

Óbudai Egyetem
Neumann János Informatikai Kar
Alkalmazott Informatikai Intézet



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

ÉLŐ TESTVIZUALIZÁCIÓ MOBIL KÖRNYEZETBEN

Szerző: **Hermann Zoltán**
mérnök informatikus BSc. szak, VII. évf.

Konzulensek: **Dr. Póser Valéria**
egyetemi docens
Mátételki Péter
MTA SZTAI
vezető fejlesztő

Budapest, 2015.

Tartalomjegyzék

1. Bevezetés	3
2. A kézfej megismerése	4
2.1. Anatómiai síkok	4
2.2. A kéz részei	5
2.2.1. Kéztő	6
2.2.2. Kézközép	6
2.2.3. Ujjak	6
2.3. Ízületek mozgása	7
2.3.1. Csukló	8
2.3.2. Kéztő	8
2.3.3. Kéztő- és kézközépcsontok közötti ízület	9
2.3.4. Kézközép-ujjízületek	9
3. A jelnyelv bemutatása	9
3.1. Jelnyelvről általában	10
3.2. Tipológia	10
3.3. A daktil jelnyelv	11
4. A meglévő alkalmazás bemutatása	13
4.1. Mozgásfelismerési technikák	13
4.2. A Tolmácskesztyű bemutatása	14
4.2.1. Architektúrális felépítés	15
A Kesztyű modul	16
A Betanító Környezet modul	16
A Mobilalkalmazás modul	16
A Backend Szerver modul	17
4.2.2. Kezelőfelület	18
5. Megvalósítás	21
5.1. Fejlesztőkörnyezetek összehasonlítása	21
5.2. Programtervezés	24
Game1 osztály	25
AnimationClip osztály	25
AnimationPlayer osztály	25

<i>Keyframe</i> osztály	25
<i>SkinningData</i> osztály	25
5.3. Implementáció	25
5.3.1. Blender	26
5.3.2. A kézmodell ellenőrzése	29
5.3.3. Microsoft XNA Game Studio 4.0	30
5.3.4. Microsoft - MonoGame.....	32
5.3.5. Integráció	37
5.4. Szoftver tesztelése.....	38
5.4.1. Szintaktikai ellenőrzés	38
5.4.2. Szemantikai ellenőrzés.....	38
5.4.3. Szoftver tesztelése emulátoron és mobileszközön	38
5.5. Végeredmény	41
5.6. Továbbfejlesztési lehetőségek.....	42
6. Összefoglalás	43
7. Abstract	44
8. Irodalomjegyzék	45
9. Ábrajegyzék.....	47
10. Táblázatjegyzék	47

1. Bevezetés

Az ember társas lény, ezért társadalmunk egyik legfontosabb eleme a kommunikáció. A beszéd- és halláskárosultak számára nem könnyű megfelelni és eleget tenni a mindennapi kötelezettségeknek. Természetesen ez nem az Ő hibájuk, egyszerűen nincs megfelelő eszközük a gondolataik közvetlen kifejezésére. Noha írásban, vagy jelnyelv használatával kapcsolatba tudnak lépni másokkal, a jelnyelvet az épek többsége nem ismeri, nem tudják használni. Az informatika fejlődése és a mérnöki találékonyság lehetővé teszi egy olyan eszköz kifejlesztését, aminek a segítségével ledönthetőek a falak a jelnyelvet nem ismerők és az azt alkalmazók között.

Az MTA SZTAKI munkatársai, egy úgynevezett Tolmácskesztyű segédeszközt fejlesztettek ki. Ez egy olyan hardver-szoftver rendszer, ami egy kézmozgást érzékelő kesztyűből és a kézjeleket felismerő és nyelvi feldolgozást végző szoftverből áll. Ez a speciális kesztyű tizenkét darab kilenc-tengelyes integrált mozgáskövető szenzorral van felszerelve, melyek egy-egy háromtengelyű gyorsulásmérőt, egy háromtengelyű giroszkópot és egy háromtengelyű magnetométert tartalmaznak. A kesztyűben lévő kéz mozgásának hatására keletkezett adatfolyamot képes vezeték nélkül továbbítani a mobil eszköz számára. A szoftvert Android és Windows Phone 8 operációs rendszerrel ellátott készülékekre telepíthetjük. Az alkalmazás az eljelelt karaktereket, szavakat megjeleníti a képernyőn, és ha szeretnénk fel is olvassa azokat. Előnye, hogy valós időben tudja elvégezni a fordítást, így nem kell külön jelnyelvi tolmács, hiszen a felek azonnal megértik egymást.

A programot egy élő testvizualizációra képes résszel még barátságosabbá lehet tenni. A cél, hogy az alkalmazást úgy fejlesszük tovább, hogy az, egy háromdimenziós emberi kézfejet mozgasson. A digitális kéznek – valós időben – követnie kell a kesztyűben lévő kéz mozgását. Mivel az alkalmazás a daktil jelnyelv kéztartásait ismeri, ezért elegendő elsősorban egy jobb kézfejet megjeleníteni. Ez a kiegészítés nem csupán a fejlesztők számára hasznos – leegyszerűsíti a hibák feltárását, mivel látják, hogy egy adott bemenetre milyen mozgást végez a kézfej –, hanem a felhasználóknak is. Gondoljunk csak bele, mennyivel gyorsabbá és szórakoztatóbbá válik így a jelnyelv elsajátítása! Fontos megjegyezni, hogy különösen ügyelni kell az implementálásnál az esetleges helytelen adatfolyamok kiszűrésére. Ha ezt nem tesszük meg, azon felül, hogy rendellenes mozdulatokat jelenít meg az applikáció, félrevezetheti a felhasználóját. Éppen ezért elengedhetetlen, hogy megismerkedjünk a kézfej anatómiájával, kinematikájával. Továbbá a dolgozatban központi szerepet tölt be a daktil jelnyelv főbb jellemzőinek leírása, valamint a már elkészített szoftver működésének bemutatása, feldolgozása.

Csak ezek összessége segítheti elő a megfelelő fejlesztői környezet kiválasztását, valamint a kézfej vizualizációjáért felelős modul megvalósítását.

2. A kézfej megismerése [1]

A későbbi fejlesztéshez elengedhetetlen, hogy tisztázzuk az anatómiában használatos síkokat és irányokat, mivel ezek alapján hivatkozunk majd a kéz egyes részeinek a mozgásakor.

A virtuális kézfej ugyanolyan felépítésű lesz, mint a valós, éppen ezért meg kell vizsgálni a csontokat, ízületeket.

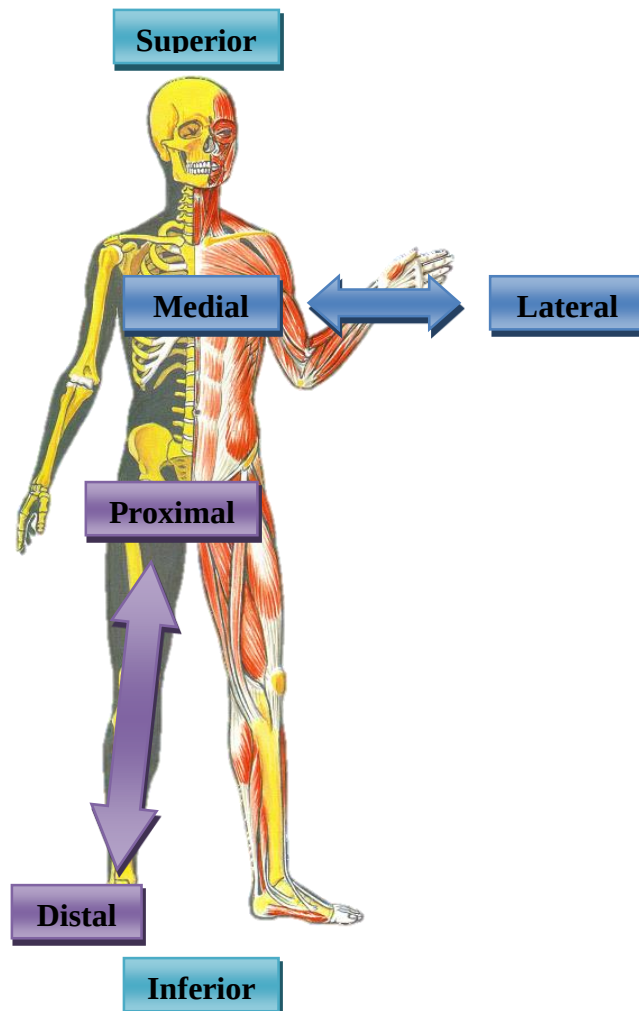
A kézfej kinematikai vizsgálata a helyes mozgástartományok meghatározására szolgál, továbbá segít a helytelen értékek kiszűrésére.

2.1. Anatómiai síkok

Az anatómiában megkülönböztetjük a nyíl- (sagittalis), a homlok- (frontalis) és a vízszintes irányú síkokat (horizontális). A nyílirányú középsík az emberi testet egy bal (sinister) és egy jobb (dexter) félre osztja. Az ezzel párhuzamos síkokat ugyancsak így hívjuk. A homlokirányú sík erre merőleges, a testet egy hátulsó (dorzális) és elülső (ventrális), illetve poszterior és anterior részekre osztja. A harmadik, horizontális sík, merőleges az előbb említettekre és a test hossz tengelyére is. Más néven a transzverzális sík a testet feji (kraniális) és farki (kaudális) részre osztja.

A következő ábra az anatómiában használt irányokat mutatja be:

1. felsőbb (superior) ⇔ alsóbb (inferior)
2. test középpontjához közelebb eső (proximal) ⇔ középponttól távolabb eső (distal)
3. oldalsó (lateral) ⇔ közbülső (medial)



1. ábra. Anatómiai irányok [2]

2.2. A kéz részei

Az emberi kézben 27 darab csont található. A kézfejet a kéztő (carpus), kézközép (metacarpus) és az ujjak (digiti manus) alkotják.

2.2.1. Kéztő

A kéztő nyolc darab kisméretű csontja két sorban ízesül. Ezek harántirányban egy közelebbi (proximalis) és egy távolabbi (distalis) sorba rendeződnek.

A proximalis sort képezik a következő csontok:

- sajkcacsont (os scaphoideum)
- félhold alakú csont (os lunatum)
- háromszögű csont (os triquetrum)
- borsócsont (os pisiforme)

A distalis csontsort alkotják az alábbiak:

- nagy trapézcsont (os trapezium / os multangulum majus)
- kis trapézcsont (os trapezoidum / os multangulum minus)
- fejescsont (os capitatum)
- horgascsont (os hamatum)

2.2.2. Kézközép

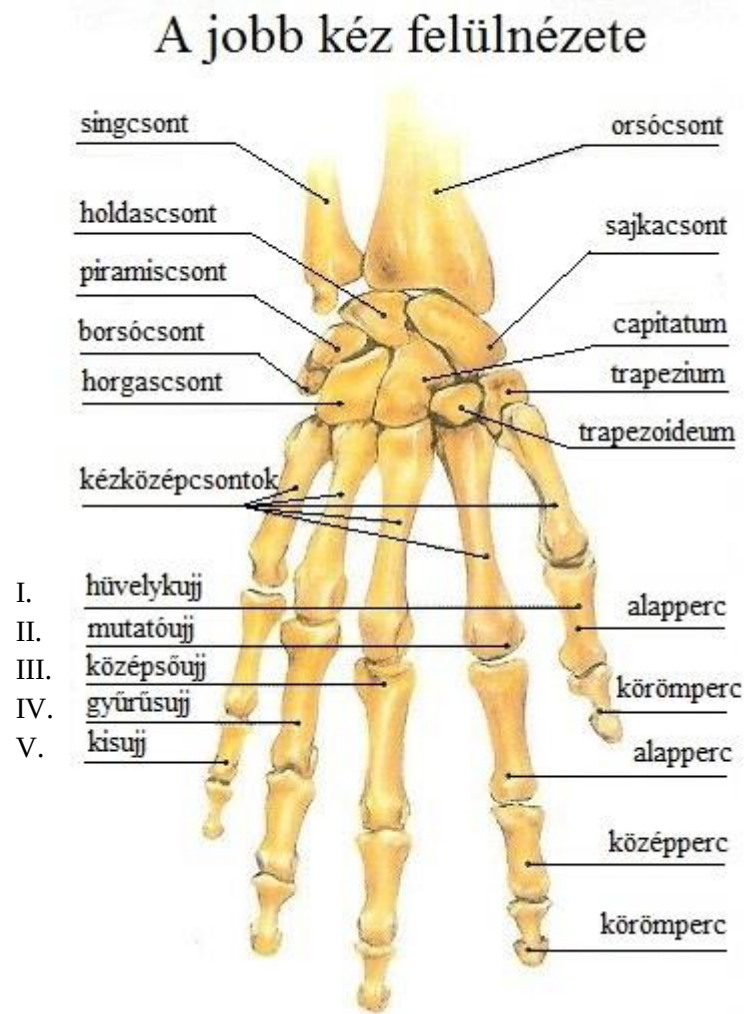
A kézközép öt darab kisméretű hosszúcsontból áll, amelyek szorosan kapcsolódnak egymáshoz és a kéztőcsontok distalis sora által alkotott ízfelszínekhez. A középcsontok distalis végén, kiváltképp a hüvelykujjnál, apró, lencse alakú csontok (ossa sesamoidea) figyelhetők meg.

2.2.3. Ujjak

Az ujjak (digi, a 3. ábrán római számokkal jelölve) csontjai úgynevezett sorokat (phalanx) alkotnak. Mindegyik három ujjperccsontból (körömperc-phalanx distalis, középperc-phalanx media, alapperc-phalanx proximalis) áll, kivételt képez a hüvelykujj, az mindössze kettőből (körömperc-phalanx distalis, alapperc-phalanx proximalis).

Az ujjak a 3. ábra alapján sorrendben:

- I. hüvelykujj (pollex)
- II. mutatóujj (index)
- III. nagyujj (digitus maximus seu medius)
- IV. gyűrűsujj (digitus anularis)
- V. kisujj (digitus minimus seu quintus)



2. ábra. A kézfej csontjai [3]

2.3. Ízületek mozgása [4]

Az ízületeket más-más mozgástartományok és elnevezések jellemzik. A mozgításba olykor több ízület is bekapcsolódik, ezért nem lehet teljes mértékben szétválasztani és külön tárgyalni ezeket.

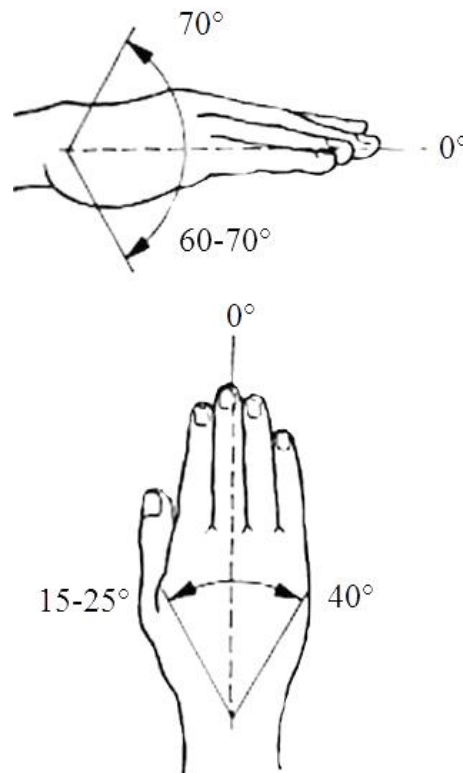
Ezért javasolt a következő felosztás:

1. Csukló ízület
2. Kéztő ízület
3. Kéztő- és kézközépcsontok közötti ízület
4. Kézközép-ujjízületek

2.3.1. Csukló

A csuklóízületet (articulatio radiocarpea) egy kéttengelyű – más néven tojásízület - alkotja. Ennek köszönhetően tengelyenként kétirányú mozgásra képes. Az első a radioulnaris tengely menti hátra- (dorsalflexio - 70°) és előrehajlítás (volarflexio - $60-70^\circ$), ami maximum $150-160^\circ$ -ot zárhat be. Ez a teljes kitérés, mikor más ízületek is bekapcsolódnak.

A második, amit a 3. ábrán is láthatunk, a dorsovolaris tengely körüli radiális távolítás (radialis abductio - $15-25^\circ$) és az ulnaris távolítás (ulnaris abductio - 40°).



3. ábra. Csuklóízület mozgástartománya

2.3.2. Kéztő

A kéztőcsontok két sorát S-alakú ízületi felszínek kapcsolják egymáshoz. A kéztőcsontok közötti ízületeket articulatio intercarpea-nak hívjuk. Feszés ízület, ezért minimális mozgással rendelkezik, viszont nagyban közreműködik a csukló mozgásában.

