace* [3] nevű laboratóriumot, mely különböző kivetítőkkel és kamerákkal képes volt élethű sziluettek szimulálásra, körülvéve a felhasználót számos interaktív kezelőfelülettel.

Számos kutatást és fejlesztést követően a következő mérföldkő 1990-ben történt, mikor Tom Caudell megalkotja az *Augmented Reality (AR)* kifejezést [4]. 1992-ben Louis Rosenberg kifejleszti az első működőképes AR rendszert az amerikai légierőnek [5], különböző gépek virtuális vezérlésének céljából. 1994-ben Julie Martin rendezi meg [5] az első kiterjesztett valóságot is körül ölelő színházi produkciót melyben akrobaták egy virtuális objektumot táncoltak körül a színpadon.

A 2000-es évek kezdetétől a kiterjesztett valóság felhasználási területe nagy mértékben teret nyert a szórakoztató iparban. 1998-ban a *Sportvision* televízió csatorna *NFL* játékában [6] először használtak vizuális jelölőt a meccs különböző elemeinek kiemelésére. A hadiparban folyamatosan fejlesztések folytak különböző vizuális segédeszközök terén, főleg a légierőben. Hirokazu Kato kifejleszti az *ARToolKit*-et [7], mely egy nyílt forráskódú programkönyvtár. Segítségével képesek vagyunk különböző számítógépes grafikák valós kameraképen való megjelenítésére és annak követésére. 2009-ben az *ARToolKit* már webes böngészőket is támogat.

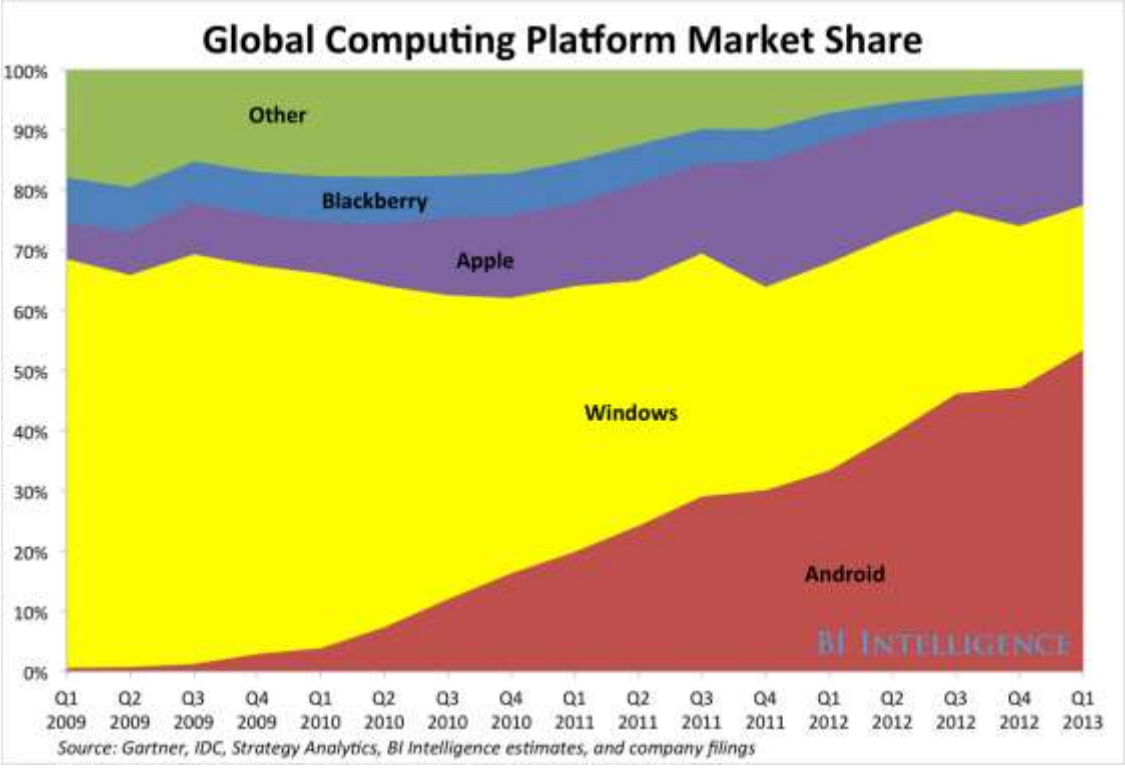
Napjainkban a hardver és szoftver rohamos fejlődésével egyre közelebb kerülnek a valósághű AR élmények. A nagy számítástechnikai cégek mind kiemelten foglalkoznak a technológiával. A Microsoft *HoloLens* [8] szemüvege külső hardver igény nélkül képes AR-tartalmat megjeleníteni, akár jelölők nélkül, a tér és a síkok érzékelésével. Az Apple és Google, a két legnagyobb okostelefon operációs rendszert az iOS-t és Androidot fejlesztő cég is rendelkezik saját AR fejlesztőkörnyezettel, melyekkel lehetőségünk van létrehozni a kiterjesztett valóság élményét.

2.1.2. Felhasználási lehetőségek

Mint ahogyan arról az előző fejezetben már szót ejtettünk, a kiterjesztett valóság első felhasználási területe a sportközvetítések voltak. Labdarugó mérkőzések kezdetekor a pályára vetített csapat címerek, kajak-kenu esetén a célba éréskor virtuálisan meghúzott jelzővonal is ezt a technológiát használja. Marketingtevékenységben is kiválóan használható, ahogyan a turizmusban az irány alapú AR adta lehetőségek által bővebb információkat kaphatunk a minket körülvevő városról. Egyéb felhasználási módok a teljesség igénye nélkül: egészségügy, ipari folyamatok (gyártás és javítás), szórakoztatás, oktatás, játék stb.

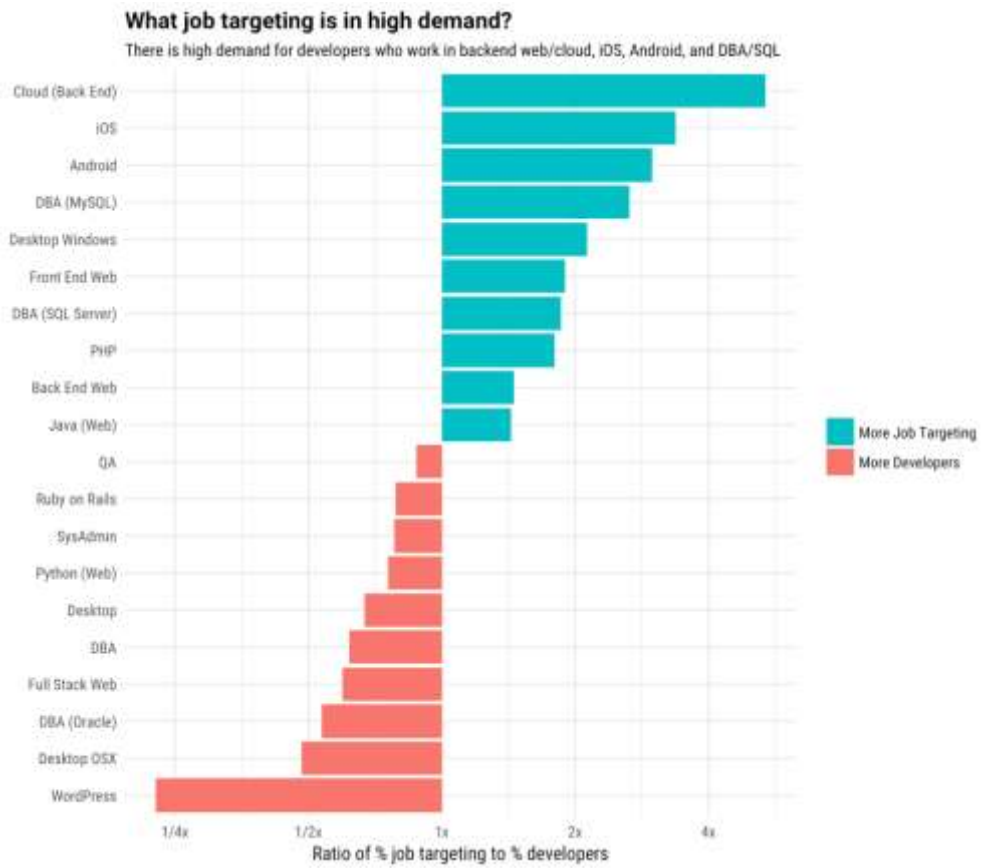
2.2. Okostelefonos alkalmazásfejlesztés mint korunk új platformja

Amikor 2008-ban az Apple elindította az App Store-t [9], a saját alkalmazás áruházukat, csak páran vették észre azt, hogy mekkora potenciál rejtőzik benne. Közel tíz évvel később mindenkinek nyilvánvalóvá vált, hogy az App Store elindítása felforgatta a komplett szoftveripart és egy új nézőpontot hozott a világba.