2.3.3. Kéztő- és kézközépcsontok közötti ízület

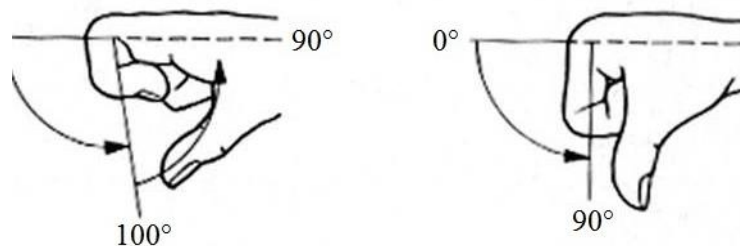
A közös kéztő-kézközép ízületek (articulatio carpometacarpeae) – hasonlóan a kéztőcsontokéhoz – feszes ízületek, ezért minimális mozgástartománnyal rendelkeznek.

A hüvelykujj a tenyér irányába befordítható, a többi ujjhoz közelíthető, szembefordítható. Ez az úgynevezett nyeregízületnek köszönhető és annak, hogy a II-V. kézközépcsontoktól eltérően az I. (os metacarpale pollicis) nem alkot ízületet a mellette lévő csontokkal. Mozgásának egyedisége abból is fakad, hogy más típusú ízülete van, mint a II-V. ujjaknak.

A nyeregízület közelítés (adductio), távolítás (abductio), hüvelykujj szembehelyezése a többi ujjal (oppositio) és visszahelyezés (repositio) mozgásokat tesz lehetővé.

2.3.4. Kézközép-ujjízületek

Az articulationes metacarpophalangeae mozgásai közé tartoznak a közelítés, távolítás – a III. ujjhoz viszonyítva –, és a hajlítás, feszítés, túlfeszítés. Míg az alapperceket csak 90°-ig lehet behajlítani, addig a középső ujjperceket kevéssel derékszögnél jobban, ahogy a 4. ábra is mutatja. A körömperc körülbelül 60°-ig hajlítható. A túlfeszítés (dorsalflexio), a kinyújtott ujjak hátrahajlítását jelenti. Habár a vizualizáció szempontjából lényegtelen, de a nők jobban hátra tudják hajlítani az ujjukat, mint a férfiak.



4. ábra. Ujjak mozgástartománya

Az ujjpercek közötti ízületek (articulationes interphalangeae) kétféle mozgásra képesek, hajlításra és feszítésre.

Az emberi kéz összesen 27 szabadságfokkal rendelkezik. Az ujjak három féle hajlító, nyújtó mozgásra és egy közelítő, illetve távolító mozgásra képesek, kivéve a hüvelykujjat – csak két hajlító, nyújtó mozgás –, mivel eggyel kevesebb ízület alkotja. A csukló összesen három szabadságfokkal bír. Ez azt jelenti, hogy például a hüvelykujj mozgását három darab változóval lehet definiálni. [5]

3. A jelnyelv bemutatása [6][7]

A vizuális kódrendszerek egyik legfejlettebb formája a jelnyelv. A 60-as évektől a szakemberek önálló nyelvként kezelik, Magyarországon 2009-től pedig hivatalos nyelvként van jelen. Főként, de nem kizárólag siketek használják.

3.1. Jelnyelvről általában

Az adott hangzó nyelv nyelvtana jellemzi, rendelkezik sajátos hangtannal, alaktannal és mondattannal. Mivel szókinccse pontos nyelvtani szabályok szerint rendeződik, emiatt nyelvjárásai lehetnek. Magyarországon összesen hét nyelvjárást különböztetünk meg. Elsőre egyszerűnek tűnik az elsajátítása, viszont közelebbről megvizsgálva beláthatjuk, hogy nehézsége vetekszik a hangzó nyelvekével.

Annak ellenére, hogy hasonlóságokat tapasztalhatunk, országonként különböző jelnyelveket figyelhetünk meg. Habár létezik egy, a beszéd- és halláskárosultak számára kialakított nemzetközi jelnyelv, a gestuno, ez éppoly nehezen tud teret nyerni, mint az eszperantó. Ezt főként olyan emberek használják, akik gyakran lépnek kapcsolatba külföldi személyekkel.

Egy tévhit szerint, a jelnyelv a gesztusok miatt lassabb, mint a beszéd. Ez részben igaz, részben nem, hiszen nem csak az ábécé betűit foglalják magukba, hanem az egyre növekvő számú fogalmakat is.

Fontos, hogy néhány szabályt betartsunk jelelés közben. Először is, törekedni kell a jel pontos kivitelezésére, másodsor, ügyeljünk a gesztikulálásra, mimikára és a testtartásra. Vannak, akik a jelelést néma artikulációval kísérik a jobb érthetőség érdekében. Ekkor figyeljünk a lassabb beszédtempóra, ne essünk a túlartikulálás hibájába és végül mindig a jelelést kísérje a beszéd.

3.2. Tipológia

A jelnyelvek típusok szerint osztályozhatóak. Ennek megfelelően beszélhetünk ragozó, hajlító, poliszintetikus, inkorporáló és izoláló jelnyelvekről. Országonként eltérhet a szórend is. Például míg az amerikai alapszórend az alany-állítmány-tárgy, addig az osztrák alany-tárgy-állítmány. A hangzó nyelvekre nem hasonlítanak azok a jelnyelvek, amiket nem ért hangzó nyelvi hatás. Ezt jól bizonyítja az is, hogy az amerikai jobban hasonlít a japánra, mint az angolra.

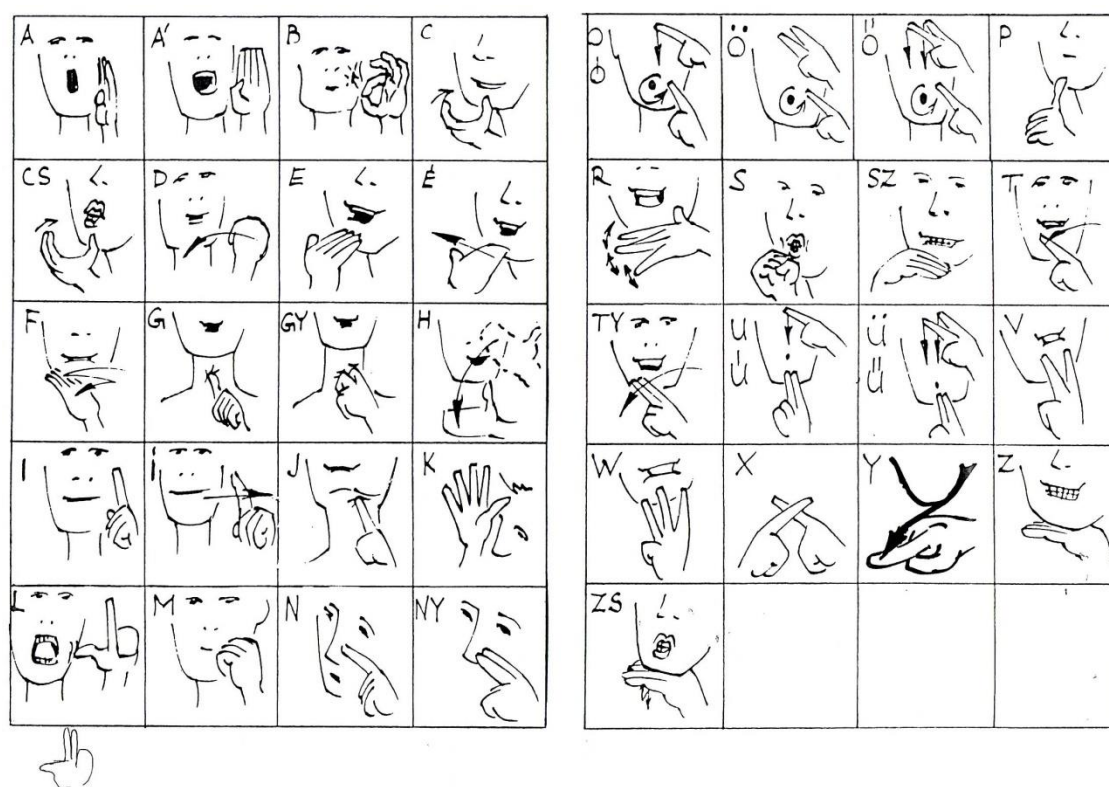
Ezeknek a nyelveknek az alapja a szimulákrum, azaz a mozdulattal felidézett mozgásos történés, vagy valaminek a jellemző mozgása, alakja, térbeli tulajdonsága. Egyszerű kifejezéseket használnak, elhagyva a felesleges toldalékokat, akár szavakat is. Például a „*Holnap majd elmegyek a boltba, veszek tejet, kenyeret.*” jelnyelvi fordításban „*Holnap fog menni vesz tej, kenyér.*”.

3.3. A daktil jelnyelv

A jelnyelvet használóknak is vannak saját argo szavaik, nem beszélve az egyedi, csak családban ismert és használt szókincsükről.

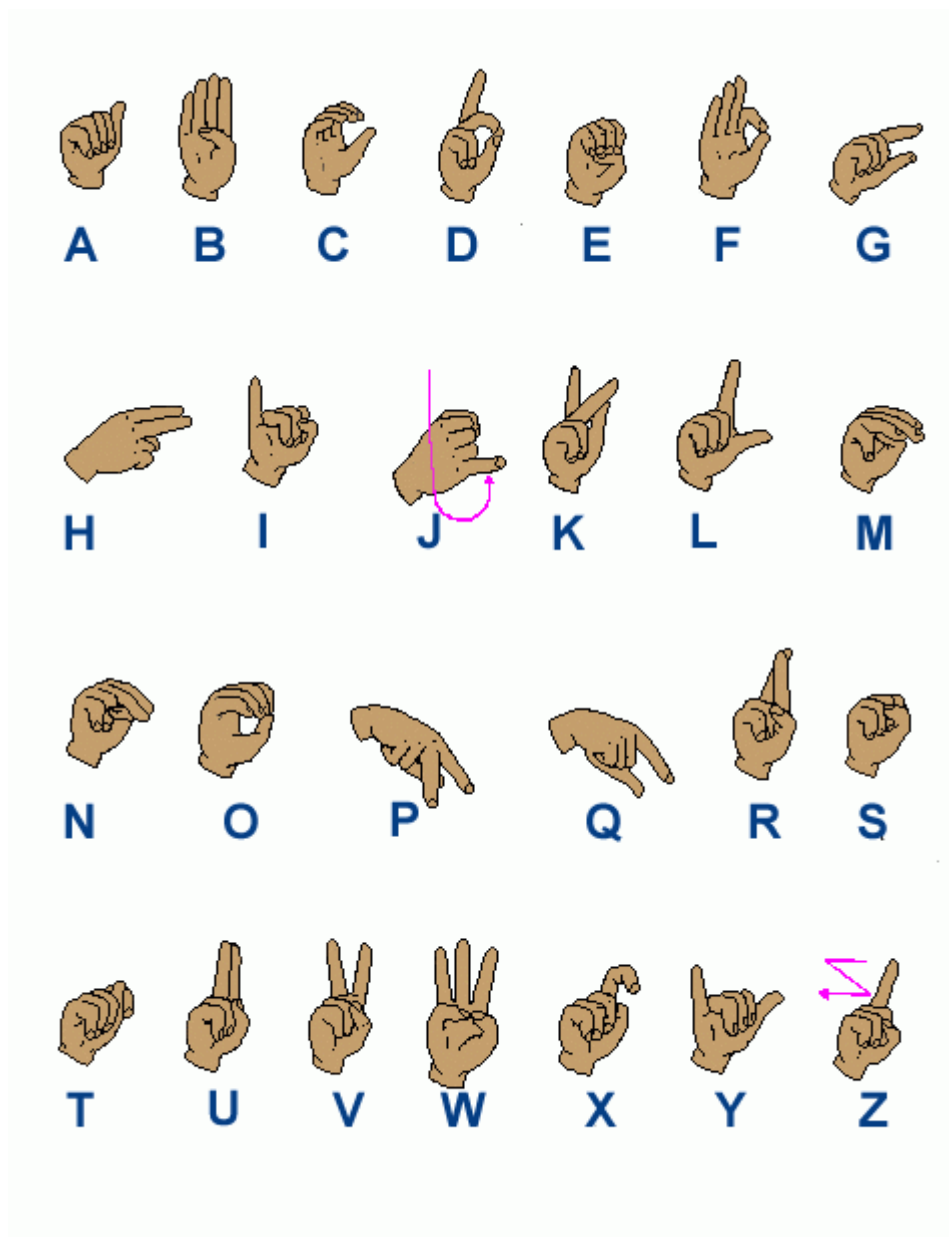
A daktil jelnyelv, vagy ujjábécé alapja, hogy egy betűhöz egy jelet rendel. Mivel ez csupán egy gesztuskészlet a hangokra, ezért nem jegyzik önálló nyelvként. A jeleket a kézfej és az ujjak megfelelő tartásával állítjuk elő. Előnyei közé tartozik, hogy a jeleléshez elegendő csak az egyik kezünket használni és nincs szükség egyéb non-verbális kommunikációra, mint például a mimika és a gesztikuláció. Jellemzi, hogy teljesen pontos, mindent ki lehet fejezni a segítségével, azonban a jelkészlete nyelvenként eltérő. Általában, ha egy fogalomnak nincs szimbolikus jele, ezt használják. Magyarországon a fiatalok körében elterjedőben van a daktil, míg a fonomimikai ábécét inkább az idősebbek alkalmazzák betűzésre.

A fonomimikai ábécé jelei a betű formájával, kiejtésével és képzésével kapcsolatos jellegzetességet mutatnak be és arra alkották, hogy a gyerekek írás- és olvasástanítását segítsék. Többek között emiatt is célszerű a daktil jelnyelv felhasználása a Tolmácskesztyű projektben, ott ugyanis csak az egyik kézfejet kell vizsgálni, nem pedig mindkét kezet és az arcot is. Az 5. ábra a fonomimikai ábécét mutatja be.



5. ábra. Fonomimikai ábécé [8]

Az alábbi ábrán észrevehető, hogy a daktil nem tesz különbséget a kis- és nagybetűk között.



6. ábra. A daktil jelnyelv ábécéje (angol) [9]

Ez a nyelv, a szöveg betűit a helyesírás szabályai szerint jelöli, figyelmen kívül hagyva azok kiejtését, emiatt a kommunikációban résztvevő feleknek ismerniük kell egy közös hangzó nyelvet, amit mindketten helyesen tudnak betűzni.

4. A meglévő alkalmazás bemutatása

A szoftver- és hardverrendszer bemutatása előtt a mozgásfelismerési technikákról is érdemes szólni. Mit nevezünk mozgásnak, hogyan lehet ezt észlelni, rögzíteni és feldolgozni.

4.1. Mozgásfelismerési technikák [10]

Egy objektum, ha megváltozik a pozíciója – egy adott vonatkoztatási rendszerben –, mozgást végez. Például egy mozgó járművön a vezető egy helyben ül, mégis megtesz egy bizonyos távolságot. Nem mindegy tehát, hogy mihez viszonyítjuk a testet, a járműhöz, vagy az aszfalthoz.

A mozgás észlelését szenzorok segítik. Ezeknek a feladata, nem villamos jelből (például termikus mennyiség, hőmérséklet) villamos jelet előállítani. Megkülönböztetünk aktív, illetve passzív érzékelőket. A különbség az, hogy az aktív szenzorok külső energiaforrást igényelnek, míg a passzív érzékelők e nélkül képesek működni.

Bár sokféle megoldás létezik a mozgás észlelésére (rádióhullám, kamera, infra, stb.), ebben a projektben a motion (mozgás/mozdulat) szenzor alapú detektálásra van szükség, mivel a hordozhatósága jelenti az egyik fő erényét.

A motion capture (digitális mozgásrögzítés, gyakran csak mocap) az objektumok mozgásának rögzítésére, térbeli elhelyezkedésük megállapítására szolgáló folyamat.

A folyamat során egy, vagy több aktort (aki a mozgást végzi) érzékelőkkel látnak el, majd nagysebességű videokamerával rögzítik a mozgást. Az érzékelők helyet foglalhatnak egy erre a célra kialakított ruhán, vagy akár közvetlenül a bőrön is. Az így keletkezett adatfolyamot egy háromdimenziós modellre tükrözik. Ennek hatására a digitális objektum is ugyanazokra a mozdulatokra képes, mint az azt produkáló aktor. Fontos megjegyezni, hogy ez az eljárás csak a mozgást rögzíti, az aktort nem. Előnyei közé tartozik, hogy a fejlesztői környezetben virtuális kamerákkal egészíthető ki a tér, így más szögekből is bemutatható a mozgás, sőt plusz animációkkal egészíthetjük ki a már meglévőt. Gyors, akár valós idejű feldolgozást tesz lehetővé, élethű, bonyolult mozgásokat rögzíthetünk segítségével.

Rengeteg területen alkalmazzák, például a videójáték gyártásban, animációs filmek készítésekor, vagy akár az orvostudományban. Mivel a Tolmácskesztyű szenzorjai pontosan meghatározzák az ujjak és a kézfej orientációját, ezért nincs szükség kamerára, azaz költséghatékonyabbá válik a munkafolyamat. Ennek köszönhetően mobilitásuk kitűnő, tömegük alacsony és a méretük csekély. Természetesen ennek a technikának is megvannak a maga hátrányai, például kisebb munkához nem feltétlenül kifizetődő a használata, gondoljunk a költséges kamerákra (marker-ek használata esetén), számítógépekre, valamint a szakemberek és az aktorok alkalmazására.

4.2. A Tolmácskesztyű bemutatása [11]

A Tolmácskesztyűt mind Windows, mind pedig Android operációs rendszert futtató mobil eszközre telepíthetjük.

Maga a kesztyű lélegző elasztikus anyagból készült, amelyen mozgáskövető szenzorok találhatóak, összesen tizenkét darab, amelyek biztosítják a kódolandó információ előállítását. Az érzékelők egy-egy háromtengelyű gyorsulásmérőt, egy giroszkópot és egy magnetométert tartalmaznak. Az abszolút háromdimenziós orientáció azonosítására jelfüziós algoritmusokat alkalmaz. Ezt felhasználva szenzorpáronként kiszámítják a relatív elfordulást (yaw), lehajlási szöget (pitch) és a billenési szöget (roll). Az ízületek behajlítottságát az érzékelő páronként kiszámolt relatív szögek adják meg.

Az alkalmazás saját gesztusleíró nyelvet használ, az úgynevezett HaGDiL-t (Hand Gesture Description Language), ami tartalmazza az emberi kézfej mozgásának és térbeli állapotának jellemzőit. A HaGDiL leíró tárolja a kézfej anatómiai jellemzőit: az ujjpercek párhuzamos és merőleges relatív helyzetét a tenyérhez viszonyítva, az ujjpercek behajlítottságát, a csukló helyzetét és a kézfej abszolút orientációját. Mindez húsz helyi értéken kerül tárolásra. Az egyedi nyelv biztosítja a felhasználó kézmozgásának a kódolását, így a szoftver tisztában van a kézfej és az ujjak aktuális helyzetével.

A rendszer, az érzékelők által előállított adatfolyamot megállás nélkül továbbítja – amennyiben állapota ezt megengedi – a mobil eszköznek, Bluetooth, vezeték nélküli adatkapcsolat alkalmazásával. [12]

A mobil eszköz, a jelfolyam fogadása után feldolgozza és értelmezi azt, annak érdekében, hogy a felhasználó által mutatott gesztussorozathoz tartozó szöveget rekonstruálhassa. A feldolgozás során először szükséges az adatfolyam szegmentálása, majd az így létrehozott nyers szöveg kontextus-érzékeny javítása, mely biztosítja az értelmes, lehetőleg hibamentes szöveg felolvasását.

A szegmentálásra azért van szükség, mert két jelformálás közti átmeneti állapot – vagyis miközben a felhasználó keze az egyik betűt formáló gesztusból átvált a másik betűt formáló jelre – nem tartalmaz releváns információt.

A kontextus-érzékeny javítási algoritmus, egy természetes nyelvi feldolgozást (*Natural Language Processing*) foglal magában. A korábban előállított nyers szöveget, ami nagy valószínűséggel tartalmaz hibákat, értelmes szöveggé formálja. Sajnos az általános helyesírás ellenőrző algoritmusok használhatatlanok, ugyanis az elgépeléstől eltérő hibák generálódnak a szegmentáció és a jelelés során. Éppen ezért figyelembe kell venni a kesztyű által készített gesztusleíró szemantikájából adódó speciális tulajdonságokat, a felhasználás tipikus jellemzőit és a beszélt nyelv sajátosságait. A nyers szöveget feldolgozó algoritmusban, az elvárásoknak megfelelően módosított Levenshtein

távolságmetrikát, n-gram adatbázis alapú keresést és igazságmátrixon (confusion matrix) alapuló súlyozási módszereket kell alkalmazni.

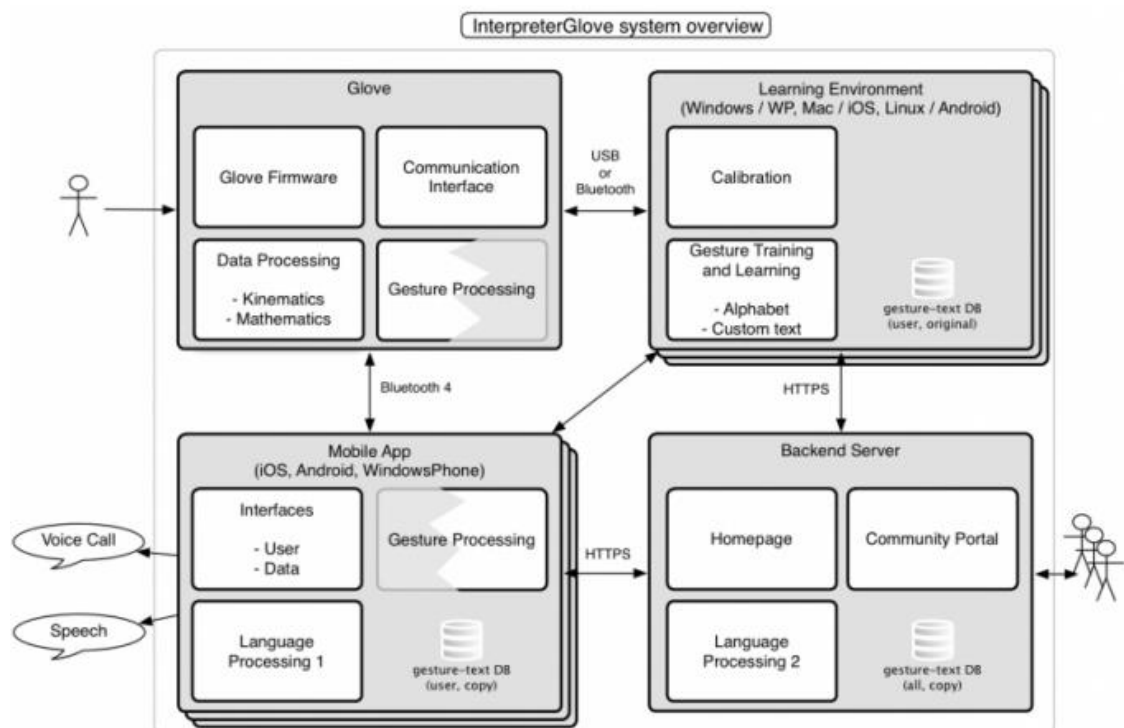
Mindkét folyamat offline (internet kapcsolattól független) módon, a szoftverbe integráltan, illetve online képes működni.

Mielőtt a felhasználó megkezdene a jelelést, szükséges a szoftver testreszabása, hogy a program fel tudja ismerni a használatára jellemző nyújtott, hajlított, nyitott, összehúzott ujjállásokat. Ez a később bemutatásra kerülő *Betanító Környezet* segítségével végezhető el.

4.2.1. Architektúrális felépítés

Az architektúrát a funkcionalitás szempontjából négy fő részre oszthatjuk (modulok), amit a 7. ábra szemléltet:

- Kesztyű
- Betanító Környezet
- Mobilalkalmazás
- Backend Szerver



7. ábra. Architektúrális felépítés [13]

A modulokat kisebb alapegységek, úgynevezett komponensek építik fel.

A Kesztyű modul

A kesztyű-hardveren futó szoftverek összességét fogja össze.

A kesztyűben található szenzorok vezérlését és a hálózati funkciók megvalósítását a *Firmware* (vezérlésért felelős program) komponens végzi.

A *Kommunikációs Interfész* biztosítja a kommunikációt a *Betanító Környezet* és a *Mobilalkalmazás* modulokkal. Az előbbivel Bluetooth 4 interfészen keresztül, az utóbbival Bluetooth, vagy kábeles interfészen keresztül kommunikál.

Az *Adatfeldolgozó* modul fogadja a szenzorokból érkező jeleket és irányvektorra alakítja azokat. Ehhez tartalmazza a kéz kinematikai modelljét, hogy a szükséges matematikai konverziókat elvégezhesse.

Amennyiben a jelfeldolgozás nem a mobil eszközön folyik, akkor az úgynevezett *Gesztusfeldolgozó* feladata lesz, hogy a kapott leírókból betűket állítson elő.

A Betanító Környezet modul

Ennek segítségével taníthatja be a felhasználó a kesztyű által – később – felismert gesztusokat, azokhoz hozzárendelve egy betűt, vagy karaktersorozatot.

Annak érdekében, hogy a betanítás „zajmentes” lehessen, a későbbi, minél pontosabb felismerés érdekében, el kell végezni a kesztyű és a szenzorok kalibrálását, ami az alaphelyzet megtalálásához szükséges.

A szoftver tartalmaz egy előre meghatározott gesztuskészletet, ami az ábécé betűit rendeli össze a daktil nyelv jeleivel, így a felhasználó új gesztusokat tud felvenni az adatbázisba. A *Gesztus Gyakorlás és Tanulás* komponens lehetővé teszi egy létező betűhöz, vagy karaktersorozathoz tartozó gesztus gyakorlását, testre szabását, vagy újradefiniálását. Magyarul, a felhasználó képes kifejezéseket, mondatokat, vagy akár hosszabb szövegeket is hozzárendelni egy-egy gesztushoz. Például „Köszönöm!”, „Jó napot kívánok”, „Egy havi felnőtt BKV bérletet legyen szíves!”.

A gesztus és szöveg összerendelése a felhasználó eszközén, a *Betanító Környezet* modulban jön létre. A *Gesztus-szöveg Adatbázis*, a *Gesztus Gyakorlás és Tanulás* komponens használata során kiegészülhet a felhasználó által bevitt új kifejezésekkel, fogalmakkal. Az adatbázis csak egy felhasználó adatait tartalmazza, amik a korábbi betanítás alapján jönnek létre.

A Mobilalkalmazás modul

Ez reprezentálja magát a szoftvert, amit a mobil eszközre telepíthet a felhasználó. A modul feladatai közé tartozik a szinkrontolmács szolgáltatás biztosítása, az interfészekon keresztüli kapcsolattartás a kesztyűvel, a szerverrel és a felhasználóval, továbbá gesztusfeldolgozást és nyelvi elemzést is végez.

A *Kesztyű* modul felé egy adatfogadó interfészt kínál, melyen keresztül képes vezérelni a kesztyűt (például a ki- és bekapcsolás), valamint fogadni az átalakított adatfolyamot.

A felhasználó számára beállítási, vezérlési és visszacsatolási célú felületet nyújt.

A *Gesztusfeldolgozó* komponens végzi a jelfeldolgozást, mely aggregációt és statisztikai adattisztítást végez az előfeldolgozás során. A *Kesztyű* biztosítja számára a gesztusleíró HaGDiL adatokat, ezeket feldolgozza, majd belőlük nyers szöveget állít elő a *Gesztus-szöveg Adatbázis* alapján. Azért nyers, mert helyessége sem morfológiailag (például hiányzó, felcserélt, vagy duplikált betűk, hiányzó szóközök, mondatjelek), sem szemantikusan nem garantálható. Ezeknek a hibáknak egy része a felhasználó miatt áll elő (helytelen nyelvtani ismeretek, hibás betűzések), másik része pedig az átmeneti állapotok mintavételezéseiből fakadóan. Az így elkészült szöveg metaadatokkal ellátott karaktersorozat, ahol annotációk segítik a további feldolgozást.

A fentebb említett hibák kijavítását a *Nyelvfeldolgozó 1* szegmens végzi, ami a természetes nyelvi feldolgozást végző komponenseket foglalja magában. Feladata, hogy a bemenetén kapott nyers szövegből morfológiailag helyes, szavakra és mondatokra bontott, a *Beszéd szintetizátor* által feldolgozható, emberek által értelmezhető szöveget alkosson. Mivel a *Beszéd szintetizátor* szorosan illeszkedik a nyelvfeldolgozó egységhez, ezért alkotórészének tekinthető. Az eljárás befejeztével, lexikai elemeket (szavak, szókapcsolatok, mondatrészek, mondatok) szolgáltat a kimenetén.

A *Mobilalkalmazás* is tartalmazza a korábban ismertetett *Gesztus-szöveg Adatbázist*, azzal a különbséggel, hogy az adatbázisban az adatok a *Backend Szerverről* származnak, vagy a kiindulási állapotot tükrözik.

A Backend Szerver modul

Mindegyik interneten elérhető komponens egy szerver foglal össze. Egy részük bárki számára publikusan elérhető, igénybe vehető (honlap, webszolgáltatások), a többi viszont csak megfelelő jogosultságokkal, autentikáció után (közösségi portál).

A *Közösségi Portál* lehetőséget biztosít a felhasználóknak, hogy saját, egyedi gesztusok és lexikai elemek cserélhessenek gazdát. Emellett az általános portálok funkcióit is magában foglalja.

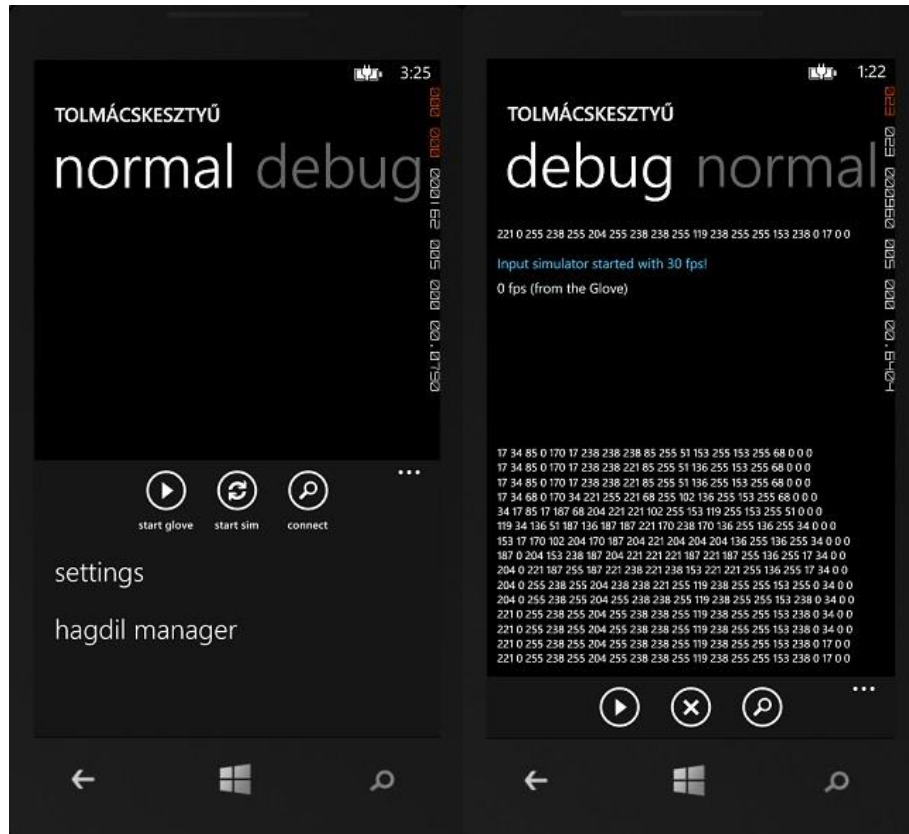
A *Honlap* célja az eszköz bemutatása, népszerűsítése és a kesztyű online megrendeléséhez szükséges felület nyújtása. Mindkét oldal akadálymentesített.

A *Nyelvfeldolgozó 2* rész leírása megegyezik a korábban említett *Nyelvfeldolgozó 1*-esével. A *Beszéd szintetizátorral* együtt a *Backend Szerveren* is helyet foglalhatnak, ezzel tehermentesítve magát az alkalmazást.

4.2.2. Kezelőfelület

A következő ábra egy Windows Phone 8.0 operációs rendszert futtató emulátoron mutatja be a programot.