[10] ***2.2.a ábra*** *- A globális számítástechnikai platform piaci részesedésének megosztása*

Az 3.a ábrán látható, hogy nem sokkal az App Store és a Google Play áruház elindulása után eltelt négy év alatt mennyit változott a globális számítástechnikai platform piaci részesedésének megosztása. Jól kivehető, hogy a Windows melynek okostelefonos operációs rendszere nem ért el átütő sikert és 2017 októberében a Microsoft hivatalosan is kijelentette, hogy a Windows Phone eddigi formájában megszűnik [11], asztali szoftver terén is rohamosan veszít részesedéséből a konkurens gyártók ellen. Tisztán látszódik a jelen felhasználóinak számítástechnikai felhasználási szokása is. Adataink tárolására már számtalan felhő alapú megoldás létezik és az informatikai óriás cégek pl. Google, Apple, Amazon, Microsoft is egyértelműen ezen technológia népszerűsítésén dolgoznak. A felhő technológiával adatainkat azonnal elérhetjük mely a hordozható telefonok rugalmasságával ötvözve még kecsegtetőbb opciót kínál a helyhez kötött, asztali számítógépekkel szemben.



[12] ***2.2.b ábra*** *- Informatikai állások kereseti arány szerint*

A fenti ábrán látható, hogy a felhő alapú technológia rohamos fejlődése és népszerűsége végett ez az a terület, ahol jelenleg a legnagyobb szükség van fejlesztőkre. Ezt követi az Apple iOS és a Google Android okostelefon operációs rendszer platformjai és utánuk ötödik helyen található az asztali Windows, mely kiválóan szemlélteti a jelenlegi felhasználói trendeket.

3. EDDIE ALKALMAZÁS TERVEZÉSI SZEMPONTJAI

Az alkalmazás célja, hogy a hagyományos értelemben vett oktatási folyamatot kiegészítse, és egy interaktív, élményalapú segédeszközt nyújtson a tanulásban. Az alkalmazás különböző digitális tartalmakat (elsődlegesen 3D objektumokat) képes megjeleníteni a felhasználó okoseszközén (okostelefon, táblagép) az AR technológia használatával.

Az AR megjelenítés erősen eszközfüggő, ezért a megfelelő fejlesztőkörnyezet kiválasztása kulcsfontosságú feladat volt. Az alkalmazás másik kulcsfontosságú tervezési szempontja az volt, hogy a tanárok és diákok számára is egyszerűen használható legyen, tehát: ne kelljen plusz eszközt vásárolni (AR szemüveg), az alkalmazás könnyen elérhető és telepíthető legyen. A tanárok szempontjából a 3D tartalom előkészítése (listák létrehozása, erről bővebben később) nem szabad, hogy megterhelő legyen.

3.1. Fejlesztőkörnyezet kiválasztása

3.1.1. AR fejlesztő könyvtárak

Az AR technológia rohamszerű fejlődésének eredménye, hogy már nem szükségesek felső-kategóriás eszközök a megjelenítéshez, így egyre jobb élményt lehet nyújtani az átlagos eszközökkel rendelkező felhasználóknak is. Az alkalmazás tervezésekor azt a cél tűztük ki, hogy nem korlátozzuk az eszközöket, minél több felhasználónak elérhető szeretnénk tenni. Ez a megfelelő fejlesztői környezet kiválasztásakor volt fontos.

Három AR fejlesztő könyvtárat szeretnénk kiemelni:

3.1.1.1. ARCore

Az *ARCore* [13] a Google által fejlesztett AR-megjelenítő SDK (software development kit, szoftverfejlesztői csomag) melynek első verziója 2018. március 1-én jelent meg. Három technológiát használ a digitális tartalom megjelenítéséhez: mozgáskövetés, környezetfelismerés (síkok, padló felismerés) és a környezeti fények feldolgozása. Használatához Android 7.0 (Nougat) vagy későbbi rendszer szükséges. Apple iOS rendszeren 11.0 vagy későbbi frissítés a követelmény. Az operációs rendszer mellett eszközfüggőség is van: Az *ARCore* kizárólag az Apple iPhone SE vagy újabb (kb. 15 különböző) eszközökön működik. Az Android rendszerű okostelefonok terén jelenleg csak kb. 100 eszköz támogatott a több mint 21700 különböző [14] okostelefon közül. Az erős eszközkorlátozás miatt az előzőekben taglalt fejlesztési szempontoknak az *ARCore* könyvtára nem felelt meg.

3.1.1.2. ARKit

Az *ARKit* [15] egy Apple által fejlesztett AR-megjelenítő könyvtár, melynek első stabil verziója az iOS 11 megjelenésével egyidőben, 2017. júniusában jelent meg. A könyvtár kizárólag iPhone és iPad eszközökkel kompatibilis. Szoftver- és hardverfüggőség itt is jelen van, ahogyan az *ARCore*-nál: iOS 11 vagy újabb rendszer és iPhone SE vagy attól újabb eszköz szükséges a futtatáshoz. Technológiailag az iPhone hardveres megoldásait használja, köztük a VIO-t (Visual Inertial Odometry) mely a környezet észleléséért felel (síkok-, tárgyak-, padló felismerése). Az *ARCore*-hoz hasonlóan képes feldolgozni a külső fényforrásokat a még szebb vizuális megjelenítés céljából. Ahogyan a Google *ARCore* függőségei, az *ARKit* kizárólagos iOS támogatása végett ez a könyvtár sem felelt meg az alkalmazás fejlesztési követelményeinek.

3.1.1.3. Vuforia

A *Vuforia* (korábban QCAR) a Qualcomm által fejlesztett AR fejlesztői platform, melynek első stabil verziója 2011. áprilisában jelent meg [16]. Marker alapú AR fejlesztésére használható SDK. Szoftver- és eszközkorlátozása az előzőekben taglalt *ARCore* és *ARKit*-hez képest jóval kedvezőbb, Android és iOS rendszereken is csak szoftverfüggősége van: Android 4.4-es verziótól, illetve iOS 9.0-tól újabb verzió szükséges. Rendelkezik egy úgynevezett *Vuforia Cloud* rendszerrel, mely lehetőséget biztosít arra, hogy valós időben változtassuk a markereket és a hozzá rendelt, kiterjesztett valóságban megjelenő 3D tartalmat. Az előzetes fejlesztési szempontokat figyelembe véve, a *Vuforia* széles eszköztámogatása és felhő-alapú rendszere miatt erre a könyvtárra esett a választásunk.

3.1.2. Okostelefonos alkalmazás fejlesztői környezete

A platform okostelefonos alkalmazás részének fejlesztésekor fontos szempont volt az, hogy a fejlesztőkörnyezetben *cross-platform* (platformfüggetlen) alkalmazásfejlesztésre legyen lehetőség. A különböző okostelefonok operációs rendszerei mind rendelkeznek hivatalosan támogatott fejlesztő környezettel. Android esetén ilyen az *Android Studio* [17], iOS esetén az *XCode* [18]. A *Windows Phone* rendszer hivatalos szoftvertámogatása megszűnt (2.2. fejezetben taglaltak szerint), így számunkra sem volt cél az arra történő fejlesztés. A fent említett fejlesztőkörnyezetek sajátossága, hogy csak a saját rendszerükkel kompatibilis, így Android és iOS rendszerre két párhuzamos fejlesztésre lenne szükség.

A folyamat optimalizálása szempontjából a *Unity [19]* fejlesztőkörnyezetét választottuk, mivel platformfüggetlen környezet révén jócskán lerövidül a fejlesztési idő.

3.1.2.a ábra – Unity fejlesztőkörnyezet ablaka

A *Unity Engine* egy játékmotor mely számítógépes játékok fejlesztésének céljából lett létrehozva. Rendkívül jól kezeli a különböző 3D formátumú fájlokat és natívan támogatja a *3.1.1.* fejezetben taglalt *Vuforia* AR-könyvtárat.

4. A WEBPLATFORM SZERKEZETI FELÉPÍTÉSE

Az Eddie webplatform részének fejlesztése során elengedhetetlen volt, hogy a felhasználó saját maga is képes legyen testre szabni a képek (*target markers*) és modellek közti kapcsolatot. Ez a kapcsolat könnyen és gyorsan megvalósítható a web alapú platformon keresztül. A platform a mai modern *Node.js - Express - JavaScript* technológiákat használja. Az események kezelését és az aszinkron *I/O műveleteket* a *Node.js* magja teszi lehetővé. Ez több *C/C++* könyvtáron alapszik. Ezen réteg felett fut a *Google V8 JavaScript* motorja, mely segítségével a *JavaScript* alkalmazás kapcsolatba tud lépni ezen könyvtárakkal egy *API*-n (Application Programming Interface, Alkalmazás Programozási Felület) keresztül. Emellett az általunk használt *Microsoft Azure* felhő-szolgáltatások natívan támogatják, így számos funkciót egyszerűen ki lehet használni.