A program indításakor a lenti képernyővel találkozhatunk (7. ábra, baloldal):



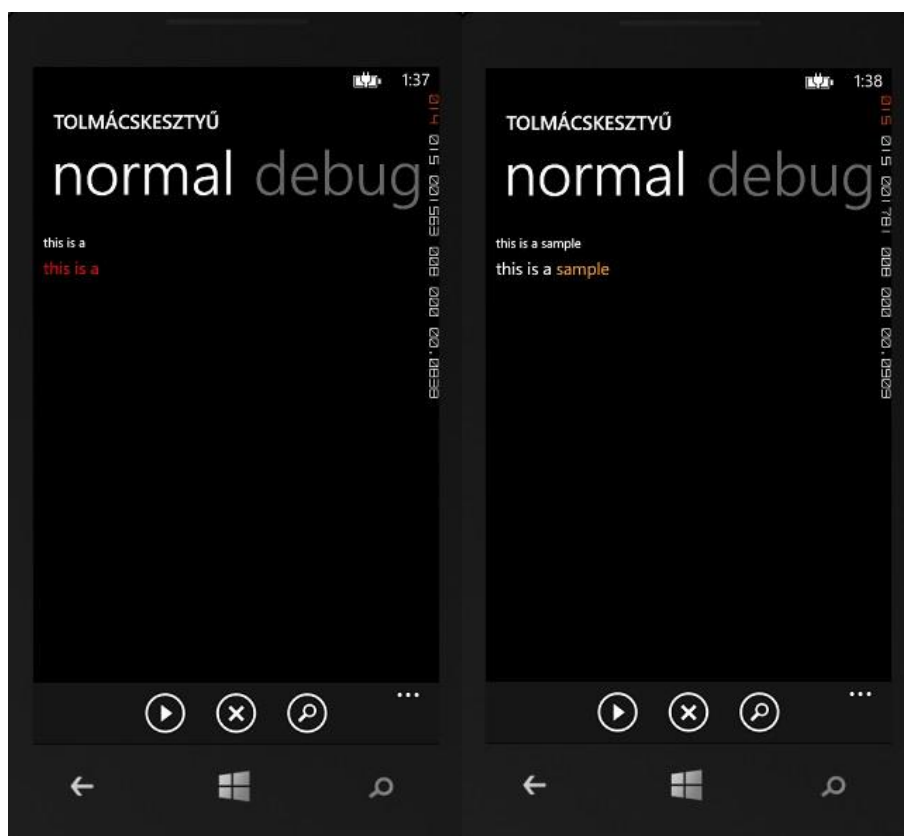
8. ábra. A Tolmácskesztyű szoftver kezelőfelülete

Az alsó három vezérlőgomb mindkét oldalon szerepel – „normal”, „debug” – az egyszerűbb kezelés érdekében. A „normal” oldal biztosítja a felhasználóknak szánt felületet, míg a „debug” oldal kizárólag a fejlesztők számára készült. A „start glove” megnyomása után a program fogadja a kesztyű által küldött adatfolyamot és feldolgozza azt. A „start sim” egy úgynevezett bemeneti szimulátort indít el. Célja, hogy ha nincs rendelkezésre álló, vagy működőképes kesztyű, akkor lehetővé tegye a tesztelést, és fejlesztést. A szoftver beolvas egy egyszerű – korábban a kesztyű által generált, majd az alkalmazás segítségével feldolgozott értékeket tartalmazó – szöveges fájlt, ezzel imitálva a kesztyű működését. Mindkét folyamatot egy újabb érintéssel lehet leállítani. Értelemszerűen egyszerre csak az egyik módszer alkalmazható. A harmadik gomb, „connect”, csatlakoztatja a mobilkészülékhez a kesztyűt. A „settings” és a „hagdil manager” egy-egy új ablakot nyitnak meg, amit lentebb ismertetek.

A „debug” első sora az éppen aktuális HaGDiL-okat mutatja, a második és harmadik a jeleket küldő technikát (jelen esetben a bemeneti szimulátor fut 30 keret/másodperces sebességgel, amit kék színnel emel ki a program) prezentálja.

Az alatta lévő sorok ugyancsak HaGDiL-okat ábrázolnak, annyi különbséggel, hogy itt láthatóak a korábban fogadott adatok is.

A 8. ábra működés közben ismerteti a „normal” oldalt:

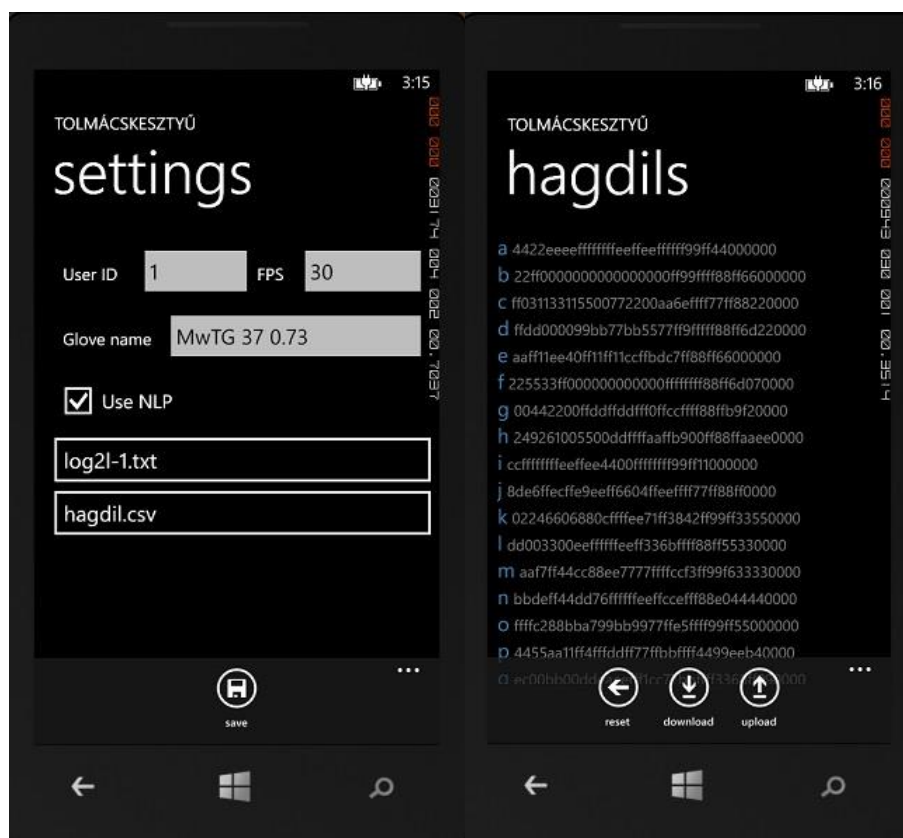


9. ábra. Normál ablak - szöveg

A kijelzőn különböző színű szavak jelennek meg. Fehérrel íródik ki permanens módon a feldolgozott karakter, amit az első sorban láthatunk. A következő sor(ok)ban sárga, vagy narancssárga színnel az aktuális szó tűnik fel, míg pirossal a Text-To-Speech (a 8.0-ás operációs rendszer olyan beépített funkciója, mely felolvassa a kijelölt szöveget) segítségével felolvasott szavak jelennek meg. Végül a feldolgozott szöveg ismét fehér színű lesz.

Két darab almenü található a programban, amit a 9. ábra szemléltet. Az első a „settings”, ahol megadhatjuk a bemeneti szimulátor olvasási sebességét (*FPS*), a felhasználói azonosítót (*User ID*), a csatlakozáshoz szükséges kesztyű nevét (*Glove name*) és bekapcsolhatjuk az online szövegjavítást (*NLP*). Az utolsó két mezőből az elsőnek a

segítségével beállíthatunk a bemeneti szimulátor számára egy HaGDiL-okat tartalmazó fájlt, amivel tesztelhetjük a szoftver működését.



10. ábra. Almenük

A másodikkal egy olyan .csv (Comma-Separated Values) fájlt állíthatunk be, ami tartalmazza a betűkhöz rendelt hexadecimális HaGDiL értékeket.

Ezeket a 9. ábra jobb oldalán láthatjuk, a „hagdils” lapon. A lista bővíthet a későbbiekben, hiszen a felhasználó képes manuálisan hozzáadni szavakat, kifejezéseket, esetleg mondatokat.

A „reset” gomb megnyomásakor a szoftver azokat az értékeket olvassa be, amiket a „settings” lapon megadott .csv fájl foglal magában.

A „download” a felhasználói azonosító alapján letölti a portálon (*Backend Server*) lévő HaGDiL-okat. Amennyiben az „upload”-ra nyomunk a portálon lévő állományt frissíti a rendszer a telefonon tárolt információk alapján.

5. Megvalósítás

A kutatómunka befejeztével, minden olyan információt sikerült elsajátítanom, melyek nagyban segítik a programfejlesztés folyamatait. A kézfej anatómiája a háromdimenziós modellezést teszi hatékonyabbá, míg a mozgásokra jellemző információk a programozáskor beállított tartományokat és a forgatáshoz szükséges, helyes tengely választásokat biztosítják.

A daktil ábécé ismerete hozzájárul a hibakereséshez, ugyanis ránézésre el lehet dönteni, hogy a vizualizációs modul által szolgáltatott kéztartás megfelel-e a valóságnak, azaz a kesztyűben lévő kéz tartása megegyezik-e a kijelzőn látottakkal.

A Tolmácskesztyű architektúrális tanulmányozása megértette velem a meglévő szoftverrendszer működését, mely a továbbfejlesztést és a vizualizációs modul integrálását könnyíti meg, hiszen számít, hogy később, az általam készített osztályok hol és hogyan tudnak bekapcsolódni a kommunikációba.

A kezelőfelület tanulmányozása elősegíti a szoftver grafikus megjelenítésének a továbbtervezését, itt a gombok, oldalak elhelyezésére gondolok elsősorban. Fontos, hogy lehetőleg ne térjen el (stílusában) az integrált modul a mostani megjelenéstől, mivel ez lényegesen rontaná a felhasználói élményt.

A feladat tehát, egy olyan modul kifejlesztése, mely a helyes működés mellett az egyező stílust is képes garantálni.

A projekt megtervezése, elkészítése előtt figyelembe kell venni néhány szempontot, ami meghatározza az elkészült munka minőségét. A helyes fejlesztői környezetek kiválasztásakor érdemes utánajárni a felhasználási feltételeknek, milyen funkciókat biztosítanak és annak, hogy mennyire támogatottak az adott termékek. Amennyiben több különálló szoftvert kell igénybe venni, javasolt ellenőrizni ezek kompatibilitását, beleértve az operációs rendszert is.

5.1. Fejlesztőkörnyezetek összehasonlítása

A kézfej elkészítéséhez egy olyan háromdimenziós szerkesztő programra van szükség, ami az elkészített objektumot képes megfelelő formátumban exportálni, lehetőséget biztosít a modellt csontokkal, textúrával és animációval kiegészíteni.

A három legismertebb környezetet célszerű összehasonlítani: 3ds Max, Blender, Maya.

1. táblázat. Modellező környezetek összehasonlítása [14]

Szoftver	Gyártó	Támogatott platformok					
		Windows	Mac OS X	Linux	Unix	BSD	Egyéb
3ds Max	Autodesk	✓	-	-	-	-	-
Blender	Blender Foundation	✓	✓	✓	✓	✓	Solaris, AmigaOS 4, MorphOS
Maya	Autodesk	✓	✓	✓	-	-	-

Az 1. táblázat jól mutatja, hogy mindhárom szoftver számos fájlkiterjesztést támogat és rengeteg funkciót biztosít, ráadásul rengeteg oktatóvideót és fórumot találhatunk a termékek használatával kapcsolatban. A döntő szempont ezért a program használatához szükséges engedély. Mivel csak a Blender-t lehet ingyenesen használni korlátlan ideig, ebből adódóan egyértelmű a választás. Ezenfelül funkcionalitásában tökéletes és a legtöbb operációs rendszeren futtatható.

A Blender számos előnnyel rendelkezik, többek között kevés helyet foglal, megannyi operációs rendszert támogat, nyílt forráskódú (tökéletesítés esélye), a rengeteg billentyűkombináció gyorsítja a szoftver kezelését, sőt hatalmas a telepíthető kiegészítők listája. Véleményem szerint pár nap alatt – az oktató videók segítségével – könnyedén elsajátítható a kezeléshez szükséges gyakorlat, még annak is, aki ezelőtt nem kezelt háromdimenziós grafikai szerkesztő programot.

A vizualizáció elsősorban Windows Phone 8.0-ás operációs rendszerre íródik, tehát a programozáshoz használt környezet – egyéb összehasonlítás nélkül – a Microsoft Visual Studio 2012.

Egy új Windows Phone 8.0 típusú projekt kezdéséhez pár függőséget kötelező telepíteni:

- Windows Phone Software Development Kit 8.0 [15]
- Windows Phone SDK 8.0 Update 3 Emulators (javasolt) [16]
- Microsoft XNA Game Studio 4.0 [17]
- Games for Windows Marketplace Client [18]

A telepítés előtt indokolt a szoftverek hardveres követelményeinek áttekintése, ezeket a következő három táblázat foglalja össze.

2. táblázat. Blender - hardveres és szoftveres követelmények [19]

Blender			
	Minimum követelmény	Ajánlott követelmény	Optimális hardver
CPU	32 bites két magos, 2 GHz-es processzor, SSE2	64-bites négy magos processzor	64-bites nyolc magos processzor
RAM	2 GB	8 GB	16 GB
GPU	OpenGL, 256 MB RAM	OpenGL, 1 GB RAM	Dual OpenGL, 3 GB RAM
OS	Windows xp-től 8.1-ig, Mac OS X 10.6-tól, bármelyik Linux		

3. táblázat. Microsoft Visual Studio - hardveres és szoftveres követelmények [20]

Microsoft Visual Studio 2012	
CPU	1,6 GHz-es, vagy ennél gyorsabb processzor
RAM	1 GB, virtuális gép esetén 1,5 GB
GPU	DirectX 9-képes videokártya, minimális felbontás: 1024 x 768
HDD	10 GB szabad tárhely, minimum 5400 RPM-es sebességű
OS	Windows 7 SP1, Windows 8, Windows 8 Server 2008 R2 SP1, Windows Server 2012

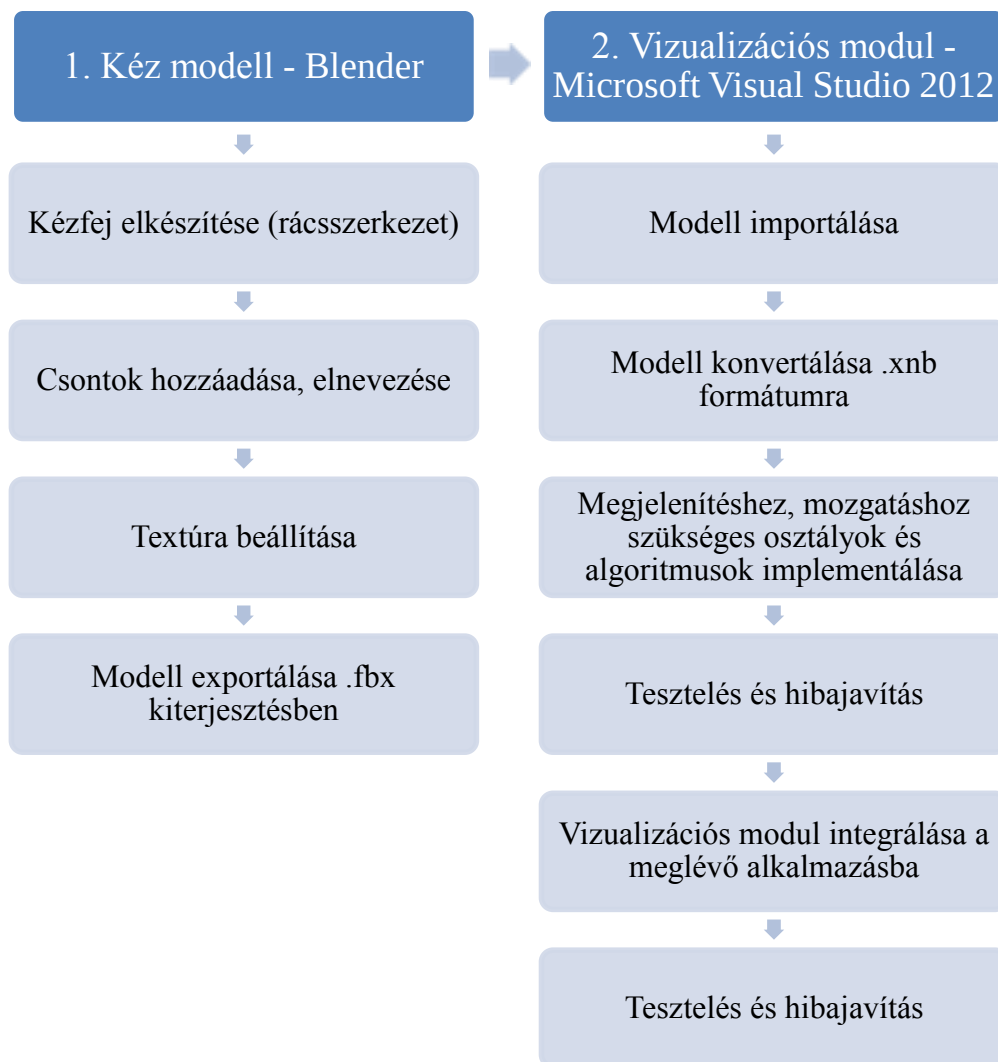
4. táblázat. Windows Phone SDK - hardveres és szoftveres követelmények [21]

Windows Phone SDK 8.0 + Windows Phone 8 emulátor	
CPU	64-bites, SLAT-képes processzor
RAM	4 GB
HDD	6,5 GB szabad tárhely
OS	Windows 8 professzionális kiadás, vagy annál jobb

Mivel a konfigurációm minden rendszerkövetelménynek megfelel, ezért nem szükséges alternatív megoldást keresnem a fejlesztői környezetek terén.