4.1.1.a. ábra *– A webplatform bejelentkezési felülete*

4.1.1. A platform használata

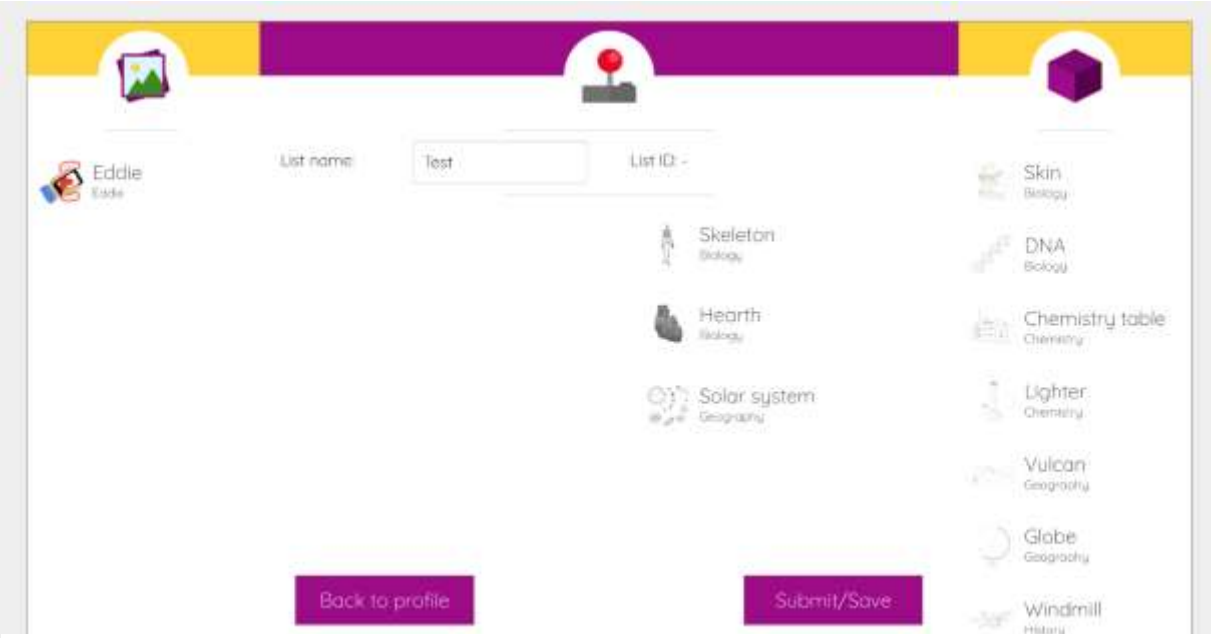
A platform megnyitását követően egy bejelentkezési felület fogadja a felhasználót, ahol lehetősége van regisztrálni a weboldalra. Ha ezt már megtette, akkor a felhasználóneve és jelszava megadását követően bejelentkezhet. A bejelentkezés további három módon lehetséges: Facebook, Google+, illetve Microsoft profil is összeköthető az alkalmazással. A szükséges adatokat az *Azure* által biztosított *CosmosDB* adatbázis-rendszer tárolja.

4.1.2. Lista létrehozás

A felhasználó profiljához tartozik még az általa feltöltött tartalom, illetve a létrehozott modell-listái. Ezeket a felhasználó egyedi azonosítója köti össze, különbözteti meg.

A bejelentkezést követően a felhasználó profilja válik láthatóvá. Itt töltheti fel az általa választott tartalmakat, megadhatja az oktatási intézményét, illetve módosíthatja a bejelentkezéshez szükséges jelszavát. Továbbá innen éri el az általa létrehozott modell-listákat, azokat szerkesztheti, vagy újat hozhat létre.

A modell-lista kezeléséhez egy egyszerű felületet szolgál. A felület jobb oldalán látható a felhasználó saját tartalma, a jobb oldalon pedig az adatbázisban található modellek. Ha a felhasználó ezekre kattint, akkor kiválasztja azokat, a lista részévé vállnak. Ha törölni szeretné a listából, akkor az adott elemre újra rá kell kattintania. A lista létrehozásakor szükséges nevet adni annak. A weboldal alján található két gomb, az egyikkel mentés nélkül vissza lehet lépni a profil oldalra, a mentés gomb menti a listában tett módosításokat.



4.1.2.a ábra *Lista létrehozása a platformon*

A mentési folyamat során ellenőrzésre kerül, hogy a felhasználó adott-e nevet a listának. Ha nem, akkor figyelmeztetést kap, hogy a mentés nem lehetséges név nélkül. Ha minden készen áll a mentés megkezdéséhez, akkor a weboldal a kiválasztott modellek és képek alapján generál egy *json* struktúrájú csomagot. Ezt a csomagot a weboldal újra töltése nélkül elküldi a szervernek, ami az adatokat eltárolja az adatbázisban.

Ezt követően a *PubNub API* segítségével üzenetet küld a weboldalnak, ami tartalmazza azt a kódot, amit a mobil alkalmazásba beírva a program az adott modell-listával köti össze magát. A weboldalon ez a kód a képernyő közepén válik láthatóvá.

A *PubNub* [20] egy olyan *API*, ami valós idejű üzenetküldésre ad lehetőséget. Ehhez a szervernek és a weboldalnak egy azonos csatornához kell csatlakoznia. Mindkét fél képes üzenetet küldeni, és egy folyamatosan futó „hallgató” folyamat segítségével fogadni is képes ezeket az üzeneteket. Ha egy meglévő modell-listát szerkeszt a felhasználó, akkor a mobilalkalmazás is kap egy üzenetet, ami arról értesít, hogy az új modell-listát szinkronizálni kell.

5. AZ OKOSTELEFONOS ALKALMAZÁS SZERKEZETI FELÉPÍTÉSE

Az alkalmazás első megnyitása során az üdvözlőképernyő várja a felhasználót. Ilyenkor két opció közül lehet választani. A *„get a fresh start”* (friss indítás) gombra kattintva, az alkalmazás betölti az alapértelmezett listát (a különböző listákról később bővebben), amihez egy alapértelmezett 3D-s emberi szív modell tartozik. A kezdőképernyőn található szövegdobozba a weboldal által generált 6 számjegyű kódot tudja írni a felhasználó. Ezt követően a *„jump”* (ugrás) gombra kattintva, az alkalmazás a beírt listaazonosító alapján szinkronizálja a lista tartalmát. A sikeres szinkronizálás után megjelenik a valós idejű kamerakép, majd a célképet (*target marker*) felismerve megjelenik a hozzá tartozó modell. További három gomb is megtalálható a képernyő alsó sávjában.

***5.a ábra** - Az alkalmazás kezdőképernyője*

A *„Sync”* (szinkronizálás) gomb ellenőrzi, hogy a legutolsó szinkronizálás óta történt-e változás a lista elemeit tekintve, és amennyiben igen, úgy frissíti a listához tartozó modelleket.

A *„Model change”* (modell váltás) gombra kattintva a képernyőn megjelenik az összes listában található modell neve. A megjeleníteni kívánt modell megváltoztatása esetén az alkalmazásnak nem kell újra megkeresnie a célképet (*target maker*), dinamikusan lecseréli azt. Amennyiben a lista csak egy modellt tartalmaz, a *„Model change”* gomb nem jelenik meg.

A *„List change”* (lista váltás) gomb lehetőséget nyújt a felhasználónak, hogy egy másik listát töltsön be az alkalmazásba. Ilyenkor visszakerül a kezdőképernyőre, ahol ismételten be tudja írni a kívánt listaazonosító kódot. Amennyiben a weboldalon megváltoztatjuk az aktuálisan használt lista tartalmát, tehát törlünk vagy hozzáadunk új 3D-s modelleket, az alkalmazásban valós idejű értesítést kapunk arról, hogy a lista módosítva lett és szinkronizálható.

5.1. AR tartalom szinkronizálása

5.1.a ábra *Az alkalmazás fő képernyője*

Az előző fejezetben taglaltak szerint a weboldalon összeállított listát az okostelefonos alkalmazásban szinkronizálni tudjuk, így megjeleníthetővé válnak az általunk választott 3D-s modellek. A „sync” gomb felhő – alkalmazás szinkronizációjáért felelős programkód részlete az *5.1.b ábrán* látható. Az *AzureCosmosDbQuery* osztály az *Azure CosmosDB NO-SQL* adatbázisban tárolt adatok lekérdezéséért felel. A *PrefabSyncControl* osztály, az előbbiekben említett, felhőben található adatbázisból lekért adatok alapján összeválogatja és betölti a lokálisan elérhető 3D modelleket. A *prefab egy, a Unity* fejlesztőkörnyezetben használt elem. Ezek olyan 3D-s tartalmak, melyeket előre definiálunk, különböző nyers 3D-s objektumok alapján. Ezekhez a nyers objektumokhoz különböző tulajdonságokat adhatunk, pl. szín, anyag, fizikai tulajdonság stb. Ezek, az akár több különálló objektumból álló csoportosulásokat hívjuk *prefabeknek.*

```
1. AzureCosmosDbQuery mAzureCosmosDbQuery = new AzureCosmosDbQuery();
2. PrefabSyncControl mPrefabSyncControl = new PrefabSyncControl();
3. StartCoroutine(mAzureCosmosDbQuery.GetRequest(mAzureCosmosDbQuery.getDocs + listId, (responseJson) => {
4.     try {
5.         mPrefabSyncControl.InitPrefabList(responseJson);
6.     } catch (Exception e) {
7.         Debug.Log("Startup sync error: " + e);
8.     }
9. }));
```

5.1.b ábra *Kollekció szinkronizálásért felelős programrészlet*

A *StartCoroutine* függvény meghívásával a függvény paramétereként megadott metódus utasításai nem a fő programszálon fognak futni, hanem a háttérben, egy külön e célra szolgáló másikon. Ezzel azt érjük el, hogy még a program a szinkronizálási feladatokat végzi, az alkalmazás kezelőfelülete használható marad. Paraméterként szükséges megadni azt a metódust melyet a háttérben szeretnék futtatni. A *GetRequest* függvény lekéri a felhő adatbázisban tárolt adatokat, melyet visszaad a fő függvény részére. Az *InitPrefabList* függvény a kapott *JSON* struktúrából összeállítja és betölti a megfelelő 3D-s modelleket.

5.2. 3D tartalom megjelenítése és váltása AR-ben

Az előző fejezetben taglalt modellek szinkronizálása és betöltése után lehetőségünk van a tartalmat böngészni és kiválasztani azt, hogy épp melyik 3D-s modell legyen aktívan látható.

```
1.  public class ChangePrefab : MonoBehaviour, IPointerClickHandler {
2.      public void OnPointerClick(PointerEventData eventData) {
3.          var parentIT = GameObject.Find("TempCR");
4.          var ListPanel = GameObject.Find("ListPanel");
5.          var selectedPrefab = PrefabListDataClass.modelList.list.Find(x => x.name.Contains(transform.Find("ModelName").
6.              GetComponent<Text>().text));
7.          parentIT.transform.Find(selectedPrefab.model_id + ("(Clone)")).SetAsFirstSibling();
8.          parentIT.transform.GetChild(0).gameObject.SetActive(true);
9.          for (int i = 1; i < parentIT.transform.childCount; i++) {
10.             parentIT.transform.GetChild(i).gameObject.SetActive(false);
11.         }
12.         ListPanel.SetActive(false);
13.     }
14. }
```

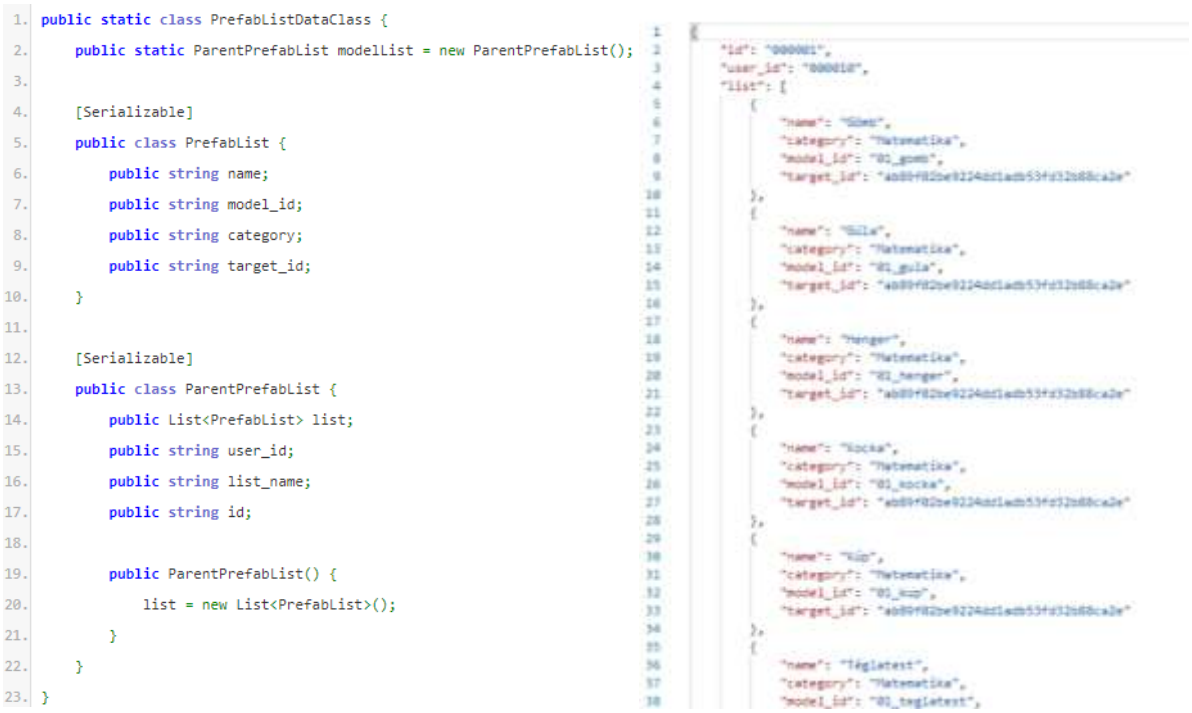
5.2.a ábra *Kollekció szinkronizálásért felelős programrészlet*

5.2.b ábra *Modell választó nézet*

Ahogyan az a fenti ábrán is látható a *ChangePrefab* osztály az alapértelmezett *MonoBehaviour* mellett implementálja az *IPointerClickHandler* interfészt is. Az interfész *OnPointerClick* metódusát megvalósítva képesek vagyunk észlelni a kattintási (jelen esetünkben érintési) interakciókat. A fenti kódrészlet az *5.1.a* ábrán látható kezdőképernyő „model change” gombján történő érintési eseményeket figyeli. A *parentIT* változó az a Unity objektum, aki alá gyerekobjektumomként betöltődik a listában található összes 3D modell. A betöltött modellek alapértelmezetten inaktívak. A megjeleníteni kívánt modellt a „model change” gomra kattintva, a megjelenő listából kiválasztva tudjuk kiválasztani. A kiválasztás után az alkalmazás az inaktív, előzetesen már betöltött modellek között megkeresi a megjeleníteni kívánt objektumot és aktívvá, az addig jelen lévőt pedig inaktívvá teszi. Ezt követően bezárja a modell választó nézetet és a kameranézetet jeleníti meg az újonnan kiválasztott elemmel.

5.3. Azure CosmosDB adatbázis-struktúra kialakítása

A *Microsoft Azure* a Microsoft *cloud computing (felhőalapú számítás)* platformja és infrastruktúrája, melynek segítségével alkalmazásokat lehet készíteni, telepíteni és futtatni a *Microsoft* által felügyelt adatközpontokon. Több mint 600 különféle szolgáltatás vehető igénybe, ezek közül az alkalmazás fejlesztése és későbbi üzemeltetése szempontjából a *virtuális gép* és a *NO-SQL* adatbázis szolgáltatásukat, a *Microsoft CosmosDB*-t vettük igénybe.



```
public static class PrefabListDataClass {
    public static ParentPrefabList modelList = new ParentPrefabList();

    [Serializable]
    public class PrefabList {
        public string name;
        public string model_id;
        public string category;
        public string target_id;
    }

    [Serializable]
    public class ParentPrefabList {
        public List<PrefabList> list;
        public string user_id;
        public string list_name;
        public string id;

        public ParentPrefabList() {
            list = new List<PrefabList>();
        }
    }
}
```

```
{
    "id": "000001",
    "user_id": "000010",
    "list": [
        {
            "name": "Gömb",
            "category": "Matematika",
            "model_id": "01_gomb",
            "target_id": "ab89f02be9224dd1adb53fd32b88ca2e"
        },
        {
            "name": "Gúla",
            "category": "Matematika",
            "model_id": "01_gula",
            "target_id": "ab89f02be9224dd1adb53fd32b88ca2e"
        },
        {
            "name": "Henger",
            "category": "Matematika",
            "model_id": "01_henger",
            "target_id": "ab89f02be9224dd1adb53fd32b88ca2e"
        },
        {
            "name": "Kocka",
            "category": "Matematika",
            "model_id": "01_kocka",
            "target_id": "ab89f02be9224dd1adb53fd32b88ca2e"
        },
        {
            "name": "Kúp",
            "category": "Matematika",
            "model_id": "01_kup",
            "target_id": "ab89f02be9224dd1adb53fd32b88ca2e"
        },
        {
            "name": "Téglatest",
            "category": "Matematika",
            "model_id": "01_teglatest",
```

5.3.a ábra *CosmosDB adatbázis struktúra egy listára vonatkozóan*

A fenti ábra jobb oldalán látható a *Microsoft Azure CosmosDB*-ben kialakított adatbázis struktúra egy adott kollekcióra vonatkozóan. Minden ilyen kollekció rendelkezik egy egyedi azonosító számmal, ezt az értéket szükséges megadnunk az alkalmazásban is a szinkronizáció végett. A *„user id”* mező jelzi, hogy az adott modell lista melyik felhasználóhoz van hozzárendelve. A „*list”* adattagban szerepelnek azok a modellek adatai, melyeket a *4.1.2.* fejezetben taglaltak szerint, a weboldalon választhatunk ki. Tárolásra kerül minden egyes elem neve, kategóriája, azonosítója és a *CosmosDB* által generált belső azonosító.

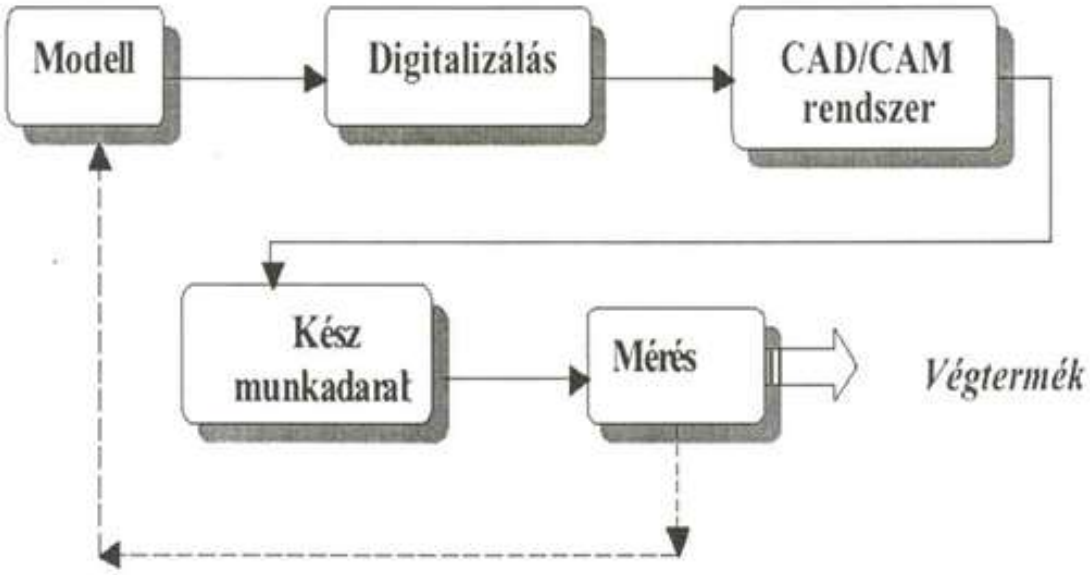
Az *5.3.a* ábra bal oldalán látható az adatok lokális tárolásáért felelős C# script. A statikus *PrefabList* osztály egyezik meg egy kiválasztott modell adataival.