5.2. Programtervezés

A cél, hogy logikailag felépüljön a végleges adatstruktúra és a szükséges algoritmusok. Az átláthatóság kedvéért a következő ábra mutatja be a teljes folyamatot:



11. ábra. Folyamatterv

A folyamatterv elkészültével fontos meghatározni, hogy milyen osztályokra, változókra van szükség a helyes megvalósítás érdekében. Összesen öt osztály együttműködése teszi lehetővé a vizualizációt:

- Game1
 - AnimationClip
 - AnimationPlayer
 - Keyframe
 - Skinningdata
- megjelenítés, mozgathoz, bemenetek kezelése
- } segédosztályok

Egy, ami magáért a megjelenítésért, mozgatásért, bemenetek kezeléséért és a képernyő frissítéséért felel, valamint négy egyéb, amik a helyes importálást segítik elő.

Game1 osztály

Ez az osztály feldolgozza a kapott HaGDiL-okat, majd egy függvény segítségével transzformálja a csontok pozícióját. Egy előre meghatározott sebességgel (Frame Per Secundum) frissíti és újrarajzolja a modellt a kijelzőre.

AnimationClip osztály

Akkor van rá szükség, ha a például a Blender-ben előre készült animáció és a modell azzal együtt lett importálva.

A konstruktoron kívül két tulajdonsággal rendelkezik, amik az animációval kapcsolatos információkat hordozzák, így például annak teljes hosszát és a csontok mozgását.

AnimationPlayer osztály

Ez az osztály akkor is használatban van, ha nincs beépített animáció.

Először is, ha a beépített konvertáló metódus hozza létre az .xnb fájlokat, akkor több olyan változónak az értéke marad kitöltetlen, amiket fel kell használni a programkód alapú animációhoz.

Másodszor, ha a modell rendelkezik beépített animációval, ennek hatására lehetne elindítani azt.

Keyframe osztály

Az osztály feladata, hogy tárolja a csontok hivatkozási, más néven index számát, az animáció kezdési idejét és a csont transzformációkat.

SkinningData osztály

Tartalmazza a csontvázat, ami azt jelenti, hogy minden csonthoz tartozik egy szám, mely tudatja a környezettel annak pontos helyét a hierarchiában. Ezen felül a csontváz aktuális pozícióját és az animációkat is magában foglalja.

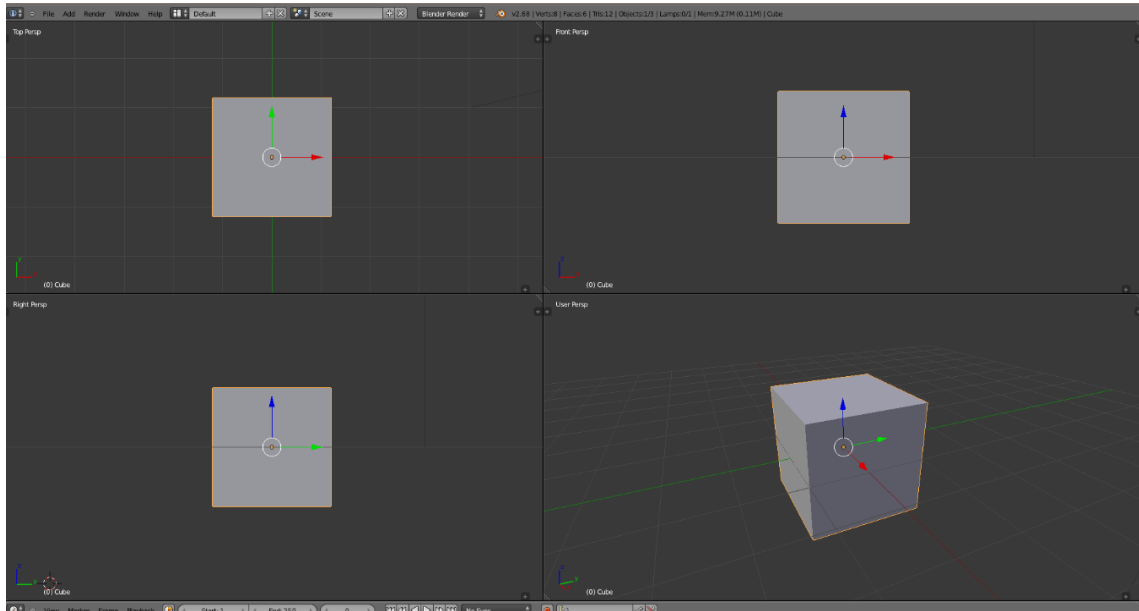
5.3. Implementáció

A megvalósítás során a folyamatterv szolgáltatja az alapot és az egyes pontjai mentén fogok haladni.

Az első feladat a kézfej elkészítése, ehhez a Blender legújabb verzióját veszem igénybe, ami jelenleg a 2.71-es verzió, 64 bites kiadás. Ezután az exportált modellt egy konverzió után tudom felhasználni a Visual Studio 2012-ben, ahol az előbb ismertetett osztályok építik fel a projektet.

5.3.1. Blender

A kezdeti testet megfelelő technikák alkalmazásával formázhatjuk (zsugorítás, nyújtás, forgatás, sokszögek duplikálása). A minél realisztikusabb végeredmény okán, fotó alapján készítettem el az átalakításokat. Ezt támogatja a program, sőt a képernyő felosztható, így biztosítja a felülnézetet, oldalnézetet és előlnézetet egy időben, amit a 12. ábrán láthatunk.



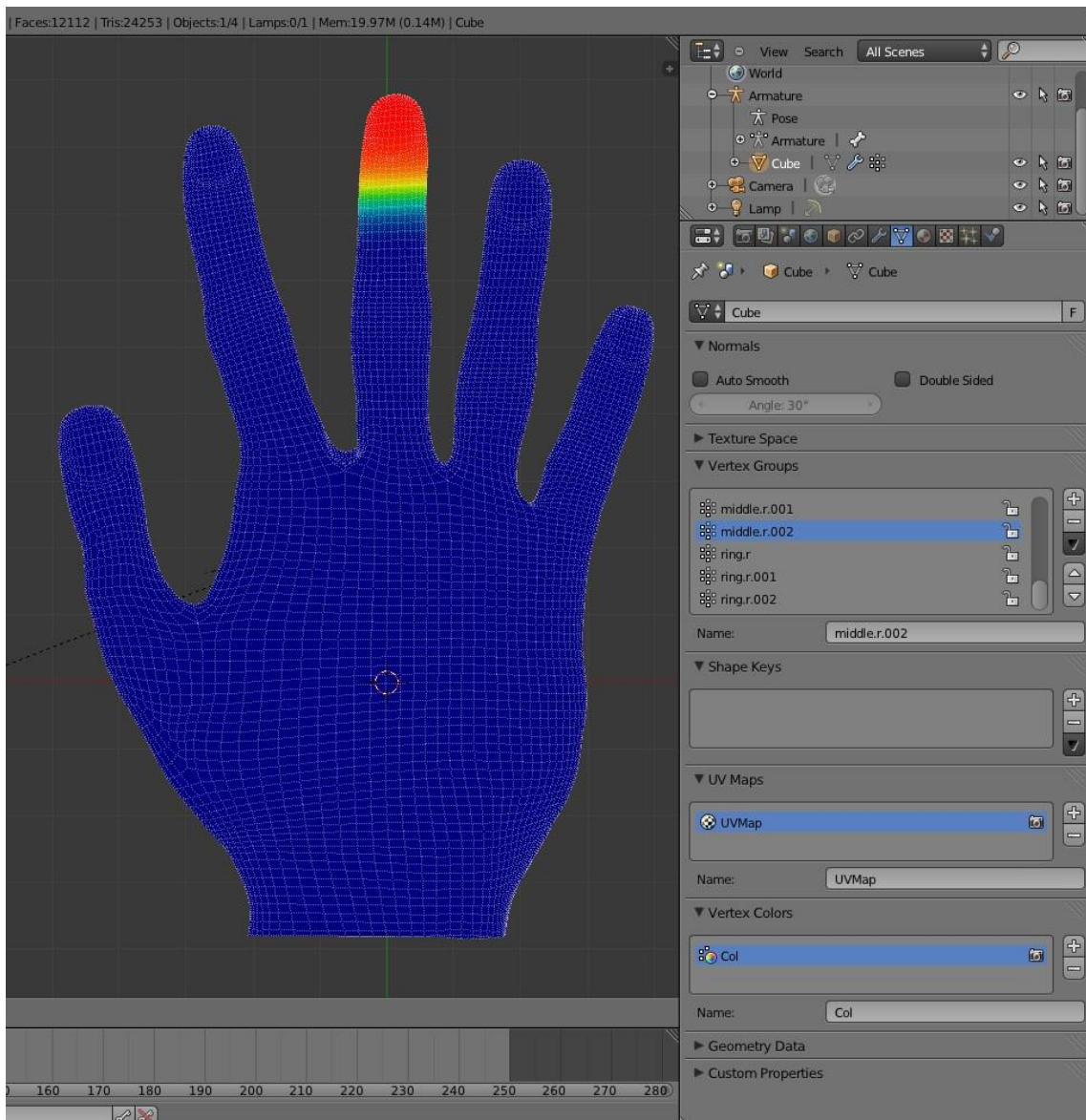
12. ábra. Blender - Kezdeti állapot

Alapértelmezetten a modell egyszínű (szürke), ezért saját textúra beállítását kívánja (*Texture Paint*, *UV Unwrap*). Kezdetben egyszerű barna színt állítottam be, majd később a kézfejemről készített képet illesztettem a testre.

A csontok hozzáadása, beállítása elengedhetetlen, hogy mozgásra bírjam a kezet. Ekkor meg kell adni a pontos hierarchiát (szülő-gyermek kapcsolatok), a csontok elforgatását (saját tengellyel rendelkeznek). Opcionális, de erősen javasolt minden csontot egyedi azonosítóval ellátni, így megkönnyítve a későbbi hivatkozásokat.

A folyamat a modellel való összekapcsolással végződik, e nélkül ugyanis csak a csontok fognak mozogni, maga a sokszögekből felépülő objektum mozdulatlan marad. Egy beépített függvény (*With Automatic Weights*) automatikusan képes kiszámolni, hogy egy csont mozgatása során a modell mennyire deformálódjon. Ajánlatos ezt alkalmazni és az előforduló hibákat kézzel kijavítani.

A színskála a deformálódás mértékét prezentálja, ahol a piros a legmagasabb érték és a kék a legalacsonyabb. A 13. ábra baloldalán, a kézfejen alkalmazott színskála szerepel.

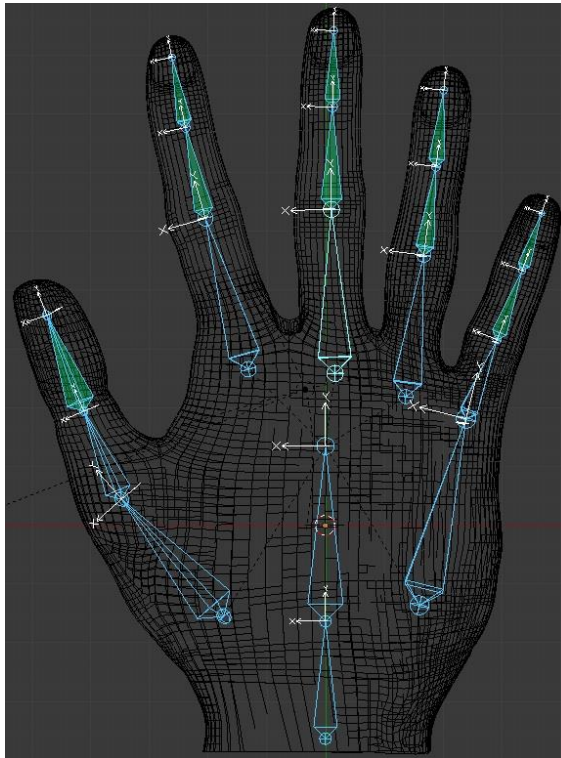


13. ábra. Blender - Deformálódás mértéke

Minden csonthoz külön-külön be kell állítani ezeket a paramétereket. A beépített algoritmus előnye, hogy szimmetrikusan állítja be az értékeket, míg a manuális megoldás esetében körülményes az arányok optimális megfestése.

A lenti ábra a befejezett kézfej rácsszerkezetét mutatja, benne a csontokkal és a hozzájuk tartozó tengelyekkel. A mutatóujj, középsőujj és a gyűrűsujj kézközépcsontjait egy darab

csont helyettesíti (kézfaj közepén), ám az ujjak alsópercei összeköttetésben vannak vele (felső szaggatott vonalak).



14. ábra. Blender - Csontokkal ellátott kézfej

Az elkészült munkát – a későbbi fejlesztést figyelembe véve – .fbx kiterjesztésben exportálom. Ennek oka, hogy a Microsoft Visual Studio ezt a fajta állományt képes kezelni.

Mentés előtti beállítások:

- -Z Forward
- Y Up
- XNA Rotation Animation Hack (Blender 2.68a)
- XNA Strict Options (Blender 2.68a)

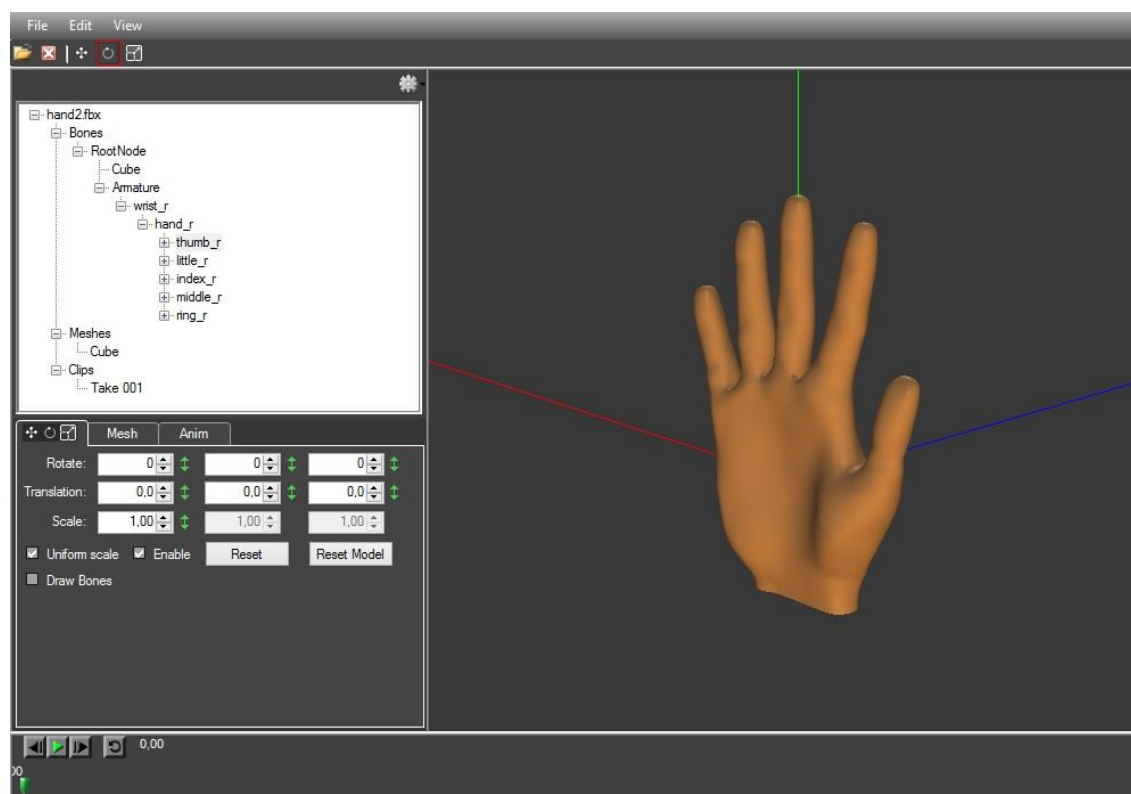
Fontos, hogy csak egy korábbi Blender verzióban érhető el a két XNA opció (Blender 2.68a) és a Microsoft Visual Studio más koordináta-rendszert használ, mint a legtöbb program, ezért a helyes exportálás érdekében a 2.68a verziót telepítettem és annak segítségével mentettem a modellt.

Szem előtt tartva a céleszközöket, csak a modell kerül exportálásra (kamera, lámpa tulajdonságok feleslegesek), így csökkentve a fájl méretét.

Az .fbx kiterjesztésű fájl felépítését tekintve egy .xml (Extensible Markup Language, Kiterjeszthető Jelölő Nyelv) dokumentum, ami tartalmazza egy objektum összes szükséges információját (csontok és sokszögek pozíciója, elnevezése, száma).

5.3.2. A kézmodell ellenőrzése

Az eredmény az XNA Model Viewer programmal ellenőrizhető (15. ábra). Ez a kiegészítő képes .fbx kiterjesztésű fájlokat betölteni és megjeleníteni. Amennyiben a modell nem rendelkezik textúrával – akár egyszínű, akár komolyabb – figyelmeztetést kapunk a hiányosságról és nem töltődik be az objektum.



15. ábra. XNA Model Viewer - Kézfej

A bal felső négyzetben a modell tulajdonságai látszódnak (csontváz, modell neve, exportált animáció), alatta pedig a manipulációs felület, melynek segítségével elforgathatjuk a csontokat XYZ tengelyek mentén, nagyíthatjuk, kicsinyíthetjük, valamint eltolhatjuk az objektumot. A *Draw Bones* kapcsoló kirajzolja, elrejtje a csontokat. Az alsó sáv lejátssza a kiválasztott animációt.

Csontok név alapján:

- wrist_r.....csukló
- hand_r.....kézközépcsontok (II., III., IV.)
- thumb_rhüvelykujj, kézközépcsont
- thumb_r_001hüvelykujj, alapperc
- thumb_r_002hüvelykujj, körömperc
- little_rkisujj, kézközépcsont
- little_r_001kisujj, alapperc
- little_r_002kisujj, középperc
- little_r_003kisujj, körömperc
- index_r.....mutatóujj, alapperc
- index_r_001.....mutatóujj, középperc
- index_r_002.....mutatóujj, körömperc
- ring_rgyűrűsujj, alapperc
- ring_r_001gyűrűsujj, középperc
- ring_r_002gyűrűsujj, körömperc

A modell vizsgálatának befejeztével környezetet váltok és megkezdem a vizualizációs modul implementációját.

5.3.3. Microsoft XNA Game Studio 4.0 [22]

A Microsoft az XNA eszközkészletet a Játékfejlesztői Konferencián, 2004 márciusában jelentette be. A .NET keretrendszeren alapuló XNA programok Windows-on, Windows Phone-on és Xbox-on képesek futni.

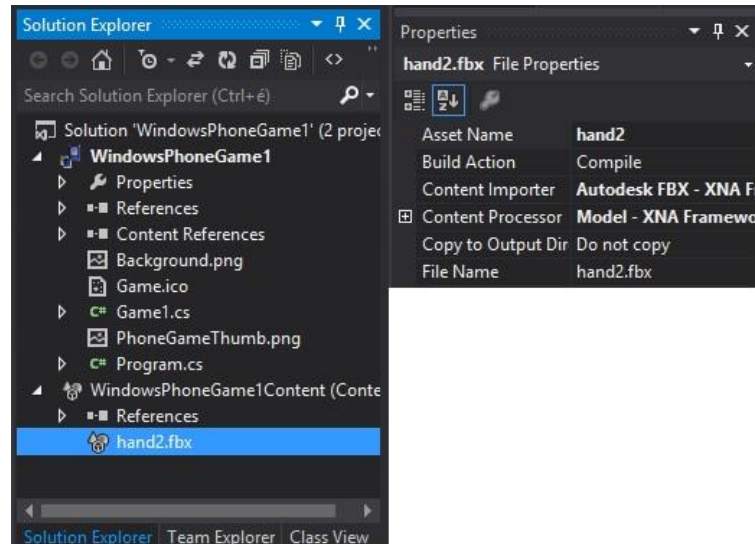
Érdemes megjegyezni, hogy az XNA támogatását a Microsoft 2014 áprilisában megszüntette, így a maximum választható Windows Phone operációs rendszer a 7.5-ös verzió. Ez a későbbiekben azért lényeges, mert a meglévő alkalmazás Windows Phone 8.0-as verzióra íródott (az XNA projektek csak felfelé kompatibilisek, visszafelé nem), ezért módosítás nélkül az XNA projektet nem lehet integrálni kompatibilitási problémák miatt.

Ahhoz, hogy a Visual Studio 2012 számára is értelmezhető legyen az .fbx kiterjesztésben mentett fájl, konvertálni kell azt, amit egy XNA – Windows Phone Game projekt hajt végre.

Egy ilyen megoldás két projektet tartalmaz, az első (WindowsPhoneGame) biztosítja a modell manipulálásáért felelős metódusokat, beállításokat (jelen esetben nem módosul a projekt által létrehozott osztály, mert a konvertáláshoz beépített eljárást alkalmazok).

A második az úgynevezett tartalom (Content) projekt, amely tartalmazza a használni kívánt fájlokat. Ezek lehetnek hangfájlok, képek, betűtípusok, illetve háromdimenziós objektumok. A projekt felépítését a 16. ábrán láthatjuk.

A tartalom projekthez, ha hozzáadjuk a korábban elkészített fájlt, ki kell választani egy legördülő listából, hogy milyen fájl típusra szeretnénk alkalmazni az átváltoztatást (Importer: Autodesk FBX – XNA Framework) és hogy milyen feldolgozással történjen (Content Processor – XNA Framework). Általában automatikusan felismeri a kiterjesztés alapján.

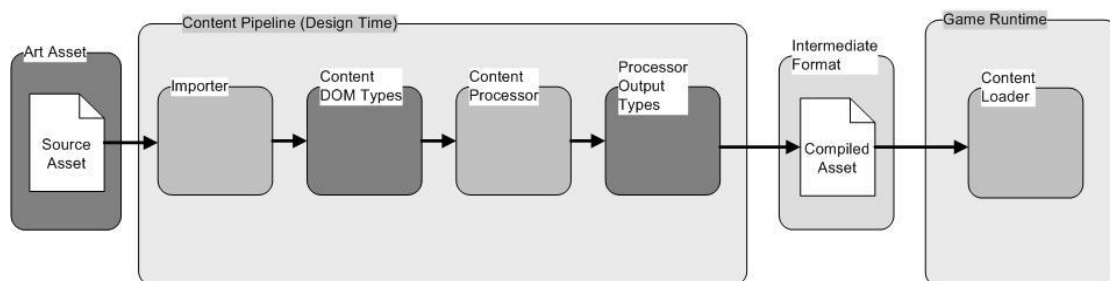


16. ábra. Microsoft Visual Studio 2012 - XNA Windows Phone Game projekt

Ezután a Build parancsot kiadva létrejön egy .xnb kiterjesztésű (bináris) fájl, amit a Studio már képes kezelni. Emellett előnye, hogy az előzőleg használt állomány méretének csupán a töredéke, körülbelül a harmada.

A létrejött fájl(oka)t a „Megoldásnév\Projektnév\bin\Windows Phone\Debug” mappában találjuk. Abban az esetben, ha a modellhez tartozott textúra, még egy állomány jön létre (általában textúranév_0.xnb).

A lenti ábrán jól látható, hogy a kezdeti fájlból miként lesz a környezet számára is értelmezhető, feldolgozható formátum. Az ezt követő folyamatokat egy másik projekt típusban kell elvégezni, ezért annak részletezését a soron lévő pontban fejtem ki.



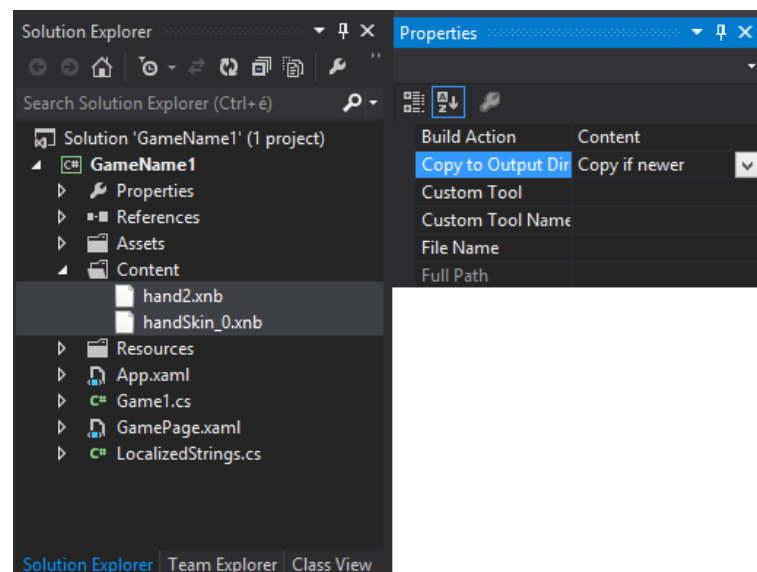
17. ábra. Microsoft Visual Studio 2012 - Konverzió [23]

5.3.4. Microsoft - MonoGame

Ahhoz, hogy Windows Phone 8.0-ára fejleszthessek, egy másik megoldást kell találni. Fejlesztők egy csoportja, a jól bevált XNA 4.0-ás API-t (Alkalmazásprogramozási Interfész) tovább korszerűsítették. Az eredmény egy platformfüggetlen környezet, amiben C# programnyelvben írhatok szoftvereket Android-ra, iOS-re, PlayStation Mobile-ra, OUYA konzolra, valamint Windows-ra és Windows Phone-ra. Továbbfejlesztésről lévén szó, olyan projekteket hozhatok létre, amik felépítésükben alig különböznek a megszokott XNA-ban látottaktól. [24]

Jelenleg a 3.2-es verzió a legfrissebb, viszont a telepítő – amit a hivatalos oldalról lehet letölteni – hibásan, a 3.0-ás verziót másolja fel a számítógépre, ami köztudottan számos hibát tartalmaz. Éppen ezért javasolt a Visual Studio NuGet Package Manager (olyan eszköz, amely leegyszerűsíti egy projekthez használt kiegészítő csomagok letöltését, telepítését, törlését, frissítését és beállítását) segítségével telepíteni, így kiszűrve az esetleges anomáliákat.

A 18. ábra egy új MonoGame Windows Phone 8.0 projektet szemléltet, ami már tartalmazza a korábban konvertált objektumokat:



18. ábra. Microsoft Visual Studio 2012 – MonoGame projekt

Itt a tartalom nem egy különálló projektben, hanem egy mappában (Content) foglal helyet. Ide kell bemásolni a kéz modellt, illetve a hozzá tartozó textúrát. A 18. ábra jobb oldalán láthatjuk a két állomány beállításait.

Ezután a 6. fejezet 2. alfejezete alapján létrehozom az öt osztályt és implementálom a szükséges metódusokat, változókat.

A *Game1* osztály kiemelt szerepet tölt be a programban, függvényeit a 19. ábra szemlélteti, melyek közül a feladat megoldása szempontjából a legfontosabbak a következők:

- *LoadContent*
- *UnloadContent*
- *Update*
- *Draw*
- *Movements*
- *DistalFlexion*
- *InermediateFlexion*
- *ProximalFlexion*
- *Crossing*
- *HandRotation*
- *FixTag*
- *BoneIndex*



19. ábra. *Game1* osztály

A *LoadContent* függvény létrehoz egy új *SpriteBatch* változót, amit később a rajzoláshoz lehet felhasználni.

Betölti a modell fájlt egy *hand* nevű változóba, majd meghívja a *FixTag* metódust, ami kijavítja az importálás hiányosságait, így a csontokkal kapcsolatos indexeléseket és a pozíciókat tároló mátrixokat. Az így kreált elemeket összefogja egy új osztályban (*SkinningData*) és a modell *Tag* tulajdonságába menti azt. A továbbiakban a *FixTag* és a *LoadContent* algoritmusra nincs szükség.

Ezekhez hasonlóan a *BoneIndex* is csupán egyszer hívódik meg az előbb ismertetett függvény végén. Feladata, hogy egy csontokat tartalmazó tömböt alkosson, ahol számít a sorrend, ugyanis a bemeneti értékek előfordulási helyének hivatkozási száma információt szolgáltat a mozgatni kívánt csontról. Például az adatfolyam negyedik eleme a mutatóujj felső ujjpercének az elmozdulási értékét adja, ezért logikus, hogy a tömbben is ez a sorrend érvényesüljön.

Az *Update* a beállított keret/másodperc alapján kicseréli a csontok pozíciójáért felelős értékeket az éppen aktuálisakra, majd frissíti mind a modellt, mind pedig a kijelzőt.

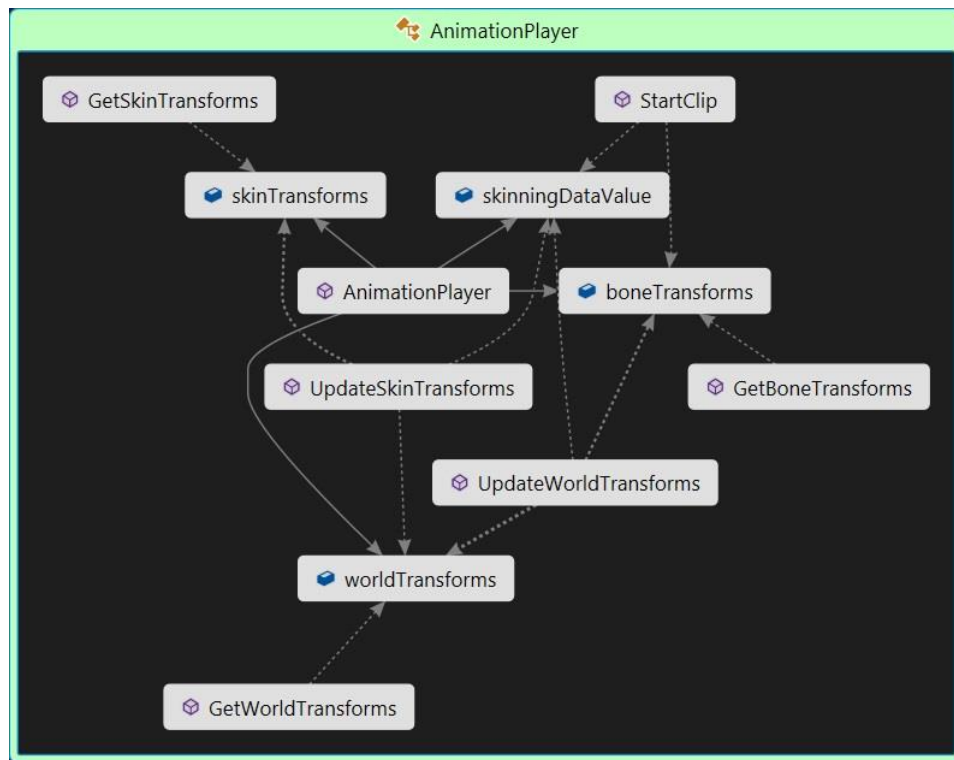
A *Draw* eljárás kezdetben feketére törli a kijelzőt, majd kiszámítja a kamera helyzetének a mátrixait (vetület, nézet). Befejezéskeppen rendereli a kézfejet, azaz árnyékokkal, tükröződésekkel teszi élethűbbé a háromdimenziós megjelenítést.