6. REVERSE ENGINEERING

A reverse engineering egy olyan eljárás, amellyel egy már kész objektumot újra tervezünk, ami nem rendelkezik tervdokumentációval, nincs számítógéppel elkészített terve, műhelyrajza, kézzel formázott mesterminta, bonyolult formájú tárgy vagy illeszkedő alkatrész. [21]

A reverse engineering folyamata:

- fizikai modell digitalizálása
- mérési pontok editálása
- mérési pontok beolvasása CAD rendszerbe
- pontokra felületi görbék illesztése
- görbékre felület vagy felületsík illesztése
- felesleges részek eltávolítása
- modell pótlása, szerkesztése
- modell gyártása
- mérés [22]



6.a. ábra *- Reverse engineering folyamata* [23]

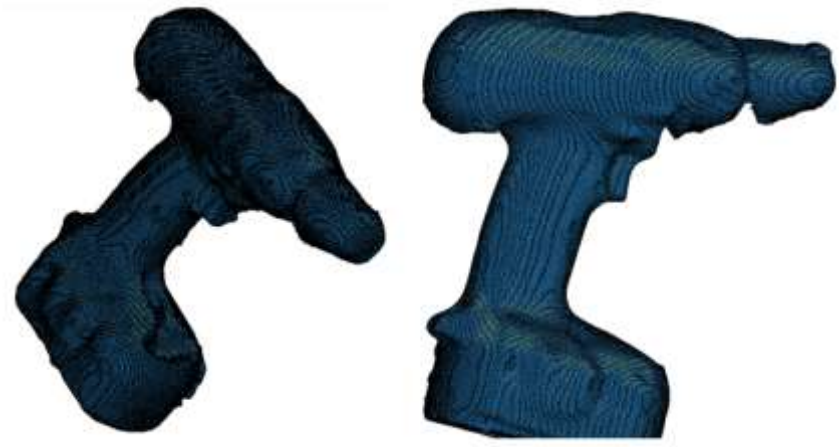
6.1. Reverse engineering bemutatása egy példán keresztül

A választott modell egy fúrógép. Ahhoz, hogy a legjobban szemléltetni lehessen a *reverse engineering* folyamatát egy olyan modellt kell választani, ami elég bonyolult, sok felületmodellezést igényel, és beszkennelhető egy egyszerűbb 3D szkennerrel is. A fúrógép befoglaló méretei: 270x200x100 mm.

A folyamatot a *Sense 3D szkennerrel* végeztük el.

***6.1.a ábra** - Választott modell*

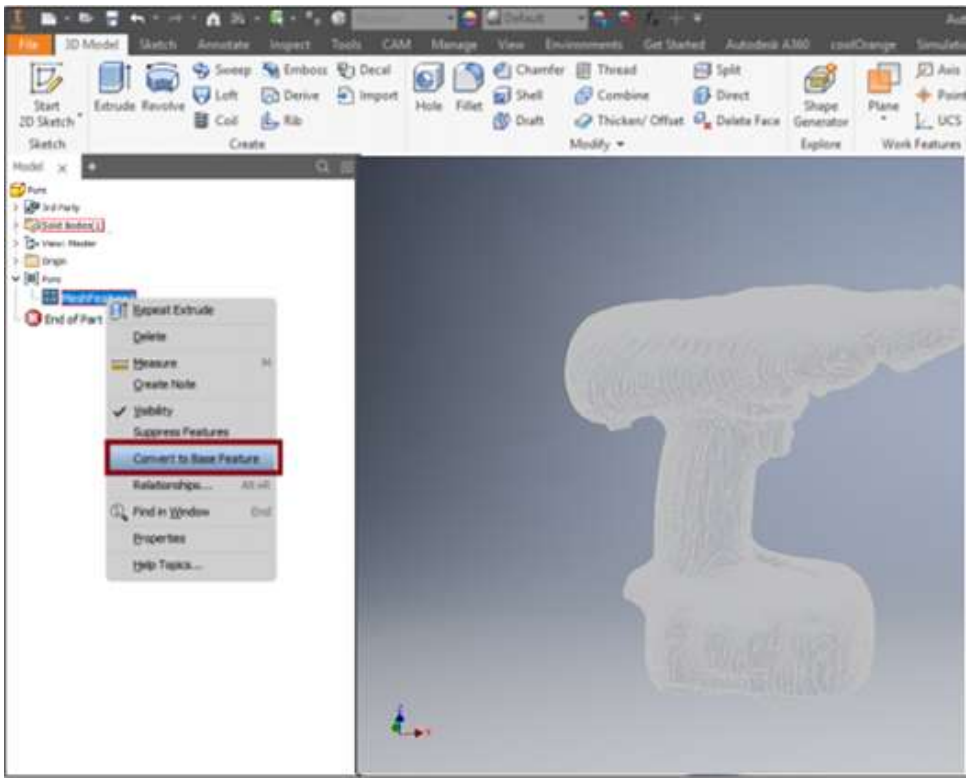
A szkennerrel körbejárjuk a tárgyat úgy, hogy az objektum minden része látható legyen a szoftveren belül. Ha ez megvan, javítások után megkapjuk a pontfelhőt. A szoftver ezután a pontokat apró háromszögekké konvertálja, így egy felületmodellt létrehozva.



***6.1.b ábra** - Háromszögesített pontháló*

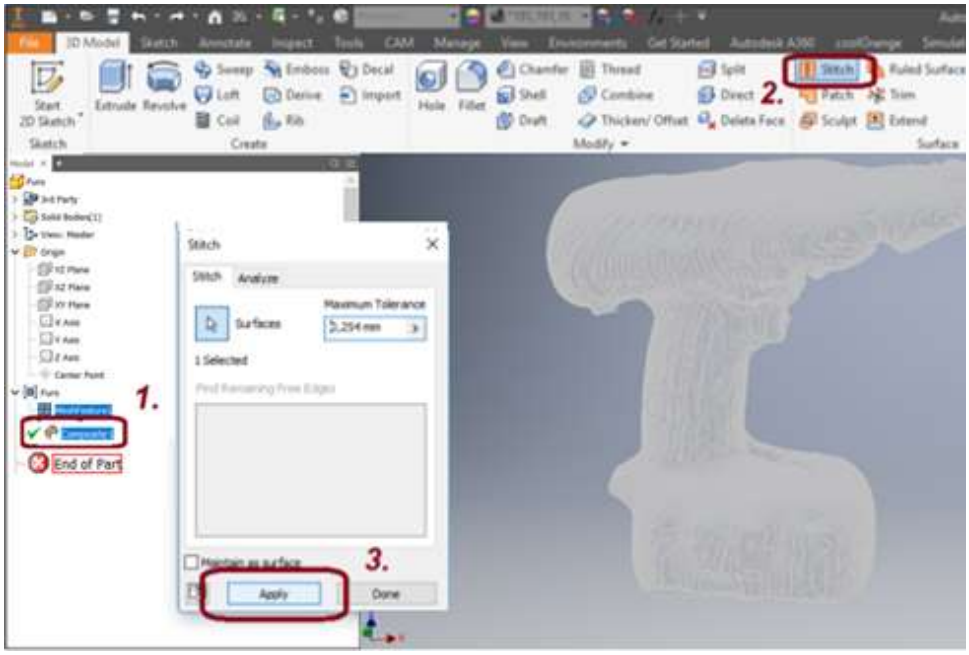
6.1.1. A geometria rekonstrukciója

A geometria rekonstrukcióját mindenképp az *Inventor*-ban kezdjük, mivel a szoftver *.fbx* és *.obj* formátumokban tudja elmenteni a modellt, és ezeket a formátumokat, veszteségmentesen ebben a CAD szoftverben tudjuk megnyitni. Ezt követően tudunk alkalmazni más szabadon választott (esetlegesen több felületmodellezési funkcióval rendelkező) CAD szoftvert. Miután megnyitottuk a fájlt, az STL hálót át kell alakítani felületmodellé.



6.1.1.a ábra *- Az STL pontháló konvertálása felületmodellé*

A konvertálás eredménye egy felületmodell lesz. Ezt célszerű (ezen lépés kihagyásával is elkészíthető az RE modell.) testmodellé konvertálni. Egy testmodellt egyszerűbb később módosítani általános CAD szoftverekkel is. Ha felületmodellként hagyjuk, akkor pedig egyéb nem CAD modellező szoftverekkel tudjuk könnyebben módosítani.