A *Movements* a csontok alapján meghívja a szükséges függvényeket és a kívánt tengely mentén forgatást végez a bemeneti paraméterek szerint:

- *DistalFlexion* - körömperc hajlítása, nyújtása
- *InermediateFlexion* - középperc hajlítása, nyújtása
- *ProximalFlexion* - alapperc hajlítása, nyújtása
- *Crossing* - ujjak keresztezése
- *HandRotation* – kézfej forgatása

Végül, de nem utolsó sorban az *UnloadContent* törli a memóriából a kézfejet tartalmazó változót, ha arra már nincs szükség. Ez azért lényeges, mert mobileszközökről lévén szó, figyelembe kell venni az erőforrások korlátozottságát.

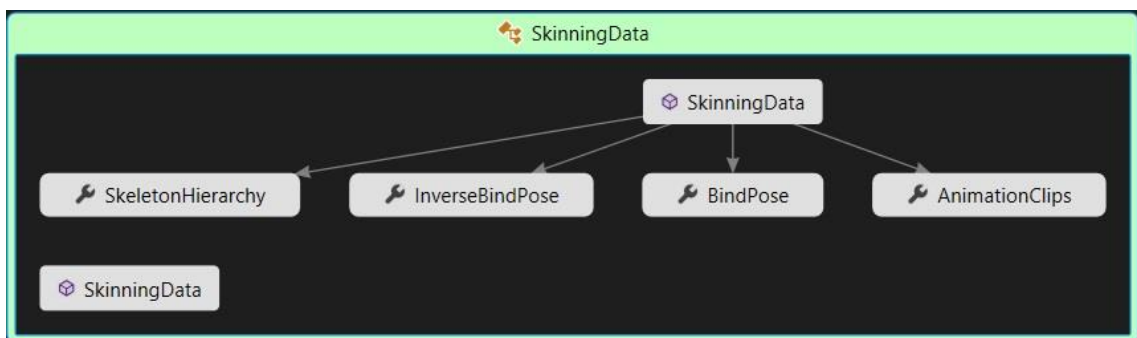
Az *AnimationPlayer* tartalmaz minden olyan metódust (20. ábra), ami hozzájárul a látványos vizualizációhoz. Amennyiben a modellhez létezik animáció (jelen esetben nem, ezért kommentként szerepel a programkódban), dekódolja azt.



20. ábra. *AnimationPlayer* osztály

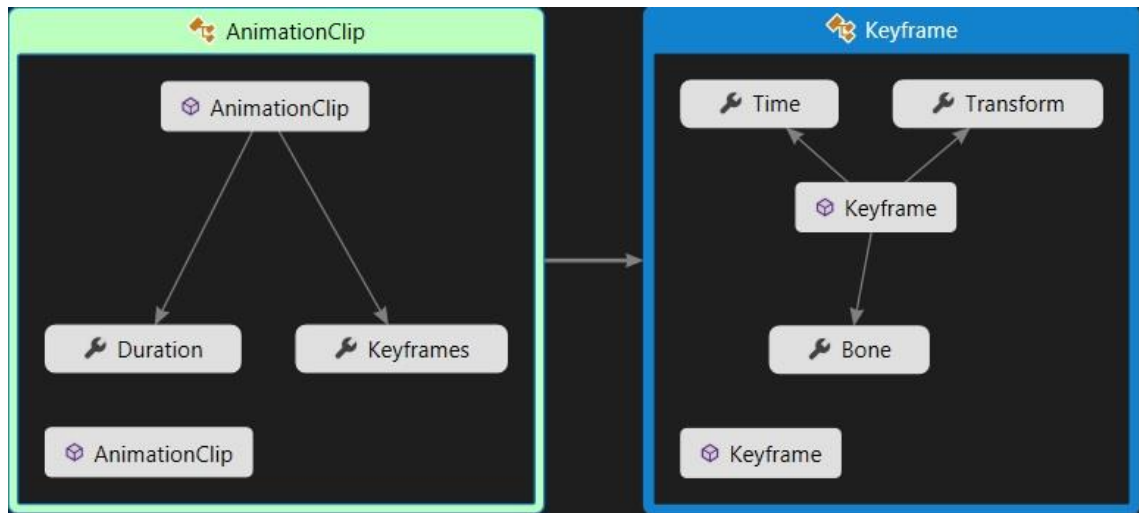
Az *Update* függvény igénybe veszi algoritmusait, melyek frissítik a csontok és a kézfelszín pontjainak pozícióját.

A *SkinningData* osztály felépítése a 21. ábrán látható. Tartalmazza a modellben szereplő animációkat, a csontvázat és a csontok koordinátáit a megfelelő mátrixokban.



21. ábra. *SkinningData* osztály

Az *AnimationClip* és *Keyframe* osztályok működése az 5. fejezet 2-dik pontjától nem tér el, ezért felépítésüket kizárólag a 22. ábra demonstrálja.



22. ábra. *AnimationClip* és *Keyframe* osztályok

Osztályok közötti kapcsolatok összefoglalva, ahol a jobbra mutató nyíl egy új osztály létrehozását jelzi:

1. *Game1* → *AnimationPlayer*, *SkinningData*
2. *AnimationPlayer* → *SkinningData*, *AnimationClip*
3. *SkinningData* → *AnimationClip*
4. *AnimationClip* → *Keyframe*

Látható, hogy a *Game1* hatására jönnek létre a segítő példányok, amik sorban hozzák létre a velük kapcsolatban álló egyéb osztályokat.

5.3.5. Integráció

Mivel a MonoGame projekt céleszköze Windows Phone 8.0, vagy annál újabb, az integráció nem okoz különösebb problémát. Az átláthatóság érdekében, a meglévő projektben, egy külön mappában foglal helyet a vizualizációs modul. A helyes működés megköveteli az egységes névterek használatát. A kezdeti Tolmácskesztyű program felépítése jóval bonyolultabb, ezért a vizualizációs modul névtére alkalmazkodik.

A szoftvert egy lappal, úgynevezett Pivot Item-mel (*handPivotItem*) kell kiegészíteni, hogy láthatóvá váljék a kézfej. Négy vezérlőgombot helyeztem el a kijelző alján, amik érintésre forgatják, közelítik, illetve távolítják a modellt.

A *Game1* osztályban két segítő algoritmusra és három segédváltozóra van szükség, ami a szenzorok által küldött adatfolyamot egy bizonyos formátumban fogadja és továbbítja a megfelelő metódusnak (*pollHagdilFromQueue*). A sikeres integrációt a 23. ábra illusztrálja.



23. ábra. Háromdimenziós kézfej (Windows Phone 8.0)

5.4. Szoftver tesztelése

A minőség és megbízhatóság érdekében szükséges a vizualizációs modul tesztelése – először önmagában, majd integrálva –, ami három fő részből tevődik össze:

- szintaktikai ellenőrzés
- szemantikai ellenőrzés
- ellenőrzés emulátoron, mobil eszközön

5.4.1. Szintaktikai ellenőrzés

A Visual Studio 2012 beépített forráskód ellenőrző funkciója valós időben figyelmezteti a fejlesztőt az elkövetett hibákról. A súlyosabb hibák (Errors) megakadályozzák a program fordítását, ezért azokat mindenképpen ki kell javítani, ha futtatni szeretnénk munkámat.

Az alacsonyabb prioritású, úgynevezett figyelmeztetéseket (Warnings), nem fontos korrigálni, csupán ajánlott. Belátható, hogy a szintaktikai ellenőrzés nem elégséges, mindössze az alapját képezi a további teszteléseknek.

5.4.2. Szemantikai ellenőrzés

A soron követő fázis teszi ki a tesztelés tekintélyes hányadát. A forráskódot sorról sorra át kell nézni, összehasonlítani a programtervezéskor használt folyamatábrát a véglegesnek vélt osztálydiagramokkal. Meg kell vizsgálni, hogy a függvények az elvártan megfelelően működnek-e, továbbá nélkülözhetetlen a kivételkezelés használata ezzel megelőzve a futásidejű végzetes hibát.

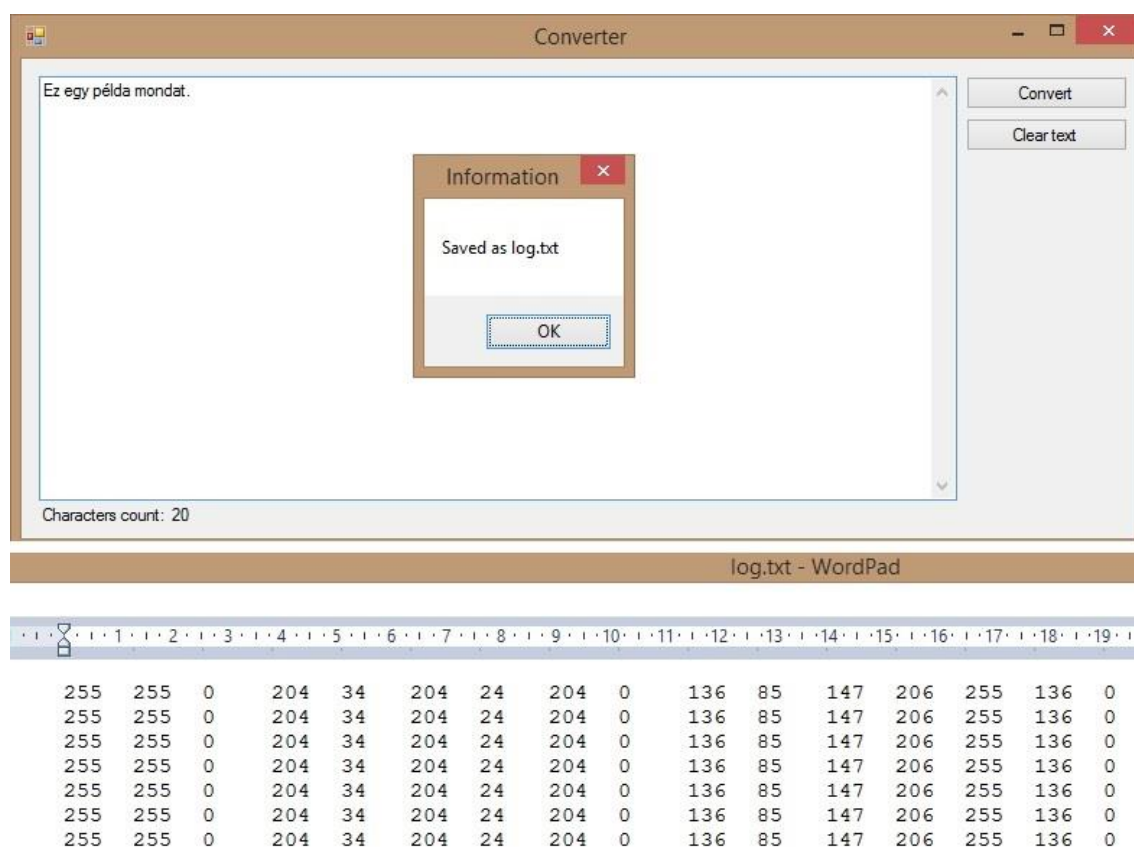
Ezenfelül létfontosságú az integrált modul működésének helyességét is garantálni, ahol azt ellenőrizzük, hogy a feldolgozó osztály által előállított értékeket megkapja-e a Game1 osztály – a korábban ismertetett függvény segítségével – és a kézfejet manipuláló gombok továbbítják-e a kérést.

Két feladattal zárul a vizsgálat. Az első, minden olyan változót, függvényt, segédosztályt eltávolítok a forráskódból, amire már nincs szükség. Jelen esetben a segédosztályokban vannak olyan változók, függvények, amiket nem használok, ezért átmenetileg kommentnek jelölöm azokat. A második ennél időigényesebb. Az összes csont mozgásának felülvizsgálata, kezdetben egyenként, majd egyszerre. Az ellenőrzés rávilágított a helytelenül deklarált tartományokra, ennek köszönhetően átadás előtt sikerült orvosolni a problémát.

5.4.3. Szoftver tesztelése emulátoron és mobil eszközön

Mivel a tesztelés nagy részét számítógépen végzem, felmerült az igény egy konvertáló segédprogramra, mely kiváltja a kesztyűt, mint bemenetet előállító eszközt. Ennek feladata, hogy a felhasználó által megadott karaktereket – amik alkothatnak akár mondatokat is – HaGDIL-okra fordítsa és egy olyan szöveges fájlt hozzon létre, amit az implementált applikáció módosítás nélkül be tud olvasni, majd bemenetként kezelni.

A 24. ábra a konvertert és az általa készített szöveges fájl tartalmát vázolja.



24. ábra. Text2Hagdil konvertáló program

A program a megadott szöveget úgy alakítja át és menti el egy szövegfájlba a háttérben futó eljárásoknak köszönhetően, hogy az emulátoron futó Tolmácskesztyű alkalmazás képes legyen beolvasni.

A *Text2Hagdil* egy egyszerű Windows Form Application (Windows-on futtatható ablakos alkalmazás), mely indításkor először beolvassa a .csv fájlban (a HaGDiL értékek egy karakter által szeparált fájlban foglalnak helyet) szereplő HaGDiL-okat, majd a szövegdobozba írt karaktereket konvertálja egy kattintásra. Az eljárás során a hexadecimális számokból decimálisakat állít elő, majd a beolvasott állomány soraiból kikeresi a karakterre illő HaGDiL-t. Annak érdekében, hogy a háromdimenziós kézfej mozgása élethű legyen, a betűk kitarását és a köztük lévő átmenetet egy segítő algoritmus biztosítja. A folyamat végeztével figyelmezteti a felhasználót a sikeres fájl létrehozásról.

Az 5. táblázat bemutatja, hogy annak ellenére, hogy az implementációt programtervezés előzte meg, a megannyi módosítás hatására több nem várt hiba került a szoftverbe. Például az, hogy a csontok hivatkozási számát helytelenül adtam meg, ki sem derül az első, illetve második tesztetéből. A másodiknál ugyanis a körömperc is behajlik a középperc függvényében, így elfedve a rendellenességet (a körömperc értékét az alappercen alkalmazza). Éppen ezért létfontosságú, hogy minden egyes részt – jelen esetben a csontokat – először külön-külön ellenőrizzek, majd egyszerre.

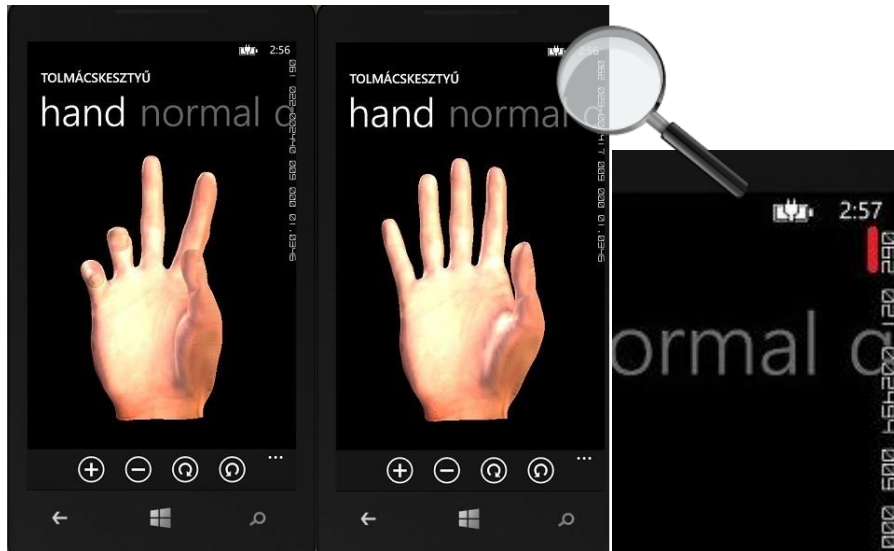
5. táblázat. Tesztelési jegyzőkönyv - részlet

ID	Teszteteset	Elvárt eredmény	Kapott eredmény	Értékelés	Teendő
1	Navigáció	Gombok lecserélődnek	Az egyik gomb nem változik	Nem megfelelő	A „zoom in” gomb megvizsgálása
2	Ujjpercek hajlítása	Ökölbe szorított kéz	Az ujjak összezárnak	Megfelelő	-
3	Középperc behajlítása	A körömperc is behajlanak	A körömperc érintik a tenyeret	Megfelelő	-
4	Ujjak összezárása	Az ujjak nem csúsznak egymásba	Az ujjak csak érintik egymást	Megfelelő	-
5	Alappercek hajlítása	Csak az alappercek hajlik, a közép- és körömperc nem	Az alapperc nyújtva, a közép – és körömperc behajlik	Nem megfelelő	Csontokat tartalmazó tömb felülvizsgálata
6	Csukló körzés	Kézfej mozgása előre, hátra, balra, jobbra	Mindkét tengely mentén helyes mozgás	Megfelelő	-
7	Mozgástartományok	Fiziológiailag helyes mozgás	Hüvelykujj kicsavarodik	Nem megfelelő	Hüvelykujj javítása Blender-ben
8	Kamera forgatás	Látszólag a kézfej elfordul	Kamera helyzetének változása (Y)	Megfelelő	-
9	Kamera közelítése, távolítása	Látszólag a kézfej nagyobb, illetve kisebb lesz	Kamera helyzetének változása (Z)	Nem megfelelő	Változtatás értéke túl magas
10	Bemeneti szimulátor	Helyes mozgás log fájl alapján	Lassú, de helyes mozgás	Részben megfelelő	FPS növelése (30 → 60)

Ezeket a problémákat viszonylag rövid idő alatt lehet orvosolni, a nagyobb gondot az jelenti, ha a folyamatban egészen vissza kell térni a kiindulópontához. A hüvelykujj javítása például megkövetelte az újbóli exportálást, konvertálást és importálást is.

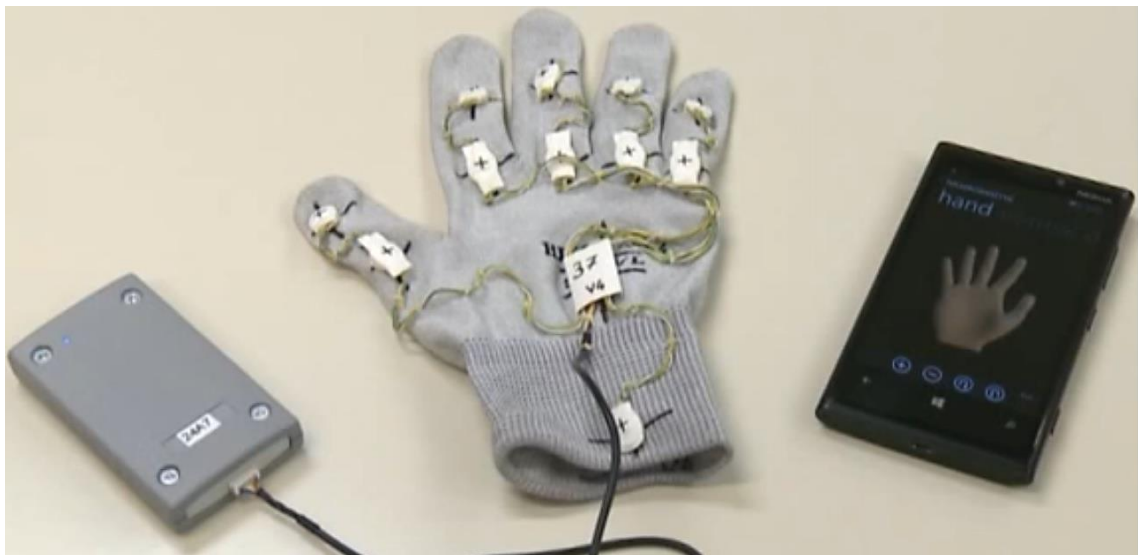
5.5. Végeredmény

A hibaellenőrzéseknek, teszteknek hála, minden lényeges rendellenességet sikerült kiszűrni, beleértve az apróbb visszasságokat, amik a tesztelési jegyzőkönyvben nem szerepelnek, mivel a helyes működést nem befolyásolták. Ezenfelül, az alkalmazott kivételkezelés biztosítja, hogy a szoftver folyamatosan tudjon működni, az elvártaknak megfelelően.



25. ábra. Textúrával ellátott modell - tenyér

Az itt látható pillanatképek a végleges, textúrával ellátott modellt ábrázolják, miközben a kesztyűben lévő kéz mozgását másolják valós időben. A jelenlegi beállítást mutatja az oldalsáv, mely szerint 62 képkocka/másodperc gyorsasággal dolgozik a vizualizáció. Az utolsó kép az elkészült rendszert ábrázolja.



26. ábra. A Tolmácskesztyű eszközrendszer [25]

5.6. Továbbfejlesztési lehetőségek

A kész szoftver működése hibamentes és kielégítő, viszont nem teljes a továbbfejlesztés nélkül. Számos olyan funkció vár megvalósításra, amik teljesebbé teszik a programot.

A gesztus alapú kamera manipuláció jelenleg fejlesztés alatt áll, célja, hogy a felhasználó a kijelzőn megjelenő gombok helyett az ujjával tudja közelíteni, távolítani, illetve forgatni a háromdimenziós modellt. Ennek köszönhetően gyorsabb és pontosabb kezelést biztosít mind a fejlesztőknek, mind a felhasználóknak.

Amennyiben garantálni szeretnénk a felhasználók részére, hogy a szótárat saját szavaikkal, kifejezéseikkel bővíthessék, a vizualizációt nyújtó lapot ki kell egészíteni egy mentés gombbal, ami tárolja az aktuális kéztartás HaGDIL-jait a .csv fájlban, majd listázza a „hagdils” menüpontban.

Mivel az applikáció nem kizárólag beszéd- és halláskárosultak számára készült, ezért tanácsos úgy módosítani a programot, hogy rendelkezzen egy oktató résszel. Azoknak kínálna alternatívát, akik könyvek, vagy internet nélkül szeretnék elsajátítani a daktil jelnyelv helyes használatát. A virtuális kéz eljeleli a bevitt szöveget, sőt a folyamatot meg is lehet fordítani úgy, hogy a szoftver kéri a felhasználót, hogy mutogassa el az általa kiírt karaktereket.

Végezetül, az Android alapú készülékekre szánt Tolmácskesztyű alkalmazás implementálása megelőzte a Windows Phone-ét, ezért a jövőben, azokra a készülékekre is lehetővé kell tenni a vizualizációs modult. Mivel a kézfej megjelenítéséért felelős programrész platformfüggetlen (MonoGame) környezetben íródott, apróbb módosításokkal integrálhatóvá és futtathatóvá lehet tenni.

6. Összefoglalás

A Tolmácskesztyű alkalmazás a beszéd- és halláskárosultak életét hivatott megkönnyíteni. A kézfej anatómiai és kinematikai tanulmányozására alapozva egy olyan vizualizációs modult tudtam az alkalmazáshoz implementálni, mely követi a kesztyűben lévő kéz mozgását, így egyaránt segítséget nyújt a program felhasználóinak és fejlesztőinek.

A fejlesztők a szenzorok által küldött értékek validálását az általam készített háromdimenziós kézfejjel felgyorsíthatják, hiszen elég ránézni a kijelzőre és azonnal eldönthetik, hogy a kesztyűben lévő kéz tartása megegyezik-e a mobil eszközön látottal. A konverter, az emulátoros hibakeresést és tesztelést teszi teljessé. A kívánt szöveget pár pillanat alatt lefordítja a Tolmácskesztyű által is értelmezhető formátumra, amit a vizualizációs modul képes módosítás nélkül megjeleníteni.

A gyorsaság és egyszerűség kedvéért, a virtuális kézfej manipulálásáért felelős gombokat (közelítés-távolítás, forgatás), egy algoritmussal ki lehet váltani, mely lehetővé teszi a felhasználó számára, hogy gesztusokkal befolyásolhassa a perspektívát. A szoftver szótára lehetőséget biztosít a saját szavakkal való feltöltésre, így a terméket alkalmazók egyedi kéztartásokhoz egyedi jelentést párosíthatnak, amiket egy gombnyomással elmenthetnek a központi szerverre. A szerver tárolja a személyre szabott szótárakat és lehetőséget biztosít a gesztusok megosztására a portálon keresztül.

Figyelembe véve, hogy az emberek különböző operációs rendszert használnak mobil eszközeiken, a vizualizációs modult elérhetővé kell tenni Android és iOS alapú készülékeken is. Mivel a modult platformfüggetlen környezetben fejlesztettem, ezért kisebb módosításokkal fel lehet készíteni nem Windows operációs rendszerre is.

Végül, számomra is rendkívül hasznos volt mind a kutatómunka, mind a programtervezés, megvalósítás. Korábban háromdimenziós szerkesztő programmal nem dolgoztam és élveztem, hogy tanulhatok olyan eljárásokat, amiket a későbbiekben is alkalmazhatok. Azt különösen kedveltem benne, hogy a fejlesztéssel számos ember hétköznapiját könnyíthetjük meg, sőt egy valós projekten dolgozhattam olyan fejlesztőkkel, mérnökökkel együtt, akik nagyban hozzájárultak a szakmai fejlődésemhez.

Ezúton szeretném megköszönni Dr. Póserné Valéria (intézményi konzulens), Mátételki Péter (külső konzulens) és az egész MTA SZTAKI Elosztott Rendszerek osztálya által nyújtott segítséget.

7. Abstract

The Interpreter Glove (Tolmácskesztyű) application is intended to ease the everyday struggle of hearing and speech impaired people. I could implement such a visualization module to the application thanks to the anatomical and kinematics examination of the hand, which can simulate the motion of the hand in the glove.

The developers can speed up the validation of the values sent by the sensors with the three-dimensional hand prepared by me, since it is serve to look at the display and it may be decided promptly, that the position of the hand in the glove is equal to the one seen on the mobile device. The converter makes the troubleshooting and testing on the emulator full. It translates the desired text in no time into the format, which can be interpreted by the Interpreter Glove. After that, the visualization module is able to display it without the need of modification.

For the sake of velocity and simplicity the manipulation buttons of the virtual hand can be substituted with an algorithm, which enables to the users to influence the perspective via gestures. The dictionary of the software provides an opportunity to upload and save own words on the server, so that, the users of the product can assign unique meanings to the unique hand positions. The server stores the personalised dictionaries and opens the door to share the gestures through the internet site.

Taking into consideration that the people use a different operating system on their mobile devices, it is necessary to make the visualisation module available on Android and iOS based apparatus. Because I developed the modul in cross-platform environment, it is possible to prepared to a non-Windows based operating system with tiny modifications.

Finally all the research work, software designing and the implementation was excessively useful for me. Previously, I haven't work in a three-dimensional editor tool and I enjoyed to learn such methods I can utilize in the future. I liked it especially, that we may facilitate the weekdays of several person with the development, moreover I could participate in a real project such engineers and developers who contributed to my vocational growth on a large scale.

Hereby I would like to thank all the support I get from Dr. Valéria Póserné (institutional consultant), Péter Mátételki (external consultant) and last but not least the whole department of the Distributed Systems of MTA SZTAKI.

8. Irodalomjegyzék

[1] Az anatómiai síkok és irányok:

Dr. Tarsoly, E.: Funkcionális anatómia. *Medicina Könyvkiadó Zrt.*, 2007.

[2] Anatómiai irányok (1. ábra)

Dr. Donáth, T.: Anatómiai atlasz. *Medicina Könyvkiadó Zrt.*, 1983.

[3] A kézfej csontjai (2. ábra)

Fordította: Karácsonyi, Z.: Anatómiai atlasz. *Holló és Társa Könyvkiadó*, 2000.

[4] Kézfej ízületei

Dr. Hollósy, T.: Anatómia (jegyzet). *PTE Általános Orvostudományi Kar*, 2005.

[5] Kézfej szabadságfoka

ElKoura, G., Singh K.: Handrix: Animating the Human Hand. *Symposium on Computer Animation*, 2003.

[6] Jelnyelv

Crystal D.: A nyelv enciklopédiája. *Osiris Kiadó*, 2003.

[7] Fonomimikai ábécé

Tomcsányiné Czukrász, R.: A fonomimikai módszer Magyarországon. *Lampel R. Könyvkiadó Vállalata*, 1934.

[8] Fonomimikai ábécéje (5. ábra)

Vincze T.: Alapfokú Kommunikációs Tanfolyam (jegyzet), *Hallássérültek Országos Szövetsége*, 1993.

[9] A daktil jelnyelv ábécéje (6. ábra)

(<http://www.deafnsobrinseattle.com/home/asl-alphabet>),
utoljára megtekintve: 2014-04-21.

[10] Motion capture

(<http://hirek.prim.hu/cikk/9405/>), utoljára megtekintve: 2014-05-01.

[11] Mátételki, P.: Mutasd a hangod – automatikus jeltolmács. *Networkshop 2014 konferencia*, Pécs, 2014. április 23-25.

[12] Tolmácskesztyű

(<http://dsd.sztaki.hu/hu/projects/TolmacsKesztyu>), utoljára megtekintve: 2014-05-10.

[13] Architektúrális felépítés (7. ábra)

(<https://kesztyu.dsd.sztaki.hu/>), utoljára megtekintve: 2014-11-01.

- [14] Modellező környezetek összehasonlítása (1. táblázat)
(http://en.wikipedia.org/wiki/Comparison_of_3D_computer_graphics_software),
utoljára megtekintve: 2014-09-21.
- [15] Windows Phone Software Development Kit 8.0
(<http://www.microsoft.com/en-us/download/details.aspx?id=35471>),
utoljára megtekintve: 2014-09-20.
- [16] Windows Phone SDK 8.0 Update 3 Emulators
(<http://www.microsoft.com/en-us/download/details.aspx?id=41559>),
utoljára megtekintve: 2014-09-20.
- [17] Microsoft XNA Game Studio 4.0
(<http://www.microsoft.com/en-us/download/details.aspx?id=23714>),
utoljára megtekintve: 2014-09-20.
- [18] Games for Windows Marketplace Client
(<http://www.xbox.com/en-US/LIVE/PC/DownloadClient>),
utoljára megtekintve: 2014-09-20.
- [19] Blender – hardveres és szoftveres követelmények (2. táblázat)
(<http://www.blender.org/download/requirements>), utoljára megtekintve: 2014-09-28.
- [20] Microsoft Visual Studio - harveres és szoftveres követelmények (3. táblázat)
(<http://www.techulator.com/resources/7422-Visual-Studio-2012-System-requirements-hardware-requirements.aspx>), utoljára megtekintve: 2014-09-28.
- [21] Windows Phone SDK - hardveres és szoftveres követelmények (4. táblázat)
(<http://www.microsoft.com/en-us/download/details.aspx?id=35471>),
utoljára megtekintve: 2014-09-23.
- [22] Microsoft XNA
(<http://msdn.microsoft.com/en-us/library/bb200104.aspx>),
utoljára megtekintve: 2014-10-22.
- [23] Microsoft Visual Studio 2012 – Konverzió (16. ábra)
(<http://msdn.microsoft.com/en-us/library/bb447745.aspx>),
utoljára megtekintve 2014-10-22.
- [24] MonoGame
(<http://www.monogame.net/about/>), utoljára megtekintve: 2014-10-22.
- [25] A Tolmácskesztyű eszközrendszer (26. ábra)
(http://www.rtlklub.hu/most/57637_rtl_hirado_-_esti_kiadas),
utoljára megtekintve: 2014-09-11.

9. Ábrajegyzék

1. ábra. Anatómiai irányok	5
2. ábra. A kézfej csontjai	7
3. ábra. Csuklóízület mozgástartománya	8
4. ábra. Ujjak mozgástartománya	9
5. ábra. Fonomimikai ábécé	11
6. ábra. A daktil jelnyelv ábécéje (angol)	12
7. ábra. Architekúrális felépítés	15
8. ábra. A Tolmácskesztyű szoftver kezelőfelülete	18
9. ábra. Normál ablak - szöveg	19
10. ábra. Almenük.....	20
11. ábra. Folyamatterv	24
12. ábra. Blender - Kezdeti állapot	26
13. ábra. Blender - Deformálódás mértéke	27
14. ábra. Blender - Csontokkal ellátott kézfej	28
15. ábra. XNA Model Viewer - Kézfej.....	29
16. ábra. Microsoft Visual Studio 2012 - XNA Windows Phone Game projekt.....	31
17. ábra. Microsoft Visual Studio 2012 - Konverzió.....	31
18. ábra. Microsoft Visual Studio 2012 – MonoGame projekt	32
19. ábra. Game1 osztály.....	33
20. ábra. AnimationPlayer osztály.....	35
21. ábra. SkinningData osztály	35
22. ábra. AnimationClip és Keyframe osztályok.....	36
23. ábra. Háromdimenziós kézfej (Windows Phone 8.0)	37
24. ábra. Text2Hagdil konvertáló program.....	39
25. ábra. Textúrával ellátott modell - tenyér.....	41
26. ábra. A Tolmácskesztyű eszközrendszer	41

10. Táblázatjegyzék

1. táblázat. Modellező környezetek összehasonlítása.....	22
2. táblázat. Blender - hardveres és szoftveres követelmények	23
3. táblázat. Microsoft Visual Studio - hardveres és szoftveres követelmények	23
4. táblázat. Windows Phone SDK - hardveres és szoftveres követelmények.....	23
5. táblázat. Tesztelési jegyzőkönyv - részlet.....	40