6.1.1.b ábra *- A felületmodell testmodellé konvertálása.*

Ezt követően, létrejön egy testmodell, amivel tudjuk folytatni a visszamodellezést az Inventor-ban, vagy más CAD szoftverben. Amennyiben más szoftverben kell folytatni a munkát, STEP fájlba kell elmenteni és így módosítható lesz bármeny CAD szoftverben.

7. TESTMODELLEZÉS ALAPJAI PÉLDÁN KERESZTÜL

Mi az Inventor-ban [30] folytattuk az előző feladat modellezését, az első lépés, a vázlatsíkok létrehozása volt. A Reverse Engineering folyamatokról általánosságban elmondható a tény, hogy a művelője próbál „az egykori tervező fejével gondolkodni” és úgy megalkotni a vázlatokat, hogy az tükrözze az eredeti modell felépítését.

A modellfában megkerestük az alap síkokat és ezt pozícionáltuk be a megfelelő helyre. Ezen létrehozott síkra elkészítjük a geometria egyik keresztmetszeti vázlatát.

Az első lépés az akkumulátor kihúzása volt. ezt egy egyszerű Kihúzás paranccsal lehet létrehozni.

7.a. ábra *- Kihúzás parancs*

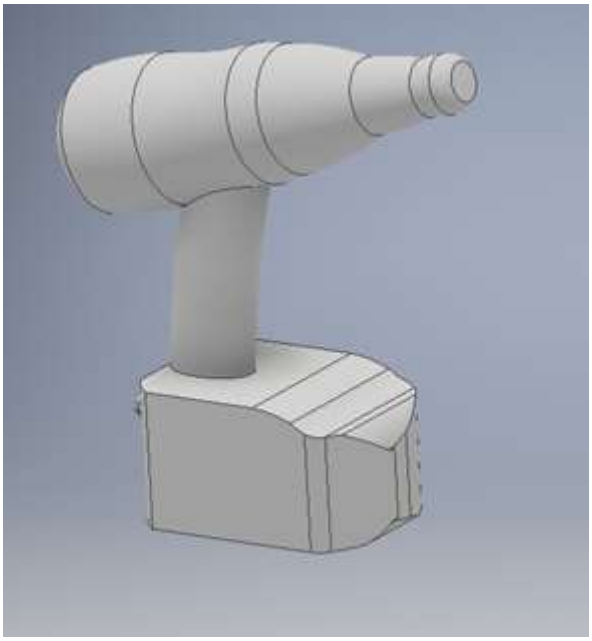
A következő lépés a fúrógép teteje volt. Mivel ez egy forgásszimmetrikus része a fúrógépnek, ezért a Forgáskihúzás paranccsal lehet létrehozni. Egy vázlatot kell létre hozni a fúró XZ síkjára. Az alakelem forgásközéppontja szögben el van döntve.

7.b. ábra *- Forgáskihúzás parancs*

Ezután az akkumulátor részén kellett javítani. Felülről egy Kivágás paranccsal lehet kivágni a megfelelő méretűre, majd a Lekerekítés paranccsal lekerekíteni az éleket.

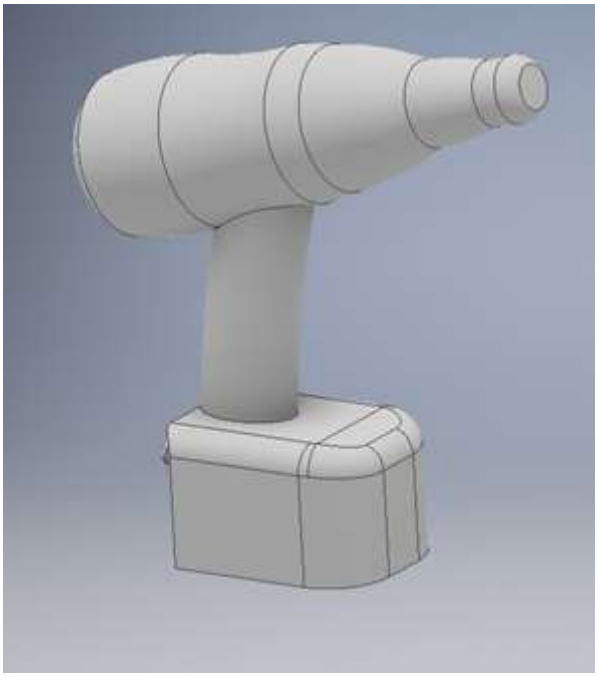
7.c. ábra *- Kivágás és Lekerekítés parancs*

A következő lépés a markolat kialakítása volt. Ezt Átvezetett kihúzással lehetett létrehozni. Először is meg kell adni egy útvonalat vázlat segítségével, ami a fúrógép közepén levő síkon van, és egy görbe. A következő lépés pedig az Átvezetett kihúzás keresztmetszete volt. Itt egy ellipszis alakú a vázlat, és párhuzamos az akkumulátor tetején levő síkra. Ez után ezzel a két paraméterrel létre lehet hozni a markolatot.



7.d. ábra *- Átvezetett kihúzás parancs*

Ezt követően az akkumulátort kellett ismételten finomítani. Pár Lekerekítést kell még rajta alkalmazni, valamint az arányok betartása miatt még egy Kivágást kellett alkalmazni.

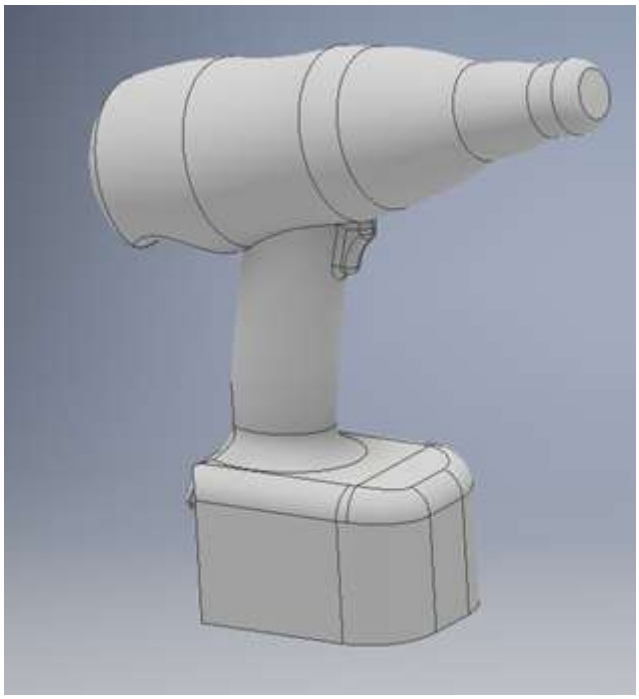


7.e. ábra *- Kivágás és lekerekítés parancs*

Ezt követően jön a nyomógomb, amit a fúrógép közepén levő síkra megrajzolt vázlatból lehet kihúzni, szimmetrikusan. Ezután ezt is le kellett kerekíteni.

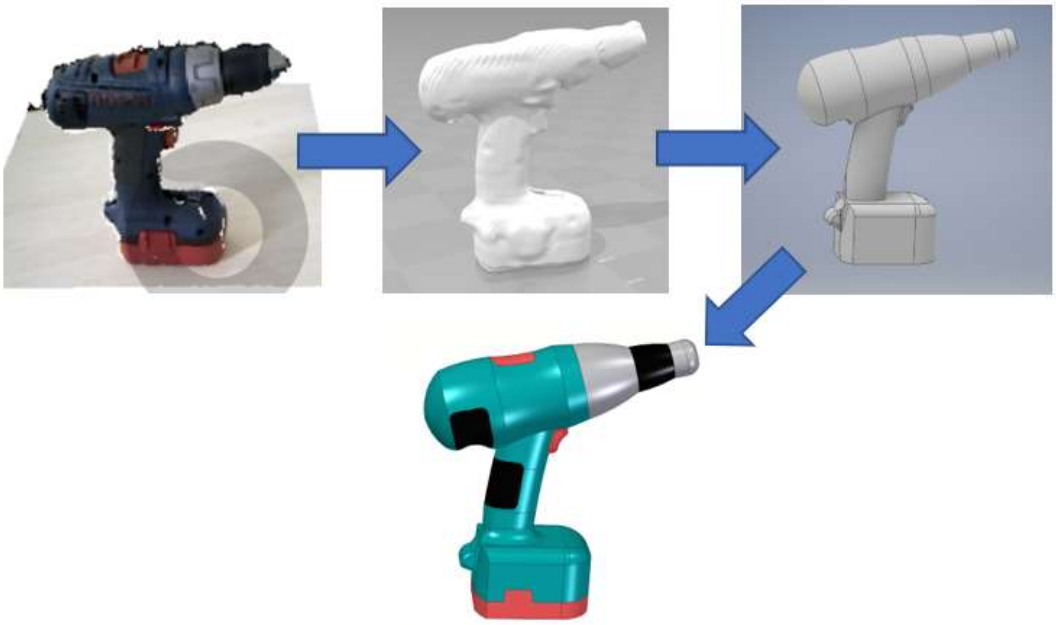
7.f. ábra *- Kihúzás és Lekerekítés parancs*

Ezután a fúrógép tetejét kellett átalakítani. A hátulján levő Kivágást az előzőhez hasonlóan a fúrógép közepén levő síkra megrajzolt vázlatból lehet kivágni, valamint a markolat és az akkumulátor találkozását kell lekerekíteni. Ezzel és még pár apróbb lekerekítéssel készen is lett a teljes modell.



7.g. ábra *- Kész modell Inventorban*

A kész modellt még be kellett színezni a valós modell alapján, ez is elvégezhető a legtöbb 3D CAD szoftverben, de vannak erre alkalmas más szoftverek is.



7.h. ábra *- Reverse engineering folyamata példán keresztül*

A feladat során bemutattuk a reverse engineering folyamatát leegyszerűsítve. Természetesen ipari környezetben pontosabb szkennert használnak, valamint az újra modellezés során sok felületmodellezést alkalmaznak.

8. 3D SZKENNELÉS ISMERTETÉSE

A 3D szkennelést főleg akkor alkalmazzák a feladat hatékonyságának javításában, amikor nem egyszerű geometriai modellről van szó, hanem egyszerű geometriai elemekből nem felépíthető modellről. Ekkor a szkennelt modell egyfajta háromdimenziós sablonként szolgálva megkönnyíti a mérnökök munkáját, valamint egy már kész módosítható modellként is lehet már rá tekinteni. [24]

A szkennelés végeredményeként létrejövő 3D modell lehet textúrázott vagy textúra nélküli *mesh* háló. Egy ember szkennelése körülbelül 5-15 percet vesz igénybe a szükséges útmutatással együtt. Maga a szkennelés körülbelül 2-3 perc. Tárgyak szkennelése esetén a folyamat több nap is lehet, amennyiben a tárgyat újra kell modellezni. [24]

A 3d szkennelés célja, hogy egy olyan pontfelhőt kapjunk, amely alkalmas a 3d modell létrehozásához, metszetek készítéséhez, pontos dokumentáláshoz, rekonstrukcióhoz, mérések elvégzéséhez, 3d nyomtatás előkészítéséhez, szabálytalan alakzatok térfogatszámításához. [24]

Leggyakoribb felhasználási területek:

- építészet
- iparialkalmazás (jármű-, gép-, eszközgyártás)
- reverse engineering (mérnöki rekonstrukciós visszafejtés) [25]

8.1. Szkennerek típusai

A kontaktszkennerek és a 3D koordinátamérő rendszereket már nagy gyakorisággal használják az iparban, mint minőségellenőrzési eszköz. Ezek az eljárások gyorsak és mikronpontosak, így megfelelő mérési adatokat adnak az ipari folyamatok optimális végrehajtásához és ellenőrzéséhez. Ezeknek az eszközöknek a beszerzési ára is igen magas, így ezeket kizárólag csak ipari felhasználásra használják. Amennyiben nincs szükség 1-2 mikronos pontosságra a feladat elvégzéséhez, léteznek alacsonyabb beszerzési árú szkennerek is, például a 10-20 mikron pontosságú telepített 3D szkennerek vagy az 50-100 mikron pontosságot nyújtó kézi 3D szkennerek. [26]

A kontaktszkennerek tapintófejének ára 80 ezer eurónál kezdődik, a telepített szkennereké 20-30 ezer euró, a kézi szkennereké pedig 5-20 ezer. Természetesen ezek az árak is az ipari felhasználású szkennerekre vonatkozik. Nem ipari felhasználásra pedig léteznek forgóasztalos és kézi szkennerek. A kézi szkennerek pontossága 1mm, míg a forgóasztalos

szkennereké 0,1mm. Mint látszik ezek pontatlanabbak, mint az ipari szkennereké, viszont ezek akár magánszemély számára is beszerezhető árú szkennerek, valamint nem ipari felhasználásra a pontosságuk is megfelelő. A megfelelő választást természetesen mindig az elvégzendő feladat határozza meg. [105]

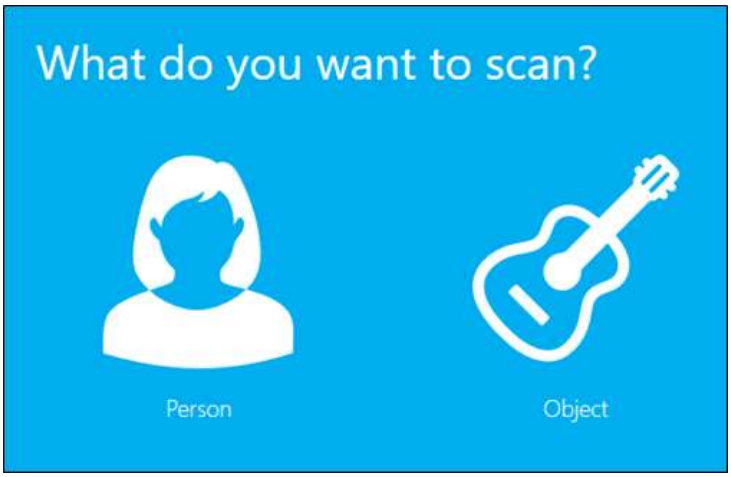
8.1.1. Sense 3D szkenner

A *Sense 3D* [27] szkenner egy viszonylag gyors, egyszerű, hordozható és praktikus szkenner, ami színes 3D-s szkennelésre is képes. Általában otthoni, iskolai és üzleti célokra használják. Ez a szkenner található meg az iskolánkban is. Előnye az alacsony beszerzési ár, a hordozhatóság és az egyszerűség, hátránya az 1mm-es pontosság. Hasznos funkciója, hogy lehet vele embert is szkennelni, akár teljes embert, vagy "portré" módban csak az ember fejét. A szkenner a *RealSense* szenzort használja, ami tartalmaz egy infra fényforrást és 2 optikai szenzort, amelyek közül az egyik a visszavert fényből térbeli pontokat készít, a másik pedig a színeket érzékeli [27].

7.1.1. ábra *- Sense 3D szkenner* [28]

8.1.2. Sense 3D szoftver

A *Sense 3D* [29] szoftver működése meglehetősen egyszerű. A kezdőképernyőn először is ki kell választani, hogy embert vagy tárgyat szeretnénk szkennelni. Ehhez fogja később igazítani a pontosságot, a táegyfelismerést és a beállított alap méretet. Ha az embert választjuk ki, akkor automatikusan ki lesz kapcsolva a tárgy felismerés, ugyanis akkor már adott, hogy mit szkennelünk be. Valamint ezután választhatjuk ki, hogy teljes testet, vagy csak arcot szeretnénk szkennelni. Ez alapján fogja meghatározni a beszkennelhető térfogat nagyságát. Tapasztalataink alapján az arc szkennelés módban a legpontosabb az eszköz. A pontosság minden esetben függ a beszkennelhető térfogat nagyságától. A 0,5*0,5*0,5 m vagy kisebb térfogat a legmegfelelőbb pontosság szempontjából [29].



8.1.2.a ábra *- Sense 3D szoftver kezdőképernyő* [29]

Amennyiben a tárgyat választjuk ki, a következő ablakban a tárgy méretét lehet majd kiválasztani. A kis méret itt is a legpontosabb. Ebben az esetben automatikusan bekapcsol a tárgyfelismerés, ez azt a célt szolgálja, hogy pontosabb legyen a szkennelés. Abban az esetben kell ezt választani, ha nem túl bonyolult tárgyat szkennelünk, egyéb esetben a tárgyfelismerés pontatlanabb szkennelést eredményez.

8.1.2.b ábra *- Sense 3D szoftver tárgy szkennelése* [29]

9. BIOLÓGIAI TÁRGYAK BESZKENNELÉSE

9.1. Megfelelő környezet és modellek kiválasztása

A szkennelésnél nagy figyelmet kell fordítani a szkennelési környezet kialakítására is. A legmegfelelőbb, ha egy síkfelületre helyezzük a beszkennelni kívánt tárgyat, mivel a szoftver automatikusan felismeri majd a síkfelületet és azt nem fogja beszkennelni, így később kevesebbet kell majd módosítani rajta. Ez csak abban az esetben teljesül, ha be van kapcsolva az automatikus tárgyfelismerés. A tárgyat úgy kell elhelyezni a síkon, hogy az a része legyen alul, amelyik a legegyszerűbb, ugyanis a tárgy alját a legnehezebb beszkennelni. A biológiai modellek szkennelése közben legtöbbször szükségünk volt állványra is, így egyszerűbben tudtuk beszkennelni a tárgyak alját is.

9.1.a ábra *- Szív modell állványon*

Fontos tényező még a fényviszony is. Meg kell találni a megfelelő fényviszonyt, ugyanis a túl sok és a túl kevés fény is pontatlansághoz vezet. A tapasztalataink alapján a legpontosabb szkennelést akkor értük el, amikor a lámpák fénye nem világított, csak az ablakon bejövő fény világította meg a testet. Ha túl erős a fény, akkor a szkenner nem érzékeli azt a felületet amelyik túl fényes, kevés fényben pedig pontatlanabb a folyamat.

Fontos még, hogy a szkennelés közben folyamatosan figyelni a kell a képernyős a folyamatot. Mivel kábellel van összekötve a laptoppal a szkenner, ezért ehhez mérten kell kialakítani a szkenneléshez használt asztalokat is.

9.1.b ábra *- Szkenner pontatlansága*

Olyan modelleket kellett választani, amelyek megfelelő szemléltetőeszközök, de figyelnünk kellett arra is, hogy a szkennerünk adottságaival megfelelően betudjuk szkennelni a tárgyakat. Olyan tárgyak, amik túl aprólékosak és nagyon kis részekből állnak, nem szkennelhetők be a *3D Sense*-el. Felületmodellezéssel elkészíthető modelleket megfelelően szkenneltünk, így a választott tárgyaink főként belső szervek és hasonló szemléltető eszközök voltak.

9.2. Szkennelési folyamat

9.2.a ábra *- Tárgy szkennelése*

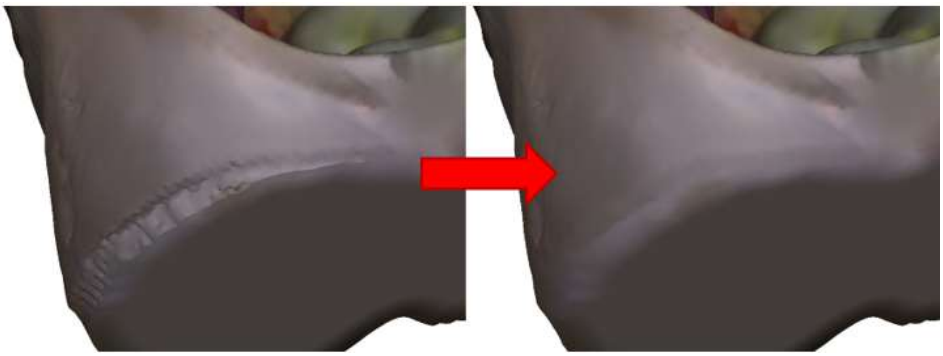
A megfelelő tárgyak és környezet kiválasztása után kezdődhetett a szkennelési folyamat. Mivel a tapasztalok azt mutatták, hogy a legpontosabb eredményt a tárgyfelismerés kikapcsolása eredményezte, így úgy kezeltük a tárgyakat, mintha egy ember fejét szkennelnénk be. Ezután a térfogatot beállítottuk a megfelelő méretűre. Mivel ez kisebb volt mindig, mint az alapbeállítás, ezért még ezzel is javítottunk egy kicsit a pontosságon.

9.3. Beszkennelt modellek szerkesztése és színezése

A szkennelés után ki kellett javítani a hibákat, valamint javítani a minőségen. Mivel a tárgyfelismerést kikapcsoltuk a pontosság javítása érdekében, így az asztal felületét is beszkenneltünk a tárggyal együtt. Ezt, és a szkennelés pontatlanságából adódó egyéb nem kívánatos felületeket ki kellett törölni.

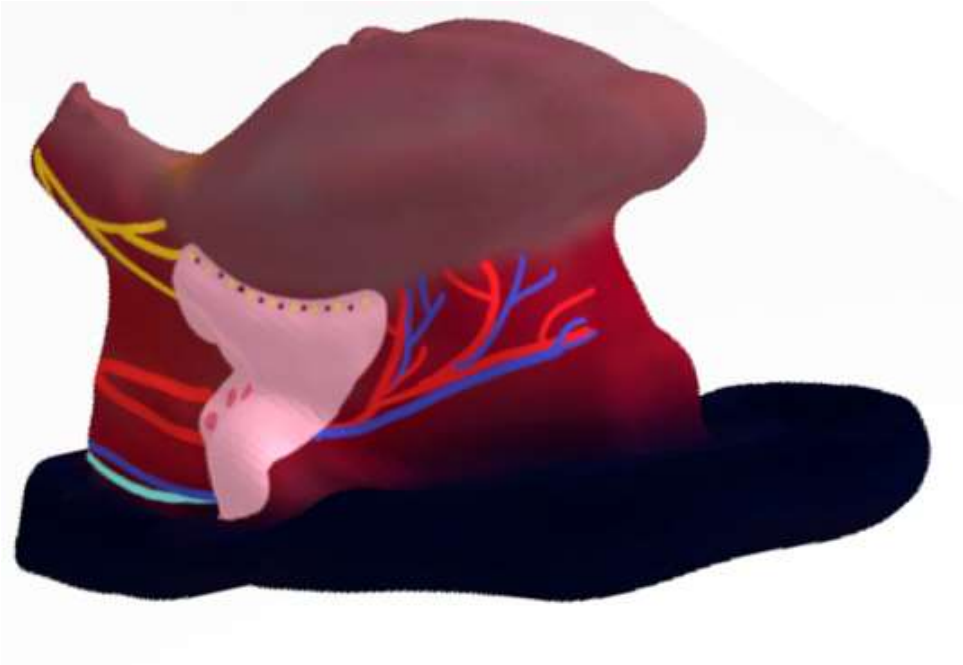
***9.3.a ábra** - Felesleges felületek levágása*

Mivel a törlés egy “lyukas” felületet eredményezett, ezért testet kellett létrehozni, ami összeközi ezeket a felületeket. Ezután ezeket még javítani kellett, mivel ezek nem valósághűen kötik össze ezeket a felületeket. A pontatlanságakat is ki lehet javítani, úgy, hogy “hozzásimítjuk” a modellhez a kiálló részeket. A kész modelleket *.fbx* és *.obj* formátumokban mentettük le.



9.3.b ábra *- Felületek javítása*

Mivel a szoftver a színeket nem tökéletesen érzékeli, ezért szín nélkül mentettük el a modelleket, hogy azokat később magunk színezzük ki. A modelleket *Windows 3D Paint*-ben színeztük ki, a lefényképezett tárgyak alapján.



9.3.c ábra *- Kész modell kiszínezve*

10. Összegzés és további tervek

Dolgozatunk során feltérképeztük a kiterjesztett valóság és a 3D szkennelés adta technológiai előnyöket és hátrányokat. Kutatásaink és kísérletezéseink során megtapasztaltuk, hogy az általunk használt 3D szkenner nem használható univerzális célokra. A kiterjesztett valóság szoftver- és hardverfüggősége a jól megválasztott, *Vuforia* AR-megjelenítő könyvtár esetén is jelentős teljesítménybéli eltéréseket mutat.

Az alkalmazás továbbfejlesztésének szempontjából elsősorban a végfelhasználók visszajelzésére lenne szükségünk. E célból szeretnék olyan workshopokat, találkozókat szervezni, ahol a diákok és a tanárok is interaktívan kipróbálhatják az alkalmazást és valós visszajelzéseket tudnak adni.

A 3D szkennelés minőségi javítása is fontos cél a megfelelő élmény átadásának szempontjából.

11. IRODALOM JEGYZÉK

[1] MILGRAM, P., TAKEMURA, H., UTSUMI, A. (1994). Augmented Reality: A class of displays on the reality-virtuality continuum. Proceedings of Telemanipulator and Telepresence Technologies (2018-11-04)

[2] https://www.colocationamerica.com/blog/history-of-augmented-reality (2018-11-04)

[3] http://thedigitalage.pbworks.com/w/page/22039083/Myron%20Krueger (2018-11-04)

[4] http://www.idemployee.id.tue.nl/g.w.m.rauterberg/presentations/hci-history/tsld096.htm (2018-11-04)

[5] http://www.augment.com/blog/infographic-lengthy-history-augmented-reality/ (2018-11-04)

[6] https://www.si.com/edge/2015/01/29/behind-nfl-yellow-first-down-line-sportsvision-technology (2018-11-04)

[7] http://www.hitl.washington.edu/artoolkit/ (2018-11-04)

[8] https://www.microsoft.com/hu-hu/hololens (2018-11-04)

[9] https://www.apple.com/hu/ios/app-store/ (2017-11-02)

[10] http://www.businessinsider.com/windows-monopoly-is-getting-destroyed-2013 (2017-11-02)

[11] https://www.theverge.com/2017/10/9/16446280/microsoft-finally-admits-windows-phone-is-dead (2017-11-02)

[12] https://stackoverflow.blog/2017/03/09/developer-hiring-trends-2017/ (2017-11-02)

[13] https://developers.google.com/ar/ (2018-11-04)

[14] http://storage.googleapis.com/play_public/supported_devices.csv (2018-11-04)

[15] https://developer.apple.com/arkit/ (2018-11-04)

[16] https://www.vuforia.com/ (2018-11-04)

[17] https://developer.android.com/studio/ (2018-11-04)

[18] https://developer.apple.com/xcode/ (2018-11-04)

[19] https://unity3d.com/ (2018-11-04)

[20] https://www.pubnub.com/ (2018-11-04)

[21] https://insights.globalspec.com/article/7367/how-does-reverse-engineering-work (2018-11-04)

[22] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3869153/ (2018-11-04)

[23] https://slideplayer.hu/slide/2124848/ (2018-11-04)

[24] http://www.freedee.hu/a-non-kontakt-3d-szkennerek-helye-a-gyartasban/ (2018-11-04)

[25] http://www.freedee.hu/3d-szkenneles/ (2018-11-04)

[26] http://www.freedee.hu/a-non-kontakt-3d-szkennerek-helye-a-gyartasban/ (2018-11-04)

[27] https://www.3dsystems.com/shop/sense?redirectFrom=cubify (2018-11-04)

[28] https://www.idig3dprinting.co.uk/shop/brand/sense-3d-scanner/ (2018-11-04)

[29] https://www.3dsystems.com/ (2018-11-04)

[30] https://www.autodesk.com/products/inventor/overview (2018-11-04)